Design Differentiated Service Multicast With Selfish Agents

WeiZhao Wang, *Student Member, IEEE*, Xiang-Yang Li, *Associate Member, IEEE*, and Zheng Sun, *Member, IEEE*

***Abstract*— Differentiated service (DiffServ) is a mechanism to provide the Quality of Service (QoS) with a certain performance guarantee. In this paper, we study how to design DiffServ multicast when every relay link is an independent selfish agent. We assume that each link e_i is associated with a (privately known) cost coefficient c_i such that the cost of e_i to provide a transmission service with bandwidth demand x is $c_i \cdot x$. Further, we assume that there is a fixed source node s and a set R of receivers, each of which requires from s data with a minimum bandwidth demand. The DiffServ multicast problem is to compute a link-weighted tree rooted at s and spanning R such that the receivers' demands are met. This generalizes the traditional link weighted Steiner tree problem. We first show that a previous approximation algorithm does not directly induce a strategyproof mechanism. We then give a new polynomial time algorithm to construct a DiffServ multicast tree whose total cost is no more than 8 times the optimal total cost when the cost coefficient of each link is known. Based on this tree, we design a truthful mechanism for DiffServ multicast, *i.e.*, we give a polynomial-time computable payment scheme to compensate all chosen relay links such that each link maximizes its profit when it declares its cost coefficient truthfully.**

***Index Terms*— DiffServ, multicast, selfish agents, algorithmic mechanism design, approximation algorithms.**

I. Introduction

The Differentiated Services framework (DiffServ) [1], [2] has been proposed to provide multiple Quality of Service (QoS) classes over IP networks. DiffServ is built upon a simple model of traffic conditioning and policing at the links of the network in addition to classifying flows into different service classes. The traffic is forwarded using simple differentiated treatments, called per-hop behaviors (PHBs), in the core of the network. This differential treatment results in differential pricing [3], which is one of the motivating factors for adopting DiffServ by major network providers and ISPs.

Multicast has been a popular mechanism for supporting group-based applications, such as video-conference and content distribution. Although multicast and DiffServ are complementary technologies, there are still some architectural conflicts between them. The first notable conflict is that multicast often requires the maintenance of per-group state information at all routers, while DiffServ usually relies on the statelessness of the core. The second notable conflict is that multicast is often based on *receiver-driven* QoS, while DiffServ is usually based on *sender-driven* QoS. Edge-based multicast (EBM) approach was proposed recently to address these possible conflicts. In this paper, we characterize the different QoS of the links by the amount of bandwidth they dedicate to the multicast transmission.

In a multicast, different receivers of a multicast group could request different bandwidth demands, which often reflect different qualities of services the receivers will get. Each link of the network may have a different cost of providing multicast with different bandwidth dedication [4]. Due to the heterogeneity in receivers' bandwidth demands, different links in a multicast tree will carry different amount of traffic such that the demand requirements of downstream receivers are satisfied. The cost of a link in a multicast tree is then the cost needed to dedicate a certain bandwidth for downstream receivers; it is typically determined by the maximum bandwidth required by downstream receivers, as well as the cost coefficient of the link (which we will define later). The DiffServ multicast problem is to compute a *tree* and the bandwidth at each link of the tree such that the receivers' bandwidth QoS demands are met. Note that the traditional Steiner tree problem for link weighted graph [5], [6], an NP-hard problem, is a special case of the problem of computing a DiffServ multicast tree with the minimum cost.

What introduces an additional degree of complexity to DiffServ multicast is that the relay links may be *non-cooperative*[1], instead of *cooperative* as assumed by previous protocols. This means that the relay links will aim to maximize their own benefits instead of the whole network's performance. We assume that a link will provide the service to receivers only if it receives a payment large enough to compensate its relay cost. To do so, each link is first asked to report its relay cost and then a payment to this link is calculated based on a certain payment scheme. It is often not in the best interests of these relay links to report their costs truthfully when they are paid whatever they ask for. Thus, instead of paying the links their *declared* costs, we should design a payment scheme that can ensure that all links reveal their true costs for their own interests, a property known as *strategyproofness*. The strategyproof mechanism for traditional multicast has been previously addressed in [7], [8]. However, unlike the traditional multicast in which every link has a *fixed* cost in the multicast transmission, each link may incur different costs

This work was supported in part by NSF Grant CCR-0311174, RGC Grant HKBU2107/04E, and HKBU Faculty Research Grant FRG/03-04/II-21. This paper was presented in part at the 1st International Conference on Algorithmic Applications in Management (AAIM 2005), Xi'an, China, June 2005.

Weizhao Wang and Xiang-Yang Li are with Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA (email: wangwei4@iit.edu, xli@cs.iit.edu).

Zheng Sun is with Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA, USA (email: sunzheng@gmail.com). Part of the research was done when the author was at Hong Kong Baptist University.

[1]It could happen when they are individually owned.

for different bandwidth demands in DiffServ multicast.

In summary, in this paper, we study two different aspects of the DiffServ multicast: the construction of the multicast tree that has low cost, and a strategyproof payment scheme. The main contributions of the paper are as follows. First of all, we show that a previous approximation algorithm does not directly induce a strategyproof mechanism. We give an alternative polynomial time algorithm to construct a DiffServ multicast tree whose total cost is no more than 8 times the optimal total cost when the cost coefficient of each link is known. We then characterize the necessary and sufficient conditions for the multicast tree construction algorithms based on which we can design a strategyproof payment scheme. Finally, we design a truthful algorithmic mechanism for DiffServ multicast, *i.e.*, we give a polynomial-time computable payment scheme to compensate all chosen relay links (by our multicast tree construction method) such that each link maximizes its profit when it declares its cost coefficient truthfully.

The rest of the paper is organized as follows. In Section II, we specify the network model, define the problem, and review the necessary technical preliminaries. We also briefly review some approximation algorithms to construct multicast trees. We study how to pay the links in Section IV after presenting our approximation algorithm for constructing the multicast tree in Section III. We conclude our paper by pointing out some possible future work in Section V.

II. Preliminaries and Previous Works

A. Algorithmic Mechanism Design

In a standard model of algorithmic mechanism design, there are n agents $\{1, 2, \cdots, n\}$. Each agent $i \in \{1, \cdots, n\}$ has some *private* information t_i, called its *type*, (*e.g.*, the cost to forward a packet for a node/link in a network environment). The types of all agents define a *profile* $t = (t_1, t_2, \cdots, t_n)$. Each agent i declares a valid type τ_i, which may be different from its actual type t_i, and the strategies of all agents define a *declared type vector* $\tau = (\tau_1, \cdots, \tau_n)$. A mechanism $M = (\mathcal{O}, \mathcal{P})$ is composed of two parts: an output function $\mathcal{O}$ that maps a declared type vector τ to an output o and a *payment* function $\mathcal{P}$ that decides the monetary payment $\mathcal{P}_i(\tau)$ for every agent i. Each agent i has a valuation function $w_i(o)$ that expressed its preference over different outcomes. Agent i's *utility* (also called *profit*) is $u_i(\mathcal{O}(\tau)) = w_i(\mathcal{O}(\tau)) + \mathcal{P}_i(\tau)$, given the declared vector type τ. An agent i is said to be *rational* if it always chooses its strategy τ_i that maximizes its utility u_i.

Let $\tau_{-i} = (\tau_1, \cdots, \tau_{i-1}, \tau_{i+1}, \cdots, \tau_n)$, *i.e.*, the strategies of all other agents except i, and let $\tau|^i t_i = (\tau_1, \tau_2, \cdots, \tau_{i-1}, t_i, \tau_{i+1}, \cdots, \tau_n)$. A mechanism is *strategyproof* if for every agent i, revealing its true type t_i will maximize its utility *regardless* of what other agents do. In this paper, we are only interested in mechanisms $M = (\mathcal{O}, \mathcal{P})$ that satisfy the following three conditions:

1) **Incentive Compatibility (IC)**: $\forall$ agent $i, \forall \tau$, $w_i(\mathcal{O}(\tau|^i t_i)) + \mathcal{P}_i(\tau|^i t_i) \geq w_i(\mathcal{O}(\tau)) + \mathcal{P}_i(\tau)$.
2) **Individual Rationality (IR)**(a.k.a., Voluntary Participation): Each agent must have a non-negative utility, *i.e.*, $w_i(\mathcal{O}(\tau|^i t_i)) + \mathcal{P}_i(\tau|^i t_i) \geq 0$.
3) **Polynomial Time Computability (PC)**: $\mathcal{O}$ and $\mathcal{P}$ are computed in polynomial time.

Notice that in a strategyproof mechanism, there is no budget-balance: the total payment to all selfish agents could be arbitrarily larger than the total declared truthful cost of all agents [9], [10]. Recently, there have been some studies to quantify the worst performances of a strategyproof mechanism [9], [10], [11], [12], or to design strategyproof mechanisms with better performances [11], [13]. In addition to strategyproof algorithm mechanisms, other approaches to deal with the selfishness of agents include auctions [14] and reputation-inference [15]. The advantage of having a strategyproof mechanism is its simplicity: it is simple to implement and it relieves all agent from guessing other's actions and possibly declaring a much higher cost. Another advantage of using a strategyproof mechanism is that, given an algorithm constructing the multicast tree with approximation ratio α, the multicast tree used will have a *real* cost no more than a certain factor α times of the minimum real cost. This is often called bounded social efficiency. If the optimal tree construction method is used, then the tree is the actual optimal tree. However, when agents declare their costs untruthfully, it will be difficult, if not impossible, to bound the total *real* cost of the output compared with the real optimal solution.

B. Problem Statement

DiffServ Multicast Tree Construction: We assume that there is a connected network $G = (V, E)$ with vertex set V, edge set E, where $|V| = n$ and $|E| = m$. Every edge e_i has a cost function $c_i \cdot x$ if x is the bandwidth e_i dedicates to a multicast transmission. Hereafter c_i is called the *cost coefficient* of the link e_i. All links' coefficients define a vector $\mathbf{c} = (c_1, c_2, \cdots, c_m)$. There is a source node s and a set of receivers $R \subset V$ that request to receive the multicast service. Every receiver $r_i \in R$ has a bandwidth demand d_i that specifies the minimum bandwidth it needs. The DiffServ multicast is also called Quality of Service Steiner Tree (QoSST) problem in [16].

A bandwidth demand is *homogeneous* if all receivers require the same bandwidth. This is the standard Steiner tree problem, for which several constant approximation algorithms [5], [6] have been proposed. For DiffServ multicast, different receivers may require different bandwidths. The DiffServ multicast problem consists of two parts: 1) a network topology rooted at the sender s that spans all receivers in the receiver set; 2) a bandwidth for each link for this multicast. The tree topology and bandwidth assignments should satisfy that for any receiver r_i, each link on the path between r_i and s in the tree has a bandwidth not smaller than d_i. Thus, for a link e_i, the bandwidth should not be smaller than the maximum bandwidth demand of its downstream receivers. The *cost* of a multicast topology T with link bandwidth vector $b = \{b_1, b_2, \cdots, b_m\}$ is $\omega(T, b, \mathbf{c}) = \sum_{e_i \in T} c_i \cdot b_i$. Given the cost coefficients vector $\mathbf{c}$ and the bandwidth demand $\mathbf{d}$ of all receivers, the DiffServ multicast problem is to construct a tree T and a bandwidth $\mathbf{b}$ such that $\omega(T, \mathbf{b}, \mathbf{c})$ is minimized.

The DiffServ multicast problem was studied before in several contexts. Maxemchuk [4] proposed a heuristic algorithm

for its solution. Some results for the case of few rates were obtained in [17], [18]. For example, for the case of two non-zero rates, a $\frac{4}{3}\alpha$-approximation algorithm was proposed [18], where $\alpha \simeq 1.549$ is the currently best approximation ratio [6] for the Steiner tree problem. Recently, Charikar *et al.* [19] gave the first constant-factor approximation algorithm for an unbounded number of rates. They achieved an approximation ratio of 4α using rounding and $e\alpha \simeq 4.211$ using randomized rounding. Recently, Karpinski *et al.* [16] gave algorithms with improved approximation factors. They achieved an approximation ratio of 1.960 when there are two non-zero rates and an approximation ratio of 3.802 when there are an unbounded number of rates. Calinescu *et al.* [20] gave a Primal-Dual algorithm with approximation ratio 4.311. Xue *et al.* [21] and Kim *et al.* [22] studied the Grade of Service Steiner Tree Problem (GOSST) in Euclidean planes.

Output and Payment Computation: Throughout this paper, we assume all the links are selfish and rational. Recall that a mechanism M consists of two parts: an output method $\mathcal{O}$ and a payment scheme $\mathcal{P}$. Each edge e_i is required to reveal its cost coefficient and it could declare a value a_i that is different from c_i. Thus, we use $\mathbf{a} = (a_1, \ldots, a_m)$ to denote the *declared* cost coefficient vector. Given receiver set R and declared cost coefficient vector $\mathbf{a}$, the output method computes a multicast tree T and a valid bandwidth vector $\mathcal{O}(R, \mathbf{a}) = (b_1, b_2, \cdots, b_m)$. Here, $b_i = \mathcal{O}_i(R, \mathbf{a})$ is the bandwidth on link e_i. After designing the output method $\mathcal{O}$, we need to design a payment scheme $\mathcal{P}$ for the links such that the mechanism $M = (\mathcal{O}, \mathcal{P})$ is truthful. Given the receiver set R and declared cost coefficient vector $\mathbf{a}$, we use $\mathcal{P}(R, \mathbf{a})$ to denote the total payment to the links, *i.e.*, $\mathcal{P}(R, \mathbf{a}) = \sum_{e_i \in E} \mathcal{P}_i(R, \mathbf{a})$. Here $\mathcal{P}_i(R, \mathbf{a})$ denotes the payment to a link e_i given the cost coefficient vector $\mathbf{a}$ and the receiver set R. Notice that the widely used VCG mechanism [23], [24], [25] can be used to design a strategyproof mechanism for the traditional multicast problem (when all receivers have the same demand) with the objective to minimize the total cost of the multicast tree when we can find the minimum-cost multicast tree. However, the optimum solution is often difficult to obtain: it is well-known that finding minimum-cost multicast tree is an NP-complete problem. Therefore, VCG mechanism for traditional multicast cannot be implemented in polynomial time unless P=NP. We thus see the trade-off of efficiency for complexity. Actually, we will show that VCG mechanism does not work for general DiffServ multicast problem.

C. Literature Review of Steiner Tree Construction

If all receivers have the same bandwidth QoS demand, the DiffServ problem becomes the standard link weighted Steiner tree problem. In link weighted Steiner tree, each link has a fixed cost c_i for a unit bandwidth and all bandwidth in the tree can be normalized to unit. Notice that c_i equals the cost coefficient in the DiffServ multicast problem, thus for notational consistency, we use $\mathbf{c}$ to denote the input for the link weighted Steiner tree problem. The link weighted Steiner tree problem enjoys several constant approximation algorithms [5], [6]. In Algorithm 1 we review a 2-approximation algorithm given in [5]. We call the tree constructed by Algorithm 1 a *Link Weighted Steiner Tree* (LST), denoted as $LST(R, \mathbf{c})$ where $\mathbf{c}$ is declared cost coefficient vector.

Algorithm 1 Construct homogeneous multicast tree [H. Takahashi and A. Matsuyama [5]]

Input: A network $G = (V, E)$, the cost coefficient vector $\mathbf{c}$, a source node s and a set of receivers R.
Output: A tree $LST(R, \mathbf{c})$ rooted at s that spans the receiver set R.

1: Initialize $LST(R, \mathbf{c}) = \emptyset$.
2: **repeat**
3: **for** each receive r_i in R **do**
4: Find the shortest path between s and r_i under link cost coefficient vector $\mathbf{c}$, denoted by $\mathsf{LCP}(s, r_i, \mathbf{c})$.
5: **end for**
6: Find the receiver r_j that is closest to the source.
7: Remove r_j from R and add the path $\mathsf{LCP}(s, r_j, \mathbf{c})$ to $LST(R, \mathbf{c})$.
8: Set all links' cost on the path $\mathsf{LCP}(s, r_j, \mathbf{c})$ as 0, *i.e.*, set $c_i = 0$ if and only if $e_i \in \mathsf{LCP}(s, r_j, \mathbf{c})$.
9: **until** R is empty.
10: Output $LST(R, \mathbf{c})$.

For DiffServ multicast, the algorithm by Charikar *et al.* [19] works as follows. Given an instance of the DiffServ multicast, they first construct the rounded-up instance by rounding up all demands of receivers to the nearest power of 2. Then they solve the standard Steiner tree problem for the receivers of each different demand separately by applying any of the well-known heuristics such as Algorithm 1. Finally, they do a "clean-up" process that transforms the graph given by the union of these Steiner trees into a tree and chooses the bandwidth of each link to be the maximum bandwidth demand of its downstream receivers. They proved that this simple approach yields a $4\alpha_{ST}$ approximation of the optimal cost, where α_{ST} is the approximation factor of the Steiner tree heuristic used. When Algorithm 1 is used as the Steiner tree heuristic, $\alpha_{ST} = 2$ and the overall approximation ratio is 8. For notational simplicity, we denote the algorithm as Charikar-Takahashi algorithm. Our algorithm is similar to Charikar-Takahashi algorithm at the first glance, but it has some key differences that will be described later.

III. A New Approximation Algorithm

In this section, we present an alternative DiffServ multicast tree construction algorithm to the algorithm in [19]. Before we present our algorithm, we define some notations that will be used later. For a set R of k receivers with bandwidth demand vector $\mathbf{d} = \{d_1, d_2, \cdots, d_k\}$, we denote the multicast tree with the minimal weight that spans R as $T^{opt}(R, \mathbf{d}, \mathbf{c})$ and the corresponding bandwidth allocation vector as $B^{opt}(R, \mathbf{d}, \mathbf{c})$, where $\mathbf{c}$ is the cost coefficient vector. If the receivers have homogenous bandwidth demand, then the minimum link weighted Steiner tree, which is denoted as $T^{\min}(R, \mathbf{c})$, does not depend on $\mathbf{d}$. Given a subset $S \subseteq R$ and

a tree T that spans R, we use T_S^{opt} to denote the subtree in T induced by S if no confusion is caused.

Given a receiver set R, a cost coefficient vector $\mathbf{c}$ and a bandwidth demand vector $\mathbf{d}$, following algorithm shows how to find a DiffServ multicast tree $\overline{DMT}(R, \mathbf{c})$ and its corresponding bandwidth allocation $\overline{B}$ with low weight. We also call this algorithm $\overline{DMT}$ if no confusion is caused. Basically, Algorithm 2 constructs a DiffServ multicast tree as follows. It first sorts the demands of all receivers in a descending order and groups them into several groups such that the largest demand in each group is at most 2 times the smallest demand in that group. Starting from the group containing the largest demand, it constructs a multicast tree to span the receivers in this group using Algorithm 1. It then marks the cost of links chosen as 0 since we will use them anyway to span these receivers and their bandwidths are enough to support any future receivers if they are also chosen later. We then process all groups in the descending order of their demands. Remember that when process the ith group, the links chosen to span any group j with $j < i$ will have cost marked as 0.

Algorithm 2 Construct DiffServ Multicast Tree

Input: A network G with coefficient vector $\mathbf{c}$, a source node s, a set of receivers R and a bandwidth demand vector $\mathbf{d}$.
Output: A tree $\overline{DMT}(R, \mathbf{c})$ spanning the receivers and a bandwidth allocation vector $\overline{B}$.

1: Sort all receivers according to their bandwidth demands in a descending order, say $R = \{r_1, r_2, \cdots, r_k\}$.
2: Initialize the tree T to empty and index $t = 1$.
3: For each link e_i, label it as WHITE and set $\overline{B}_i = 0$.
4: **repeat**
5: Let r_j be the first receiver in the receiver set R and find the maximum index k such that $d_k \geq \frac{d_j}{2}$.
6: Set the cost coefficient of each BLACK link as 0, *i.e.*, $c_i = 0$ if e_i is BLACK.
7: Let $R_t = \{r_j, \cdots, r_k\}$ and find the spanning tree $T_t = LST(R_t, \mathbf{c})$ using Algorithm 1.
8: Remove R_t from R and mark all links in tree $LST(R_t, \mathbf{c})$ as BLACK.
9: Set $T = T \bigcup T_t$.
10: **for** each link $e_i \in T_t$ **do**
11: Set $\overline{B}_i = d_j$.
12: **end for**
13: Set $t = t + 1$.
14: **until** the receiver set R is empty.
15: Output $\overline{DMT}(R, \mathbf{c}) = T$ and bandwidth vector $\overline{B}$.

The major difference of this algorithm compared with the Charikar algorithm is that, instead of computing several trees *independently* and then combining them to make the final DiffServ multicast tree, we construct a single tree directly. The receiver set is divided into subsets, each containing receivers with demands in a particular range. These subsets are handled in multiple rounds, in a descending order according to their bandwidth demand ranges. In each round, all receivers in a subset are connected to the DiffServ multicast tree being built. The links picked in earlier rounds will be used in later rounds, without additional costs involved, to connect receivers with lower demands.

Notice that, as indicated by Line 11 of Algorithm 2, for each link e_i added into T in round t the bandwidth allocation of e_i is set to be the maximum bandwidth demand among all receivers in R_t. This may be more than necessary; after all, e_i will not be relaying packets for all of them. Indeed, one can design the following Algorithm 3, which constructs the same tree as Algorithm 2 does, and yet allocates less bandwidth on each link e_i by setting the bandwidth allocation to be maximum bandwidth demand of e_i's downstream receivers. In order to distinguish these two algorithms, we use DMT to denote the tree constructed by Algorithm 3. As minor (and harmless) as this modification seems to be, Algorithm 3 does not induce a truthful payment scheme. In the next section, we will use this algorithm as an example to show how to use a general criterion to determine the truthfulness of a payment scheme induced by a given algorithm.

Algorithm 3 Construct DiffServ multicast Tree with Less Bandwidth Allocation

Input: A network G with coefficient vector $\mathbf{c}$, a source node s, a set of receivers R and a bandwidth demand vector $\mathbf{d}$.
Output: A tree $DMT(R, \mathbf{c})$ spanning the receivers and a bandwidth allocation vector B.

1: Compute a multicast tree T using Algorithm 2.
2: **for** each link e_i in tree T **do**
3: Find the maximal bandwidth demand of e_i's downstream receivers, say r_j.
4: e_i allocates a bandwidth $B_i = d_j$.
5: **end for**
6: Output $DMT(R, \mathbf{c})$ and bandwidth vector B.

We have the following theorem for the approximation bound of Algorithm 2 and Algorithm 3. Although there are only subtle differences between these two algorithms presented here and the one in [19], the proof is not as obvious as that one.

Theorem 1: Both Algorithm 2 and Algorithm 3 construct a tree whose cost is at most 8 times the cost of the minimal cost DiffServ multicast tree.

PROOF. The proofs for both algorithms are similar. Since the cost of the tree constructed by Algorithm 2 is not smaller than the cost of the tree constructed by Algorithm 3, in the following we only prove the case for Algorithm 2. For notational convenience, we use T and B to denote the tree and bandwidth allocation vector computed by Algorithm 2. Remember that T_i is the tree found in the i^{th} iteration by applying Algorithm 1. Without loss of generality, we assume that there are ℓ iterations in Algorithm 3. Let $R_1, R_2, \cdots, R_\ell$ be a partition of receiver set R, and let $R_i^{\max}$(respectively $R_i^{\min}$) be the maximum (respectively minimal) bandwidth demand in the receiver set R_i. For notational simplicity, we use $T_{R_i}^{opt}$ and B^{opt} to denote $T_{R_i}^{opt}(R, \mathbf{d}, \mathbf{c})$ and $B^{opt}(R, \mathbf{d}, \mathbf{c})$ respectively.

Recall that each link in tree $T_{R_1}^{opt}$ should be able to supply

a bandwidth larger than $R_1^{\min}$. Thus,

$$\begin{aligned}
& \omega(T_1, B, \mathbf{c}) \le \omega(T_1, \langle R_1^{\max}\rangle, \mathbf{c}) \\
= \;& R_1^{\max}\omega(T_1, \langle 1\rangle, \mathbf{c}) \\
\le \;& 2R_1^{\max} \cdot \omega(T^{\min}(R_1, \mathbf{c}), \langle 1\rangle, \mathbf{c}) \\
\le \;& 2R_1^{\max} \cdot \omega(T_{R_1}^{opt}, \langle 1\rangle, \mathbf{c}) \\
= \;& 2R_1^{\max} \cdot \sum_{e_i \in T_{R_1}^{opt}} c_i \\
\le \;& 4R_1^{\min} \cdot \sum_{e_i \in T_{R_1}^{opt}} c_i \\
= \;& 4\omega(T^{opt}(R_1), \langle R_1^{\min}\rangle, \mathbf{c}) \\
\le \;& 4\omega(T_{R_1}^{opt}, B^{opt}, \mathbf{c})
\end{aligned}$$

For set R_2, we have

$$\begin{aligned}
& \omega(T_2, B, \mathbf{c}) \le \omega(T_2, \langle R_2^{\max}\rangle, \mathbf{c}) = R_2^{\max}\omega(T_2, \langle 1\rangle, \mathbf{c}) \\
\le \;& 2R_2^{\max} \cdot \omega(T^{\min}(R_2, a), \langle 1\rangle, \mathbf{c}) \\
\le \;& 2R_2^{\max} \cdot \omega(T_{R_2}^{opt}, \langle 1\rangle, \mathbf{c}) \\
\le \;& 2R_2^{\max} \cdot \omega(T_{R_1\cup R_2}^{opt}, \langle 1\rangle) \\
\le \;& 2R_2^{\max} \cdot [\omega(T_{R_1}^{opt}, \langle 1\rangle, \mathbf{c}) + \omega(T_{R_2}^{opt} - T_{R_1}^{opt}, \langle 1\rangle, \mathbf{c})] \\
= \;& 2R_2^{\max} \cdot \sum_{e_i \in T_{R_1}^{opt}} c_i + 4 \sum_{e_i \in T_{R_2}^{opt} - T_{R_1}^{opt}} c_i \cdot R_2^{\min} \\
\le \;& 2R_1^{\min} \cdot \sum_{e_i \in T_{R_1}^{opt}} c_i + 4 \sum_{e_i \in T_{R_2}^{opt} - T_{R_1}^{opt}} c_i \cdot R_2^{\min} \\
= \;& 2\omega(T_{R_1}^{opt}, \langle R_1^{\min}\rangle, \mathbf{c}) + 4\omega(T_{R_2}^{opt} - T_{R_1}^{opt}, \langle R_2^{\min}\rangle, \mathbf{c}) \\
\le \;& 2\omega(T_{R_1}^{opt}, B^{opt}, \mathbf{c}) + 4\omega(T_{R_2}^{opt} - T_{R_1}^{opt}, B^{opt}, \mathbf{c})
\end{aligned}$$

Similarly, for any set R_i ($1 \le i \le l$) we have

$$\omega(T_i, B, \mathbf{c}) \le 4\sum_{j=1}^{i} \frac{1}{2^{i-j}} \omega(T_{R_j}^{opt} - \bigcup_{k=1}^{j-1} T_{R_k}^{opt}, B^{opt}, \mathbf{c})$$

Summing the inequalities for i from 1 to ℓ, we obtain

$$\begin{aligned}
& \omega(\overline{DMT}(R, \mathbf{c}), B, \mathbf{c}) \\
= \;& \omega(\bigcup_{i=1}^{\ell} T_i, B, \mathbf{c}) \le \sum_{i=1}^{\ell} \omega(T_i, B, \mathbf{c}) \\
\le \;& 4 \cdot \sum_{i=1}^{\ell}\sum_{j=1}^{i} \frac{1}{2^{i-j}} \cdot \omega\left(T^{opt}(R_j) - \bigcup_{k=1}^{j-1} T_{R_k}^{opt}, B^{opt}, \mathbf{c}\right) \\
= \;& 4 \cdot \sum_{i=1}^{\ell} \left[\omega\left(T_{R_i}^{opt} - \bigcup_{j=1}^{i-1} T_{R_j}^{opt}, B^{opt}, \mathbf{c}\right) \cdot \sum_{k=0}^{l-i} 2^{-k}\right] \\
\le \;& 8 \cdot \sum_{i=1}^{\ell} \left[\omega\left(T_{R_i}^{opt} - \bigcup_{j=1}^{i-1} T_{R_j}^{opt}, B^{opt}, \mathbf{c}\right)\right] \\
= \;& 8 \cdot \omega\left(T^{opt}(\bigcup_{i=1}^{l} R_i), B^{opt}, \mathbf{c}\right) \\
= \;& 8 \cdot \omega(T_R^{opt}, B^{opt}, \mathbf{c})
\end{aligned}$$

This finishes our proof. □

IV. Payment for Selfish Links

In this section, we first show that VCG mechanism does not work for any algorithms that we proposed before. In light of the failure of the VCG mechanism, some truthful mechanisms that are not based on VCG are needed. Instead of simply presenting a truthful payment scheme for a specific DiffServ multicast tree construction algorithm, such as Algorithm 2, we study a general framework to design a truthful payment scheme for any given tree construction algorithm. In Subsection IV-B, we fist give a necessary and sufficient condition for the existence of a truthful payment scheme for a given tree construction algorithm. In the meanwhile, we also present a truthful payment scheme if it exists. We then apply this general framework to the DiffServ multicast tree constructed by Algorithm 2 and design a truthful payment scheme. In this section, we need to distinguish between the declared cost coefficient vector $\mathbf{a}$ and actual cost coefficient vector $\mathbf{c}$.

A. Failure of VCG mechanism

Arguably the most positive result in mechanism design is what is usually called the generalized Vickrey-Clarke-Groves (VCG) mechanism by Vickrey [23], Clarke [24], and Groves [25]. Although the family of VCG mechanisms is powerful, but it has its limitations. To use VCG mechanism, we have to compute the exact output that maximizes the total valuation of all agents. In our case, we need to find the tree with the minimum cost that is computationally intractable. Most often, replacing the optimal algorithm with non-optimal approximation usually leads to untruthful mechanisms if VCG mechanism is used [26].

Unfortunately, if we insist on using VCG mechanism for those algorithms we proposed above, none of the resulting mechanisms is truthful. Moreover, even for the special case when bandwidth on each link and for each receiver are homogeneous, VCG mechanism still fails. Recall that when bandwidths are homogenous, Algorithm 3, Algorithm 2 and Charikar-Takahashi Algorithm are exactly Algorithm 1. Thus, in the following we focus on Algorithm 1.

Given a receiver set R and declared vector $\mathbf{a}$, if we apply VCG mechanism to Algorithm 1, the payment to an edge $e_i \in LST(R, \mathbf{a})$ is

$$\mathcal{P}_i(d) = \omega(LST(R, \mathbf{a}|^i\infty), \langle 1\rangle, \mathbf{a}) - \omega(LST(R, \mathbf{a}), \langle 1\rangle, \mathbf{a}) + c_i.$$

Next we show that this mechanism does *not* satisfy IR property, i.e., it is possible that some edges have negative utility if each link reveals its actual cost coefficient. Figure 1 illustrates

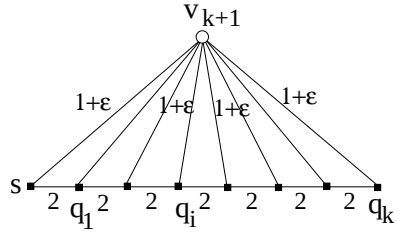


Fig. 1. Here q_i, $1 \le i \le k$ are receivers; the cost coefficient of each link $v_{k+1}q_i$ and $v_{k+1}s$ is $1+\epsilon$, where ϵ is a small positive real number. The cost coefficient of each link q_iq_{i+1} and sq_1 is 2.

the example with terminal s being the source terminal. In this example, each link reveals its actual cost coefficient, *i.e.*, $a_i = c_i$. It is not difficult to show that, in the first round, link sq_1 is selected to connect terminals s and q_1 with cost 2; in round r, we will select link $q_{r-1}q_r$ to connect to q_r with cost 2. Thus, the tree $LST(R, \mathbf{c})$ is path $sq_1q_2 \cdots q_k$, whose cost is $2k$. When link $e_1 = sq_1$ is not used, it is easy to see that the tree $LST(R, \mathbf{a}|^i\infty)$ only uses terminal v_{k+1} to connect all receivers with total cost $(k+1)(1+\epsilon)$. Thus, the utility of link $e_1 = sq_1$ is $(k+1)(1+\epsilon) - 2k = k\epsilon - k + 2$, which is negative when $\epsilon < \frac{k-2}{k}$. Thus, the VCG based mechanism is not truthful.

B. General Framework

From the definition of the truthfulness, we can fix the graph G, the receiver set R and bandwidth demand $\mathbf{d}$. Thus, for our notational convenience, we use $\mathcal{O}^{\mathcal{A}}(\mathbf{a})$ to denote the bandwidth vector computed by an algorithm $\mathcal{A}$ for links, where $b_i = \mathcal{O}_i^{\mathcal{A}}(\mathbf{a})$ is the bandwidth on link e_i.

Here, we assume that $\mathcal{O}_i^{\mathcal{A}}(\mathbf{a})$ is *piecewise continuous* with respect to any variable a_j, *i.e.*, a finite number of piece-wise linear functions. The only possible types of discontinuities for a piecewise continuous function are removable and step discontinuities. In the following we give a definition that is critical to the presentation of our general framework.

*Definition 1 (***Monotone Non-increasing Property (MNP)):** An algorithm $\mathcal{A}$ is said to satisfy the *monotone non-increasing property* if for every link e_i and any two of its possible coefficients $a_{i_1} < a_{i_2}$, $\mathcal{O}_i^{\mathcal{A}}(\mathbf{a}|^i a_{i_1}) \geq \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_2})$.

Now we are ready to present the necessary and sufficient condition for the existence of truthful mechanism given an algorithm $\mathcal{A}$ that computes the bandwidth. This theorem is similar to the forklore for the binary demand games.

Theorem 2: For a given algorithm $\mathcal{A}$, there exists a payment scheme $\mathcal{P}$ such that the mechanism $M = (\mathcal{A}, \mathcal{P})$ is truthful if and only if $\mathcal{A}$ satisfies MNP.

PROOF. First, we prove that if there exists a strategyproof mechanism $M = (\mathcal{A}, \mathcal{P})$ then $\mathcal{A}$ satisfies MNP. We consider two coefficients profile $a|^i a_{i_1}$ and $a|^i a_{i_2}$ where $a_{i_1} \leq a_{i_2}$.

Consider the case when link e_i has coefficient a_{i_1}. Remember $\mathcal{P}$ is strategyproof, thus if link e_i lies its coefficient to a_{i_2}, its utility should not increase. Thus, we have

$$\begin{aligned} & \mathcal{P}_i(\mathcal{A}, a|^i a_{i_1}) - a_{i_1} \cdot \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_1}) \\ \geq\ & \mathcal{P}_i(\mathcal{A}, a|^i a_{i_2}) - a_{i_1} \cdot \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_2}). \end{aligned}$$

Now consider the case when link e_i actually has cost coefficient a_{i_2}. Similarly, we have

$$\begin{aligned} & \mathcal{P}_i(\mathcal{A}, a|^i a_{i_2}) - a_{i_2} \cdot \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_2}) \\ \geq\ & \mathcal{P}_i(\mathcal{A}, a|^i a_{i_1}) - a_{i_2} \cdot \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_1}) \end{aligned}$$

Combining the above two inequalities, we obtain

$$\begin{aligned} & a_{i_2} \cdot [\mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_1}) - \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_2})] \qquad (1) \\ \geq\ & \mathcal{P}_i(\mathcal{A}, a|^i a_{i_1}) - \mathcal{P}_i(\mathcal{A}, a|^i a_{i_2}) \\ \geq\ & a_{i_1} \cdot [\mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_1}) - \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_2})] \end{aligned}$$

Thus, we have $\mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_1}) \geq \mathcal{O}_i^{\mathcal{A}}(a|^i a_{i_2})$ as $a_{i_1} \leq a_{i_2}$. This proves that $\mathcal{A}$ satisfies MNP.

To prove that if $\mathcal{A}$ satisfies MNP then there exists a strategyproof payment $\mathcal{P}$, we prove it by construction. For a link e_i, we first fix a_{-i} and use x to denote cost vector $a|^i x$ if no confusion is caused. From the assumption that $\mathcal{A}$ satisfies MNP, function $\mathcal{O}_i^{\mathcal{A}}(x)$ is non-increasing. Recall that $\mathcal{O}_i^{\mathcal{A}}(x)$ is a piecewise continuous function. We let $x_1 < x_2 \cdots < x_m$ be the points at which $\mathcal{O}_i^{\mathcal{A}}(x)$ is not continuous, and introduce a dummy point $x_{m+1} = \infty$. We define a function $\kappa_i(x)$ such that, for $x_p < x \leq x_{p+1}$,

$$\kappa_i(x) = x \cdot \mathcal{O}_i^{\mathcal{A}}(x) + \int_x^{x_{p+1}} \mathcal{O}_i^{\mathcal{A}}(y)dy + \sum_{j=p+1}^{m} \int_{x_j}^{x_{j+1}} \mathcal{O}_i^{\mathcal{A}}(y)dy.$$

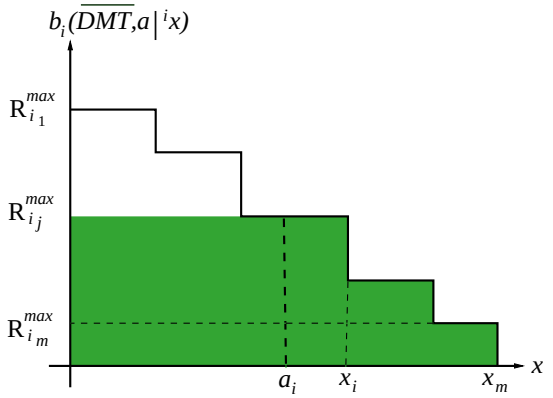


Fig. 2. Bandwidth allocation function $\mathcal{O}_i^{\overline{DMT}}(\mathbf{a}|^i x)$.

In Figure 2, $\kappa_i(x)$ corresponds to the area of the shaded region. Given an algorithm $\mathcal{A}$ and a coefficient vector $\mathbf{a}$, Algorithm 4 defines the payment based on algorithm $\mathcal{A}$.

Algorithm 4 Payment Scheme based on $\mathcal{A}$

Input: Algorithm $\mathcal{A}$ and declared coefficient vector $\mathbf{a}$.
Output: The payment scheme $\mathcal{P}$.

1: **for** each link i **do**
2: Fix a_{-i}. The payment to i is $\mathcal{P}_i(\mathcal{A}, a) = \kappa_i(a_i)$.
3: **end for**

Thus, we only need to prove the payment scheme computed by Algorithm 4 is truthful. See Lemma 7 in the appendix for the proof of this statement. This finishes the proof of the theorem. □

We note that the above theorem applies to any problem (*e.g.*, job scheduling) when the cost of an agent is of format $c_i \cdot b_i$, where c_i is a privately known cost-coefficient and b_i is its load computed by output method. Actually, Archer and Tardos [27] proved a similar result for job scheduling. If we require that a link e_i that has 0 bandwidth should receive 0 payment (which is called *normalized* payment scheme), then we have the following theorem.

Theorem 3: Given an algorithm $\mathcal{A}$ satisfying MNP, the payment scheme defined by Algorithm 4 is the *only* normalized truthful payment scheme.

PROOF. In inequality 1, substitute x for a_{i_1} and $x+\delta$ for a_{i_2} we obtain $(x+\delta)(\mathcal{O}_i^{\mathcal{A}}(x)-\mathcal{O}_i^{\mathcal{A}}(x+\delta)) \geq \mathcal{P}_i(x)-\mathcal{P}_i(x+\delta) \geq x(\mathcal{O}_i^{\mathcal{A}}(x)-\mathcal{O}_i^{\mathcal{A}}(x+\delta))$. When $\mathcal{O}_i^{\mathcal{A}}(x)$ is continuous at x, we can set $\delta \to 0$ and obtain

$$(x+\delta)\cdot d(-\mathcal{O}_i^{\mathcal{A}}(x)) \geq d(-\mathcal{P}_i(x)) \geq x\cdot d(-\mathcal{O}_i^{\mathcal{A}}(x)) \quad (2)$$

From equation 2, if x is continuous in (l,u), then we obtain

$$\begin{aligned} -p_i(x)|_l^u &= p_i(l)-p_i(u) \\ &= \int_l^u x d(-\mathcal{O}_i^{\mathcal{A}}(x)) \\ &= -\int_l^u x d(\mathcal{O}_i^{\mathcal{A}}(x)) \\ &= -[x\mathcal{O}_i^{\mathcal{A}}(x)|_l^u - \int_l^u \mathcal{O}_i^{\mathcal{A}}(x)dx] \\ &= l\cdot\mathcal{O}_i^{\mathcal{A}}(l) - u\cdot\mathcal{O}_i^{\mathcal{A}}(u) + \int_l^u \mathcal{O}_i^{\mathcal{A}}(x)dx \end{aligned}$$

Set $l=x_j$ and $u=x_{j+1}$ $(1\leq j\leq q)$, we obtain

$$\begin{aligned} &\mathcal{P}_i(x_j)-\mathcal{P}_i(x_{j+1}) \\ =\ & x_j\cdot\mathcal{O}_i^{\mathcal{A}}(x_j) - x_{j+1}\cdot\mathcal{O}_i^{\mathcal{A}}(x_{j+1}) + \int_{x_j}^{x_{j+1}} \mathcal{O}_i^{\mathcal{A}}(x)dx \end{aligned}$$

Assume $x_p \leq a_i < x_{p+1}$, then summing j from $p+1$ to q we have

$$\begin{aligned} &\mathcal{P}_i(x_{p+1}) = \mathcal{P}_i(x_{p+1}) - \mathcal{P}_i(x_{q+1}) \\ =\ & \sum_{j=p+1}^{q} p_i(x_j) - p_i(x_{j+1}) \\ =\ & \sum_{j=p+1}^{q} [x_j\cdot\mathcal{O}_i^{\mathcal{A}}(x_j) - x_{j+1}\cdot\mathcal{O}_i^{\mathcal{A}}(x_{j+1})] \\ & + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} \mathcal{O}_i^{\mathcal{A}}(x)dx \\ =\ & x_{p+1}\cdot\mathcal{O}_i^{\mathcal{A}}(x_{p+1}) + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} \mathcal{O}_i^{\mathcal{A}}(x)dx \end{aligned}$$

Let $l=a_i$ and $u=x_{p+1}$, we have $\mathcal{P}_i(a_i)-\mathcal{P}_i(x_{p+1}) = a_i\cdot\mathcal{O}_i^{\mathcal{A}}(a_i) - x_{p+1}\cdot\mathcal{O}_i^{\mathcal{A}}(x_{p+1}) + \int_{c_i}^{x_{p+1}} \mathcal{O}_i^{\mathcal{A}}(y)dy$. Combining the above two equations we get

$$\begin{aligned} &\mathcal{P}_i(a_i) \\ =\ & x_{p+1}\cdot\mathcal{O}_i^{\mathcal{A}}(x_{p+1}) + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} \mathcal{O}_i^{\mathcal{A}}(x)dx + a_i\cdot\mathcal{O}_i^{\mathcal{A}}(a_i) \\ & -x_{p+1}\cdot\mathcal{O}_i^{\mathcal{A}}(x_{p+1}) + \int_{a_i}^{x_{p+1}} \mathcal{O}_i^{\mathcal{A}}(x)dx \\ =\ & a_i\cdot\mathcal{O}_i^{\mathcal{A}}(a_i) + \int_{a_i}^{x_{p+1}} \mathcal{O}_i^{\mathcal{A}}(y)dy + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} \mathcal{O}_i^{\mathcal{A}}(x)dx \end{aligned}$$

This finishes our proof. □

We then summarize the general framework to design a truthful payment scheme $\mathcal{P}$, such that $M=(\mathcal{O}^{\mathcal{A}},\mathcal{P})$ is truthful, for a given output algorithm $\mathcal{O}^{\mathcal{A}}$ that constructs a DiffServ multicast tree and outputs the bandwidth allocation for DiffServ multicast.

1) *Check whether the bandwidth function of algorithm $\mathcal{O}^{\mathcal{A}}$ satisfies MNP. If not then return, else continue.*
2) *Compute the bandwidth function $\mathcal{O}^{\mathcal{A}}(\mathbf{a})$.*
3) *Design the payment according to Algorithm 4.*

C. Design Truthful Mechanism

We first show that, there is no truthful payment scheme based on Algorithm 3 and Algorithm by Charikar *et al.* [19].

Theorem 4: There is no truthful mechanism that uses either Algorithm 3 or Charikar-Takahashi algorithm as its output method.

According to Theorem 2, it suffices to prove the following Lemma.

Lemma 5: Neither Algorithm 3 nor Charikar-Takahashi algorithm satisfies MNP.

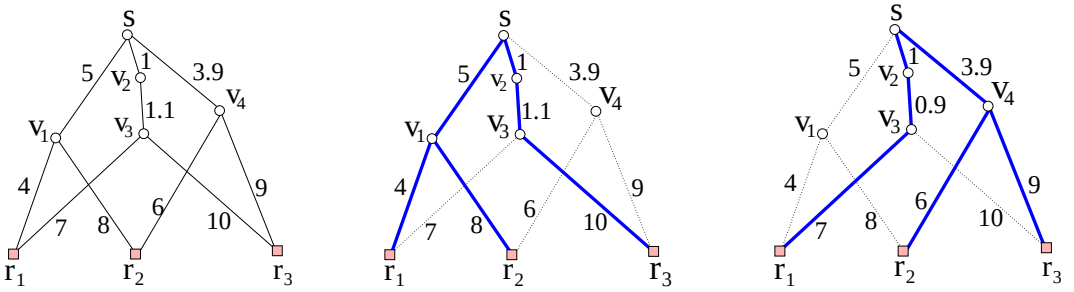
PROOF. We prove it by presenting a counter example illustrated by Figure 3. A network G has three receivers r_1, r_2, r_3 with bandwidth demand $d_1=d_2=1$ and $d_3=1.5$. Under this bandwidth demand vector, all receivers' demand will be rounded to the same value and only one iteration of Algorithm 1 is needed. Thus, Both Algorithm 3 and Charikar-Takahashi algorithm output the same tree and bandwidth vector. The coefficient of the link is described in Figure 3 (a). Now we apply Algorithm 3 to network G. In the first iteration, path sv_1r_1 is chosen (with cost 9); in the second iteration, path v_1r_2 is chosen (with cost 8); and in the last iteration, path $sv_2v_3r_3$ is chosen (with cost 12.1). The final tree shown in Figure 3 (b). The bandwidth allocation of link v_2v_3 is 1.5. Consider the scenario when the coefficient of link v_2v_3 changes from 1.1 to 0.9 while other coefficients remain the same. When apply Algorithm 3 to network G, in the first iteration, path $sv_2v_3r_1$ is chosen (with cost 8.9); in the second iteration, path sv_4r_2 is chosen (with cost 9.9); and in the last iteration, path v_4r_3 is chosen (with cost 9). The new spanning tree topology is shown in Figure 3 (c). The bandwidth on v_2v_3 becomes 1, which is decreased compared with the former case when the coefficient is 1.1. This contradicts the MNP property and finishes our proof. □

The above example also shows that there is no strategyproof mechanism for the DiffServ multicast tree construction method presented in [19] when the bandwidth on any link is taken as the maximum bandwidth demand of its downstream receivers. Meanwhile, we can show that there exists a truthful payment scheme for Algorithm 2 with the following theorem.

Theorem 6: Algorithm 2 satisfies MNP.

PROOF. Given a link e_i, if it does not appear in the tree $\overline{DMT}(R,\mathbf{a})$ then $\mathcal{O}_i^{\overline{DMT}}(\mathbf{a})=0$. Otherwise, if $e_i \in T_j - \bigcup_{k=1}^{j-1} T_k$, *i.e.*, in iteration j, the link e_i is added to the spanning tree $\overline{DMT}(R,\mathbf{a})$ for the first time, then $\mathcal{O}_i^{\overline{DMT}}(\mathbf{a}) = R_j^{\max}$. When e_i declares a smaller coefficient $\underline{a_i}$, we show by cases that its bandwidth does not become smaller.

Case 1: e_i is added to the spanning tree $\overline{DMT}(R,\mathbf{a})$ before the j^{th} iteration of REPEAT loop in iteration j. Without loss of generality, we assume that e_i is added to $\overline{DMT}(R,\mathbf{a})$ in iteration $j' \leq j$. Remember that the partition of R does not



(a) Original network G (b) The spanning tree when $a_2 = 1.1$ (c) New tree when $a_2 = 0.9$

Fig. 3. The spanning tree constructed by Algorithm 3.

depend on coefficient vector $\mathbf{a}$, thus $\mathcal{O}_i^{\overline{DMT}}(\mathbf{a}|^i\underline{a_i}) = R_{j'}^{\max} \geq R_j^{\max} = \mathcal{O}_i^{\overline{DMT}}(\mathbf{a})$.

Case 2: e_i is not added to the spanning tree $\overline{DMT}(R, \mathbf{a})$ before the j^{th} iteration of REPEAT loop in Algorithm 2. In this case, each link's label does not change in the beginning of iteration j. Following we show that the link e_i *must* in the tree $LST(R_j, \mathbf{a}|^i\underline{a_i})$. From our assumption, e_i is in the tree $LST(R_j, \mathbf{a})$ and without loss of generality, we assume that e_i is in path $\mathsf{LCP}(s, r_k, \mathbf{c})$ that is selected in the ℓ^{th} iteration of Algorithm 1. If e_i is selected before ℓ^{th} iteration of Algorithm 1 when we constructing tree $LST(R_j, \mathbf{a}|^i\underline{a_i})$, then the argument is proven. Otherwise, we can assume that e_i is not selected before iteration ℓ. Notice that if e_i is not selected before ℓ^{th} iteration, each path selected before ℓ^{th} iteration should be the same. In other words, in the beginning of iteration ℓ of Algorithm 1, (1) the receiver set R is the same; (2) the cost of each link except e_i is the same. Thus, if $\mathbf{c}$ is the cost vector in the beginning of iteration ℓ when input is $\mathbf{a}$, then $\mathbf{c}|^i\underline{a_i}$ is the cost vector in the beginning of iteration ℓ when input is $\mathbf{a}|^i\underline{a_i}$. It is not difficult to observe that the path $\mathsf{LCP}(s, r_k, \mathbf{c})$ is decreases by $a_i - \underline{a_i}$ if $\mathbf{c}|^i\underline{a_i}$ is the cost vector. On the other hand, the cost of any path is decreased at most $a_i - \underline{a_i}$ if $\mathbf{c}|^i\underline{a_i}$ is the cost vector instead of $\mathbf{c}$. Thus, path $\mathsf{LCP}(s, r_k, \mathbf{c})$ is also selected under cost vector $\mathbf{c}|^i\underline{a_i}$. Thus, e_i is selected by Algorithm 2 before iteration j, which means that $\mathcal{O}_i^{\overline{DMT}}(a|^i\underline{a_i}) \geq R_j^{\max}$.

This proves that $\mathcal{O}_i^{\overline{DMT}}(\mathbf{a})$ does not decrease when a_i decreases. Thus, Algorithm 2 satisfies MNP. □

Algorithm 5 Payment based on Algorithm 1

Input: A network G with link cost coefficient vector $\mathbf{a}$, a source node s and a receiver set R.
Output: A truthful payment based on Algorithm 1.

1: Compute $LST(R, \mathbf{a})$ using Algorithm 1.
2: **for** each link $e_i \in \mathsf{LCP}(R, \mathbf{a})$ **do**
3: Set $\text{temp}_i = a_i$ for each link e_i.
4: Set $a_i = \infty$, and $p_i = 0$.
5: **repeat**
6: Find the shortest path $\mathsf{LCP}(s, r_i, \mathbf{a})$ between s and r_i for each receive r_i in R.
7: Find the receiver r_j that is closest to the source.
8: Set $a_i = 0$ and find the shortest path $\mathsf{LCP}(s, r_i, \mathbf{a})$ between s and r_i for each receive r_i in R.
9: Find the receiver r'_j that is closest to the source and let $|\mathsf{P}'_j|$ be the cost of the shortest path path between s and r_j'.
10: Set $a_i = \infty$.
11: Remove r_j from R and add the path $\mathsf{LCP}(s, r_i, \mathbf{a})$ to $LST(R, \mathbf{a})$.
12: Set all links' cost on the path $\mathsf{LCP}(s, r_j, \mathbf{a})$ as 0, *i.e.*, set $a_i = 0$ if and only if $e_i \in \mathsf{LCP}(s, r_j, \mathbf{a})$.
13: Set $p_i = \max\{p_i, |\mathsf{LCP}(s, r_j, \mathbf{c})\| - |\mathsf{P}'_j|\}$.
14: **until** R is empty.
15: Set $a_i = \text{temp}_i$ for each link e_i.
16: **end for**
17: Set $\mathcal{P}_i(R, \mathbf{a}) = p_i$ and output $\mathcal{P}$.

In order to find the truthful payment for Algorithm 2, we should find the bandwidth output function $\mathcal{O}_i^{\overline{DMT}}(\mathbf{a}|^i x)$ for every link e_i first. Recall that for every link e_i, the bandwidth could only be a real value that is equal to $R_j^{\max}$ for some index j. Let $x_1^i < x_2^i < \cdots < x_q^i$ be the points at which $\mathcal{O}_i(\overline{DMT}, \mathbf{a}|^i x)$ is not continuous, then the bandwidth allocation function $\mathcal{O}_i(\overline{DMT}, \mathbf{a}|^i x)$ should be a constant, say y_j^i in (x_j^i, x_{j+1}^i) as shown in the Figure 2. In order to find the values of these discontinuous points, we first need to compute the truthful payment for standard Steiner tree problem. Following we brief review the algorithm to compute the payment for Algorithm 1. Please refer for [8] for more details.

For clarity of the notation, we use $\kappa_i(R, \mathbf{a})$ to denote the payment computed for a link e_i by Algorithm 5 and study how to find the bandwidth allocation function for Algorithm 2. Algorithm 6 shows how we can find the bandwidth-allocation function.

With the bandwidth allocation function $\mathcal{O}_i^{\overline{DMT}}(\cdot)$, we give our truthful payment scheme, as illustrated by Algorithm 7, by using the general framework. The proof of the correctness of these algorithms are either straightforward or omitted here due to space limit. Notice that since Algorithm 3 often constructs a multicast tree with less cost than the multicast tree constructed by Algorithm 2, the multicast "*principal*" may jump to Algorithm 3 after all links declared their cost coefficients. This will be prevented from the untruthfulness of Algorithm 3 when links knew that the "principal" will jump to Algorithm 3, *i.e.*, links could lie to gain more benefits. When links are unaware of this jump, we need a trusted third-

Algorithm 6 Bandwidth Output Function for Algorithm 2

Input: A network G with declared link cost vector $\mathbf{a}$, a source node s and a receiver set R with demand vector $\mathbf{d}$.
Output: The bandwidth output function for Algorithm 2.

Compute $\overline{DMT}(R, \mathbf{a})$ and bandwidth vector $\overline{B}$.
for each link e_i in $\overline{DMT}(R, \mathbf{a})$ **do**
  Label each link it as WHITE.
  Set $\text{temp}_i = a_i$ for each link e_i.
  Set $a_i = \infty$ and index $t = 1$.
  Initialize the list $X^i = \emptyset$, $Y^i = \emptyset$, $up = 0$, and $q = 0$.
  repeat
    Let r_j be the first receiver in the receiver set R and find the maximum index k such that $d_k \geq \frac{d_j}{2}$.
    Set the cost coefficient of each BLACK link as 0, *i.e.*, $a_i = 0$ if e_i is BLACK.
    Let $R_t = \{r_j, \cdots, r_k\}$ and find the spanning tree $LST(R_t, \mathbf{c})$ using Algorithm 1.
    Remove $LST(R_t, \mathbf{a})$ from R and mark all links in tree $LST(R_t, \mathbf{a})$ as BLACK.
    Set $T = T \bigcup T_t$ and compute $\kappa_i(R_t, \mathbf{a})$ using Algorithm 5.
    if $\kappa_i(R_t, \mathbf{a}) > up$ **then**
      Set $q = q + 1$ and $up = \kappa_i(R_t, \mathbf{a})$.
      Set $x_q^i = \kappa_i(R_t, \mathbf{a})$ and $y_q^i = R_t^{\max}$.
      Add x_q^i to set X^i and y_q^i to Y^i.
    end if
    Set $t = t + 1$.
  until R is empty
  Set x_0^i=0 and $x^{q+1} = \infty$.
  for $i = 1$ to $q + 1$ **do**
    Set $\mathcal{O}_i^{\overline{DMT}}(R, \mathbf{a}|^i x) = y_j^i$ for $x_{j-1}^i \leq x < x_j^i$.
  end for
  Set $a_i = \text{temp}_i$ for each link e_i.
end for

Algorithm 7 Payment Scheme for Algorithm 2

Input: A network G with cost coefficient vector $\mathbf{a}$, a source node s and a receiver set R with demand vector $\mathbf{d}$.
Output: A truthful payment $\mathcal{P}^{\overline{DMT}}$ for Algorithm 2.

Compute the multicast tree $\overline{DMT}(R, \mathbf{a})$ by applying Algorithm 2.
Compute the bandwidth allocation function for tree $\overline{DMT}(R, \mathbf{a})$ by applying Algorithm 6.
for each link e_i **do**
  if e_i is in tree $\overline{DMT}(R, \mathbf{a})$ **then**
    Find an index j such that $x_j^i < a_i \leq x_{j+1}^i$. Then the payment is $\mathcal{P}_i^{\overline{DMT}}(R, \mathbf{a}) = \sum_{k=j+1}^{|X^i|-1} y_k^i \cdot (x_{k+1}^i - x_k^i) + (x_{j+1}^i - a_i) \cdot y_j^i$.
  else
    $\mathcal{P}_i^{\overline{DMT}}(R, \mathbf{a}) = 0$.
  end if
end for

party to prevent the principal from changing the multicast-tree construction algorithm.

D. Performance Improvement and Special Case

In essence, Algorithm 2 converts the original instance of the DiffServ multicast problem to a "rounded-up" one, with bandwidth demand vector forming a geometric sequence of ratio 2. According to the result of Charikar *et al.* [19], the approximation ratio of 8 of Algorithm 2 can be improved (while still using Algorithm 1 for computing approximately optimal Steiner trees) if the "randomized bucketing" technique is used. Specifically, a number y is picked randomly with a uniform distribution in the range $[0, 1]$, and the (non-zero) bandwidth demands of all receivers are rounded up to the nearest e^{y+i}. (Note that the ratio of the geometric sequence is e instead of 2.) The *expected* approximation ratio is $e \cdot 2 \simeq 5.437$.

Here we argue that we can also convert the mechanism described above for DiffServ multicast to a randomized one with an expected approximation ratio of 5.437, while maintaining strategyproofness. First of all, in Algorithm 2, we group the receivers according to their bandwidths: a receiver with minimum bandwidth (the minimum bandwidth is called "start point" in literatures) that is not grouped yet is chosen and each receiver whose bandwidth is at most 2 times the minimum bandwidth is fallen into the same group. On the other hand, if we use e^y as the start point for some fixed y and replacing the ratio of 2 by e for the geometric sequence (of rounded up bandwidth demands) should not affect strategyproofness. Furthermore, the randomized process also does not encourage untruthfulness of the links: if for any fixed start point e^y, the links find no incentive to lie, nor will they find incentives to lie when such start point is randomly selected.

Charikar *et al.* [19] also proposed a de-randomized process to replace the above random selection of start point e^y, with the cost of an increased time complexity. For each distinct bandwidth demand d_i, the same algorithm is invoked with $y_i = \ln d_i - \lfloor \ln d_i \rfloor$. It is claimed that there is at least one y_i such that the solution for $y = y_i$ has a cost no more than the expected cost of the solution for a randomly picked y. Therefore, we can simply pick the best solution (with the minimum cost) among all solutions computed using different y. A similar technique is used for the case with only two non-zero rates for bandwidth demands [18], improving the approximation bound to $\frac{4}{3} \cdot 2 \simeq 2.667$. The common characteristic of the two algorithms is to compute multiple DiffServ multicast trees using different methods (or same method but with different parameters), and pick the one with the smallest cost. Although this approach (*i.e.*, taking the best output of several outcomes and using a certain combination of the payments for these separated games as its final payment) works for binary selection problems under certain conditions [28], [29], a problem arises when it comes to determining the payments to the links for DiffServ multicast.

In the network shown in Figure 4 (a), receiver r_1 has bandwidth demand $d_1 = 1$ unit and each of receivers r_2, r_3, r_4 has bandwidth demand $d_2 = 4$. Let $R_1 = \{r_1\}$ and $R_2 = \{r_2, r_3, r_4\}$, and let $\mathbf{c}$ be the cost vector shown in Figure 4 (a). Let $\mathbf{c}'$ be the cost vector we get by changing the cost of edge sv_1 from $1.5 + \epsilon$ to $1.5 - \epsilon$ while keeping

all other links' costs unchanged. Figure 4 shows that the tree $LST(R, c)$ and tree $LST(R, c')$ are the same. We have $\omega(LST(R, \mathbf{c}), b) = \omega(LST(R, \mathbf{c}'), b') = 5.5 \cdot d_2 = 22$. Figure 4 (c) shows the tree $LST(R_1, \mathbf{c}) \cup LST(R_2, \mathbf{c})$ and its weight is $1.5 \cdot d_1 + (5 + \epsilon) \cdot d_2 = 21.5 + 4\epsilon < \omega(LST(R, \mathbf{c}), b)$ for small ϵ. Thus, when sv_1 has cost $1.5 + \epsilon$, it has bandwidth $d_2 = 4$. Now consider the cost vector $\mathbf{c}'$. Figure 4 (c) shows the tree $LST(R_1, \mathbf{c}') \cup LST(R_2, \mathbf{c}')$ and its weight is $1.5 \cdot d_1 + (6 + 3\epsilon) \cdot d_2 = 25.5 - 12\epsilon > \omega(LST(R, \mathbf{c}'), b)$ for small ϵ. Thus, when sv_1 has cost $1.5 - \epsilon$, its bandwidth is 0. This shows that the tree output by the algorithm in [18] violates the MNP property, which implies that there is no truthful payment.

V. Conclusion

In this paper, we studied the DiffServ multicast problem in a game theoretic context, where network links are selfish agents who would demand payments to at least cover their costs when relaying data packets, and may lie about their actual costs in order to maximize their gains. We show that a naive conversion of the previously known 8-approximation algorithm does not work. We then propose an alternative approximation algorithm for DiffServ multicast with the same approximation bound. We also introduced a general method to convert any DiffServ multicast algorithm satisfying the Monotone non-increasing Property to a strategyproof mechanism, and applied it to the algorithm we proposed.

The strategyproof payment scheme is not the end story for designing protocols for DiffServ multicast. The very natural question to ask is how these payments can be split among the receivers, which is known as the *multicast payment sharing* problem [7]. Several criteria [30], [7] for the *fairness* of sharing have been proposed in previous work, and we would like to design payment sharing schemes that are considered to be fair according to these criteria. Another important work is to design distributed implementations of our proposed strategyproof mechanism, which could be based on some results in [31], [32]. Last, but not the least, since strategyproof mechanisms will often pay more than what the agents declared, it is an interesting future work to design scheme that could result in a less total payment by the multicast principal at some certain equilibrium of the agents' declaration. Some initial work has been done in this direction for unicast [14], [33], [34].

References

[1] K. Nichols, S. Blake, D.B.: Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. In: IETF RFC 2474. (1998)

[2] K. Nichols, S.B.: Differentiated services operational model and definitions. In: Networking. (1993)

[3] Wang, X., Schulzrinne, H.: Pricing network resources for adaptive applications in a differentiated services network. In: INFOCOM. (2001) 943–952

[4] Maxemchuk, N.F.: Video distribution on multicast networks. IEEE JSAC (1997)

[5] Takahashi, H., Matsuyama, A.: An approximate solution for the steiner problem in graphs. Mathematical Japonica **24** (1980) 573–577

[6] Robins, G., Zelikovsky, A.: Improved steiner tree approximation in graphs. In: Proceedings of ACM/SIAM Symposium on Discrete Algorithms. (2000) 770–779

[7] Wang, W., Li, X.Y., Sun, Z., Wang, Y.: Design multicast protocols for non-cooperative networks. In: IEEE INFOCOM. (2005)

[8] Wang, W., Li, X.Y., Wang, Y.: Truthful mutlicast in selfish wireless networks. In: ACM MobiCom 2004. (2004)

[9] Talwar, K.: The price of truth: Frugality in truthful mechanisms. In: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag (2003) 608–619

[10] Archer, A., l. Tardos: Frugal path mechanisms. In: SODA. (2002) 991–998

[11] Karlin, A., Kempe, D., Tamir, T.: Beyond vcg: Frugality of truthful mechanisms. In: To appear in FOCS. (2005)

[12] Calinescu, G.: Bounding the payment of approximate truthful mechanisms. In: ISAAC. (2004) 221–233

[13] Elkind, E., Sahai, A., Steiglitz, K.: Frugality in path auctions. In: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2004) 701–709

[14] N. Immorlica, D. Karger, E.N., Sami, R.: First-price path auctions. In: Proceedings of the 6th ACM Conference on Electronic Commerce. (2005)

[15] Buchegger, S., Boudec, J.Y.L.: A robust reputation system for p2p and mobile ad-hoc networks. In: Second Workshop on the Economics of Peer-to-Peer Systems, in conjunction with ACM Sigcomm. (2004)

[16] Karpinski, M., Mndoiu, I.I., Olshevsky, A., Zelikovsky, A.: Improved approximation algorithms for the quality of service steiner tree problem. In: WADS. (2003)

[17] Balakrishnan, A., Magnanti, T., Mirchandani, P.: Modeling and heuristic worst-case performance analysis of the two-level network design problem. Management Science (1994) 846–867

[18] Balakrishnan, A., Magnanti, T., Mirchandani, P.: Heuristics, lps, and trees on trees: Network design analyses. Operations Research (1996) 478–496

[19] Charikar, M., Naor, J.S., Schieber, B.: Resource optimization in qos multicast routing of real-time multimedia. In: IEEE INFOCOM. (2000)

[20] Calinescu, G., Fernandes, C., Mndoiu, I., Olshevsky, A., Yang, K., Zelikovsky, A.: Primal-dual algorithms for qos multimedia multicast. In: GlobeCom. (2003)

[21] Xue, G., Lin, G.H., Du, D.Z.: Grade of service steiner minimum trees in the euclidean plane. Algorithmica (2001) 479–500

[22] Kim, J., Cardei, M., Cardei, I., Jia, X.: A polynomial time approximation scheme for the grade of service steiner minimum tree problem,. Journal of Global Optimization **24** (2002) 439–450

[23] Vickrey, W.: Counterspeculation, auctions and competitive sealed tenders. Journal of Finance (1961) 8–37

[24] Clarke, E.H.: Multipart pricing of public goods. Public Choice (1971) 17–33

[25] Groves, T.: Incentives in teams. Econometrica (1973) 617–631

[26] Nisan, N., Ronen, A.: Algorithmic mechanism design. In: Proceedings of the 31st Annual Symposium on Theory of Computing, Atlanta, Georgia, United States (1999) 129–140

[27] Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science, Las Vegas, Nevada, IEEE Computer Society (2001) 482–491

[28] Mu'alem, A., Nisan, N.: Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. In: Proceedings of the 18th National Conference on Artificial Intelligence, Edmonton, Alberta, Canada, American Association for Artificial Intelligence (2002) 379–384

[29] Kao, M.Y., Li, X.Y., Wang, W.: Towards truthful mechanisms for binary demand games: A general framework. In: ACM EC. (2004)

[30] Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the cost of multicast transmissions. Journal of Computer and System Sciences **63** (2001) 21–41

[31] Wang, W., Li, X.Y.: Truthful low-cost unicast in selfish wireless networks. In: 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN) of IPDPS 2004, to appear. (2004)

[32] Wang, W., Li, X.Y.: Low-cost routing in selfish and rational wireless ad hoc networks. IEEE Transactions on Mobile Computing (2005) to appear.

[33] Wang, W., Li, X.Y., Chu, X.: Nash equilibria, dominant strateies in routing. In: Workshop for Internet and Network Economics. (2005)

[34] Kao, M.Y., Li, X.Y., Wang, W.: Output truthful versus input truthful: A new concept for algorithmic mechanism design (2005) Submitted for publication.

Appendix

Lemma 7: Algorithm 4 defines a truthful payment scheme.

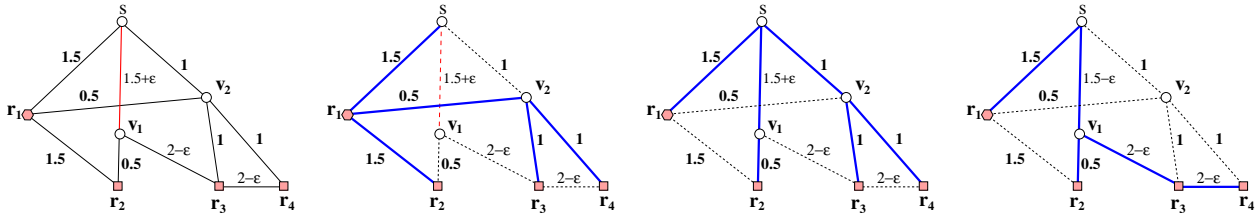


Fig. 4. An example to show that simply choosing the best solution may not work. Here (a) denotes the original network G, (b) is the tree $LST(R,c)(LST(R,c'))$, (c) is the tree $LST(R_1,c)\cup LST(R_1,c)$ and (d) is the tree $LST(R_1,c')\cup LST(R_1,c')$.

PROOF. Hereafter, we always fix a_{-i}, *i.e.*, we are interested only in a_i. Notice that when e_i reveals its true coefficient a_i, its utility is $u_i(a_i)=\mathcal{P}_i(a_i)-a_i\cdot\mathcal{O}_i^{\mathcal{A}}(a_i)=\kappa_i(a_i)-a_i\cdot\mathcal{O}_i^{\mathcal{A}}(a_i)$ $=\int_{a_i}^{x_{p+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy$. Remember that $\mathcal{O}_i^{\mathcal{A}}(y)$ is non-negative. Thus $u_i(a_i)\geq 0$, which implies that payment scheme 4 satisfies IR. To prove that payment scheme 4 satisfies IC, we prove it by cases.

Case 1: Node i lies its cost upward to $\overline{a_i}$. In this case, we assume $x_{p'}<\overline{a_i}\leq x_{p'+1}$. Since $a_i<\overline{a_i}$, $p\leq p'$. The utility of node i becomes

$$\begin{aligned}u_i(\overline{a_i}) &= p_i(\overline{a_i})-a_i\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})=\xi(\overline{a_i})-a_i\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})\\ &= \int_{\overline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &\quad +\overline{a_i}\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})-a_i\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})\end{aligned}$$

There are two subcases here. If $p<p'$ then

$$\begin{aligned}&u_i(a_i)\\ =\;& \int_{a_i}^{x_{p+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ =\;& \int_{a_i}^{x_{p+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p+1}^{p'-1}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &+\int_{x_{p'}}^{\overline{a_i}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\int_{\overline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ \geq\;& \mathcal{O}_i^{\mathcal{A}}(\overline{a_i})\cdot[(x_{p+1}-a_i)+(\sum_{j=p+1}^{p'-1}(x_{j+1}-x_j))+(\overline{a_i}-x_{p'})]\\ &+\int_{\overline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ =\;& b_i(\overline{a_i})\cdot(\overline{a_i}-a_i)+\int_{\overline{a_i}}^{x_{p+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ =\;& u_i(\overline{a_i})\end{aligned}$$

If $p=p'$ then

$$\begin{aligned}u_i(a_i) &= \int_{a_i}^{x_{p+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &= \int_{a_i}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &= u_i(\overline{a_i})-(\overline{a_i}-a_i)\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})+\int_{a_i}^{\overline{a_i}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &\geq u_i(\overline{a_i})-(\overline{a_i}-a_i)\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})+(\overline{a_i}-a_i)\cdot\mathcal{O}_i^{\mathcal{A}}(\overline{a_i})\\ &= u_i(a_i)\end{aligned}$$

Thus, link e_i have no incentive to lie its coefficient upward.

Case 2: Link e_i lies its coefficient downward to $\underline{a_i}$. In this case, we assume $x_{p'}<\underline{a_i}\leq x_{p'+1}$. Since $a_i>\underline{a_i}$, $p\geq p'$. The utility of node i becomes $u_i(\underline{a_i})=\mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\int_{\underline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy$.

There are two subcases here also. If $p>p'$ then

$$\begin{aligned}&u_i(\underline{a_i})\\ =\;& \mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\int_{\underline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &+\sum_{j=p'+1}^{p-1}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\int_{x_p}^{a_i}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &+\int_{a_i}^{x_{p+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy+\sum_{j=p+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ \leq\;& \mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\int_{\underline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(\underline{a_i})dy\\ &+\sum_{j=p'+1}^{p-1}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(\underline{a_i})dy+\int_{x_p}^{a_i}\mathcal{O}_i^{\mathcal{A}}(\underline{a_i})dy+u_i(a_i)\\ =\;& \mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(a_i-\underline{a_i})+u_i(a_i)\\ =\;& u_i(a_i)\end{aligned}$$

If $p=p'$ then

$$\begin{aligned}&u_i(\underline{a_i})\\ =\;& \mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\int_{\underline{a_i}}^{x_{p'+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ &+\sum_{j=p'+1}^{m}\int_{x_j}^{x_{j+1}}\mathcal{O}_i^{\mathcal{A}}(y)dy\\ =\;& \mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\int_{\underline{a_i}}^{a_i}\mathcal{O}_i^{\mathcal{A}}(y)dy+u_i(a_i)\\ \leq\;& \mathcal{O}_i^{\mathcal{A}}(\underline{a_i})\cdot(\underline{a_i}-a_i)+\int_{\underline{a_i}}^{a_i}\mathcal{O}_i^{\mathcal{A}}(\underline{a_i})dy+u_i(a_i)\\ =\;& u_i(a_i)\end{aligned}$$

This proves that node i does not have incentive to lie downward. Thus, the payment scheme 4 satisfies IC. Therefore, the payment scheme 4 is truthful. □