Design Differentiated Service Multicast With Selfish Agents

WeiZhao Wang[1], Xiang-Yang Li*[1], and Zheng Sun**[2]

[1] Illinois Institute of Technology, Chicago, IL, USA, {lixian,wangwei4}@iit.edu
[2] Hong Kong Baptist University, Hong Kong, China, sunz@comp.hkbu.edu.hk

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 the DiffServ multicast when the participants are selfish. We assume that the cost of a link e_i to provide a multicast service with bandwidth demand x is $a_i \cdot x$. This generalizes the traditional link weighted Steiner tree problem. The main contribution of the paper is as follows. This paper studies the strategyproof mechanism design and fair payment sharing scheme for DiffServ multicast. First of all, we show that a previous approximation method does not imply a strategyproof mechanism. We then give a polynomial time method to construct a multicast tree whose cost is no more than 8 times of the optimal 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 each chosen relay links such that each link maximizes its profit when it declares its coefficient truthfully. We also study how to share the payment to relay links among the given set of receivers who require the multicast service. Both positive results and negative results are presented.

Keywords: DiffServ, multicast, selfish agents, algorithm mechanism design, cost sharing, fair.

1 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 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. Each link of the network may have different costs of providing multicast with different bandwidth dedication [4]. Due to the heterogeneity in receivers' demand requirements, different links in a multicast tree will carry different 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. This is often the maximum bandwidth required by downstream receivers. Then the DiffServ multicast problem is to find a *tree* and the bandwidth reservation at each link such that the receivers' bandwidth QoS demand are met. Recall that the traditional Steiner tree problem is NP-hard for both node weighted [5, 6] and link weighted graph [7, 8], and it is a special case of constructing the DiffServ multicast tree with the minimum cost.

What makes things more complicated is that the links that relay the packets may be *non-cooperative*, instead of *cooperative* assumed by previous protocols. This means that the relay links will aim to maximize their own benefits

* The research of the author was supported in part by NSF under Grant CCR-0311174.
** The research of the author was supported in part by Grant FRG/03-04/II-21 and Grant RGC HKBU2107/04E.

instead of the whole network's performance. We assume that a link will provide the service to receivers only if they received 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 some mechanisms. It is not often in the best interests of these relay links to report their costs truthfully when we pay whatever they asked. Thus, instead of paying the links their *declared* costs, we should design some payment scheme that can ensure all links reveal their true costs out of their own interests, which is known as *strategyproof*. The strategyproof mechanism for traditional multicast has been previously addressed in [9, 10]. However, unlike the traditional multicast in which every link has a *fixed* cost in the multicast transmission, each link may incur different costs for different bandwidth demands in DiffServ multicast. Furthermore, the strategyproof payment scheme is not the end story for the DiffServ multicast. A natural question to ask is that how these payments (or costs if the links are indeed cooperative to report their true costs always) are *fairly* shared among the receivers, which is known as the *multicast payment sharing* problem. In summary, in this paper, we study three different aspects of the DiffServ multicast: the construction of the multicast tree that has low cost, a strategyproof payment scheme, and a fair payment sharing scheme.

The main contribution of the paper is as follows. First of all, we show that a previous approximation method does not imply a strategyproof mechanism. We then characterize the necessary and sufficient conditions about the multicast tree construction methods such that we can design a strategyproof payment scheme based on this. We give a polynomial time method to construct a multicast tree whose cost is no more than 8 times of the optimal when the cost coefficient of each link is known. We then design a truthful algorithm 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 coefficient truthfully. We also study how to share the payment to relay links among the given set of receivers who require the multicast service. Both positive results and negative results are presented.

The rest of the paper is organized as follows. In Section 2, we specify the network model, define the problem, and review the necessary technical preliminaries. We also briefly review some approximation methods to construct the multicast tree. We study how to pay the links in Section 3 and how to share the payment in Section 4 after presenting our approximation method for constructing the multicast tree. We conclude our paper by pointing out some possible future researches in Section 5.

2 Preliminaries and Previous Works

2.1 Algorithmic Mechanism Design

In a standard model of algorithm 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. its cost to forward a packet in a network environment. All agents' type defines 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 all agents' strategy defines 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 $p_i = \mathcal{P}_i(\tau)$ for every agent i. Each agent i has a valuation function $w_i(t_i, o)$ that expressed its preference over different outcomes. Agent i's *utility* or called *profit* is $u_i(t_i, o) = w_i(t_i, o) + p_i$, given output o and payment p_i. An agent i is said to be *rational* if it always chooses its strategy τ_i to maximize 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 $\tau|^i t_i = (\tau_1, \tau_2, \cdots, \tau_{i-1}, t_i, \tau_{i+1}, \cdots, \tau_n)$. 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 $i, \forall \tau$, $w_i(t_i, \mathcal{O}(\tau|^i t_i)) + p_i(\tau|^i t_i) \geq w_i(t_i, \mathcal{O}(\tau)) + p_i(\tau)$
2. **Individual Rationality (IR)**(a.k.a., Voluntary Participation): Each agent must have a non-negative utility, i.e., $w_i(t_i, \mathcal{O}(\tau|^i t_i)) + p_i(\tau|^i t_i) \geq 0$.
3. **Polynomial Time Computability (PC)**: $\mathcal{O}$ and $\mathcal{P}$ are computed in polynomial time.

2.2 Problem Statement

Differentiated 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 $c_i = a_i x$ where x is the bandwidth e_i dedicates to the multicast transmission. Hereafter a_i is called the cost coefficient of the link e_i. All links' coefficients

define a vector $a = (a_1, a_2, \cdots, a_m)$. 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 [11].

A bandwidth demand is homogeneous if all receivers require the same bandwidth. This is the standard Steiner tree problem and several constant approximation algorithms [7, 8] have been proposed. For differentiated multicast (also called heterogeneous hereafter), different receivers may require different bandwidth. The differentiated 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 reservation for each link for this multicast. The tree topology and bandwidth reservation should satisfy that for any receiver r_i, each link on the tree path between r_i and s has a bandwidth reservation not smaller than d_i. Thus, for a link e_i, the reserved bandwidth should not be smaller than the maximum bandwidth demand of its downstream receivers. The weight of a multicast topology T with link bandwidth reservation vector $b = \{b_1, b_2, \cdots, b_m\}$ is $\omega(T, b) = \sum_{e_i \in T} c_i = \sum_{e_i \in T} a_i \cdot b_i$. Given the cost coefficients vector a of all links and the bandwidth demand d of all receivers, the differentiated multicast problem is to construct a tree T and a bandwidth reservation b with the minimum cost $\omega(T, b)$.

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 [12, 13]. Specifically, an algorithm was suggested in [13] for the case of two non-zero rates with approximation ratio of $\frac{4}{3}\alpha$, where $\alpha \simeq 1.549$ is currently the best approximation ratio [8] of an algorithm for the Steiner tree problem. Recently, Charikar *et al.* [14] 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.* [11] 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 is an unbounded number of rates. Calinescu *et al.* [15] gave a Primal-Dual algorithm with approximation ratio 4.311. Xue *et al.* [16] and Kim *et al.* [17] studied the Grade of Service Steiner Tree Problem (GOSST) in Euclidean planes.

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 $\mathcal{P}$. Thus, after designing a method $\mathcal{O}$ to construct a multicast tree, we need to design a payment scheme $\mathcal{P}$ for the links such that the mechanism $M = (\mathcal{O}, \mathcal{P})$ is truthful. We use $\mathcal{P}(R, a)$ to denote the total payment to the links, *i.e.*, $\mathcal{P}(R, a) = \sum_{e_i \in E} \mathcal{P}_i(R, a)$. Here $\mathcal{P}_i(R, a)$ denotes the payment to a link e_i given the cost coefficient vector a and the receiver set R.

Payment Sharing: For a given set of receivers R, after we calculate the payment for every link, it is natural to ask who will pay these payments. Two possible payment models have been proposed in the literature.

1. *Outside bank* or *Group payment model*: an outside bank or an organization to which the receivers belong will pay all these relay agents.
2. *Payment sharing model*: each receiver i should pay a *reasonable* sharing ξ_i of the total payment. We will address what we mean "reasonable" (or called *fair*) later.

For outside bank model, the only thing we should care is how to find the truthful mechanism for the multicast, which will be addressed in Section 3. In practice, it is often the case that the receivers have to share the payments among themselves. Thus, we will study how to share the payments fairly. Notice that the payment sharing is different from the traditional cost sharing studied in [18–20], which assumes that costs of relay links are public and the multicast topology is a fixed tree. Given a network G with coefficient a and a set of potential receivers R, we let $\xi(i, R) \geq 0$ denote how much receiver r_i is charged. We will give both negative and positive results on designing fair payment sharing mechanisms.

2.3 Literature Review of Steiner Tree Construction

Given a homogeneous bandwidth demand d, the weight of a tree T is $\omega(T, d) = \sum_{e_i \in T} c_i = \sum_{e_i \in T} a_i \cdot d = d \sum_{e_i \in T} a_i = d \cdot \omega(T, \langle 1 \rangle)$. Thus, in order to minimize the weight of the tree that spans all receivers, we can normalize the demand of every receiver to 1. Therefore, we can define cost vector $c = a$ and the problem becomes the standard link weighted Steiner tree problem, which enjoys several constant approximation metods [7, 8]. In Algorithm 1 we review a 2-approximation method given in [7]. We call the tree constructed by Algorithm 1 a *Link Weighted Steiner Tree* (LST), denoted as $LST(R, c)$.

The method by Charikar *et al.* [14] 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

Algorithm 1 Construct homogeneous multicast tree

Input: A network $G = (V, E)$, c is the cost vector of the links, a source node s and a receiver demand vector d.
Output: A spanning tree $LST(R, c)$ rooted at s that spans the receiver set R.

1: Initialize $LST(R, c) = \emptyset$.
2: **repeat**
3: **for** each receive r_i in R **do**
4: Find the least cost path $\mathsf{LCP}(s, r_i, G)$ between s and r_i.
5: **end for**
6: Find the receiver r_j with the minimum cost of the shortest path $\mathsf{LCP}(s, r_j, G)$.
7: Remove r_j from R and add $\mathsf{LCP}(s, r_j, G)$ to $LST(R, c)$.
8: Set all links' costs on $\mathsf{LCP}(s, r_j, G)$ as 0.
9: **until** R is empty.
10: Output $LST(R, c)$.

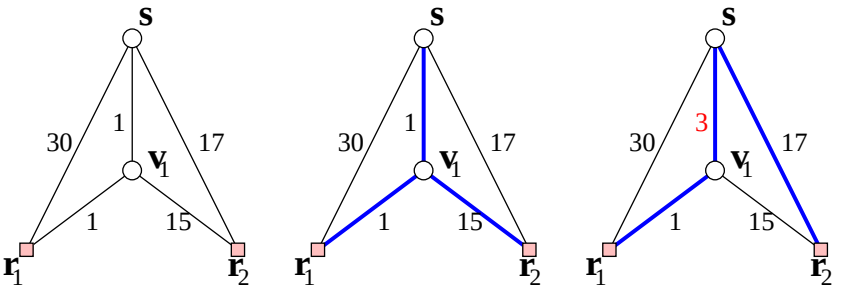
the standard Steiner tree problem for the receivers of each different demand separately by applying any of the well-known heuristics. Finally, they do a "clean-up" process that transforms the graph given by the union of these Steiner trees into a tree. They proved that this simple approach yields a $4\alpha_{ST}$ approximation of the optimal cost, where α_{ST} is the approximation factor of the used Steiner tree heuristic. Our algorithm is similar to this approach at the first glance, but it has some key differences, which will be described later.

3 Payment for Selfish Links

In this section, we discuss how to design a truthful payment scheme for links when they are selfish. For the multicast when the receivers have a homogeneous bandwidth demand, in [10], Li and Wang proved that the VCG mechanism [21–23] is not truthful if the tree is computed by Algorithm 1. In light of the failure of VCG mechanism, they proposed a truthful payment scheme for any round-based method constructing a multicast tree.

To construct a truthful payment scheme for heterogeneous multicast, one naive approach seems to be combining the algorithm of [14] with a truthful payment scheme for homogeneous multicast. More specifically, for each distinct bandwidth demand rate in the rounded-up instance, a homogeneous multicast tree is constructed and the payment for each selected link is determined. The union of these multicast trees, after the clean-up process described in [14], is the final heterogeneous multicast tree, with the payment of each link set to be the maximum of its payments in all homogeneous multicast trees computed. Although this approach (*i.e.*, taking the union of partial outcomes and using the maximum payment of each agent over all partial outcomes as its final payment) works for binary selection problems (see [24, 10] for more details), for differentiated multicast the resulting payment scheme is no longer truthful, as demonstrated by the example in Figure 1. Figure 1.a shows the original network G containing two receivers r_1 and r_2, with bandwidth demands $d_1 = 1$ and $d_2 = 10$ respectively. For bandwidth demand rate 1, links sv_1 and v_1r_1 are selected, and for bandwidth demand rate 10, links sv_1 and v_1r_2 are selected. The final heterogeneous tree is shown in Figure 1.b (no clean-up process is necessary). The payment to sv_1 is $\max\{29, 2\cdot 10\} = 29$, while its cost is $1\cdot 10 = 10$, giving a utility of 19. However, if sv_1 reports $a_1 = 3$ instead (see Figure 1.c), its payment is still 29 (as it no longer needs to relay for r_2 with bandwidth rate 10), and yet its cost is only 1, giving an utility of 28. The reason why this approach does not work for DiffServ multicast is that the cost of a link here is no long a fixed number: it depends on the outcome of the game. Thus, although a link may not be able to change the payment it would get from the mechanism by lying its cost coefficient, it could reduce its final cost it will incur through lying. As a consequence, it still increases its utility by lying.

In this section, instead of simply presenting a truthful payment scheme for a specific tree construction method, such as Algorithm 3, we study how to design a truthful payment scheme for any given service differentiated multicast tree. In Subsection 3.1, we fist give a necessary and sufficient condition for the existence of a truthful payment scheme when given a multicast tree construction method. 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 3 and design a truthful payment scheme.



(a) Original network G (b) The spanning tree when $a_1 = 1$ (c) New tree when $a_1 = 3$

Fig. 1. The naive mechanism is not strategyproof.

3.1 General Framework

From the definition of the truthfulness, we can fix the graph G, the receiver set R and bandwidth demand d. Thus, for our notational convenience, we use $b(\mathcal{A}, a) = \{b_1(\mathcal{A}, a), \cdots, b_m(\mathcal{A}, a)\}$ to denote the bandwidth reservation vector computed by an algorithm $\mathcal{A}$, where $b_i(\mathcal{A}, a)$ is the bandwidth reserved at link e_i.

We assume that the $b_i(\mathcal{A}, a)$ is *piecewise continuous* with respect to any variable a_j, *i.e.*, a finite number of piecewise 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-increase Property (MNP)). *An algorithm $\mathcal{A}$ is said to satisfy the* monotone non-increase property *if for every link e_i and two of its possible coefficients $a_{i_1} < a_{i_2}$, $b_i(\mathcal{A}, a|^i a_{i_1}) \geq b_i(\mathcal{A}, a|^i a_{i_2})$.*

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

Theorem 1. *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 actually 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

$$\mathcal{P}_i(\mathcal{A}, a|^i a_{i_1}) - a_{i_1} \cdot b_i(\mathcal{A}, a|^i a_{i_1}) \geq \mathcal{P}_i(\mathcal{A}, a|^i a_{i_2}) - a_{i_1} \cdot b_i(\mathcal{A}, a|^i a_{i_2}).$$

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

$$\mathcal{P}_i(\mathcal{A}, a|^i a_{i_2}) - a_{i_2} \cdot b_i(\mathcal{A}, a|^i a_{i_2}) \geq \mathcal{P}_i(\mathcal{A}, a|^i a_{i_1}) - a_{i_2} \cdot b_i(\mathcal{A}, a|^i a_{i_1})$$

Combining the above two inequalities, we obtain

$$a_{i_2} \cdot [b_i(\mathcal{A}, a|^i a_{i_1}) - b_i(\mathcal{A}, a|^i a_{i_2})] \geq \mathcal{P}_i(\mathcal{A}, a|^i a_{i_1}) - \mathcal{P}_i(\mathcal{A}, a|^i a_{i_2}) \geq a_{i_1} \cdot [b_i(\mathcal{A}, a|^i a_{i_1}) - b_i(\mathcal{A}, a|^i a_{i_2})] \quad (1)$$

Thus, we have $b_i(\mathcal{A}, a|^i a_{i_1}) \geq b_i(\mathcal{A}, a|^i a_{i_2})$ 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 $b_i(\mathcal{A}, x)$ is non-increasing. Recall that $b_i(\mathcal{A}, x)$ is a piecewise continuous function. We let $x_1 < x_2 \cdots < x_m$ be the points at which $b_i(\mathcal{A}, x)$ is not continuous, and introduce a dummy point $x_{m+1} = \infty$. We define a function $\kappa_i(x)$ such that, for $x_p < x \leq x_{p+1}$,

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

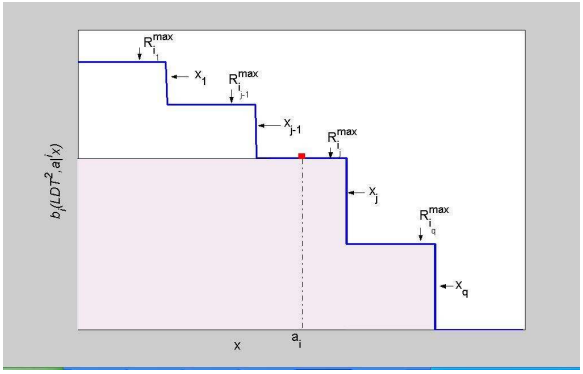


Fig. 2. Bandwidth allocation function $b_i(\overline{DMT}, a|^i x)$.

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

Algorithm 2 Payment Scheme based on $\mathcal{A}$

Input: Algorithm $\mathcal{A}$ and coefficient vector 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 2 is truthful. See Lemma 3 in the appendix for the proof of this statement. This finishes the proof of the theorem. □

If we specify that if a link e_i has 0 bandwidth reservation then it should receive 0 payment (which is called *normalized* payment scheme), then we have the following theorem. See appendix for the proof of the theorem.

Theorem 2. *Given an algorithm $\mathcal{A}$ satisfying MNP, the payment scheme defined by Algorithm 2 is the* only *normalized truthful scheme.*

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

1. Check whether the bandwidth allocation of algorithm $\mathcal{A}$ satisfies MNP. If not then return, else continue.
2. Find the bandwidth reservation $b(\mathcal{A}, a)$.
3. Design the payment according to Algorithm 2.

3.2 Design Truthful Mechanism

First of all, we would like to see whether we could design a payment scheme based on the methods presented before, especially the first constant-factor approximation method presented by Charikar *et al.* [14]. Let $T_1, T_2, \cdots, T_k$ be k different Steiner trees constructed by their method for k different demand values. For each tree T_i, we can define a strategyproof mechanism based on the criteria characterized in [10, 9]. Let $p_{e,i}$ be the payment to link e based on tree T_i and $b_{e,i}$ be the bandwidth reservation on link e based on T_i. For the union of these trees, it is unclear how a payment $\mathcal{P}$ could be defined such that mechanism $(\bigcup_{i=1}^{k} T_i, \mathcal{P})$ is truthful. Notice that when all trees are zero-one demand games, we can simply pay each link the maximum payment it could get from these k separated trees. However, here these trees are not zero-one demanded: a link has a bandwidth reservation. The profit of a link e is its final payment p_e received minus its cost $a_e \cdot b_e$, where b_e is its final bandwidth reservation. Here it is possible that the tree T_i has the maximum

payment $p_{e,i}$ to link e, while the value b_e could be different from $b_{e,i}$. Thus, simply taking maximum payment will not guarantee strategyproof here. Further more, even if we can define a strategyproof mechanism for output $\bigcup_{i=1}^{k} T_i$, it is still not clear how to extend it to a strategyproof mechanism for the output computed by "clean-up".

Thus, in this paper, we take a different approach by redesigning some new DiffServ multicast tree construction methods. Before we present our algorithm, we give some notations that will be used later. Given a network G with edge cost vector c and receiver set R, we use $T^{\min}(R, c)$ to denote the minimum weight Steiner tree where c is the cost vector of the links in the network. For a receiver set R with bandwidth demand vector $d = \{d_1, d_2, \cdots, d_k\}$, we denote the multicast tree with the minimal weight that spans R as T^{opt} and the corresponding bandwidth allocation vector as B^{opt}. Given a subset $S \subseteq R$, for notational simplicity, we use $T^{opt}(S)$ to denote the subtree in T^{opt} induced by S if no confusion is caused.

Remember that the cost function of a link e_i is $f_i(x) = a_i x$. Given a network G, a receiver set R, a cost coefficient vector a and a bandwidth demand vector d, the following algorithm shows how to find a DiffServ multicast tree $DMT(a, d)$ and its corresponding bandwidth allocation B with low weight. We also call this algorithm DMT if no confusion is caused.

Algorithm 3 Construct Differentiated Multicast Tree

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

1: Sort all receivers according to their bandwidth demands. Without loss of generality, we can assume that the receivers $R = \{r_1, r_2, \cdots, r_k\}$ are sorted in a descending order of their bandwidth demands.
2: Initialize the tree T to empty, set $t = 1$, and label all links in the tree WHITE.
3: **repeat**
4: Let r_j be the first receiver in the receiver set R.
5: Find the maximal index k such that $d_k \geq \frac{d_j}{2}$.
6: Set the cost of each WHITE link as $c_i = a_i \cdot d_j$ and each BLACK link as $c_i = 0$.
7: Let $R_t = \{r_j, \cdots, r_k\}$ and find the spanning tree $T_t = LST(R_t, c)$ using any Steiner tree heuristic, such as Algorithm 1.
8: Remove R_t from R and mark all links in tree T_t as BLACK.
9: Set $T = T \bigcup T_t$.
10: Set $t = t + 1$.
11: **until** the receiver set R is empty.
12: **for** each link e_i in tree T **do**
13: Find the maximal bandwidth demand of e_i's downstream receivers, say r_j.
14: e_i allocates a bandwidth $B_i = d_j$.
15: **end for**
16: Output tree T and bandwidth vector B.

The major difference of this method compared with the method presented by Charikar *et al.* [14] is that we directly construct a tree. Instead of rounding the demands up to the nearest power of 2, we divide the demands into several segments such that, in each segment, the ratio of the maximum demand over the minimum demand is at most 2. We also have the following theorem[3].

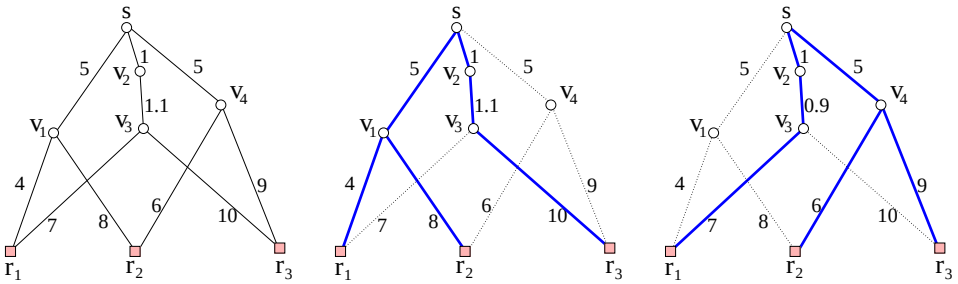
Theorem 3. *Algorithm 3 constructs a tree whose weight is at most $4\alpha_{ST}$ times the weight of the minimal cost DiffServ multicast tree T^{opt}.*

With the general framework, we would like to design a truthful payment scheme based on Algorithm 3. However, the following lemma shows that there is no such truthful payment.

Lemma 1. *Algorithm 3 does not satisfy MNP.*

PROOF. We prove it by presenting an example here. A network G has three receivers r_1, r_2, r_3 with bandwidth demand $d_1 = d_2 = 1$ and $d_3 = 2$. The coefficient of the link is described in Figure 3 (a). When we apply Algorithm

[3] Although there is a subtle difference between the algorithm presented here and the one in [14], the proof is not as obvious as that one. The proof is omitted here due to space limit.



(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.

3 to network G, we obtain a tree shown in Figure 3 (b). Let agent 2 be link v_2v_3. The bandwidth allocation of link $e_2 = v_2v_3$ is 2. Consider the scenario when the coefficient of link e_2 changes from 1.1 to 0.9 while other coefficients remain the same. The new spanning tree topology constructed by Algorithm 3 is shown in Figure 3 (c). The bandwidth allocation of e_2 becomes 1, which decreases by half compared with the bandwidth reservation with coefficient 1.1. This finishes our proof. □

From Theorem 1 and Lemma 1, we have the following theorem directly.

Theorem 4. *There is no truthful mechanism M based on Algorithm 3.*

In light of the negative result in Theorem 4, we would like to design another algorithm for constructing the differentiated service multicast tree that satisfies MNP and, in the meanwhile, has a weight that is not too large compared with the optimal. With a little modification of the Algorithm 3, we present a new method to construct the multicast tree in Algorithm 4. The trees constructed by Algorithm 3 and Algorithm 4 are the same, and the only difference is on the bandwidth allocation. In Algorithm 3, the bandwidth of a link is set to the maximum bandwidth demand of its downstream receivers. In Algorithm 4, the bandwidth of a link is set to a bandwidth greater than the bandwidth set in Algorithm 3. In order to distinguish these two algorithms, we use $\overline{DMT}$ to denote the tree constructed by Algorithm 4. We can show that our new algorithm achieves the same approximation ratio as Algorithm 3.

Theorem 5. *Algorithm 4 satisfies MNP and it constructs a tree whose weight is at most 8 times the weight of the minimal cost differentiated service tree $T^{opt}(a, d)$.*

PROOF. The proof of 8-approximation ratio is similar to the proof of Theorem 3 and is thus omitted here. We focus on the proof that Algorithm 4 satisfies MNP. Given a link e_i, if it does not appear in the tree $DMT(a, d)$ then $b_i(\overline{DMT}, a) = 0$. 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 $DMT(a, d)$ for the first time, then $b_i(\overline{DMT}, a) = R_j^{\max}$. When e_i has a smaller coefficient $\underline{a_i}$, we show by cases that it will have a larger bandwidth reservation.

Case 1: e_i is added to the spanning tree $DMT(a, d)$ no later than iteration j. Without loss of generality, we assume that e_i is added to $DMT(a, d)$ in iteration $j' \leq j$. Remember that the partition of R does not depend on coefficient vector a, thus $b_i(\overline{DMT}, a|^i\underline{a}) = R_{j'}^{\max} \geq R_j^{\max} = b_i(\overline{DMT}, a)$.

Case 2: e_i is not added to the spanning tree $DMT(a, d)$ before iteration j. In this case, every link's label does not change in the beginning of iteration j. For Algorithm 1, it has been proven in [10] that if any link originally in $LST(R, c)$ reduces its cost from c_i to c_i', then it is still in $LST(R, c|^i c_i')$. Thus, the resulting spanning tree T_j still has the link e_i in it, which means that $b_i(\overline{DMT}, a|^i\underline{a_i}) = R_j^{\max}$ keeps the same.

This proves that $b_i(\overline{DMT}, a)$ does not decrease when a_i decreases. Thus, Algorithm 4 satisfies MNP. □

From Theorem 1 and Theorem 5, we know that there exists a truthful payment for Algorithm 4. In order to find the truthful payment for Algorithm 4, we should find the bandwidth allocation function $b_k(\overline{DMT}, a|^k x)$ for every link e_k 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 $x_1^k < x_2^k < \cdots < x_q^k$ be the points at which $b_k(\overline{DMT}, a|^k x)$ is not continuous, then the bandwidth allocation function $b_k(\overline{DMT}, a|^k x)$ should be a constant, say y_j^k in (x_j^k, x_{j+1}^k) as shown in the Figure 2. In order to find the values of

Algorithm 4 Construct Multicast Tree with MNP

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

1: Sort all receivers according to their bandwidth demands in an descending order, say $R = \{r_1, r_2, \cdots, r_k\}$.
2: Initialize the tree T to empty and index $i = 1$.
3: **for** each link e_j **do**
4: Label it as WHITE and set $\overline{B}_j = 0$.
5: **end for**
6: **repeat**
7: Let r_j be the first receiver in the receiver set R.
8: Find the maximal index k such that $d_k \geq \frac{d_j}{2}$.
9: Set the cost of each WHITE link e_t as $c_t = a_t \cdot d_t$ and each BLACK link as 0.
10: Let $R_i = \{r_j, \cdots, r_k\}$ and find the spanning tree $T_i = LST(R_i, c)$ using Algorithm 1.
11: Remove R_i from R and mark all links in tree T_i as BLACK.
12: Set $T = T \bigcup T_i$.
13: **for** each link e_k in T_i **do**
14: **if** $\overline{B}_k = 0$ **then**
15: Set $\overline{B}_k = d_j$.
16: **end if**
17: **end for**
18: Set $i = i + 1$.
19: **until** the receiver set R is empty.
20: Output T as $\overline{DMT}$ and $\overline{B}$.

these discontinuous points, we first need to compute the truthful payment for standard Steiner tree problem. Please refer for [10] for more details. We use $\tau(c_{-i}, R)$ to denote the payment computed for a link e_i based on a Steiner tree heuristic and study how to find the bandwidth allocation function for Algorithm 4. Algorithm 5 shows how we can find the bandwidth-allocation function.

With the bandwidth allocation function $b_k(\mathcal{A}, a|^k x)$, we give our truthful payment scheme by following the general framework illustrated by Algorithm 6. The proof of the correctness of these algorithms are either straightforward or omitted here due to space limit.

3.3 Performance Improvement and Special Case

In essence, Algorithm 4 converts the original instance of the differentiated 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.* [14], the approximation ratio of 8 of Algorithm 4 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 nodes 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 differentiated multicast to a randomized one with an expected approximation ratio of 5.437, while maintaining strategyproofness. First of all, it is easy to see that using a "start point" of e^y 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.* [14] 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 [13], improving the approximation bound to $\frac{4}{3} \cdot 2 = 2.667$. The common characteristic of the two algorithms is to compute multiple differentiated multicast trees using different methods (or

Algorithm 5 Bandwidth Allocation Function for Algorithm 4

Input: A network G with link cost vector c, a source node s and a receiver set R with demand vector d.
Output: The bandwidth allocation function for Algorithm 4.

Apply Algorithm 4. Let ℓ be the number of iterations in Algorithm 4.
for every link e_k in $DMT(a, d)$ **do**
  Set $c_k = \infty$ and apply Algorithm 4 again.
  At the beginning of each iteration i, compute the value $\tau_k^i(a_{-k}, R_i)$.
  Initialize the list $X^k = \emptyset$, $Y^k = \emptyset$, $up = 0$, and $q = 0$.
  for $i = 1$ to ℓ **do**
    if $\tau_k^i(a_{-k}, R_i) > up$ **then**
      $q = q + 1$.
      Set $x_q^k = \tau_k^i(a_{-k}, R_i)$ and $y_q^k = R_i^{\max}$.
      Add x_q^k to set X^k and y_q^k to Y^k.
    end if
  end for
  Set x_0^k=0 and $x^{q+1} = \infty$.
  for $i = 1$ to $q + 1$ **do**
    Set $b_k(\mathcal{A}, a|^k x) = y_i^k$ for $x_{i-1}^k \le x < x_i^k$.
  end for
end for

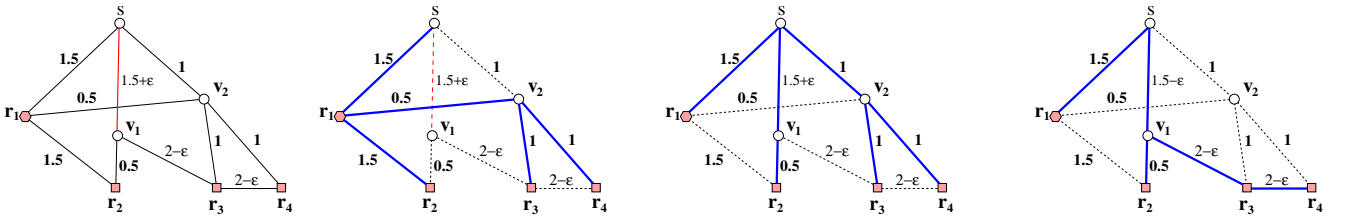
Algorithm 6 Payment Scheme for Algorithm 4

Input: A network G with link cost vector c, a source node s and a receiver set R with demand vector d.
Output: A payment scheme for Algorithm 4.

Compute the multicast tree $\overline{DMT}$ by applying Algorithm 4.
Compute the bandwidth allocation function for tree $\overline{DMT}$ by applying Algorithm 5.
for each link e_k **do**
  if e_k is in tree $\overline{DMT}$ **then**
    Find i such that $x_i^k < a_k \le x_{i+1}^k$. Then the payment is $\mathcal{P}_k(a) = \sum_{j=i+1}^{|X^k|-1} y_j^k \cdot (x_{j+1}^k - x_j^k) + (x_{i+1}^k - a_k) \cdot y_i^k$.
  else
    $\mathcal{P}_k(a) = 0$.
  end if
end for

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 the some combination of the payments for these separated games as its final payment) works for binary selection problems under certain conditions [25, 24], 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 receivers r_2, r_3, r_4 has bandwidth demand $d_2 = 4$. Let $R_1 = \{r_1\}$ and $R_2 = \{r_2, r_3, r_4\}$, c be the cost vector shown in Figure Figure 4 (a). If we change the cost of edge sv_1 from $1.5 + \epsilon$ to $1.5 - \epsilon$ while keep all other links' cost unchanged, the cost vector is denoted as c'. Figure 4 shows that the tree $LST(R, c)$ and $LST(R, c')$ is the same. We have $\omega(LST(R, c), b) = \omega(LST(R, c'), b') = 5.5 \cdot d_2 = 22$. Figure 4 (c) shows the tree $LST(R_1, c) \cup LST(R_2, c)$ and its weight is $1.5 \cdot d_1 + (5 + \epsilon) \cdot d_2 = 21.5 + 4\epsilon < \omega(LST(R, c), b)$ for small ϵ. Thus, when sv_1 has cost $1.5 + \epsilon$, it has bandwidth reservation $d_2 = 4$. Consider the cost vector c', Figure 4 (c) shows the tree $LST(R_1, c') \cup LST(R_2, c')$ and its weight is $1.5 \cdot d_1 + (6 + 3\epsilon) \cdot d_2 = 25.5 - 12\epsilon > \omega(LST(R, c'), b)$ for small ϵ. Thus, when sv_1 has cost $1.5 - \epsilon$, it has bandwidth reservation 0. This shows that the tree output by the algorithm in [13] violates the MNP property, which implies that is not such truthful payment.



(a) Original network G (b) Tree $LST(R, c)(LST(R, c'))$ (c) Tree $LST(R_1, c) \cup LST(R_1, c)$ (d) Tree $LST(R_1, c') \cup LST(R_1, c')$.

Fig. 4. An example to show that simply choose the best solution may not work.

4 Payment Sharing

4.1 Preliminaries of Sharing Scheme

In this section, we assume that each receiver is willing to pay whatever a share of total payment/cost computed as long as it is *fair* under some definitions. If the relay links are cooperative in declaring their truthful costs, *i.e.*, the costs of relay links are publically known, we essentially will study how to share the costs of the multicast tree among receivers fairly. If the relay links are selfish, then we have to share the payments to these relay links. For fair cost sharing, most of the literatures [18–20] used the *Equal Link Split Downstream* (ELSD) sharing scheme to charge receivers: the *cost* of each link is shared *equally* among all its downstream receivers. However, if we simply use the ELSD as our charging scheme to share the payment, it usually is not reasonable in common sense.

Consider a set U of n players. For a subset $S \subseteq U$ of players, let $C(S)$ be the *cost* [4] of providing service to S. Here $C(S)$ could be the minimum cost, denoted by $\mathrm{OPT}(S)$, or the cost computed by some algorithm $\mathcal{A}$, denoted by $\mathcal{A}(S)$. We always assume that the cost function $C(S)$ is *cohesive*, *i.e.*, for any two disjoint subsets S_1 and S_2, $C(S_1 \cup S_2) \leq C(S_1) + C(S_2)$. A cost sharing scheme is simply a function $\xi(i, S)$ with $\xi(i, S) = 0$ for $i \notin S$, for every set $S \subseteq U$ of players. An obvious criterion is that the sharing method should be *fair*. While the definition of budget-balance is straightforward, defining fairness is more subtle: many fairness concepts were proposed in the literature, such as *core* and *bargaining set* [26]. We call a charging scheme ξ *reasonable* or *fair* if it satisfies the following criteria.

1. **Budget Balance** (BB): The payment to all relay agents should be shared by all receivers, *i.e.*, $\mathcal{P}(R, a) = \sum_{r_i \in R} \xi(i, R)$. When budget-balance cannot be met, we relax it to β-budget-balanced. For all receivers R, $\beta \cdot C(R) \leq \sum_{i \in R} \xi(i, R) \leq C(R)$, for some given parameter $\beta \leq 1$. Equivalently, if we divide the shares by β, we would require that the total cost shares of all receivers are at least the cost of providing the service, but do not exceed $\frac{1}{\beta}$ of that.

[4] Here the cost is generic. It could be the payment to the links here.

2. **No Positive Transfer** (NPT): Any receiver r_i's sharing should not be negative. In other words, we don't pay the receiver to receive.
3. **fairness under core** (Core): For any subset $S \subseteq R$, $\sum_{i \in S} \xi(i, S) \leq \mathrm{OPT}(S)$. In other words, the cost shares paid by any subset of players should not exceed the minimum cost of providing the service to them alone, hence they have no incentives to secede.
4. **Cross-monotonicity** (CM): For any two subsets $S \subseteq T$ and $i \in S$, $\xi(i, S) \geq \xi(i, T)$. In other words, the cost share of a player i should not go up if more players require the service. This is also called *population monotone*.

When each receiver has a maximum payment it is willing to pay to receive the multicast service, several other properties could also be required. Let z_i be the willing payment by receiver r_i to receive the multicast service. Let z be the vector of the willing payments of all receivers. Some common requirements are

1. **Voluntary Participation** (VP): $z_i - \xi_z(i, S) \geq 0$ for any $S \subseteq R$. Users are always free to not receive the transmission and not be charged, which would result in an individual welfare of zero; the network can't force a user to be worse off than this baseline option.
2. **Consumer Sovereignty** (CS): If the bids of all other players are fixed, for every player i, there exists a threshold τ_i such that player i is guaranteed to get the service when its bid is at least τ_i.
3. **Group Strategyproof** (GS): No group of receivers can increase their welfare by lying about their utilities w.

In this paper, we will study the payment sharing that satisfies a subset or all of the above properties. Notice that the definition of "reasonable" can be changed due to different requirements. For example, a common criterion for multicast charging scheme is to maximize *network welfare*: select a subset of receivers such that the network welfare is maximized. Here, the network welfare is defined as the total valuations of all selected receivers minus the cost of the network providing service. Then instead of sharing the payment (or costs) among all receivers, we can only share it among the selected receivers.

It was proved in [18] that a cost-sharing mechanism satisfies BB and GS if it satisfies the BB and cross-monotone (CM) property. They also [18] offered a characterization of a whole class of budget-balanced and group strategyproof mechanisms.

4.2 Payment Sharing for DiffServ Multicast

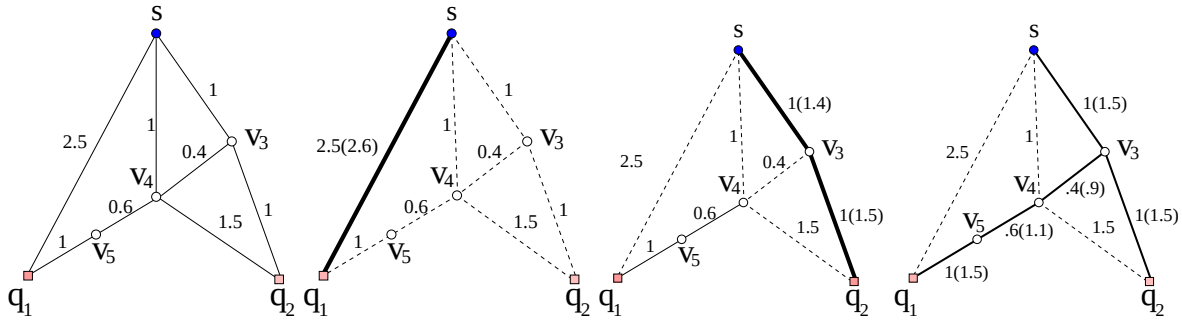
In this paper, we assume that each receiver's willing payment is infinity. We obtain the following negative result for multicast with tree construction method (illustrated by Algorithm 4) and payment scheme (illustrated by Algorithm 6).

Theorem 6. *There is no payment-sharing mechanism satisfies both BB and CM for differentiated multicast if we use tree construction method illustrated by Algorithm 4 and payment scheme illustrated by Algorithm 6.*

PROOF. Recall that the standard Steiner tree heuristic LST and its coupled payment scheme is a special case of tree construction algorithm 4 and payment scheme 6 when the bandwidth demand is homogeneous. Thus, in order to prove the above theorem, we prove that there is no payment-sharing mechanism satisfies both BB and CM for multicast with homogeneous bandwidth demand if we use tree construction algorithm 1 and payment scheme 4. We prove this by presenting a counter example. In a network G, the bandwidth demand is 1 and the costs of links are shown in Figure 5. The tree $LST(r_1, c)$ is shown in Figure 5 and the payment $\mathcal{P}(r_1, c) = 2.6$. We assume f is the payment-sharing scheme satisfying BB and GS. From the characterization of the payment-sharing satisfying BB and GS, we obtain $f_1(r_1, c) = \mathcal{P}(r_1, c) = 2.6$. The tree $LST(r_2, c)$ is shown in Figure 5 and the total payment $\mathcal{P}(r_2, c) = 1.4 + 1.5 = 2.9$. Similarly, we have $f_2(r_2, c) = 2.9$. The tree $LST(r_1 \cup r_2, c)$ is shown in Figure 5 and the total payment $\mathcal{P}(r_1 \cup r_2, c) = f_1(r_1 \cup r_2, c) + f_2(r_1 \cup r_2, c) = 6.5$. Remember that $f_i(R+j) \leq f_i(R)$ for all $i, j \in P$, thus $f_1(r_1 \cup r_2, c) \leq f_1(r_1, c) = 2.6$ and $f_2(r_1 \cup r_2, c) \leq f_2(r_1, c) = 2.9$. Therefore, $f_1(r_1 \cup r_2, c) + f_2(r_1 \cup r_2, c) = 6.5 \leq 2.9 + 2.6 = 5.5$, which is a contradiction. This finishes our proof. ◻

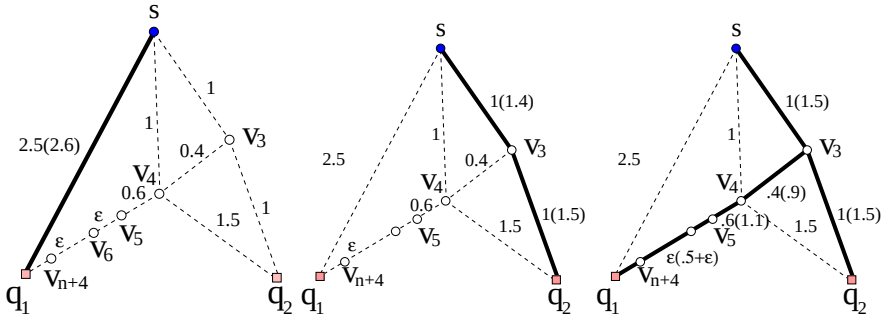
With the negative result from Theorem 6, we have to relax the requirement of BB for the payment sharing scheme if cross-monotone is needed. Given a payment sharing scheme, if the total charge from the receivers is at least β times of the total payment to the links, then we call this payment achieves a β-budget-balance. If it is both β-budget-balance and in the core, then it is called β-core. We first present the following result about the β-core payment sharing scheme.

Lemma 2. *There is no β-core payment sharing scheme for $\beta = \Omega(1/n)$ if Steiner tree heuristic LST is used.*



(a) The network G and the costs (b) The tree $LST(r_1, c)$ and the payments (c) The Tree $LST(r_2, c)$ and the payments (d) The tree $LST(r_1 \cup r_2, c)$ and the payments

Fig. 5. A counter example.



(a) The tree $LST(r_1, c)$ and the payments (b) The Tree $LST(r_2, c)$ and the payments (c) The tree $LST(r_1 \cup r_2, c)$ and the payments

Fig. 6. A bad example.

PROOF. We briefly show it by the example shown in Figure 6. The receivers q_1 and q_2 have demand 1. There are $n-1$ nodes between nodes q_1 and v_4: v_5, v_6, $\cdots$, v_{n+4}. Every link $v_i v_{i+1}$, for $i \geq 5$ has cost $\epsilon = \frac{1}{n}$. The payment to each link is shown in Figure 6. If q_1 and q_2 play along, we have $\xi(q_1, \{q_1\}) \leq 2.6$ and $\xi(q_2, \{q_2\}) \leq 2.9$. If q_1 and q_2 are receivers, we have $\xi(q_1, \{q_1, q_2\}) + \xi(q_2, \{q_1, q_2\}) \leq 5.5$ from the CM property. On the other hand, notice that the total payment to links by providing service to q_1 and q_2 are $C(q_1 \cup q_2) = 5 + n \cdot (0.5 + \epsilon) = 6 + 0.5n$. Thus, if a payment sharing scheme is β-core we should have $\beta = \Omega(1/n)$. □

Currently, we are able to design a payment sharing scheme that is $\frac{1}{n^2}$-core for any strategyproof payment schemes for DiffServ multicast. The detailed methods are omitted here due to space limit.

4.3 Cost Sharing for DiffServ Multicast

In this subsection, we study how to share the cost of DiffServ multicast among the set of receivers fairly. For the cost sharing scheme, we first compute the *LCPT cost sharing*, then divide by $\frac{1}{r}$ where r is the number of the receivers. The algorithm is summarized as following.

Although Algorithm 7 use the structure LCPT, but the actual multicast routing tree is still constructed by Algorithm 4. Regarding the cost sharing scheme 7, we have the following theorem. The proof of this theorem is similar to [9] and thus is omitted.

Theorem 7. *The cost sharing scheme 7 is cross-monotonic and $\frac{1}{8r}$-budget balance, where r is the number of the receivers.*

When the DiffServ multicast tree is actually constructed based on LCPT, then the above cost-sharing scheme is budget-balanced and in the core.

Theorem 8. *The cost sharing scheme 7 is cross-monotonic, budget-balanced, and in the core if the DiffServ multicast tree is actually constructed based on LCPT.*

Algorithm 7 Cost Sharing Scheme for DiffServ Multicast

Input: A network $G = (V, E)$, cost coefficients vector a, receiver set R, and the demand vector d.
Output: A cost sharing method ξ that is fair.

1: Set the cost $c_i = a_i$ for each link e_i.
2: Find the shortest path between source node s and each receiver $r_i \in R$.
3: Union all these pathes to form a tree called Least Cost Path Tree (LCPT).
4: **for** every link e_i in the LCPT **do**
5: Sort e_i's downstream receivers according to their demands in an ascending order. If two or more receivers have the same value, the receiver with smaller ID ranks first. Let $\{r_{\sigma_0}, r_{\sigma_1}, \cdots, r_{\sigma_{q(e_i)}}\}$ be the downstream receivers. Here, we add a dummy receiver with demand $d_{\sigma_0} = 0$ to ranking σ.
6: For receivers that are not downstream receivers of e_i, its sharing is 0.
7: For a receiver q_{σ_k} who is a downstream receiver of e_i, its sharing is:

$$f^i_{\sigma_k}(R, a) = \sum_{x=1}^{k} \frac{a_k \cdot (d_{\sigma_x} - d_{\sigma_{x-1}})}{q(e_i) - x + 1} \tag{2}$$

8: **end for**
9: The final sharing of receiver r_i is

$$\xi(i, R) = \frac{\sum_{e_j \in E} f^j_i(R, a)}{|R|}$$

Based on this cost sharing scheme, we can also design a payment sharing scheme: treat the cost sharing of a receiver simply as its shared payment. Then we can prove the following theorem.

Theorem 9. *The payment sharing scheme induced from the cost sharing scheme 7 is cross-monotonic and $\frac{1}{8r\cdot\gamma}$-budget balanced, where r is the number of the receivers and γ is the overpayment ratio of our DiffServ multicast.*

PROOF. First, we prove that it is cross-monotonic. It was proved in [9] that $f^j_i(R, a) \geq f^j_i \sigma_k(R', a)$ for any receiver r_i, link e_j and $R \in R'$. Thus, we have

$$\frac{f^j_i(R, a)}{|R|} \geq \frac{f^j_i(R', a)}{|R'|}$$

for any receiver r_i, link e_j and $R \in R'$. Therefore, $\xi(i, R') = \frac{\sum_{e_j \in E} f^j_i(R', a)}{|R'|} \leq \frac{\sum_{e_j \in E} f^j_i(R', a)}{|R'|} \leq \xi(i, R)$. This proves that the cost sharing scheme is cross-monotonic. Following we prove that payment sharing scheme is $\frac{1}{8r\cdot\gamma}$-budge balanced. In other word, for any receiver set R, we should prove that $\frac{\mathcal{P}(R,a)}{8r\cdot\gamma} \leq \sum_{r_i \in R} \xi(i, R) \leq \mathcal{P}(R, a)$. For the tree LCPT constructed in Algorithm 7, if we assign every link e_i in LCPT a cost $a_i \cdot b_i$, where b_i is the maximum demand of e_i's downstream receivers, the cost of the tree LCPT is denoted as $LCPT(R, a)$. Let $T^{opt}(R, a)$ be the tree with the minimal cost, then $|\mathsf{LCP}(s, r_i, G)| \leq \omega(T^{opt}(R, a))$ for any $r_i \in R$. Thus, we have

$$|LCPT(R, a)| \leq \sum_{r_i \in R} |\mathsf{LCP}(s, r_i, G)| \leq r \cdot \omega(T^{opt}(R, a))$$

Recall that $\sum_{r_i \in R} \xi(i, R) = \frac{|LCPT(R,a)|}{r} \leq \omega(T^{opt}(R, a)) < \mathcal{P}(R, a)$, which proves one direction. For another direction, remember that $|LCPT(R, a)| \geq T^{opt}(R, a) \geq \frac{\omega(DMT(a,d),\overline{B})}{8} \geq \frac{\mathcal{P}(R,a)}{8\gamma}$. Thus, we have $\sum_{r_i \in R} \xi(i, R) = \frac{|LCPT(R,a)|}{r} \geq \frac{\mathcal{P}(R,a)}{8\gamma}$, which finishes our proof. □

5 Conclusion

In this paper, we studied the differentiated 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; the mechanism is either not strategyproof, or the resulting network structure is not a tree. We then propose an alternative approximation algorithm for differentiated multicast with the same approximation bound. We also introduced a general method to convert any differentiated multicast algorithm satisfying the Monotone Non-increase Property to a strategyproof mechanism, and applied it to the algorithm we proposed. Finally, we showed how the payments to the links can be shared fairly among nodes demanding multicast services.

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. Klein, P., Ravi, R.: A nearly best-possible approximation algorithm for node-weighted steiner trees. J. Algorithms **19** (1995) 104–115
6. Guha, S., Khuller, S.: Improved methods for approximating node weighted steiner trees and connected dominating sets. In: Foundations of Software Technology and Theoretical Computer Science. (1998) 54–65
7. Takahashi, H., Matsuyama, A.: An approximate solution for the steiner problem in graphs. Mathematical Japonica **24** (1980) 573–577
8. Robins, G., Zelikovsky, A.: Improved steiner tree approximation in graphs. In: Proceedings of ACM/SIAM Symposium on Discrete Algorithms. (2000) 770–779
9. Wang, W., Li, X.Y., Sun, Z., Wang, Y.: Design multicast protocols for non-cooperative networks. In: IEEE INFOCOM. (2005) Accepted for publication.
10. Wang, W., Li, X.Y., Wang, Y.: Truthful mutlicast in selfish wireless networks. In: ACM MobiCom 2004. (2004)
11. Karpinski, M., Mandoiu, I.I., Olshevsky, A., Zelikovsky, A.: Improved approximation algorithms for the quality of service steiner tree problem. In: WADS. (2003)
12. 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
13. Balakrishnan, A., Magnanti, T., Mirchandani, P.: Heuristics, lps, and trees on trees: Network design analyses. Operations Research (1996) 478–496
14. Charikar, M., Naor, J.S., Schieber, B.: Resource optimization in qos multicast routing of real-time multimedia. In: IEEE INFOCOM. (2000)
15. Calinescu, G., Fernandes, C., Mandoiu, I., Olshevsky, A., Yang, K., Zelikovsky, A.: Primal-dual algorithms for qos multimedia multicast. In: GlobeCom. (2003)
16. Xue, G., Lin, G.H., Du, D.Z.: Grade of service steiner minimum trees in the euclidean plane. Algorithmica (2001) 479–500
17. 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
18. Moulin, H., Shenker, S.: Strategyproof sharing of submodular costs: Budget balance versus efficiency. In: Economic Theory. Volume 18. (2001) 511–533
19. Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the cost of multicast transmissions. Journal of Computer and System Sciences **63** (2001) 21–41
20. Herzog, S., Shenker, S., Estrin, D.: Sharing the "cost" of multicast trees: an axiomatic analysis. IEEE/ACM Transactions on Networking **5** (1997) 847–860
21. Vickrey, W.: Counterspeculation, auctions and competitive sealed tenders. Journal of Finance (1961) 8–37
22. Clarke, E.H.: Multipart pricing of public goods. Public Choice (1971) 17–33
23. Groves, T.: Incentives in teams. Econometrica (1973) 617–631
24. Kao, M.Y., Li, X.Y., Wang, W.: Polynomial-time truthful mechanisms for binary selection problems: A general design framework (2004) Submitted for publication.
25. 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
26. Osborne, M.J., Rubinstein, A.: A course in game theory. The MIT Press (2002)

6 Appendix

Theorem 3 Algorithm 3 constructs a tree whose weight is at most 8 times the weight of the minimal cost differentiated service tree T^{opt}.

PROOF. For notational convenience, we use T and B to denote the tree and bandwidth allocation vector output by Algorithm 3. Remember that T_i is the tree found in the ith iteration by applying Algorithm 1. Without loss of generality, we assume that there are l iterations in Algorithm 3. $R_1, R_2, \cdots, R_l$ is a partition of receiver set R, and we use $R_i^{\max}$(respectively $R_i^{\min}$) to denote the maximum (respectively minimal) bandwidth demand in the receiver set R_i.

Notice that every link in $T^{opt}(R_1)$ should be able to supply a bandwidth larger than $R_1^{\min}$, then

$$\begin{aligned}
&\omega(T_1, B) \leq \cdot\omega(T_1, \langle R_1^{\max}\rangle)\\
&\leq 2R_1^{\max} \cdot \omega(T^{\min}(R_1, a), \langle 1\rangle)\\
&\leq 2R_1^{\max} \cdot \omega(T^{opt}(R_1), \langle 1\rangle)\\
&= 2R_1^{\max} \cdot \sum_{e_i \in T^{opt}(R_1)} a_i \leq 4R_1^{\min} \cdot \sum_{e_i \in T^{opt}(R_1)} a_i\\
&= 4\omega(T^{opt}(R_1), \langle R_1^{\min}\rangle)) \leq 4\omega(T^{opt}(R_1), B^{opt})
\end{aligned}$$

For set R_2, we have

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

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

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

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

$$\begin{aligned}
&\omega(T, B) = \omega(\bigcup_{i=1}^{l} T_i, B) \leq \sum_{i=1}^{l} \omega(T_i, B)\\
&\leq \sum_{i=1}^{l} 4 \sum_{j=1}^{i} \frac{1}{2^{i-j}} \omega(\langle T^{opt}(R_j) - \bigcup_{k=1}^{j-1} T^{opt}(R_k), B^{opt})\\
&= 4\sum_{i=1}^{l} [\omega(\langle T^{opt}(R_i) - \bigcup_{j=1}^{i-1} T^{opt}(R_j), B^{opt}) \cdot \sum_{k=0}^{l-i} 2^{-k}]\\
&= 8\sum_{i=1}^{l} [\omega(\langle T^{opt}(R_i) - \bigcup_{j=1}^{i-1} T^{opt}(R_j), B^{opt})]\\
&= 8\omega(T^{opt}(\bigcup_{i=1}^{l} R_i), B^{opt})) = 8\omega(T^{opt}(R), B^{opt}))
\end{aligned}$$

This finishes our proof. □

Lemma 3. *Algorithm 2 defines a truthful payment scheme.*

PROOF. Hereafter, we always fix a_{-i}, *i.e.*, we are interested only in a_i. To simplify our notation, we denote $b_i(\mathcal{A}, a_i)$ as $b_i(a_i)$. Notice that when e_i reveals its true coefficient a_i, its utility is

$$u_i(a_i) = \mathcal{P}_i(a_i) - a_i \cdot b_i(a_i) = \kappa_i(a_i) - a_i \cdot b_i(a_i) = \int_{a_i}^{x_{p+1}} b_i(y)dy + \sum_{j=p+1}^{m} \int_{x_j}^{x_{j+1}} b_i(y)dy$$

Remember that $b_i(y)$ is non-negative. Thus $u_i(a_i) \geq 0$, which implies that payment scheme 2 satisfies IR. To prove that payment scheme 2 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

$$u_i(\overline{a_i}) = p_i(\overline{a_i}) - a_i \cdot b_i(\overline{a_i}) = \xi(\overline{a_i}) - a_i \cdot b_i(\overline{a_i}) = \int_{\overline{a_i}}^{x_{p'+1}} b_i(y)dy + \sum_{j=p'+1}^{m} \int_{x_j}^{x_{j+1}} b_i(y)dy + \overline{a_i} \cdot b_i(\overline{a_i}) - a_i \cdot b_i(\overline{a_i})$$

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

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

If $p = p'$ then

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

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 $u_i(\underline{a_i}) = b_i(\underline{a_i}) \cdot (\underline{a_i} - a_i) + \int_{\underline{a_i}}^{x_{p'+1}} b_i(y)dy + \sum_{j=p'+1}^{m} \int_{x_j}^{x_{j+1}} b_i(y)dy$.

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

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

If $p = p'$ then

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

This proves that node i does not have incentive to lie downward. Thus, the payment scheme 2 satisfies IC. Therefore, the payment scheme 2 is truthful. □

Theorem 2 Given an algorithm $\mathcal{A}$ satisfying MNP, the payment scheme defined by Algorithm 2 is the *only* normalized truthful scheme.

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

$$(x+\delta)\cdot d(-b_i(x)) \geq d(-\mathcal{P}_i(x)) \geq x \cdot d(-b_i(x)) \tag{3}$$

From equation 3, if x is continuous in (l,u), then we obtain

$$-p_i(x)|_l^u = p_i(l)-p_i(u) = \int_l^u x d(-b_i(x)) = -\int_l^u x d(b_i(x)) = -[xb_i(x)|_l^u - \int_l^u b_i(x)dx] = l\cdot b_i(l) - u\cdot b_i(u) + \int_l^u b_i(x)dx$$

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

$$\mathcal{P}_i(x_j) - \mathcal{P}_i(x_{j+1}) = x_j \cdot b_i(x_j) - x_{j+1}\cdot b_i(x_{j+1}) + \int_{x_j}^{x_{j+1}} b_i(x)dx$$

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

$$\mathcal{P}_i(x_{p+1}) = \mathcal{P}_i(x_{p+1}) - \mathcal{P}_i(x_{q+1}) = \sum_{j=p+1}^{q} p_i(x_j) - p_i(x_{j+1})$$

$$= \sum_{j=p+1}^{q} [x_j \cdot b_i(x_j) - x_{j+1}\cdot b_i(x_{j+1})] + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} b_i(x)dx = x_{p+1}\cdot b_i(x_{p+1}) + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} b_i(x)dx$$

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

$$\mathcal{P}_i(a_i) = x_{p+1}\cdot b_i(x_{p+1}) + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} b_i(x)dx + a_i\cdot b_i(a_i) - x_{p+1}\cdot b_i(x_{p+1}) + \int_{a_i}^{x_{p+1}} b_i(x)dx$$

$$= a_i\cdot b_i(a_i) + \int_{a_i}^{x_{p+1}} b_i(y)dy + \sum_{j=p+1}^{q}\int_{x_j}^{x_{j+1}} b_i(x)dx$$

This finishes our proof. □