

# Low-Cost Routing in Selfish and Rational Wireless Ad Hoc Networks

WeiZhao Wang\*    Xiang-Yang Li\*

**Abstract**—Numerous routing protocols have been proposed for wireless networks. A common assumption made by the majority of these protocols is that each wireless node will follow the prescribed protocol without any deviation. This may not be true in practice since wireless nodes could be owned by users who perform in their own interests. We then have to design routing protocols that still work properly even for networks composed of selfish nodes. In this paper, we propose a unicast routing protocol to address this issue under the assumption that all networking nodes are rational. Here a node is rational if it always chooses a strategy that maximizes its benefit.

We assume that each node has a privately known cost of relaying a unit of data for other nodes. In our protocol, each wireless node has to declare a cost for forwarding a unit of data. When a node wants to send data to the access point, it first computes the least cost path to the access point and then computes a payment to each node on this path. We present a pricing mechanism such that the profit of each relay node is maximized when it declares its true cost. We also give a time optimal method to compute the payment in a centralized manner. We then discuss in detail how to implement the routing protocol in the distributed manner. We conduct extensive simulations to study the ratio of the payment by a source node over the total cost incurred by all relay nodes.

We find that this ratio is small in practice. Our protocol works when the wireless nodes will *not* collude and we show that no truthful mechanism can avoid the collusion of any pair of two nodes. We also give truthful mechanism when a node only colludes with its neighbors.

**Index Terms**—Non-cooperative computing, unicast, game theory, wireless ad hoc networks.

## I. INTRODUCTION

Wireless networks have become increasingly important with the requirement for enhanced data and multimedia communications in ad hoc environments. While infrastructured networks are common, there are a growing number of applications that require multi-hop infrastructureless mobile wireless networks. In a multi-hop wireless network, each wireless node can only send signal to nodes within some transmission range.

A source node communicates with far off destinations by using intermediate nodes as relays. A common assumption made by the majority of the wireless ad hoc routing protocols is that each wireless node will follow the prescribed protocols without any deviation, e.g., a node is always assumed to relay data packets for other nodes if it is asked to do so by the routing protocols. However, this may not be true in reality: the wireless nodes could be owned by individual users and thus they will perform in their own interests; the wireless nodes are often powered by batteries only, thus it is not in the best interest of a wireless node to always forward the data packets for

other nodes. If a node refuses to relay the data while the routing protocol assumed that it will, the throughput of the network may decrease and even the network connectivity may be broken *de facto*. In other words, selfish wireless nodes may hinder the functioning of the network completely. Thus, a stimulation mechanism is required to encourage the users to provide service to other nodes.

Dealing with selfish users has been well-studied in game theory and economics. Recently, there have been a sequence of results [1], [2], [3], [4], [5], [6] published in the theoretical computer science area that tried to solve various problems when the agents are selfish and rational. Here an agent is *rational* if it always chooses a strategy that maximizes its own gain. A common setting in all these results is that each agent incurs a cost if it is selected to provide the service. For example, in wireless networks, each node will incur an energy cost (and possibly memory cost) when it is asked to relay the data for other nodes. Several protocols [7], [8], [9], [10], [11], [12], [13] have also been proposed recently to address the non-cooperative issue in wireless ad hoc networks. Some protocols [7], [10], [11] pay the relay node a virtual *nugget* when it relays a data packet for other nodes and charge the source node  $h$  *nuggets* when it initiates a unicast with  $h$ -hops. However, the concept of *nugget* does not reflect the actual cost of a node relaying the data packet, and thus nodes may still refuse to relay data packets. In this paper, we are interested in designing a routing protocol that will compensate a relaying node with a monetary value that is at least its actual cost.

We consider a set of wireless nodes  $V = \{v_0, v_1, \dots, v_{n-1}\}$ . Each node  $v_i$  is associated with an average cost  $c_i$  to forward a data packet and this cost is only known to node  $v_i$  itself. Here the cost  $c_i$  may also include the minimum amount of profit  $z_i$  that node  $v_i$  wants to make if it is chosen to relay packets for other nodes. For simplicity, in this paper, we assume that the cost  $c_i$  is *fixed* and node  $v_i$  knows this monetary value. This is also a common assumption made in all mechanism design results [4], [6], [14] that can achieve strategyproofness. If its cost changes, node  $v_i$  has to declare this new cost as we will see later so its profit is maximized. Notice that our results can be simply extended to the scenario when  $c_i$  reflects the cost of node  $v_i$  relaying data for one communication session instead of a data packet.

We assume that, in order to stimulate cooperation among all wireless nodes, every wireless node is willing to pay other nodes for relaying its data to and from the access point (or other wireless nodes in general). For simplicity, we use the access point as the sole destination. We will use node  $v_0$  to represent the access point of the wireless network to the wired net-

\*Department of Computer Science, Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616. Emails: wangwei4@iit.edu, xli@cs.iit.edu. The work of the second author is partially supported by NSF CCR-0311174.

work and, in addition, the routing from each node to the access point is connection-oriented. Our routing protocol works as follows. First, each node  $v_j$  in the network declares a cost  $d_j$  for relaying a data packet of unit size, which could be different from its true cost  $c_j$ . Secondly, the source node  $v_i$  computes the least cost path (LCP), denoted by  $\mathbf{P}(v_i, v_0, d)$ , from  $v_i$  to the access point  $v_0$  according to the declared cost vector  $d = (d_0, d_1, \dots, d_{n-1})$ . After that, a payment  $p_i^j(d)$  to node  $v_j$  is computed (either in a centralized or in a distributed way) for every node  $v_j$ . The data is then routed along the computed LCP. Each node on the LCP is asked to relay the data packets and is compensated by the computed payment. The *utility* or called *profit* by some researchers of node  $v_j$  is  $u^j(d) = p_i^j(d) - c_j$  if node  $v_j$  relays data for  $v_i$ ; otherwise it is  $p_i^j(d)$ . Naturally, it is preferred that each node  $v_j$  declares a cost  $d_j = c_j$  (called *truthful or strategyproof* in this paper). Since we assumed that wireless nodes are selfish and rational, there is no guarantee that any wireless node will reveal its cost truthfully unless it is convinced that it cannot do better by declaring a different cost. The first objective of this paper is then to design a payment scheme such that each node  $v_j$  will always maximize its profit when it declares its true cost, i.e.,  $d_j = c_j$ , no matter what other wireless nodes do. The second objective of this paper is to implement the protocol efficiently and truthfully.

By assuming that the nodes will not collude with each other, we present a strategyproof pricing mechanism such that the profit of each wireless node is maximized when it declares its true cost. In our protocol, the payment to a node  $v_k$  on the LCP is  $d_k$  plus the difference between the cost of the least cost path without using  $v_k$  and the cost of the least cost path. The payment to any node not on the LCP is always 0. Our protocol does not need the routing to be performed repeatedly. To simplify our analysis, we assume that the network topology remains the same for the period of routing of current traffic request from a source node to the access point. Our main contributions of this paper are follows. First, we present a centralized method to compute the payment in time  $O(n \log n + m)$ , where  $n$  is the number of wireless nodes, and  $m$  is the number of wireless links. Clearly this is asymptotically optimum since we have to spend  $O(n \log n + m)$  time just to compute a least cost path for routing. Secondly, we discuss in detail how to implement our routing protocol in a distributed manner. Notice that it is not straightforward that we can implement a protocol in a distributed manner, even without considering the communication complexity. The difficulty here is how to rely on these selfish wireless nodes to *truthfully* implement our protocols that will prevent them from lying about its cost. Notice that since we pay more than its actual cost of a node to relay the data packets, the total payment to all nodes relaying packets is clearly at least the same as the actual cost of the least cost path. It is easy to construct a worst case example in which we could pay much more than the actual cost. We then conduct extensive simulations to study the ratio of the payment to all relay nodes by the source node over the total cost incurred by all relay nodes. We studied both the ratio averaged for all source nodes and the maximum ratio among all source nodes. We find that these ratios are all small when the cost of wireless nodes are randomly distributed in an interval.

The rest of the paper is organized as follows. In Section II, we briefly discuss what is algorithmic mechanism design and what is the network model used in this paper, and formally define the problem we want to solve. We present our pricing mechanism in Section III. We discuss in detail how to compute the payment fast and how to compute it in a distributed manner. In Section IV, we show that there is no pricing mechanism that can prevent an arbitrary pair of nodes from colluding. We then give a mechanism that can prevent the neighbors from colluding with each other. We also study the case when the cost is incurred on communication links instead of wireless nodes. We conduct extensive simulations to show that the overpayment to the relay nodes according to the pricing scheme is small compared with the actual cost of the least cost path. We review the prior arts on dealing with non-cooperative computing and wireless networks in Section V. We conclude our paper in Section VI with a discussion of possible future works.

## II. TECHNICAL PRELIMINARIES

### A. Algorithmic Mechanism Design

Conventionally, in economics and mechanism design theory, the scenarios in which the agents act according to their own self-interests are modeled as follows. There are  $n$  agents and each agent  $i$ , for  $i \in \{1, \dots, n\}$ , has some private information  $t^i$ , called its *type*. The type  $t^i$  is a node's cost to forward a packet in unicast scenario. All agents' types  $t = (t^1, t^2, \dots, t^n)$  define a type vector, which is called the *profile*. Given a reported profile, there is an output specification  $\mathcal{O}$  that maps each type vector  $t$  to an allowed output. For each possible output  $o$  agent  $i$ 's preferences are given by a valuation function  $w^i$  that assigns a real number  $w^i(t^i, o)$ , which does not depend on other agents' types.

Given a reported profile  $a = (a^1, \dots, a^n)$ , a mechanism defines an *output*  $\mathcal{O}(a)$  and a *payment* vector  $p(a) = (p^1(a), \dots, p^n(a))$ , where  $p^i = p^i(a)$  is the money given to each participating agent  $i$ . Agent  $i$ 's *utility* is  $u^i(a) = w^i(t^i, \mathcal{O}(a)) + p^i(a)$ . By assumption of rationality, agent  $i$  always tries to maximize its utility  $u^i(a)$  by choosing its action  $a^i$ . A mechanism satisfies the *incentive compatibility* (IC) if each agent maximizes its utility by reporting its type  $t^i$  regardless of what other agents do. A mechanism satisfies the *individual rationality* (IR) (or called *voluntary participation*) if each agent's utility of participating in the action is non-negative. A mechanism is *strategy-proof* (or called *truthful*) if it satisfies both IC and IR properties.

One of the most celebrated positive results in mechanism design is what is usually called the family of Vickrey-Clarke-Groves (VCG) mechanisms by Vickrey [16], Clarke [17], and Groves [18]. VCG mechanisms apply to mechanism design maximization problems where the objective function  $g(o, t)$  is the sum of all agents' valuations, i.e.,  $g(o, t) = \sum_i w^i(t^i, o)$ , and the set of possible outputs is assumed to be finite. This maximization mechanism design problem is often called *utilitarian*. A mechanism  $M = (\mathcal{O}(t), p(t))$  belongs to the family of VCG mechanisms if (1) the output  $o = \mathcal{O}(t)$  computed based on the type vector  $t$  maximizes the objective function  $g(o, t) = \sum_i w^i(t^i, o)$ , and (2) the payment to an agent  $i$  is

of format  $p^i(t) = \sum_{j \neq i} w^j(t^j, \mathcal{O}(t)) + h^i(t^{-i})$ . Here  $h^i(\cdot)$  is an arbitrary function of  $t^{-i}$ . Groves [18] proved that a VCG mechanism is truthful. Green and Laffont [19] proved that, under mild assumptions, VCG mechanisms are only truthful implementations for utilitarian problems.

Let  $a^{-i}$  denote the vector of strategies of all agents except  $i$ , i.e.,  $a^{-i} = (a^1, a^2, \dots, a^{i-1}, a^{i+1}, \dots, a^n)$ . Let  $a^i | b = (a^1, a^2, \dots, a^{i-1}, b, a^{i+1}, \dots, a^n)$ , i.e., each agent  $j \neq i$  uses strategy  $a^j$  except that the agent  $i$  uses strategy  $b$ .

### B. Network Model

We consider a set  $V = \{v_0, v_1, \dots, v_{n-1}\}$  of  $n$  wireless nodes. Here  $v_0$  is used to represent the access point (AP) of the wireless network. Let  $G = (V, E)$  be the communication graph defined by  $V$ , where  $E$  is the set of links  $(v_i, v_j)$  such that the node  $v_i$  can communicate directly with the node  $v_j$ . We assume that  $G$  is node bi-connected. In other words, the remaining graph, by removing any node  $v_i$  and its incident links from  $G$ , is still connected. The bi-connectivity of the communication graph  $G$  will prevent the monopoly of nodes as will see later in addition to provide fault tolerance.

In wireless networks, if two nodes cannot communicate directly, they communicate through multi-hop wireless links by using some intermediate nodes to relay the message. Consequently, each node in the wireless network also acts as a router to forward data packets for other nodes. We assume that each wireless node  $v_i$  has a fixed cost  $c_i$  of relaying/sending a data packet to any (or all) of its outgoing neighbors. This cost  $c_i$  is a private information, only known to node  $v_i$ . All  $n$  nodes together define a cost vector  $c = (c_0, c_1, \dots, c_{n-1})$ , which is the profile of the network  $G$ . When a node  $v_i$  wants to make a guaranteed profit  $z_i$ , its declared ‘‘cost’’ should be  $c_i + z_i$ . It will be seen later that node  $v_i$  makes a profit at least  $z_i$  when it is chosen to relay the data. In the remaining of our study, we assume that the source node will always pay the relay nodes for relaying its data, i.e., the source node has an infinitely large valuation of the data being sent to the destination. Our results can be easily extended to deal with the case when the source node  $v_i$  has a fixed valuation  $b_i$  of the data being sent to the destination. The source node chooses to send the data via relay nodes if and only if the total payments to all these relay nodes is no more than  $b_i$ . It is easy to show that the source node will also be truthful about  $b_i$ .

In this paper we restrict our attentions to a unicast between any node  $v_i$  and the access point  $v_0$  only. Our results can be easily extended to the routing between an arbitrary pair of source node  $v_i$  and destination node  $v_j$ .

### C. Statement of Problem

If a node  $v_i$  wants to send data to the access point  $v_0$ , typically, the least cost path (with minimum total relaying cost) from node  $v_i$  to node  $v_0$ , denoted by  $\mathbf{P}(v_i, v_0, c)$ , is used to route the packets. Consider a path  $\Pi(i, 0) = v_{r_s}, v_{r_{s-1}}, \dots, v_{r_1}, v_{r_0}$  connecting node  $v_i$  and node  $v_0$ , i.e.,  $v_{r_s} = v_i$  and  $v_{r_0} = v_0$ , and node  $v_{r_j}$  can send signal directly to node  $v_{r_{j-1}}$ . The cost of the path  $\Pi(i, 0)$  is defined as  $\sum_{j=1}^{s-1} v_{r_j}$ , which excludes the costs of the source node and the target node.

To stimulate cooperation among all wireless nodes, node  $v_i$  pays some nodes of the network to forward the data to the access point. Thus, each node  $v_j$  on the network declares a cost  $d_j$ , which is its claimed cost to relay the packets. Note that here  $d_j$  could be different from its true cost  $c_j$ . Then node  $v_i$  computes the least cost path  $\mathbf{P}(v_i, v_0, d)$  to connect to the access point  $v_0$  according to the declared cost vector  $d = (d_0, d_1, \dots, d_{n-1})$ . For each node  $v_j$ , a payment  $p_i^j(d)$  is computed according to the declared cost vector  $d$ . The utility of node  $v_j$  is  $u^j(d) = p_i^j(d) - x_j(i) \cdot c_j$ , where  $x_j(i) \in \{0, 1\}$  indicates whether  $v_j$  relays the packet for  $v_i$ . We always assume that the wireless nodes are rational: it always tries to maximize its utility  $u^j(d)$ .

We assume that the cost  $c_i$  is based on per packet or per session, whichever is appropriate. If the cost is per packet and a node  $v_i$  wants to send  $s$  packets to the access point  $v_0$  in one session, then the actual payment of  $v_i$  to a node  $v_k$  is  $s \cdot p_i^k$  for that session.

If the payment scheme is not well-designed, a node  $v_j$  may improve its utility by lying its cost, i.e., declares a cost  $d_j \neq c_j$ . The objective of this paper is then to design a payment scheme such that each node  $v_j$  maximizes its utility as long as it declares its true cost, i.e.,  $d_j = c_j$ . Using the standard assumption from economic model, we assume that the wireless nodes do *not* collude to improve their utilities. We will relax this assumption later.

## III. THE PRICING MECHANISM

### A. Payment Scheme

Assume that the node  $v_i$  has to send packets to  $v_0$  through the relay of some other nodes. It pays these relay nodes to compensate their costs for carrying the transit traffic incurred by  $v_i$ . The output  $\mathcal{O}(d)$  of the mechanism is the path connecting  $v_i$  and  $v_0$  with the minimum cost, which is known as  $\mathbf{P}(v_i, v_0, d)$ . The payment to a node  $v_k$  is 0 if  $v_k \notin \mathbf{P}(v_i, v_0, d)$ . Otherwise, its payment is  $p_i^k(d) = \|\mathbf{P}_{-v_k}(v_i, v_0, d)\| - \|\mathbf{P}(v_i, v_0, d)\| + d_k$ . Here  $\mathbf{P}_{-v_k}(v_i, v_0, d)$  denotes the least cost path between node  $v_i$  and  $v_0$  without using node  $v_k$ , and  $\|\Pi\|$  denotes the total cost of a path  $\Pi$ .

This payment falls into the VCG mechanism, so it is strategy-proof. In other words, if  $d_k = c_k$ , node  $v_k$  maximizes its utility  $p_i^k(d) - x_k(i) \cdot c_k$ . Every node participating in the relay will have a *non-negative* profit; every node that does not relay the traffic will have profit 0. Notice that when a node originally is not chosen with its true cost, its profit will become *negative* if it tries to lie its cost such that it is chosen. Clearly, to make sure that the payment is well-defined, we need that the network is bi-connected, i.e., the path  $\mathbf{P}_{-v_k}(v_i, v_0, d)$  does exist. Otherwise, node  $v_k$  can charge a monopoly price since it is a critical node to connect  $v_i$  and  $v_0$ .

Designing a truthful payment scheme for unicast is straightforward. One of our main contributions in this paper is a time optimal centralized method to compute the payment, which will be described in detail in the following subsection.

### B. Fast Payment Computing

Assume that the access point has collected all nodes’ costs, and the network structure  $G$ . Then this access point can com-

pute the payment to all relay nodes in a centralized manner. The very naive way to calculate the payment for all nodes on  $P(v_i, v_0, d)$  is to calculate every node's payment using Dijkstra's algorithm. In the worst case there will be  $O(n)$  nodes on  $P(v_i, v_0, d)$ , so this naive algorithm will result in a time complexity  $O(n^2 \log n + nm)$ . In [20], Hershberger and Suri provided a fast payment calculation algorithm for *edge weighted* graph (by assuming the edges are rational agents). Nardelli, Proietti and Widmayer [21] studied a similar question of finding the most vital node of a shortest path in an edge weighted graph, and gave a method to do so in time  $O(m + n \log n)$ . Borrowing some ideas from [20], we present an  $O(n \log n + m)$  time complexity algorithm for fast payment calculation in a *node weighted* graph.

Consider a node  $v_k \in P(v_i, v_0, d)$  and we want to compute  $\|P_{-v_k}(v_i, v_0, d)\|$ . The basic idea of our algorithm is for a pair of nodes  $v_a, v_b$  such that  $v_a v_b \in G$ , we calculate the path  $P_{-v_k}(v_i, v_a, d)$  and  $P_{-v_k}(v_b, v_0, d)$  separately. Then by concatenating  $P_{-v_k}(v_i, v_a, d)$ , link  $v_a v_b$  and  $P_{-v_k}(v_b, v_0, d)$  we obtain the path with the minimum cost from  $v_i$  to  $v_0$  without node  $v_k$  and having  $v_a v_b$  on it. Choosing the minimal cost path for all edges  $v_a v_b \in G$ , we find  $\|P_{-v_k}(v_i, v_0, d)\|$ . See Figure 1 for an illustration.

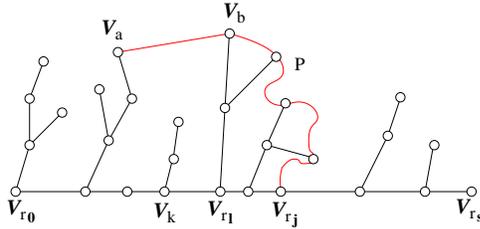


Fig. 1. Computing  $v_k$ -avoiding shortest path.

Algorithm 1 computes the payment for all nodes on the least cost path. The correctness of this algorithm comes from the following observations of the shortest  $v_k$ -avoiding path.

*Lemma 1:* Assume that, for a node  $v_{r_l} \in P(v_i, v_0, d)$ ,  $P_{-v_{r_l}}(v_i, v_0, d) = v_{l_0} v_{l_1} \cdots v_{l_{t-1}} v_{l_t}$ , where  $v_{l_0} = v_i$  and  $v_{l_t} = v_0$ . If  $v_{l_a}.level \geq l$  then  $v_{l_b}.level \geq l$  for all  $b > a$ .

*Proof.* We prove it by contradiction. Assume that there exists a pair of nodes  $a$  and  $b$  such that  $v_{l_b}.level < l$ ,  $v_{l_a}.level \geq l$  and  $b > a$ . Notice that  $P(v_i, v_{l_b}, d)$  doesn't contain node  $v_{r_l}$  since  $v_{l_b}.level < l$ . Thus, replacing path  $v_{l_0} v_{l_1} \cdots v_{l_{t-1}} v_{l_t}$  by path  $P(v_i, v_{l_b}, d)$  concatenated with  $v_{l_{b+1}} \cdots v_{l_{t-1}} v_{l_t}$  will result in a  $v_{r_l}$ -avoiding path with a smaller weight since the path  $P(v_i, v_{l_b}, d) \subset SPT(v_i)$  will only use nodes with level at most  $v_{l_b}.level < l$ , while the subpath  $v_{l_0} v_{l_1} \cdots v_{l_{b-1}} v_{l_b}$  uses node  $v_{l_a}$  with level at least  $l$ . Notice that  $P(v_i, v_{l_b}, d)$  is the least cost path connecting  $v_i$  and  $v_{l_b}$ . This finishes our proof.  $\square$

*Lemma 2:* For a node  $v_k$  such that  $v_k.level = l$ , then  $P(v_k, v_0, d)$  cannot contain any relay node  $v_{r_a}$  with  $a < l$ .

*Proof.* Again, we prove it by contradiction. Assume there exists a node  $v_k$  such that  $v_k.level = l$ , and  $P(v_k, v_0, d)$  contains a node  $v_{r_a}$  with  $a < l$ . Obviously,  $P(v_i, v_k, d)$  contains path  $v_{r_0} v_{r_1} \cdots v_{r_{l-1}} v_{r_l}$ . For simplicity, we assume that path

### Algorithm 1 Fast VCG Payment Computing

- 1: Find the Shortest Path Tree (SPT)  $SPT(v_i)$  and  $SPT(v_j)$  rooted at  $v_i$  and  $v_0$  respectively. Assume that  $P(v_i, v_0, d) = v_{r_0} v_{r_1} \cdots v_{r_{s-1}} v_{r_s}$ , where  $v_{r_0} = v_i$  and  $v_{r_s} = v_0$ . For a node  $v_k \in P(v_i, v_0, d)$ , let  $L(v_k)$  be the cost of LCP from  $v_i$  to  $v_k$  and  $R(v_k)$  be the cost of LCP from  $v_k$  to  $v_0$ .
- 2: **for** every node  $v_k$  in  $G$  **do**
- 3: Find a node  $v_{r_l} \in P(v_i, v_0, d)$  such that removing node  $v_{r_l}$  causes that node  $v_k$  neither connects to  $v_i$  nor connects to  $v_j$  in the tree  $SPT(v_i)$ .
- 4: Set  $v_k.level = l$ .
- 5: Mark all nodes WHITE.
- 6: Mark all nodes whose level is  $s$  BLACK.
- 7: **for**  $l = s - 1$  down to 1 **do**
- 8: **repeat**
- 9: For each WHITE node  $v_k$  whose level is  $l$ , find its BLACK neighbor  $v_a$  such that  $d_a + \|P_{-v_{r_l}}(v_a, v_0, d)\|$  is minimized.
- 10: Set  $R^{-l}(v_k) = c_a + \|P_{-v_{r_l}}(v_a, v_0, d)\|$ ;
- 11: For all nodes  $v_k$  with level  $l$ , find the node with minimum  $R^{-l}(v_k)$ . Let this node be  $v_j$ . Mark  $v_j$  BLACK.
- 12: **until** all nodes with level  $l$  except  $v_{r_l}$  are BLACK.
- 13: Mark node  $v_{r_l}$  as BLACK.
- 14: For node  $v_k$  with level  $l$ , find a neighbor  $v_s$  whose  $c^{-l}(v_k) = L(v_s) + R^{-l}(v_k) + d_s + d_k$  is minimized among all neighbors with level smaller than  $l$ . Denote such minimum value as  $c^{-l}(v_k)$ .
- 15: Among all nodes with level  $l$ , choose the node  $v_k$  whose  $c^{-l}(v_k)$  is minimal and set  $c^{-l}$  to this value, i.e.,  $c^{-l} = \min_{v_k.level=l} c^{-l}(v_k)$ .
- 16: For each node  $v_{r_l} \in P(v_i, v_0, d)$ , where  $l$  starts from  $s - 1$  to 1, we use a heap  $H$  to find the path containing an edge  $v_a v_b$  with minimum cost such that  $v_a.level < l < v_b.level$ . The heap  $H$  has nodes  $\overline{v_a v_b}$  corresponding to all such edges  $v_a v_b \in G$ . The value of a node  $\overline{v_a v_b}$  is  $L(v_a) + R(v_b) + d_a + d_b$ . Find the node with the minimal value in  $H$  and compare this value with  $c^{-l}$ , and the minimal of these two values is set as  $\|P_{-v_{r_l}}(v_i, v_0, d)\|$ .
- 17: Calculate the payment to node  $v_{r_l}$  as follows

$$p_i^{r_l} = \|P_{-v_{r_l}}(v_i, v_0, d)\| - \|P(v_i, v_0, d)\| + d_{r_l}$$

$P(v_i, v_k, d)$  is composed of two subpaths  $v_{r_0} v_{r_1} \cdots v_{r_{l-1}} v_{r_l}$  and  $P_1$  as shown in Figure 2.

Similarly, we assume that  $P(v_k, v_0, d)$  is composed of two subpaths  $P_2$  and  $v_{r_a} v_{r_{a+1}} \cdots v_{r_{s-1}} v_{r_s}$ . Let  $P_3$  be the subpath  $v_{r_a} v_{r_{a+1}} \cdots v_{r_{l-1}} v_{r_l}$ . It is easy to show that  $\|P_2\| + \|P_3\| \leq \|P_1\|$  from the property of least cost path  $P(v_i, v_k, d)$ , and  $\|P_1\| + \|P_3\| \leq \|P_2\|$  from the property of least cost path  $P(v_k, v_0, d)$ . Consequently, we have  $\|P_3\| \leq 0$ , which is a contradiction. This finishes our proof.  $\square$

Similarly, we have

*Lemma 3:* Consider the path  $P_{-v_{r_l}}(v_k, v_0, d)$ . If there exists a node  $v_{k'}$  on this path with  $v_{k'}.level < v_k.level$ , then node  $v_k$  cannot appear on  $P_{-v_{r_l}}(v_i, v_0, d)$ .

The proof of this lemma is omitted due to space limit.

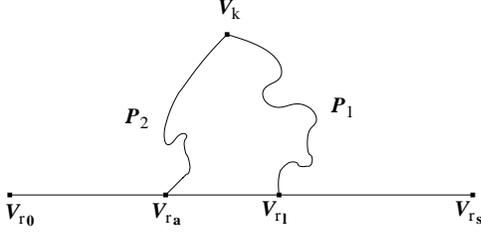


Fig. 2. Observation of  $v_{r_l}$ -avoiding shortest path.

Lemma 3 allows us to only focus on the path connecting  $v_k$  and  $v_0$  that avoids nodes  $v_{k'}$  with  $v_{k'}.level < v_k.level$ .

Now we analyze the time complexity of this algorithm. First, the shortest path tree can be calculated in  $O(n \log n + m)$ , and with the SPT it only takes us linear time to find  $L(v_k)$  and  $R(v_k)$ . Clearly, we can compute the level for all nodes in time  $O(n)$ . We can calculate the value  $R^{-l}(v_k)$  for all nodes  $v_k$  in time  $O(n \log n + m)$ . Assume the number of nodes with label  $l$  is  $n_l$  and node  $v_k$ 's degree is  $deg(v_k)$ , then it will take at most  $\sum_{v_k.level=l} deg(v_k) + n_l \log n_l$  time to find  $c^{-l}$ . Summing  $l$  from 1 to  $s-1$ , the time complexity as  $\sum deg(v_k) + \sum_{l=1}^{s-1} n_l \log n_l < 2m + n \log n$ . Thus, computing  $R^{-l}(v_k)$  and  $c^{-l}$  will take time complexity of  $O(n \log n + m)$ . The heap operations have at most  $m$  insertions,  $m$  deletions, and  $n$  extract-min operations, which takes time  $O(m + n \log n)$ . Overall, the time complexity is still  $O(m + n \log n)$ .

### C. Distributed Algorithm for Payment Calculation

Unlike wired or cellular networks, wireless ad hoc networks are lack of a centralized authority. Thus, it is more desirable to compute the payment in a distributed manner. Next, we discuss how to compute the payment of a node to all the relay nodes truthfully in a distributed manner. Assume that there is a fixed destination node  $v_0$ . Our distributed algorithm will compute the payment of each node  $v_i$  to all its relay nodes. The distributed algorithm has two stages. Firstly, all nodes together find the Shortest Path Tree (SPT) rooted at node  $v_0$ . We assume that the SPT tree does not have a loop. This step can be easily implemented using Dijkstra's algorithm, so we omit this one. Assume that we already formed a shortest path tree  $T$  rooted at node  $v_0$ , and every node knows its parent and children in tree  $T$ . Secondly, every node  $v_i$  computes its payment  $p_i^k$  in a distributed manner using Algorithm 2, which is based on the algorithm presented in [22].

Whenever some entry  $p_i^k$  stored at node  $v_i$  changes, the entry  $p_i^k$  is sent to all neighbors of  $v_i$  by node  $v_i$ . When the network is static, the price entries decrease monotonically and converge to stable values after a finite number of rounds (at most  $n$  rounds).

### D. Compute the Payment Truthfully

While it is quite obvious to conceive that the node  $v_i$  has the incentive not to correctly calculate his payment  $p_i^k$  in the second stage, it is not so straightforward to notice that the node  $v_i$  also has the incentive to lie about his shortest path even in the first stage. We give an example to show that even we can guarantee that node  $v_i$  calculates his payment truthfully in the

---

### Algorithm 2 Distributed payment computing by a node $v_i$

---

- 1: Set  $p_i^k \leftarrow \infty$ , if  $v_k \in \mathbf{P}(v_i, v_0, d)$ ; otherwise,  $p_i^k \leftarrow 0$ .
  - 2: Broadcasts its entries  $p_i^k$  to its neighbors.
  - 3: **while**  $v_i$  receives an updated price from a neighbor  $v_j$  **do**
  - 4:   **if**  $v_j$  is the parent of  $v_i$  **then**
  - 5:      $p_i^k \leftarrow \min(p_i^k, p_j^k)$  if  $v_k \in \mathbf{P}(v_i, v_0, d)$ .
  - 6:   **else if**  $v_i$  is the parent of  $v_j$  **then**
  - 7:      $p_i^k \leftarrow \min(p_i^k, p_j^k + d_i + d_j)$  if  $v_k \in \mathbf{P}(v_i, v_0, d)$ .
  - 8:   **else**
  - 9:     for every  $v_k \in \mathbf{P}(v_i, v_0, d)$ ,  $v_i$  updates  $p_i^k$  as follows.
  - 10:     **if**  $v_k \in \mathbf{P}(v_j, v_0, d)$  **then**
  - 11:        $p_i^k \leftarrow \min(p_i^k, p_j^k + d_j + \|\mathbf{P}(v_j, v_0, d)\| - \|\mathbf{P}(v_i, v_0, d)\|)$ ;
  - 12:     **else**
  - 13:        $p_i^k \leftarrow \min(p_i^k, d_k + d_j + \|\mathbf{P}(v_j, v_0, d)\| - \|\mathbf{P}(v_i, v_0, d)\|)$
  - 14:     Broadcasts its entries  $p_i^k$  to its neighbors.
- 

second stage, it is not unnecessary for us to worry about nodes' lying in the first stage. In Figure 3, the shortest path between  $v_0$  and  $v_1$  should be  $v_1 v_4 v_3 v_2 v_0$ . It is easy to calculate that  $v_1$ 's payments to nodes  $v_2$ ,  $v_3$  and  $v_4$  are all exactly 2. Then the overall payment by node  $v_1$  is 6. If node  $v_1$  lies that it is not a neighbor of  $v_4$ , then its shortest path becomes  $v_1 v_5 v_0$ . Now it only needs to pay  $v_5$  5 to send a packet. Thus, node  $v_1$  benefits by lying about its neighborhood connection information, which consequently changes the SPT. This problem rises from the fact that the least cost path is not necessarily the path that you pay the least. This observation also raises the concern whether a distributed method presented in [22] does compute the payment correctly since a node can lie not only about its cost, but also about its neighborhood information.

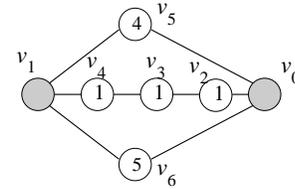


Fig. 3. The node has the incentive to lie about his shortest path

Notice that the distributed method to compute the payment relies on the selfish node  $v_i$  to calculate the payment  $p_i^k$  to node  $v_k$ , which cannot prevent node  $v_i$  from manipulating the calculation in its favor. In [22], the authors suggested to use the following approach: all agents are required to sign all messages that they sent and to verify all messages that they received from their neighbors. They claimed that their protocol can be modified so that all forms of cheating by agents are detectable. Observe that, they did not consider the possible scenario when an agent could lie about the topology (the scenario we just discussed in previous paragraph). Notice that even using their approach, all nodes must keep a record of messages sent to and received from its neighbors so that an audit can be performed later if a disagreement happens.

Next, we present a new distributed method that prevents nodes from lying about the topology, and mis-calculating the

payment, and it does not need store all messages. The nodes use Algorithm 3 to compute the shortest distance to source and then use Algorithm 4 to compute the payment to the relay nodes. It is easy to verify that they are truthful and no node will lie about its neighbor information and will follow the payment calculation procedure. For the example illustrated in Figure 3, node  $v_1$  has to use the shortest path  $v_1v_4v_3v_2v_0$  to compute the payment according to our new protocol since node  $v_4$  knows the existence of this shortest path and it will detect the lie by node  $v_1$  if node  $v_1$  chose to use path  $v_1v_5v_0$  instead. The problem remaining is how to make it more efficient.

---

**Algorithm 3** Truthful Distributed SPT Construction by  $v_i$

---

- 1: Every node  $v_i$  has two variables:  $D(v_i)$  stores the shortest distance to  $v_0$  and  $FH(v_i)$  stores its parent on SPT. Initially, if  $v_0$  is  $v_i$ 's neighbor then set  $D(v_i)$  to 0 and  $FH(v_i)$  to  $v_0$ ; else set  $D(v_i)$  to  $\infty$  and  $FH(v_i)$  to NULL. Node  $v_i$  broadcasts its information to its neighbors.
  - 2: **while**  $v_i$  received information from its neighbor  $v_j$  **do**
  - 3:   **if**  $D(v_i) > D(v_j) + c_j$  **then**
  - 4:      $D(v_i) \leftarrow D(v_j) + c_j$ , and  $FH(v_i) \leftarrow v_j$ ;
  - 5:     **if**  $v_i \neq FH(v_j)$  and  $D(v_i) + c_i < D(v_j)$ ; or  $v_i = FH(v_j)$  and  $D(v_i) + c_i \neq D(v_j)$  **then**
  - 6:       node  $v_i$  contacts  $v_j$  directly using reliable and secure connection, asking  $v_j$  to update his  $D(v_j)$  to  $D(v_i) + c_i$  and  $FH(v_j)$  to  $v_i$ . After the necessary updating,  $v_j$  must broadcast his information.
  - 7:   Node  $v_i$  broadcasts its information to its neighbors.
- 

---

**Algorithm 4** Truthful Distributed Payment Computation by  $v_i$

---

- 1: Set  $p_i^k \leftarrow \infty$ , if  $v_k \in \mathbf{P}(v_i, v_0, d)$ ; otherwise,  $p_i^k \leftarrow 0$ .
  - 2: Broadcasts its entries  $p_i^k$  to its neighbors.
  - 3: **while**  $v_i$  received information  $p_j^k$  from its neighbor  $v_j$  **do**
  - 4:    $v_i$  updates  $p_i^k$  using Algorithm 2 (steps 4-15).
  - 5:   when  $p_i^k$  changes,  $v_i$  broadcasts the value of  $p_i^k$ , and the ID of the node  $p_j$  that triggered this change.
  - 6:   if  $v_i$  triggered the change for this  $p_j^k$  from  $v_j$ ,  $v_i$  recalculates  $p_j^k$  for  $v_j$  using Algorithm 2 (steps 4-15) to verify it. If his answer and the payment sent from  $v_j$  do not match, node  $v_i$  then notifies  $v_j$  and other nodes. Node  $v_j$  will then be punished accordingly, e.g.,  $v_j$  is dropped from the network by all nodes.
- 

## IV. OTHER ISSUES OF PRICING MECHANISM

### A. Collusion of Nodes

So far we have assumed that the wireless nodes do *not* collude to improve their utilities. In practice, the nodes may collude with each other in the hope to gain as a group. There are several possible ways such that some nodes can collude. For example, if two nodes  $v_{k_1}$  and  $v_{k_2}$  know that the removal of them will disconnect some nodes from the access point, then these two nodes can collude to declare arbitrarily large costs and charge a monopoly price together. Notice that, by declaring much higher costs together, one node's utility may decrease,

but the sum of their utilities is guaranteed to increase (thus, they share the increased utilities).

We point out here that the collusion of nodes discussed here is different from the traditional *group strategyproof* concept studied in [23], [24]. A pricing mechanism is said to be group strategyproof in [23], [24] if any subset of agents colludes, then each agent of this subset cannot improve its utility without decreasing the utility of some other agent. Clearly, this formulation of group strategyproofness cannot capture the scenario when the profit can be transferred among all colluding nodes, which happens very often in real world. We then formally define what is *k-agents strategyproof* mechanism as follows.

*Definition 1:* A mechanism is said to be *k-agents strategyproof* if, when any subset of agents of size  $k$  colludes, the overall utility of this subset is made worse off by misreporting their types. A mechanism is *true group strategyproof* if it is *k-agents strategyproof* for any  $k$ .

Clearly, we cannot design a *true group strategyproof* mechanism for the unicast routing problem studied here: if all nodes but node  $v_i$  collude and declare arbitrarily high costs, then node  $v_i$  has to pay a payment arbitrarily higher than the actual payment it needs to pay if these nodes do not collude.

Directly from the incentive compatibility property, for any truthful mechanism, we have:

*Lemma 4:* Assume node  $v_i$ 's valuation is of the form  $w^i(o, c_i)$ . For any strategyproof mechanism, if the output  $o$  keeps unchanged, then the *payment* and *utility* of node  $v_i$  do not depend on  $c_i$ .

Furthermore, for any 2-agents strategyproof mechanism, we have a stronger conclusion:

*Lemma 5:* Suppose every node  $v_i$ 's valuation is of the form  $w^i(o, c_i)$ . For any 2-agents strategyproof mechanism, as long as the output  $o$  doesn't change, node  $v_i$ 's *payment* and *utility* do not depend on the profile  $c$ .

*Proof.* From lemma 4, we know  $w^i(o, c_i)$  does not depend on  $c_i$  since we assume the output  $o$  is not changed when node  $v_i$  declares  $d_i$  instead of  $c_i$ . Thus, we just need to prove that its utility doesn't depend on any other nodes' declared cost. We prove it by contradiction. Assume its utility  $u^i(d)$  depends on a node  $v_k$ 's declared cost  $d_k$ , then there exist two numbers  $d_{k_1} \neq d_{k_2}$  such that  $u^i(d|^{k_1}d_{k_1}) \neq u^i(d|^{k_2}d_{k_2})$ . Without loss of generality we assume that  $u^i(d|^{k_1}d_{k_1}) > u^i(d|^{k_2}d_{k_2})$ . From lemma 4, we have  $u^k(d|^{k_1}d_{k_1}) = u^k(d|^{k_2}d_{k_2})$  since output  $o$  is not changed. Consider the case with original profile  $c = d|^{k_2}d_{k_2}$ . Node  $v_i$  can ask  $v_k$  to lie its cost to  $d_{k_1}$ , thus increase  $v_i$ 's utility while keeping  $v_k$ 's utility unchanged, which violates the incentive compatibility of 2-agents strategyproof mechanism.  $\square$

In the following discussions, we restrict our attention to the unicast scenario. Remember that  $x_k$  denotes whether a node  $v_k$  is on the least cost path or not. Let  $D_1^k$  be the set of profiles such that  $x_k = 1$ , i.e., node  $v_k$  is on the LCP;  $D_0^k$  be the set of profiles such that  $x_k = 0$ , i.e., node  $v_k$  is not on the LCP. Clearly,  $D_1^k \cup D_0^k$  comprises all possible profiles. From lemma 5, we have the following:

*Lemma 6:* Assume that  $\mathcal{A}$  is a 2-agents strategyproof mechanism for unicast and its output  $\mathcal{O}$  is the LCP connecting the

source and target. For any node  $v_k$ , if  $x_k$  doesn't change, then  $p^k$  calculated by  $\mathcal{A}$  is independent of  $d$ .

**Proof.** We prove it by contradiction. Suppose node  $v_k$ 's payment depends on  $d$ , then there exist two profiles  $(d^i d_{i_1})$  and  $(d^i d_{i_2})$  such that  $d_{i_1} > d_{i_2}$ ,  $p^k(d^i d_{i_1}) \neq p^k(d^i d_{i_2})$ , and  $x_k(d^i d_{i_1}) = x_k(d^i d_{i_2})$ . There are two cases here.

Case 1:  $x_i(d^i d_{i_1}) = x_i(d^i d_{i_2})$ . Clearly, the LCP path remains the same when node  $v_i$  declares cost  $d_{i_1}$  or  $d_{i_2}$ . From Lemma 5, we have  $p^k(d^i d_{i_1}) = p^k(d^i d_{i_2})$ , which is a contradiction. Thus, this case is impossible.

Case 2:  $x_i(d^i d_{i_1}) \neq x_i(d^i d_{i_2})$ . Since  $d_{i_1} > d_{i_2}$ , this case means that when  $d_i = d_{i_1}$ ,  $v_i$  is not on the LCP, and when  $d_i = d_{i_2}$ ,  $v_i$  is on the LCP. Now fixing  $d^{-i}$  and increasing node  $v_i$ 's declared cost from  $d_{i_2}$  to  $d_{i_1}$ , there must exist  $a_i \in [d_{i_2}, d_{i_1}]$  such that  $v_i$  is on LCP if  $d_i < a_i$ ,  $v_i$  is not on LCP if  $d_i > a_i$ , and it is unknown when  $d_i = a_i$ . For node  $v_i$ , its utility and payment do not depend on its own declared cost  $d_i$ . From Lemma 5, its payment is a constant  $\underline{P}_i$  when  $d_i < a_i$  (since the output remains the same for every  $d_i < a_i$ ) and another constant  $\overline{P}_i$  when  $d_i > a_i$ . From the incentive compatibility of node  $v_i$ , we have  $\underline{P}_i - d_i \geq \overline{P}_i$ , for any  $d_i \leq a_i$ , since we have to prevent node  $v_i$  from lying its cost from  $d_i$  to a number larger than  $a_i$ . Similarly, to prevent  $v_i$  from lying down its cost from a number larger than  $a_i$  to a  $d_i \leq a_i$ , we need  $\underline{P}_i - d_i \leq \overline{P}_i$ , for any  $d_i \leq a_i$ . Thus, we have  $\underline{P}_i - \overline{P}_i = a_i$ .

Suppose  $p^k(d^i d_{i_1}) = p^k(d^i d_{i_2}) + \delta$ . We first consider the case  $\delta < 0$ . Considering the graph with profile  $c = (d^i d_{i_1})$ , clearly node  $v_i$  is not on the LCP and its utility is  $\overline{P}_i$ . Thus, the sum of node  $v_k$  and  $v_i$ 's utility, when node  $v_i$  declares cost  $d_{i_1}$ , is  $u^k(d^i d_{i_1}) + u^i(d^i d_{i_1}) = p^k(d^i d_{i_1}) + \overline{P}_i - x_k \cdot c_k$ , where  $x_k = x_k(d^i d_{i_1}) = x_k(d^i d_{i_2})$ . Now consider the scenario when  $v_i$  declares its cost as  $d_{i_2} \leq a_i$ . The sum of node  $v_k$  and  $v_i$ 's utility becomes (since  $x_k$  remains the same)

$$\begin{aligned} & u^k(d^i d_{i_2}) + u^i(d^i d_{i_2}) \\ &= p^k(d^i d_{i_2}) + \underline{P}_i - x_k c_k - d_{i_2} \geq p^k(d^i d_{i_2}) + \overline{P}_i - x_k c_k \\ &= p^k(d^i d_{i_1}) + \overline{P}_i - x_k c_k - \delta > u^k(d^i c_i) + u^i(d^i c_i) \end{aligned}$$

This implies that  $v_i$  and  $v_k$  can benefit together by asking  $v_i$  to lie its cost from  $d_{i_1}$  to  $d_{i_2}$ .

We then consider the case  $\delta > 0$ . Consider the graph with profile  $c = (d^i c_i)$ , where  $c_i = a_i - \epsilon$ , and  $0 \leq \epsilon \leq \min\{\frac{\delta}{2}, a_i - d_{i_2}\}$ . Clearly node  $v_i$  is on the LCP and its utility is  $\underline{P}_i - c_i$ . Thus, the sum of node  $v_k$  and  $v_i$ 's utility, when  $d_i = c_i$ , is

$$\begin{aligned} & u^k(d^i c_i) + u^i(d^i c_i) = p^k(d^i c_i) + \underline{P}_i - x'_k c_k - c_i \\ &= p^k(d^i d_{i_2}) + \overline{P}_i - x'_k c_k + \epsilon, \end{aligned}$$

where  $x'_k = x_k(d^i c_i)$ . Now consider the scenario when  $v_i$  declares its cost as  $d_{i_1}$ . Notice that  $d_{i_1} \geq a_i$ . The sum of node  $v_k$  and  $v_i$ 's utility becomes

$$\begin{aligned} & u^k(d^i d_{i_1}) + u^i(d^i d_{i_1}) \\ &= p^k(d^i d_{i_1}) + \overline{P}_i - x_k c_k = p^k(d^i d_{i_2}) + \overline{P}_i - x_k c_k + \delta \\ &> p^k(d^i d_{i_2}) + \overline{P}_i - x'_k c_k + \epsilon = u^k(d^i c_i) + u^i(d^i c_i) \end{aligned}$$

The last inequality comes from  $x_k(d^i d_{i_1}) = x_k \leq x'_k = x_k(d^i c_i)$  (proof follows) and  $\delta > \epsilon$ . This implies that  $v_i$  and

$v_k$  can benefit together by asking  $v_i$  to lie its cost from  $a_i - \frac{\delta}{2}$  to  $d_{i_1}$ .

At last, we prove that when node  $v_i$  declares a cost  $b_i \in [d_{i_2}, a_i]$  while  $d^{-i}$  is fixed,  $x_k(d^i b_i) \geq x_k(d^i d_{i_1})$ . We only have to prove for the case  $x_k(d^i d_{i_1}) = 1$  and we prove it by contradiction. Assume that there is a  $b_i \in [d_{i_2}, a_i]$  such that  $x_k(d^i b_i) = 0$ . Let  $\Pi(d)$  be the total cost of a path  $\Pi$  under cost profile  $d$ . Let  $\Pi_1$  be the least cost path connecting the source and target using profile  $d^i b_i$ . Observe that  $v_i \in \Pi_1$ . By assumption,  $v_k \notin \Pi_1$ . Let  $\Pi_2$  be the least cost path connecting the source and target using profile  $d^i d_{i_2}$ . Remember that  $v_i \in \Pi_2$  and  $v_k \in \Pi_2$ . Thus,  $\Pi_1$  and  $\Pi_2$  are different paths. From the optimality of  $\Pi_1$  under cost profile  $d^i b_i$ , we have  $\Pi_1(d^i b_i) < \Pi_2(d^i b_i)$ . From the optimality of  $\Pi_2$  under cost profile  $d^i d_{i_2}$ , we have  $\Pi_1(d^i d_{i_2}) > \Pi_2(d^i d_{i_2})$ . On the other hand,

$$\begin{aligned} & \Pi_1(d^i b_i) = \Pi_1(d^i d_{i_2}) + b_i - d_{i_2} \\ &> \Pi_2(d^i d_{i_2}) + b_i - d_{i_2} = \Pi_2(d^i b_i) > \Pi_1(d^i b_i), \end{aligned}$$

which is a contradiction. This finishes all our proof.  $\square$

The above lemma implies that, for any 2-agents strategyproof mechanism for unicast, the payment to any node  $v_k$ , regardless of the cost profile, is a constant as long as  $v_k$  is on the LCP; the payment to any node  $v_k$ , regardless of the cost profile, is another constant as long as  $v_k$  is not on the LCP.

**Theorem 7:** There is no 2-agents strategyproof mechanism for unicast problem if the output is the LCP.

**Proof.** We prove it by contradiction. Assume  $\mathcal{A}$  is a 2-agents strategyproof mechanism. From Lemma 6, we know that if node  $v_k$  is on LCP then its payment is  $\underline{P}$ , else its payment is  $\overline{P}$ . Now consider a profile  $d$  of their declared costs, and  $v_k$  is on LCP. Given a fixed  $d^{-k}$ , there exists  $a_k > 0$  such that  $v_k$  is on LCP if and only if  $d_k \leq a_k$ . It is easy to see that  $a_k = \mathbf{P}_{-v_k}(v_i, v_0, d) - \mathbf{P}(v_i, v_0, d^k 0)$  and  $\overline{P} - \underline{P} = a_k$  (otherwise node  $v_k$  can lie about its cost to improve its utility). In other words,  $\overline{P} - \underline{P} = \|\mathbf{P}_{-v_k}(v_i, v_0, d) - \mathbf{P}(v_i, v_0, d^k 0)\|$  depends on  $d$ , which is a contradiction to the requirement that both  $\overline{P}$  and  $\underline{P}$  are fixed constants. This finishes our proof.  $\square$

Theorem 7 relieves us from designing any  $k$ -agents strategyproof when the objective is to use the least cost path for routing. In the following discussions, we study how to design a truthful mechanism such that it can prevent nodes from colluding with its one-hop neighbors. Notice that the VCG payment scheme discussed in subsection III-A does not prevent a node from colluding with its neighbors at all. It is not difficult to construct an example such that, for a node  $v_k \in \mathbf{P}(v_i, v_0, d)$ , the path  $\mathbf{P}_{-v_k}(v_i, v_0, d)$  uses a node  $v_t$  that is a neighbor of  $v_k$  and  $v_t \notin \mathbf{P}(v_i, v_0, d)$ . Then  $v_t$  can lie its cost up to increase the utility of node  $v_k$ .

Assume that node  $v_i$  pays other nodes to relay the data to another node  $v_j$ . Let  $N(v_k)$  be the set of neighbors of node  $v_k$ , including node  $v_k$  itself. Thus, to have a payment scheme that prevents collusion between any two neighboring nodes, it is necessary that the graph resulted by removing  $N(v_k)$  still has a path connecting  $v_i$  and  $v_j$ . Therefore, we assume that graph  $G \setminus N(v_k)$  is connected for any node  $v_k$ . Similar to the payment scheme presented in subsection III-A when nodes do

not collude, we design the following payment scheme  $\tilde{p}$  that avoids the collusion between any two neighboring nodes. The payment  $\tilde{p}_i^k(d)$  to a node  $v_k$  is

$$\tilde{p}_i^k(d) = \|\mathbf{P}_{-N(v_k)}(v_i, v_0, d)\| - \|\mathbf{P}(v_i, v_0, d)\| + x_k(i) \cdot d_k,$$

where  $\mathbf{P}_{-N(v_k)}(v_i, v_0, d)$  is the least cost path connecting  $v_i$  and  $v_j$  in graph  $G \setminus N(v_k)$  without using any node in  $N(v_k)$ . Notice that the payment to a node  $v_k \notin \mathbf{P}(v_i, v_0, d)$  could be positive when node  $v_k$  has a neighbor on  $\mathbf{P}(v_i, v_0, d)$ . This is a sharp difference to our first payment scheme based on VCG.

We then prove that the payment scheme  $\tilde{p}$  is indeed truthful.

*Theorem 8:* The payment scheme  $\tilde{p}$  is a strategyproof mechanism that prevents any two neighboring nodes from colluding. Proof. Clearly, each individual node will be truthful since our mechanism belongs to the family of VCG mechanisms. Notice for any two neighboring nodes  $v_k$  and  $v_l$ , their utilities can be written as

$$\begin{cases} u^k(c) &= \sum_{t=0}^{n-1} v^t(o(c), c_t) + h^{-k}(c^{-N(v_k)}) \\ u^l(c) &= \sum_{t=0}^{n-1} v^t(o(c), c_t) + h^{-l}(c^{-N(v_l)}) \end{cases}$$

Summarizing them, we get  $u^l(c) + u^k(c) = 2 \sum_{t=0}^{n-1} v^t(o(c), c_t) + h^{-k}(c^{-N(v_k)}) + h^{-l}(c^{-N(v_l)})$ . Notice that  $h^{-k}(c^{-N(v_k)}) + h^{-l}(c^{-N(v_l)})$  doesn't depend on  $d_l$  and  $d_k$  since  $v_k$  and  $v_l$  are neighbors of each other. In addition,  $\sum v^i(o(c), c_i)$  is maximized when they reveal their true costs. Thus,  $v_k$  and  $v_l$  will maximize their total utilities by revealing their true costs.  $\square$

It is easy to show that the above payment scheme is optimum in terms of the payment to each individual node when the output is the least cost path. The proof is omitted due to space limit.

Furthermore, we can extend the above scheme to a more general case when we want to prevent some groups of nodes from colluding. Let  $\{Q(v_1), Q(v_2), \dots, Q(v_n)\}$  be a set of subsets of nodes, i.e.,  $Q(v_k) \subset V$ . We then show how to design a truthful mechanism such that any node  $v_k$  can not collude with other nodes in  $Q(v_k)$  to increase their total utilities. For simplicity, assume that  $v_k \in Q(v_k)$ , for  $1 \leq k \leq n$ . It is easy to show that the following mechanism is truthful: 1) the output is the least cost path connecting the source  $v_i$  and destination  $v_0$ ; 2) the payment  $\tilde{p}_i^k(d)$  to a node  $v_k$  is  $\tilde{p}_i^k(d) = \|\mathbf{P}_{-Q(v_k)}(v_i, v_0, d)\| - \|\mathbf{P}(v_i, v_0, d)\| + x_k(i) \cdot d_k$ . Obviously, we need graph  $G \setminus Q(v_k)$  to be connected for any node  $v_k$ .

### B. Link Cost Instead of Node Cost

So far, we assumed that the cost of a node forwarding data to any neighbor is same, i.e., the cost is incurred on each node. However, each node could have different costs of forwarding data to different neighbors by using power adjustment technique. Thus, we assume that each wireless node  $v_i$  has a private type  $c_i = (c_{i,0}, c_{i,1}, \dots, c_{i,n-1})$ . Here  $c_{i,j}$  is its power cost to support the link to a node  $v_j$ . If node  $v_i$  cannot reach node  $v_j$ , then the power cost is assumed to be  $\infty$ . Obviously,  $c_{i,i} = 0$ .

Notice that this model of network is different from the network models used by previous strategyproof pricing mechanisms [4], [22] for unicast. In their models, either a link is an agent, which has a computational power and private cost type,

or a node is an agent, which has computational power and the private scalar cost to carry the transit traffic. In our new model here, we treat each node as an agent and it has some private type which is a vector. The valuation of a node is solely determined by which incident link is used in the optimal solution. Specifically, given an output (a path  $v_{i_s}, v_{i_{s-1}}, \dots, v_{i_1}, v_{i_0}$  connecting node  $v_i$  to  $v_0$ , where  $v_i = v_{i_s}$  and  $v_0 = v_{i_0}$ ), the valuation  $w^{i_k}(c^{i_k}, o)$  of the node  $v_{i_k}$  is  $-c_{i_k, i_{k-1}}$ . Given the declared types by all wireless nodes, the mechanism will compute a path that maximizes the valuation of all nodes, and a pricing scheme  $p$  that is strategyproof.

The strategyproof pricing mechanism works as follows. First, each node  $v_i$  declares its cost vector  $d_i$ , which is a  $n$ -ary vector itself. We then define a directed and weighted graph  $G = (Q, E, W)$ , where the weight of a directed link  $v_i v_j$  is  $d_{i,j}$ . A least cost directed path  $\mathbf{P}(v_i, v_0, d)$  is computed to connect  $v_i$  to  $v_0$ , which is the output. Let  $x_{k,j}(d, i, 0)$  be the indicator of whether a directed link  $v_k v_j$  is on the directed path from  $v_i$  to  $v_0$ . The payment  $p_i^k(d)$  to node  $v_k$  is  $\sum_j x_{k,j}(d, i, 0) d_{k,j} + \Delta_{i,k}$ . Here  $\Delta_{i,k}$  is the improvement of the least cost path from  $v_i$  to the access point due to the existence of node  $v_k$ . In other words,  $\Delta_{i,k} = \sum_{r,j} x_{r,j}(d^{i_k}, i, 0) d_{r,j} - \sum_{r,j} x_{r,j}(d, i, 0) d_{r,j}$ . Notice, to calculate the least cost  $v_k$ -avoiding-path, we set  $d_{k,j} = \infty$  for each node  $v_j$ .

It is not difficult to show that, under the above payment model, every node (except the source node) could not lie about its cost vector to improve its profit. We assume that the source node will not lie about its cost vector. Then the above payment scheme is truthful. We can show that the fast payment scheme based on Algorithm 1 can be modified to compute the payment in time  $O(n \log n + m)$  when each node is an agent in a link-weighted directed network. Notice that, in this network model, the source node could lie about its cost vector such that its total payment to all relay nodes will decrease.

### C. Ratio of Total Payment Over Total Cost of the Path

Clearly, node  $v_i$  pays each node on  $\mathbf{P}(v_i, v_0, c)$  more than its actual cost to make sure that it will not lie about its cost. The overpaid value is the improvement of the least cost path due to the existence of node  $v_k$ . It is not difficult to construct a network example such that the over-payment of a node  $v_i$  could be arbitrarily large. But on the other hand, when we conducted extensive simulations to study the amount of overpayment when the cost of each node is chosen independently and uniformly from a range and the network topology is a random graph, the result shows that the large overpayment usually won't happen in the real world.

Let  $p_i = \sum_{v_k \in \mathbf{P}(v_i, v_0, c)} p_i^k(c)$ , i.e., the total payment of node  $v_i$  to the relay nodes. The metrics of the overpayment used in our simulations are *Total Overpayment Ratio (TOR)*, *Individual Overpayment Ratio (IOR)*, and *Worst Overpayment Ratio*. The TOR of a graph is defined as  $\sum_i p_i / \sum_i \|\mathbf{P}(v_i, v_0, c)\|$ , i.e., the total payment of all nodes over the total cost of all LCPs. The IOR of a graph is defined as  $\frac{1}{n} \sum_i \frac{p_i}{\|\mathbf{P}(v_i, v_0, c)\|}$ , i.e., the average overpayment ratio over all  $n$  nodes. The worst overpayment ratio is defined as  $\max_i \frac{p_i}{\|\mathbf{P}(v_i, v_0, c)\|}$ , i.e., the maximum overpayment ratio over all  $n$  nodes. We found that the IOR and

TOR are almost the same in all our simulations and they take values around 1.5. In all our simulations, the average and the maximum are taken over 100 random instances.

In the first simulation, we randomly generate  $n$  nodes uniformly in a  $2000m \times 2000m$  region. The transmission range of each node is set as  $300m$ . The cost of each node  $v_i$  to forward a packet to any neighbor is  $c_1 + c_2 \cdot r^\kappa$ , where  $r$  is fixed at 300. Here random number  $c_1$  takes value from 300 to 500 and  $c_2$  takes a random value from 10 to 50. The ranges of  $c_1$  and  $c_2$  we used here reflect the actual power cost in one second of a node to send data at  $2Mbps$  rate. The number of nodes in our simulations varies among 100, 150, 200,  $\dots$ , 500. We choose two different  $\kappa$  values 2 and 2.5. Figure 4 (a) illustrates the difference between IOR and TOR when graph model is UDG and  $\kappa = 2$ . We found that these two metrics are almost the same and both of them are stable when the number of nodes increases. Figure 4 (d) illustrates the overpayment respecting to the hop distance to the source node. The average overpayment ratio of node stays almost stable regardless of the hop distance to the source. The maximum overpayment ratio decreases when the hop distance increases, which is because large hop distance to the source node will smooth off the oscillation of the relay costs' difference: for a node closer to the access point, the second shortest path could be much larger than the shortest path, which in turn incurs large overpayment; for a node far away from the access point, the second shortest path has total cost almost the same as the shortest path, which in turn incurs small overpayment. Keep in mind that the payment indeed increases when the hop distance to the source increases. Figure 4 (b) and (c) illustrate the overpayment for UDG graph when  $\kappa = 2$  and  $\kappa = 2.5$  respectively.

In our second simulations, we study the performance of our payment methods when each individual nodes have different transmission ranges instead of the same transmission ranges studied before. The cost  $c_i$  of a node  $v_i$  to send a packet to its neighbors is  $c_1 + c_2 \|r_i\|^\kappa$ . Here the transmission range  $r_i$  of node  $i$  takes a random value from [100, 500]. Figure 4 (e) and (f) illustrate the overpayment for random networks when  $\kappa = 2$  and  $\kappa = 2.5$  respectively. Similar observations were obtained for these simulations.

#### D. Other Issues about the Pricing Mechanism

**Other possible attacks:** There are some other attacks possible to the scheme. A source node may refuse to pay by claiming that he is not the source of the communication and thus should not pay for it. To counter this attack, we require that each node sign the message when it initiates the message, the relay nodes will verify the signature.

Another possible attack is *free riding*: a relay node  $v_k$  on the route  $P(v_i, v_0, c)$  may attempt to piggyback data on the packets sent between the initiator  $v_i$  with the goal of not having to pay for the communications to node  $v_0$ . To counter this attack, the initiator  $v_i$  pays the relay node  $v_k$  only when it receives a signed acknowledgment from the access point  $v_0$ .

**Where to pay:** We briefly discuss how the payment is charged. All payment transactions are conducted at the access point  $v_0$ . Each node  $v_i$  has a secure account at node  $v_0$ . There are two scenarios here. First scenario is that the access point

$v_0$  receives a data from  $v_i$ . Node  $v_0$  verifies the truthfulness of the source and then pays each node  $v_k$  on the LCP  $p_i^k$  and charges that from node  $v_i$ . The second scenario is that a node  $v_i$  retrieves data from node  $v_0$ . A relay node  $v_k$  on the LCP will send a signed acknowledgment after relaying the data to the next hop  $v_j$ . The acknowledgment to  $v_0$  by  $v_k$  includes the authentication from  $v_j$  that  $v_k$  does send data to  $v_j$ . Node  $v_0$  then pays node  $v_k$  and charges node  $v_i$  accordingly after receiving this signed acknowledgment.

A better way to prevent these attacks described before may be to combine the efficient pricing mechanism presented here with the payment management method presented by Jacobsson [8], which we leave as a future work.

**Mobility and dynamic cost:** Our protocols assumed that the network topology does not change for the period time of computing the least cost path, computing the payment, and routing packets from the source to the access point. When wireless nodes move, the above protocol still works as long as the network topology does not change. When the mobility triggers the change of the network topology, our protocol has to update the path, and the payment calculated accordingly. A node will be compensated for only the portion of the traffic it relayed.

For simplicity, we assumed that the cost of each node is fixed for the moment. When the cost of a node changes, the node has to declare its new cost. This will trigger the change of the least cost paths and consequently the payment to nodes in the network. The cost could change due to its battery power changes, the competition among other nodes changes, and so on.

So far, we only concentrate on one session of routing. For the successive sessions, the source node and the access point already know the "actual" cost of all relay nodes. It is natural to ask how much we should pay the relay nodes for later sessions now? One may think that we could pay these nodes their declared costs now. It is not difficult to construct an example such that this simple mechanism is not truthful: a relay node could misreport its cost initially; although its payment for the first session could decrease, but the gain in later sessions is large enough to cover this initial loss. We actually can prove that for successive sessions the source node still has to pay the relay nodes based on the payment scheme described before. The result is omitted here due to space limit. Notice that another reason we could not pay a relay node  $v_k$  its declared cost  $d_k$  is that node  $v_k$  could simply refuse to relay since it gains nothing by relaying.

**Resale the path:** Another possible collusion happens after the payment is calculated and during the process of actually routing the packets. Let  $p_i = \sum_{k=0}^{n-1} p_i^k$ , i.e., the total payment of node  $v_i$  to all relay nodes on the least cost path  $P(v_i, v_0, c)$ . Assume that  $p_i > p_j + \max(p_j^j, c_j)$  for some neighbor  $v_j$  of  $v_i$ . Notice that  $\max(p_j^j, c_j) = x_j(i)p_j^j + (1 - x_j(i))c_j$  since if  $v_j$  is on LCP  $P(v_i, v_0, c)$ , then  $p_j^j \geq c_j$  and  $p_j^j = 0 < c_j$  otherwise. Here  $x_j(i)$  is the indicator function whether node  $v_j$  is on  $P(v_i, v_0, c)$ . Then,  $v_i$  and  $v_j$  can collude in favor of them as follows: (1)  $v_j$  sends the data packets for  $v_i$  and  $v_j$  pays all relay nodes on path  $P(v_i, v_0, c)$ ; (2)  $v_i$  pays  $v_j$  the cost  $p_j + \max(p_j^j, c_j)$ , which covers the payment by  $v_j$ ; (3)  $v_i$  and  $v_j$  split the difference  $p_i - (p_j + \max(p_j^j, c_j))$ , which is the saving of node  $v_i$  from colluding with node  $v_j$ . Notice that it

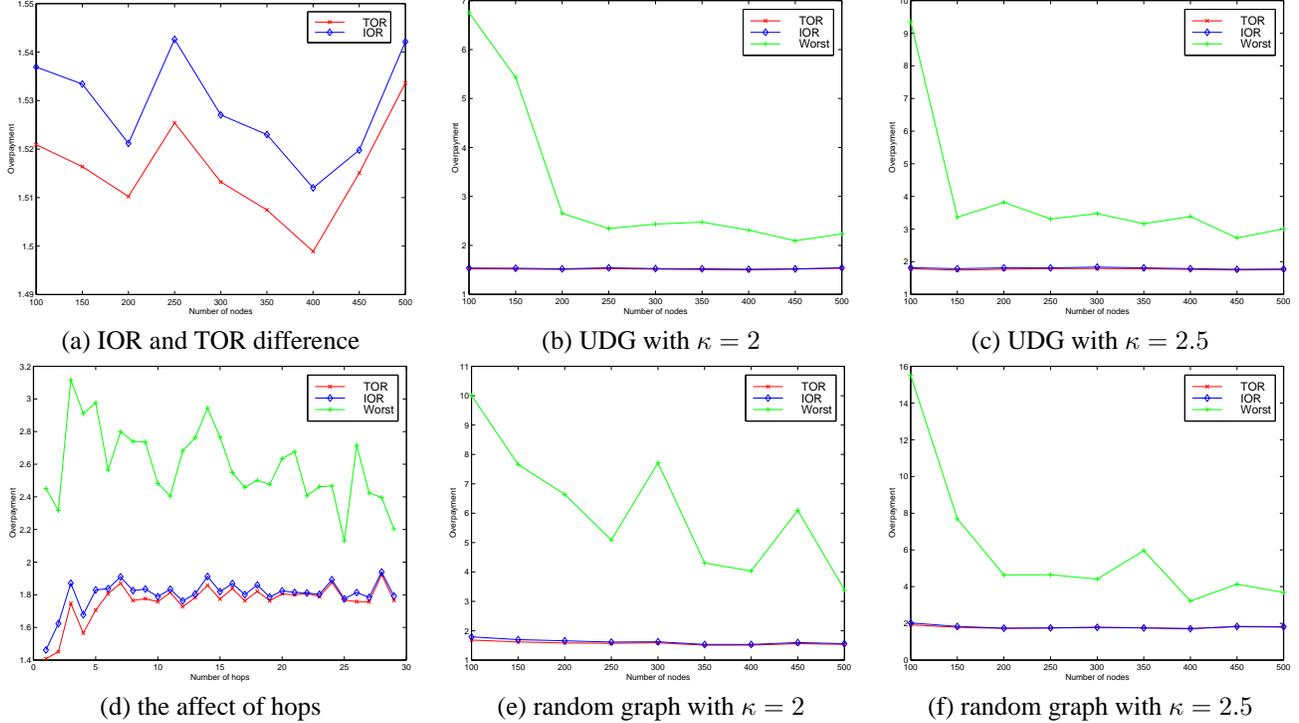


Fig. 4. Overpayment ratios IOR, TOR and the worst ratio for UDG and random graphs.

is possible that  $p_i > p_j + \max(p_i^j, c_j)$  for some neighbor  $v_j$  of  $v_i$ . Figure 5 illustrates such an example of such collusion. It

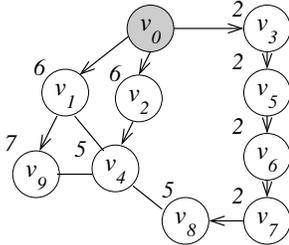


Fig. 5. An example of wireless network and the node cost. The directed links form the shortest path tree from  $v_0$  to every  $v_i$ .

is easy to compute that  $p_8 = 20$ ,  $p_4 = 6$  and  $p_8^4 = 0$ . Notice  $c_4 = 5$ . Thus,  $v_8$  can ask  $v_4$  to forward the data packets using its LCP to  $v_0$ . Node  $v_8$  pays node  $v_4$  a price  $6 + 5 = 11$  to cover its payment  $p_4$  and its cost  $c_4$ , and half of the savings, which is 4.5. Thus, the total payment of node  $v_8$  is only 15.5 now, which is less than  $p_8$  and node  $v_4$  also increases its utility from 0 to 4.5. Currently, we are not aware of any method that can prevent this from happening. We may argue, on the other side, that it may be not necessary to prevent this from happening since this actually encourages some kind of cooperations among nodes.

## V. RELATED WORK

Routing has been part of the algorithmic mechanism-design from the beginning. Nisan and Ronen [4] provided a polynomial-time strategyproof mechanism for optimal route selection in a centralized computational model. In their formulation, each edge  $e$  of the graph is an agent and has a private cost

$c_e$  of sending a unit data along this edge. Their payment scheme falls into the family of VCG mechanisms. Feigenbaum *et. al* [22] then addressed the truthful low cost routing in a different network model. They assumed that each node  $k$  incurs a transit cost  $c_k$  for each transit packet it carries. Their payment scheme again is also a VCG mechanism. They also gave a distributed method such that each node  $i$  can compute the payment to node  $k$  for carrying the transit traffic from node  $i$  to node  $j$  if node  $k$  is on the LCP from  $i$  to  $j$ . Since the mechanism is truthful, any node cannot lie its cost to improve its profit in their distributed algorithm. However, as they pointed out [22], it is unclear how to prevent these selfish nodes from running a different algorithm in computing a payment that is more favorable to themselves since we have to rely on these nodes to run the distributed algorithm, although we know that the nodes will input their true values. Andereg and Eidenbenz [25] recently proposed a routing protocol for wireless ad hoc networks based on the VCG mechanism. They assumed that each node is a selfish agent, and incurs a privately known cost when communicating with a neighbor. In their model, the cost from a node  $v$  to a node  $u$  may be different from the cost from node  $u$  to a node  $v$ , i.e., being asymmetric. They did not consider how to compute the payment efficiently. Moreover, we can show that the distributed implementation, which computes the payment defined by VCG mechanisms, does not work when the underlying network is a directed graph.

Some researchers use totally different methods to deal with selfish wireless networks. We briefly review some of them as follows. Marti *et al.* [9] proposed a scheme based on credibility of nodes. They call a node *misbehaving* if it originally agrees to relay traffic but does not actually. The routing pro-

protocol will avoid using these misbehaving nodes. Buttyan *et al.* [7], [8], [11], [10] presented a sequence of methods to stimulate cooperation among nodes and to prevent nodes from overloading the network. A key idea behind their protocols is that we should compensate nodes providing a service, and charge nodes receiving a service. In their protocols, each node maintains a counter, called *nuglet counter*, in a tamper resistant hardware module. The value of nuglet always remains positive. When a node sends packets to other node, it pays each relay node 1 nuglet, and its nuglet counter is decreased by the hops of the path used. These approaches can be viewed as a fixed price payment. Srinivasan *et al.* [12] proposed two acceptance algorithms, which are used by each wireless node to decide whether to relay the traffic for other node. The methods work per session basis. These methods try to balance the energy consumed by a node for relaying with energy consumed by other nodes in relaying its traffic. In [13], Srinivasan *et al.* proposed an acceptance algorithm called GTFT and they proved that GTFT results in Nash equilibrium. They assumed that each path is  $l$  hops long and the  $l$  relay nodes are chosen with equal probability from the remaining  $n - 1$  nodes. Salem *et al.* [26] presented a charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In their network model, there is a base-station to forward the packets. They use symmetric cryptography to cope with the lying. To counter several possible attacks, it pre-charges some nodes and then refunds them only if a proper acknowledgment is received.

## VI. CONCLUSION

In this paper we gave a strategyproof pricing mechanism that stimulates cooperation for unicast among wireless ad hoc networks. In our strategyproof scheme, each node  $v_k$  first declares its cost of relaying data for other nodes; every node  $v_i$  then computes the least cost path to the access point  $v_0$ ; a payment is also computed for each relay node on the least cost path. We presented the first centralized algorithm with optimal time-complexity to compute such payment. We also discussed in detail how to implement this scheme on each selfish node in a distributed manner. We showed that although each node is selfish, the proposed scheme guarantees that each node will declare its true cost and also follow the designed protocol. As all truthful mechanisms, the proposed scheme pays each relay node more than its declared cost to prevent it from lying. We conducted extensive simulations and found that practically the overpayment is small when the cost of each node is a random value between some range.

Our protocol assumes that nodes will not collude. We showed that *no* truthful mechanism can prevent all pairs of nodes from colluding to improve their utilities. We designed a truthful payment scheme that can prevent nodes from colluding with its neighbors. Our payment scheme is optimum in terms of the individual payment.

In this paper, we assumed that the cost of a node is fixed (at least for one communicating session). In practice, the cost of a node may be dynamic, e.g., it may depend on other ongoing traffics, depend on its remaining battery power, or even depend on the competition by other nodes. A successful study of the static cost model will pave a way for studying the dynamic cost

model. We leave designing payment schemes for dynamic cost model as a future work.

## REFERENCES

- [1] Tim Roughgarden, "Stackelberg scheduling strategies," in *ACM Symposium on Theory of Computing*, 2001, pp. 104–113.
- [2] Tim Roughgarden, "Designing networks for selfish users is hard," in *IEEE Symposium on Foundations of Computer Science*, 2001, pp. 472–481.
- [3] Tim Roughgarden and Eva Tardos, "How bad is selfish routing?," in *IEEE Symposium on Foundations of Computer Science*, 2000, pp. 93–102.
- [4] Noam Nisan, "Algorithms for selfish agents," *Lecture Notes in Computer Science*, vol. 1563, pp. 1–15, 1999.
- [5] Joan Feigenbaum and Scott Shenker, "Distributed algorithmic mechanism design: Recent results and future directions," in *the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, ACM Press, New York, 2002, pp. 1-13.
- [6] Daniel J. Lehmann, Liaden Ita O'Callaghan, and Yoav Shoham, "Truth revelation in approximately efficient combinatorial auctions," in *ACM Conference on Electronic Commerce*, 1999, pp. 96–102.
- [7] L. Buttyan and J. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *ACM/Kluwer Mobile Networks and Applications*, vol. 5, no. 8, October 2003.
- [8] Markus Jakobsson, Jean-Pierre Hubaux, and Levente Buttyan, "A micro-payment scheme encouraging collaboration in multi-hop cellular networks," in *Proceedings of Financial Cryptography*, 2003.
- [9] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. of MobiCom*, 2000.
- [10] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J. P. Hubaux, and J. Y. Le Boudec, "Self-organization in mobile ad-hoc networks: the approach of terminodes," *IEEE Communications Magazine*, vol. 39, no. 6, June 2001.
- [11] L. Buttyan and J. P. Hubaux, "Enforcing service availability in mobile ad-hoc wans," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, 2000, pp. 87–96.
- [12] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao, "Energy efficiency of ad hoc wireless networks with selfish users," in *European Wireless Conference 2002 (EW2002)*, 2002.
- [13] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao, "Cooperation in wireless ad hoc wireless networks," in *IEEE Infocom*, 2003.
- [14] Kamal Jain and Vijay V. Vazirani, "Applications of approximation algorithms to cooperative games," in *ACM Symposium on Theory of Computing*, 2001, pp. 364–372.
- [15] Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker, "Sharing the cost of multicast transmissions," *Journal of Computer and System Sciences*, vol. 63, no. 1, pp. 21–41, 2001.
- [16] W. Vickrey, "Counterspeculation, auctions and competitive sealed tenders," *Journal of Finance*, pp. 8–37, 1961.
- [17] E. H. Clarke, "Multipart pricing of public goods," *Public Choice*, pp. 17–33, 1971.
- [18] T. Groves, "Incentives in teams," *Econometrica*, pp. 617–631, 1973.
- [19] J. Green and J. J. Laffont, "Characterization of satisfactory mechanisms for the revelation of preferences for public goods," *Econometrica*, pp. 427–438, 1977.
- [20] John Hershberger and Subhash Suri, "Vickrey pricing in network routing: Fast payment computation," in *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 252–259.
- [21] Enrico Nardelli, Guido Proietti, and Peter Widmayer, "Finding the most vital node of a shortest path," *Theor. Comput. Sci.*, vol. 296, no. 1, pp. 167–177, 2003.
- [22] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, "A BGP-based mechanism for lowest-cost routing," in *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing.*, 2002, pp. 173–182.
- [23] Kamal Jain and Vijay V. Vazirani, "Group strategyproofness and no subsidy via lp-duality," 2002.
- [24] Herve Moulin and Scoot Shenker, "Strategyproof sharing of submodular costs: Budget balance versus efficiency," in *Economic Theory*, 2002, Available in preprint form at <http://www.aciri.org/shenker/cost.ps>.
- [25] Luzi Anderegg and Stephan Eidenbenz, "Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *Proceedings of the 9th ACM annual international conference on Mobile computing and networking (MobiCom)*, 2003, pp. 245–259.
- [26] Naouel Ben Salem, Levente Buttyan, Jean-Pierre Hubaux, and Markus Jakobsson, "A charging and rewarding scheme for packet forwarding in multi-hop cellular networks," in *ACM MobiHoc*, 2003.