Truthful Low-Cost Unicast in Selfish Wireless Networks

WeiZhao Wang* Xiang-Yang Li*

Abstract— **Much of the existing work in wireless ad hoc networking assumes that each individual wireless node (possibly owned by selfish users) will follow prescribed protocols without deviation. In this paper, we address the issue of user cooperation in selfish and rational wireless ad hoc networks using an incentive approach. Each node has a cost (known only to itself) of relaying unit data for other nodes. In our protocols, each wireless node declares a cost for forwarding a unit data for any other node. When a node wants to send data to the access point, it first computes the least cost path to the access point and computes a payment to each node on the path. We present a strategyproof pricing mechanism such that the profit of each wireless node is maximized only 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 algorithm in the distributed manner. We conduct extensive simulations to study the relation of the total payment of a node to the total cost incurred by all relay nodes and found that the ratio of the total payment over the total cost is small. Our protocol works when the wireless nodes will *not* collude and we show that no truthful mechanism can avoid the collusion of arbitrary two nodes. We also give truthful mechanism when a node only colludes with its neighbors.**

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

I. Introduction

In a 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. Traditionally, it is assumed that nodes in wireless ad hoc networks will always relay packets for each other thus ensuring the network connectivity and throughput. However, the limitation of energy supply raises concerns about this traditional belief.

Consider a user in a campus environment equipped with a laptop. The user might expect that his battery-powered laptop will last without recharging until the end of the day. When he participates in various ad hoc networks, he will be expected to relay traffic for other users. If he accepts all relay requests, he might run out of energy prematurely. Therefore, to extend his lifetime, he might decide to reject all relay requests. If every user argues in this fashion, then the throughput that each user receives will drop dramatically. Srinivasan *et al.* [1] studied the trade-off between an individual user's lifetime and throughput. Here, we argue that the node may refuse to relay the data packets for other nodes at all. For example, when a student seldom uses the network, it is not in his/her interest to relay the packets for other nodes since it only consumes its battery faster. Clearly, selfish wireless nodes may hinder the functioning of the network completely. Thus, a stimulation mechanism is required to encourage users to provide service to other nodes. Cooperation among nodes in an ad hoc network has been previously addressed in [2], [3], [4], [5], [6], [7], [1].

In this paper, we consider a set $V = \{v_0, v_1, \cdots, v_{n-1}\}$ of n wireless nodes (e.g., students on a campus). Each node v_i, depending on its type (e.g., laptop, PDA, cell phone), is associated with an average cost c_i to forward a data packet, and this cost is only known to node v_i. We assume that, 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. Here v_0 is used to represent the access point of the wireless network to the wired network. In addition, we assume that the routing from each node to the access point is connection-oriented. Each node v_j on the network declares a cost d_j for relaying a data packet, which could be different from its true cost c_j. Then node v_i computes the shortest path $\mathbf{P}(v_i, v_0, d)$ to connect the access point v_0 according to the declared cost vector $d = (d_0, d_1, \cdots, d_{n-1})$. For each node v_j, node v_i computes a payment $p_i^j(d)$ according to the declared cost vector d. The *utility*, in standard economic model, of node v_j is $u^j = p_i^j(d) - c_j$ 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 $d_j = c_j$. Since users are self-interested and rational, there is no guarantee that they will reveal its cost truthfully unless they are convinced that they cannot do better by declaring a different cost. The objective of this paper is then to design a payment scheme such that each node v_j has to declare its true cost, i.e., $d_j = c_j$, to maximize its utility. Then we study in detail how to implement the protocol efficiently and truthfully.

By assuming the nodes will not collude, we present a strategyproof pricing mechanism such that the profit of each wireless node is maximized only when it declares its true costs. In our protocol, when a node wants to send data to the access point, it first asks all nodes to declare its cost of forwarding, and computes the least cost path (LCP) to the access point. It then computes a payment to each node v_k on the LCP, which 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. We show that our payment scheme is strategyproof, i.e., the profit of each wireless node is maximized only when it declares its true costs. We present a time optimal method to compute the payment in a centralized manner. We then discuss in detail how to implement the algorithm in the distributed manner. We conduct extensive simulations to study the relation of the total payment of a node to the total cost incurred by all relay nodes.

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, formally define the problem we want to solve, and then review the related work. 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 even the individual nodes may deviate from the protocol and show that it is correct as long as the nodes

*Department of Computer Science, Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616. Emails: wangwei4@iit.edu, xli@cs.iit.edu

are rational and no collusion between nodes. We also conduct simulations to show that the overpayment to the relay nodes according to the pricing scheme is small compared to the actual cost of the least cost path. We conclude our paper in Section IV with a discussion of possible future works.

II. Technical Preliminaries

A. Algorithmic Mechanism Design

The purpose of this subsection is to review the basics of algorithmic mechanism design. Readers familiar with this should skip to the next subsection. Refer the early papers of Nisan and Ronen [8], Feigenbaum, and Shenker [9] and Feigenbaum, Papadimitriou, and Shenker [10] for more detailed description.

A standard economic model for analyzing scenarios in which the agents act according to their own self-interest is as follows. There are n agents. Each agent i, for $i \in \{1, \cdots, n\}$, has some private information t^i, called its *type*. The type t^i could be its cost to forward a packet in a network environment, could be its willing payment for a good in an auction environment. Then the set of n agents define a type vector $t = (t^1, t^2, \cdots, t^n)$, which is called the *profile*. There is an output specification $o = (o_1, o_2, \cdots, o_n)$ that maps each type vector t to a set of allowed outputs. Agent i's preferences are given by a valuation function w^i that assigns a real number $w^i(t^i, o)$ to each possible output o. Here, we assume that the valuation of an agent does not depend on other agents' types. Everything in the scenario is public knowledge except the type t^i, which is a private information to agent i.

A mechanism defines, for each agent i, a set of strategies A^i. For each strategy vector $a = (a^1, \cdots, a^n)$, i.e., agent i plays strategy $a^i \in A^i$, the mechanism computes an *output* $o = o(a^1, \cdots, a^n)$ and a *payment* vector $p = (p^1, \cdots, p^n)$, where $p^i = p^i(a)$. Here the payment p^i is the money given to each participating agent i. If $p^i < 0$, it means that the agent has to pay $-p^i$ to participate in the action. Agent i's *utility* is $u^i = w^i(t^i, o) + p^i$. By assumption of rationality, agent i always tries to maximize its utility u^i. A mechanism is *strategy-proof* if the types are part of the strategy space A^i and each agent maximizes its utility by giving its type t^i as input *regardless* of what other agents do. Let a^{-i} denote the vector of strategies of all other agents except i, i.e., $a^{-i} = (a^1, a^2, \cdots, a^{i-1}, a^{i+1}, \cdots, a^n)$. Let $a|^i b = (a^1, a^2, \cdots, a^{i-1}, b, a^{i+1}, \cdots, a^n)$, i.e., each agent $j \neq i$ uses strategy a^j except that the agent i uses strategy b. The following are some natural constraints which any for strategy-proof mechanism must satisfy:

1. **Incentive Compatibility (IC)**: The payment should satisfy the incentive compatibility, i.e., for each agent i,

$$w^i(t^i, o(a|^i t^i)) + p^i(a|^i t^i) \geq w^i(t^i, o(a|^i a^i)) + p^i(a|^i a^i).$$

Thus, revealing the type t^i is the *dominating strategy*.

2. **Individual Rationality (IR)**: It is also called Voluntary Participation. Every participating agent must have non-negative utility, i.e., $u^i(t, p) \geq 0$.

3. **Polynomial Time Computability (PC)**: All computation is done in polynomial time.

VCG mechanism: Arguably the most important positive result in mechanism design is what is usually called the generalized Vickrey-Clarke-Groves (VCG) mechanism by Vickrey [11], Clarke [12], and Groves [13]. A direct revelation mechanism $m = (o(t), p(t))$ belongs to the VCG family if (1) the output $o(t)$ 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 agent i is $p^i(t) = \sum_{j \neq i} w^j(t^j, o(t)) + h^i(t^{-i})$. Here $h^i()$ is an arbitrary function of t^{-i}. It is proved by Groves [13] that a VCG mechanism is truthful. Green and Laffont [14] proved that, under mild assumptions, VCG mechanisms are the only truthful implementations for utilitarian problems, i.e., $g(o, t) = \sum_i w^i(t^i, o)$.

An output function of a VCG mechanism is required to maximize the objective function (minimization problem can be treated as maximization easily). This makes the mechanism computationally intractable in many cases. Notice that replacing the optimal algorithm with non-optimal approximation usually leads to untruthful mechanisms.

B. Network Model

We consider a set $V = \{v_0, v_1, \cdots, v_{n-1}\}$ of n wireless nodes. Here v_0 is used to represent the access point (AP) of the wireless network to the wired network if it presents. 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 the graph G is node biconnected. In other words, the remaining graph, by removing any node v_i and its incident links from the graph G, is still connected. The bi-connectivity of the communication graph G will prevent the monopoly on the network as will see later in addition to provide fault tolerance.

We also assume that each wireless node has an omni-directional antenna and a single transmission of a node can be received by *any* node within its vicinity, i.e., all its neighbors in G. A node v_j can receive the signal from another node v_i if node v_j is within the transmission range of the sender v_i. Otherwise, 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, forwarding 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. In the terminology of economic theory, c_i is the type of node v_i. All n nodes together define a cost vector $c = (c_0, c_1, \cdots, c_{n-1})$, which is the profile of the network G.

In this paper we restrict our attentions to a unicast between any node v_i and access point v_0 only, it is not very different to generalize to arbitrary node between any pair of nodes v_i and 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 $\mathsf{P}(v_i, v_0, c)$, is used to route the packets. Consider a path $\Pi(i, 0) = v_{r_s}, v_{r_{s-1}}, \cdots v_{r_1}, v_{r_0}$ 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 $\sum_{j=1}^{s-1} c_{r_j}$, which excludes the cost of the source

and the target nodes.

To stimulate cooperation among all wireless nodes, node v_i pays some nodes of the network to forward the data for node v_i 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 $\mathsf{P}(v_i, v_0, d)$ to connect the access point v_0 according to the declared cost vector $d = (d_0, d_1, \cdots, d_{n-1})$. For each node v_j, node v_i computes a payment $p_i^j(d)$ according to the declared cost vector d. The *utility*, in standard economic model, of node v_j is $u^j = p_i^j(d) - x_j(i) \cdot c_j$, where $x_j(i)$ indicates whether v_j relays the packet. We always assume that the wireless nodes are rational: it always tries to maximize its utility u^j.

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 will be $s \cdot p_i^k$.

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 such that $d_j \neq c_j$. The objective of this paper is then to design a payment scheme such that each node v_j has to declare its true cost, i.e., $d_j = c_j$, to maximize its utility. Using the standard assumption from economic model, we assume that the wireless nodes do *not* collude to improve their utility.

D. Related Work

Routing has been part of the algorithmic mechanism-design from the beginning. Nisan and Ronen [8] provided a polynomial-time strategyproof mechanism for optimal route selection in a centralized computational model. In their formulation, the network is modelled as an abstract graph $G = (V, E)$. Each edge e of the graph is an agent and has a private type t^e, which represents the cost of sending a message along this edge. The mechanism-design goal is to find an Least Cost Path (LCP) $\mathsf{P}(x, y)$ between two designated nodes x and y. The valuation of an agent e is $-t^e$ if the edge e is part of the path $\mathsf{P}(x, y)$ and 0 otherwise. Nisan and Ronen used the VCG mechanism for payment. The payment to agent e is 0 if e is not on the LCP $\mathsf{P}(x, y)$, and the payment is $D_{G-\{e\}}(x, y) - D_G(x, y)$ if e is on $\mathsf{P}(x, y)$. Here $D_{G-\{e\}}(x, y)$ is the cost of the LCP through G when edge e is not presented and $D_G(x, y)$ is the cost of the LCP $\mathsf{P}(x, y)$ through G. Clearly, there must have two node disjoint paths connecting x and y to prevent the monopoly. The result in [8] can be easily extended to deal with all-to-all traffics instead of the fixed source and destination node.

Feigenbaum *et. al* [15] then addressed the truthful low cost routing in a different network model. They assume that each node k incurs a transit cost c_k for each transit packet it carries. For any two nodes i and j of the network, $T_{i,j}$ is the intensity of the traffic (number of packets) originating from i and destined for node j. They present a strategyproof payment scheme such that each node k is given a payment p^k to compensate it for carrying transit traffic. Their scheme again is essentially the VCG mechanism. They gave a distributed method such that each node i can compute a number $p_{ij}^k > 0$, which is the payment to node k for carrying the transit traffic from node i to node j if node k is on the LCP $\mathsf{P}(i, j)$. The algorithm converges to a stable state after d' rounds, where d' is the maximum of diameters of graph G removing a node k, over all k. However, they [15] agreed that, *"one important issue that is not yet completely resolved is the need to reconcile the strategic model with the computational model"*. On one hand, the nodes *"may have incentives to lie about costs in order to gain financial advantage"*, and the strategyproof mechanism (essentially, a VCG mechanism) removes these incentives. On the other hand, it is these nodes that *"implement the distributed algorithm we have designated to compute this mechanism; even if the nodes input their true costs, what is to stop them from running a different algorithm that computes prices more favorable to them?"*.

Anderegg and Eidenbenz [16] recently proposed a routing protocol for wireless ad hoc networks based on VCG mechanism. They assumed that each link has a cost and each node is a selfish agent. They showed that the total payment over the cost of the shortest path is bounded by a constant factor of $\frac{\max c_i}{\min c_i}$ by assuming that every node can adjust its transmission range.

There is a vast literature on the mechanism design or implementation paradigm in which some mechanisms are designed to achieve the socially desirable outcomes in spite of user selfishness. Some of these approaches use Nash equilibrium rather than strategyproofness. That is, they assume that simultaneous selfish play leads to a self-consistent equilibrium, called a *Nash Equilibrium*, in which no agent can improve its utility by deviating from its current strategy when other agents keep their strategies. The Nash implementation approach involves designing resources allocation mechanisms with Nash equilibriums that yield the socially desirable outcomes. In contrast, strategyproofness ensures that no matter how other agents behave, truthful revelation is the optimal (or called dominant) strategy for each agent. In other words, they do not have to exhaust their computational power to find a better strategy. Notice that since Nash equilibrium has a weak requirement on the strategies used by the agents, it often can achieve a much wider variety of outcomes.

Some researchers use totally different methods to deal with the selfish wireless networks. We briefly review some of them as follows.

In [4], nodes, which agree to relay traffic but do not, are termed as misbehaving. They used *Watchdog* and *Pathrater* to identify misbehaving users and avoid routing through these nodes. The former runs on every node keeping track of how the other nodes behave; the latter uses this information to calculate the route with the highest reliability. Notice that this method ignores the reason why a node refused to relay the transit traffic for other nodes. A node will be wrongfully labelled as misbehaving when its battery power cannot support many relay requests and thus refused to relay.

In [2], [3], [6], [5], a secure mechanism to stimulate nodes to cooperate and to prevent them from overloading the network is presented. The key idea is that nodes providing a service should be remunerated, while nodes receiving a service should be charged. Each node maintains a counter, called *nuglet counter*, in a tamper resistant hardware module. The nuglet counter decreases when the node wants to send a packet as originator and increased when the node relays a packet. The value of nuglet remains positive, which means that if a node wants

to send packets as originator, it must have forwarded enough packets for other node. To jump-start the system, each node is initially assigned a positive nuglet value. When a node wants to send packets to other node, it pays each relay node 1 nuglet, and its nuglet counter is decreased by the hops of the path used. Based on this concept, they proposed an acceptance algorithm to decide whether to accept or reject a packet relay request. The acceptance algorithm at each node attempts to balance the number of packets it has relayed with the number of its packets that have been relayed by others.

The approaches presented in [2], [3], [6], [5] can be viewed as a fixed price payment.For a selected path connecting the source and the target node, each node on such path is paid *one* nuglet and the source (or target) is charged h nuglets, where h is the number of relay nodes on the path. If a node does not initiate a traffic, then it does not have any incentives to relay the traffic for other nodes since nuglet does not have actual monetary value. If the nuglet reflects actual monetary value, then a node may still refuse to relay the packet if its actual cost is higher than the monetary value of the nuglet.

In [7], two acceptance algorithms are proposed, which are used by the network nodes to decide whether to relay traffic on a per session basis. The goal of these algorithms is to balance [1] the energy consumed by a node in relaying traffic for others with energy consumed by other nodes in relaying traffic and to find an optimal trade-off between energy consumption and session blocking probability. By taking decisions on a per session basis, the per packet processing overhead of previous schemes is eliminated. In [1], a distributed and scalable acceptance algorithm called GTFT is proposed. They proved that GTFT results in Nash equilibrium and proved that the system converges to the rational and optimal operating point. We emphasize, however, that all the above algorithms are based on heuristics and lack a formal framework to analyze the optimal trade-off between lifetime and throughput. More importantly, in their network model, 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, which is unrealistic.

In [17], Salem *et al.* presented a novel 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 count several possible attacks, it pre-charges some nodes and then refunds them only if a proper acknowledgment is received. The basic payment scheme is still based on nuglets.

III. The Pricing Mechanism

A. Payment Scheme

Assume that the node v_i has to send packet to v_j 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 $o(d)$ of the algorithm is the path connecting v_i and v_j with the minimum cost, which is known as $\mathsf{P}(v_i, v_j, d)$. Our payment scheme is also based on the VCG. The payment for node v_k is 0 if $v_k \notin \mathsf{P}(v_i, v_j, d)$. Otherwise, its payment is

$$p_i^k(d) = \|\mathsf{P}_{-v_k}(v_i, v_j, d)\| - \|\mathsf{P}(v_i, v_j, d)\| + d_k$$

Here $\mathsf{P}_{-v_k}(v_i, v_j, d)$ denotes the least cost path between node v_i and v_j if we remove node v_k from the original graph, and $\|\Pi\|$ denotes the total cost of a path Π. 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 $p_i^k(d) - x_k \cdot c_k$. Here $x_k = 1$ if $v_k \in \mathsf{P}(v_i, v_j, d)$, otherwise $x_k = 0$.

B. Fast Payment Computing

The very naive way to calculate the payment for all nodes on the $\mathsf{P}(v_i, v_j, d)$ is to calculate every node's payment using Dijkstra's algorithm. In the worst case there will be $O(n)$ nodes on the $\mathsf{P}(v_i, v_j, d)$, so this naive algorithm will result in a time complexity $O(n^2 \log n + nm)$. In [18], Hershberger and Suri provided a fast payment calculation algorithm for *edge weighted* graph (by assuming the edges are rational agents). Nardelli, Proietti and Widmayer [19] 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 [18], we present an $O(n \log n + m)$ time complexity algorithm for fast payment calculation in a *node weighted* graph. Consider a node $v_k \in \mathsf{P}(v_i, v_j, G)$ and we want to compute $\|\mathsf{P}(v_i, v_j, G \backslash v_k)\|$. 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 $\mathsf{P}(v_i, v_a, G \backslash v_k)$ and $\mathsf{P}(v_b, v_j, G \backslash v_k)$ separately. Then we get the path with the minimum cost from v_i to v_j without node v_k and having $v_a v_b$ on it. Choosing the minimal value among all edges $v_a v_b \in G$, we get $\|\mathsf{P}(v_i, v_j, G \backslash v_k)\|$. Our method works as follows.

Algorithm 1: Fast VCG Payment Computing

1. First we calculate the Shortest Path Tree (SPT) rooted at v_i and v_j respectively and denote them as $SPT(v_i)$ and $SPT(v_j)$. We also assume that

$$\mathsf{P}(v_i, v_j, G) = v_{r_0} v_{r_1} \cdots v_{r_{s-1}} v_{r_s},$$

where $v_{r_0} = v_i$ and $v_{r_s} = v_j$. For a node $v_k \in \mathsf{P}(v_i, v_j, G)$, 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_j.

2. This step calculates a level (denoted as $v_k.level$) for every node v_k on the graph G. If removing a node $v_{r_l} \in \mathsf{P}(v_i, v_j, G)$ would cause node v_k neither connects to v_i nor connects to v_j in the tree $SPT(v_i)$, we set $v_k.level = l$.

3. For every node $v_k \notin \mathsf{P}(v_i, v_j, G)$, we find $\mathsf{P}(v_k, v_j, G \backslash v_{r_l})$, where $l = v_k.level$. Let $R^{-l}(v_k)$ denote $\|\mathsf{P}(v_k, v_j, G \backslash v_{r_l})\|$. All such paths can be found in total time $O(n \log n + m)$ using the following approach.

We find paths $\mathsf{P}(v_k, v_j, G \backslash v_{r_l})$, where l starts from s to 1. Assume that we have found such LCPs for all nodes with level $l \geq h$, and we start to find the LCPs for nodes with level $h-1$. From Lemma 2, we know that $\mathsf{P}(v_k, v_j, G \backslash v_{r_l})$ does not contain any node with level less than l. We find the node v_k with level $h-1$ such that $c_a + \|\mathsf{P}(v_a, v_j, G)\|$ is minimum, where

[1] It is impossible to strictly balance the number of packets a node has relayed for other nodes and the number of packets of this node relayed by other nodes since, in a wireless ad hoc network, majority of the packet transmissions are relayed packets. For example, consider a path of h hops. $h-1$ nodes on path relay the packets for others. If the average path length of all routes is h, then $1 - 1/h$ fraction of the transmissions are transit traffics.

$v_k v_a \in G$ and v_a is processed. Then $\|\mathsf{P}(v_k, v_j, G \backslash v_{r_{h-1}})\| = c_a + \|\mathsf{P}(v_a, v_j, G)\|$ and we mark node v_k processed. Repeat the above step until all such least cost paths for all nodes have been found.

4. For node v_k and each of its neighbors v_s such that $v_s.level < v_k.level$, we calculate $L(v_s) + R^{-l}(v_k) + c_s + c_k$. Choose the neighbor v_s with the minimum value of $L(v_s) + R^{-l}(v_k) + c_s + c_k$ and denote it as $c^{-l}(v_k)$, where l is the level of node v_k. Among all nodes with label l, choose the one with the minimum $c^{-l}(v_k)$ and set c^{-l} to this value, i.e.,

$$c^{-l} = \min_{v_k.level=l} c^{-l}(v_k).$$

5. For each node $v_{r_l} \in \mathsf{P}(v_i, v_j, d)$, 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 $v_a.level < l < v_b.value$. 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 $L(v_a) + R(v_b) + c_a + c_b$. An edge $v_a v_b$ is added to H at most once and deleted from H once. 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 $\|\mathsf{P}_{-v_{r_l}}(v_i, v_j, d)\|$.

6. Calculate the payment to node v_{r_l} as follows

$$p_{ij}^{r_l} = \|\mathsf{P}_{-v_{r_l}}(v_i, v_j, d)\| - \|\mathsf{P}(v_i, v_j, d)\| + d_{r_l}$$

The correctness of this algorithm comes from the following observations of the shortest v_k-avoiding path.

Lemma 1: Assume that, for a node $v_{r_l} \in \mathsf{P}(v_i, v_j, d)$,

$$\mathsf{P}_{-v_{r_l}}(v_i, v_j, d) = v_{l_0} v_{l_1} \cdots v_{l_{t-1}} v_{l_t},$$

where $v_{l_0} = v_i$ and $v_{l_s} = v_j$. 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 a and b such that $v_{l_b}.level < l$, $v_{l_a}.level \geq l$ and $b > a$. Notice that $\mathsf{P}(v_i, v_{l_b}, G)$ 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 $\mathsf{P}(v_i, v_{l_b}, G)$ concatenated with $v_{l_{b+1}} \cdots v_{l_{t-1}} v_{l_t}$ will result in a v_{r_l}-avoiding path with smaller weight since the path $\mathsf{P}(v_i, v_{l_b}, G) \subset SPT(v_i)$ 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. This finished our proof. ■

Lemma 2: For a node v_k such that $v_k.level = l$, then $\mathsf{P}(v_k, v_j, G)$ cannot contain any 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 $\mathsf{P}(v_k, v_j, G)$ contains a node v_{r_a} with $a < l$. Obviously, $\mathsf{P}(v_i, v_k, G)$ contains path $v_{r_0} v_{r_1} \cdots v_{r_{l-1}} v_{r_l}$. For simplicity, we assume that path $\mathsf{P}(v_i, v_k, G)$ 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 III-B. Similarly, we assume that $\mathsf{P}(v_k, v_j, G)$ 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 $\|P2\| + \|P_3\| \leq \|P_1\|$ from the property of least cost path $\mathsf{P}(v_i, v_k, G)$, and $\|P1\| + \|P_3\| \leq \|P_2\|$ from the property of least cost path $\mathsf{P}(v_k, v_j, G)$. Consequently, we have $\|P_3\| \leq 0$, which is a contradiction. This finishes our proof. ■

Similarly, we have

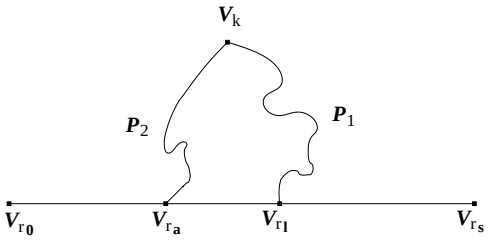


Fig. 1. Observation of v_{r_l}-avoiding shortest path.

Lemma 3: Consider the path $\mathsf{P}_{-v_{r_l}}(v_k, v_j, G \backslash v_{r_l})$. If there exists a node $v_{k'}$ on this path with $v_{k'}.level < v_k.level$, then node v_k cannot appear on the $\mathsf{P}_{-v_{r_l}}(v_i, v_j, G \backslash v_{r_l})$.

The proof of this lemma is omitted due to space limit. Lemma 3 allows us to only focus on the path connecting v_k and v_j 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 $R(v_k)$. As discussed, 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 $\sum_{v_l.level=l} deg(v_k) + n_l \log n_l$ to find the c^{-l}. Summing l from 1 to $s-1$, we get the time complexity as $\sum deg(v_k) + \sum_{l=1}^{s-1} n_l \log n_l < m + n \log n$. Thus, the third and fourth steps will take time complexity of $O(n \log n + m)$ also. The fifth step will have at most m insertions, and m deletions, 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 in the wired or cellular network, wireless ad hoc networks are lack of a centralized authority. Thus, it is more desirable to compute this payment p_i^k to a relay node v_k in a distributed manner. In the following sections, 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. First, 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. Let $c(i, 0)$ be the cost of the shortest path from node v_0 to node v_i. Second, every node v_i computes its payment p_i^k in a distributed manner which is based on the algorithm in Feigenbaum *et al.* [15].

The first stage can be easily implemented using Dijkstra's algorithm, so we omit this one. In the second stage, we first form a shortest path tree T root at node v_0, and every node knows its parent and children in tree T. Initially at node v_i, each entry p_i^k is set to ∞, if v_k is on the LCP $\mathsf{P}(c, i, 0)$; otherwise, p_i^k is set to 0. Every node now broadcasts its entries p_i^k to its neighbors. When node v_i receives an updated price from a neighbor v_j, it updates the price entries as follows:

1. If v_j is the parent of v_i, node v_i then updates

$$p_i^k = \min(p_i^k, p_j^k) \text{ if } v_k \in \mathsf{P}(v_i, v_0, c).$$

2. If v_i is the parent of v_j, node v_i then updates

$$p_i^k = \min(p_i^k, p_j^k + c_i + c_j) \text{ if } v_k \in \mathsf{P}(v_i, v_0, c).$$

3. If nodes v_i and v_j are not adjacent in tree T, for every $v_k \in \mathsf{P}(c, i, 0)$, node v_i then update p_i^k as follows. If $v_k \in \mathsf{P}(v_j, v_0, c)$ then

$$p_i^k = \min(p_i^k, p_j^k + c_j + c(j,0) - c(i,0));$$

If $v_k \notin \mathsf{P}(v_j, v_0, c)$ then

$$p_i^k = \min(p_i^k, c_k + c_j + c(j,0) - c(i,0)).$$

Whenever any entry p_i^k of 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 finite number of rounds (at most n rounds).

D. Compute the Payment Truthfully

In the previous subsection, we presented a distributed method to compute the payment of p_i^k based on a strategyproof pricing mechanism for unicast routing. Notice that this algorithm 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 [15], the authors pointed out that if agents *are required to sign all of the messages that they send and to verify all of the messages that they receive from their neighbors*, then the protocol can be modified so that all forms of cheating are detectable. Notice that even using this 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.

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 second stage, it is not unnecessary for us to worry about nodes' lying in the first stage. In Figure 2, the shortest path between v_0 and v_1 should be $v_1v_4v_3v_2v_0$, it is easy to calculate v_1's payment to v_2, v_3 and v_4 is 2, so the overall payment is 6. If v_0 lies that it is not a neighbor of v_4, then its shortest path becomes $v_1v_5v_0$, now it only need 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.

To solve this, we present a new distributed method as follows.

Algorithm 2: **Modified Distributed Algorithm**

First Stage:

1. For every node v_i, it has two entries: $D(v_i)$ stores the shortest distance to v_0 and $FH(v_i)$ stores its corresponding first hop neighbor. 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 ∞ and $FH(v_i)$ to NULL. Every node broadcasts its information to its neighbors.

2. For every node v_i, when it receives information from its neighbor v_j, it compares $D(v_i)$ with $D(v_j) + c_j$: if $D(v_i) > D(v_j) + c_j$ then set $D(v_i)$ to $D(v_j) + c_j$ and $FH(v_i)$ to v_j. There are two cases here:

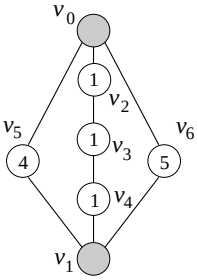


Fig. 2. The node has the incentive to lie about his shortest path

The first case is that $v_i \neq FH(v_j)$: If $D(v_i) + c_i < D(v_j)$ then 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 broadcasts this information.

The second case is that $v_i = FH(v_j)$: If $D(v_i) + c_i \neq D(v_j)$ then node v_i contacts v_j directly using reliable and secure connection asking node 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, node v_j broadcasts this information.

Second Stage:

1. For every node v_i, when its entry for p_i^k changes, it not only broadcasts the value of p_i^k, it should also broadcast that which node p_j that triggers this change.

2. When v_i receives information p_j^k from its neighbor v_j, it updates the p_i^k using the updating algorithm presented in previous subsection. Additionally, if v_i triggers the change for p_j^k, it should recalculate p_j^k for v_j using the updating algorithm in previous section to verify it. If his answer and the payment sent from its neighbor does not match, node v_i then notifies v_j and other nodes. Node v_j will then be punished accordingly.

It is easy to verify that this algorithm is truthful and no node can lie about its neighbor information and do not follow the payment calculation procedure. The problem remaining is how to make it more efficient.

E. Collusion of Nodes

Using the standard assumption from economic model, we assumed that the wireless nodes do *not* collude to improve their utilities. 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 [20], [21]. A pricing mechanism is said to be group strategyproof in [20], [21] 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 cost, 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 d_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 d.

Proof: From lemma 4, we know $w^i(o, c_i)$ does not depend on d_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 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 exists $d_{k_1} \neq d_{k_2}$ and $u^i(d|^k d_{k_1}) \neq u^i(d|^k d_{k_2})$. Without loss of generality we assume that $u^i(d|^k d_{k_1}) > u^i(d|^k d_{k_2})$. From lemma 4, we have $u^k(d|^k d_{k_1}) = u^k(d|^k d_{k_2})$ since output o is not changed. Consider the case with original profile $c = d|^k d_{k_2}$. 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. ■

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, $D_1^k \bigcup D_0^k$ 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 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 exists two profile $(d|^i d_{i_1})$ 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 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

$$u^k(d|^i d_{i_1}) + u^i(d|^i d_{i_1}) \quad = \quad p^k(d|^i d_{i_1}) + \overline{P_i} - x_k \cdot c_k,$$

where $x_k = x_k(d|^i d_{i_1}) = x_k(d|^i d_{i_2})$. Now consider 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)

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

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

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

where $x'_k = x_k(d|^i c_i)$. Now consider 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

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

The last inequality comes $x_k(d|^i d_{i_1}) = x_k \leq x'_k = x_k(d|^i c_i)$ (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 Π under cost profile d. Let Π_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 Π_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, Π_1 and Π_2 are different paths. From the optimality of Π_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 Π_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 finished our proof. ■

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 considering a profile d of their declared cost, and v_k in on LCP. Fixing 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 $a_k = \mathsf{P}(v_i, v_j, G\backslash v_k) - \mathsf{P}(v_i, v_j, G|^k 0)$ and $\overline{P} - \underline{P} = a_k$ (otherwise node v_k can lie about its cost to improve its utility). In other words, $\Delta_P = \overline{P} - \underline{P} = |\mathsf{P}(v_i, v_j, G\backslash v_k) - \mathsf{P}(\underline{v_i}, v_j, G|^k 0)|$ depends on d, which is a contradiction to both $\overline{P}$ and $\underline{P}$ are constants. This finishes our proof. ■

Theorem 7 relieves us from designing any strategy-proof mechanism for arbitrary $k-$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 $v_k \in \mathsf{P}(v_i, v_j, G)$, the path $\mathsf{P}_{v_k}(v_i, v_j, G)$ uses a node v_t that is a neighbor of v_k and $v_t \notin \mathsf{P}(v_i, v_j, G)$. 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 $G\backslash N(v_k)$ 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

$$\tilde{p}_i^k(d) = \|\mathsf{P}_{-N(v_k)}(v_i, v_j, d)\| - \|\mathsf{P}(v_i, v_j, d)\| + d_k,$$

where $\mathsf{P}_{-N(v_k)}(v_i, v_j, d)$ is the least cost path connecting v_i and v_j in graph $G\backslash N(v_k)$. Notice that the payment to a node $v_k \notin \mathsf{P}(v_i, v_j, d)$ could be positive when node v_k has a neighbor on $\mathsf{P}(v_i, v_j, d)$.

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

Theorem 8: The payment scheme $\tilde{p}$ is a strategyproof mechanism that prevent *any* two neighboring nodes from colluding.

Proof:

Notice for any two neighboring node v_k and v_l

$$u^k(c) = \sum v^k(o(c), c_k) + h^{-k}(c^{-N(v_k)})$$
$$u^l(c) = \sum v^l(o(c), c_l) + h^{-l}(c^{-N(v_l)})$$

Sum them we got

$$u^l(c) + u^k(c) = 2\sum v^i(o(c), c_i) + + h^{-k}(c^{-N(v_k)}) + h^{-l}(c^{-N(v_l)})$$

Notice that $h^{-k}(c^{-N(v_k)}) + h^{-l}(c^{-N(v_l)})$ doesn't depends on d_l and $\sum v^i(o(c), c_i)$ is maximized, so v_k and v_l will maximize their utility by revealing their true cost.

■

It is easy to show that the above payment scheme is optimum in terms of the payment to each individual node. Generally, let $\{Q(v_1), Q(v_2), \cdots, Q(v_n)\}$ be a set of subsets of nodes, i.e., $Q(v_i) \subset V$. We then show how to design a truthful mechanism such that any node v_k can *not* collude with another node in $Q(v_k)$ to increase their total utilities. For simplicity of notation, assume that $v_i \in Q(v_i)$, for $1 \leq i \leq n$. It is easy to show that the following mechanism is truthful: 1) the output is the least cost connecting the source v_i and destination v_j; 2) the payment $\tilde{p}_i^k(d)$ to a node v_k is

$$\tilde{p}_i^k(d) = \|\mathsf{P}_{-Q(v_k)}(v_i, v_j, d)\| - \|\mathsf{P}(v_i, v_j, d)\| + d_k.$$

Here, obviously, we need graph $G\backslash Q(v_k)$ to be connected for any node v_k.

F. Link Cost Instead of Node Cost

Energy conservation is a critical issue in *ad hoc* wireless network for the node and network life, as the nodes are powered by batteries only. So far, we assumed that the cost of a node forwarding data to any neighbor is same. However, each node could have different cost of forwarding data to different neighbor by using power adjustment technique. In the most common power-attenuation model, the power, denoted by $p(e)$, needed to support a link $e = v_i v_j$ is $\alpha + \beta\|v_i v_j\|^\kappa$, where $\|v_i v_j\|$ is the Euclidean distance between v_i and v_j, α, β, κ are positive real constants dependent on the wireless transmission environment and κ is often between 2 and 5. This power consumption $\beta\|v_i v_j\|^\kappa$ is typically called *path loss*, and α is the overhead cost for each device to receive and then process the signal. We assume that the parameter κ is the same for all wireless nodes, but different node may have different values of α and β. Obviously, given the position of a neighbor v_j, the power cost of node v_i

sending signal to node v_j is uniquely determined by parameters α, β and the position of node v_i. Thus, we assume that each wireless node v_i has a private type $c_i = (c_{i,0}, c_{i,2}, \cdots, c_{i,n-1})$. 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 ∞. Obviously, $c_{i,i} = 0$.

Notice that this model of network is different from the network models used by previous strategyproof pricing mechanisms [8], [15] for unicast. In their models, either a link is an agent, which has 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 $v_{i_s}, v_{i_{s-1}}, \cdots v_{i_1}, v_{i_0}$ connecting node v_i to v_0, where $v_i = v_{i_s}$ and $v_0 = v_{i_0}$), the valuation $w^{i_k}(t^{i_k}, o)$ of the node v_{i_k} is $-c_{i_k,i_{k-1}}$. Given the declared types by all wireless nodes, the pricing 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 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_iv_j is $d_{i,j}$. A least cost directed path $\mathsf{P}(v_i, v_0, d)$ 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_kv_j is on the directed path from v_i to v_0. The payment $p_i^k(d)$ of node v_i 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,

$$\Delta_{i,k} = \sum_{r,j} x_{r,j}(d|^k\infty, i, 0)d_{r,j} - \sum_{r,j} x_{r,j}(d, i, 0)d_{r,j}.$$

Notice, to calculate the least cost v_k-avoiding-path, we set $d_{k,j} = \infty$ for each node v_j. It is easy to write the above payment in VCG format $\sum_{j\neq k} w^j(d_j, o(t)) + h^k(d^{-k})$, which implies that the 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.

G. Ratio of Total Payment Over Total Cost of the Path

Remember that the payment of a node v_i to a node v_k on the LCP $\mathsf{P}(v_i, v_0, c)$ is

$$p_i^k(d) = \|\mathsf{P}_{-v_k}(v_i, v_0, d)\| - \|\mathsf{P}(v_i, v_0, d)\| + d_k$$

Clearly, node v_i pays each node on the LCP $\mathsf{P}(v_i, v_0, c)$ 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, while 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.

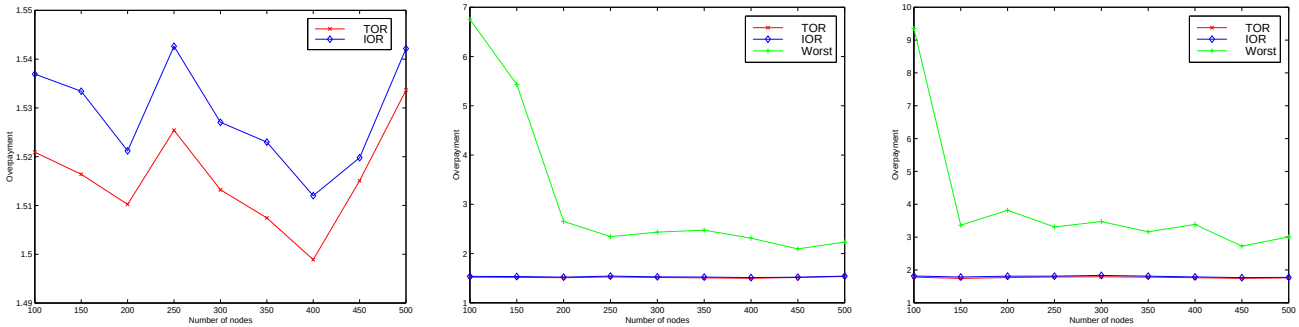
Let $p_i = \sum_{v_k\in\mathsf{P}(v_i,v_0,c)} p_i^k(c)$, i.e., the total payment of node v_i to the relay nodes. The metrics of the overpayment used in our simulations are *Total Overpayment Ratio* (TOR), *Individual Overpayment Ratio* (IOR), and *Worst Overpayment Ratio*. The TOR of a graph is defined as $\sum_i p_i / \sum_i c(i, 0)$, i.e., the total payment of all nodes over the total cost of all LCPs. The IOR of a graph is defined as $\frac{1}{n}\sum_i \frac{p_i}{c(i,0)}$, i.e., the average overpayment ratio over all n nodes. The worst overpayment ratio is defined as $\max_i \frac{p_i}{c(i,0)}$, 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 another node v_j is $\|v_iv_j\|^\kappa$. The number of nodes in our simulations varies among 100, 150, 200, $\cdots$, 500. We choose two different κ values 2 and 2.5. Figure 3 (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 3 (d) illustrates the overpayment respecting to the hop distance to the source node. The average overpayment ratio of a 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 cost difference: for node closer to the source node, the second shortest path could be much larger than the shortest path, which in turn incurs large overpayment; for node far away from the source, 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 overpayment indeed increases when the hop distance to the source increases. Figure 3 (b) and (c) illustrate the overpayment for UDG graph when $\kappa = 2$ and $\kappa = 2.5$ respectively.

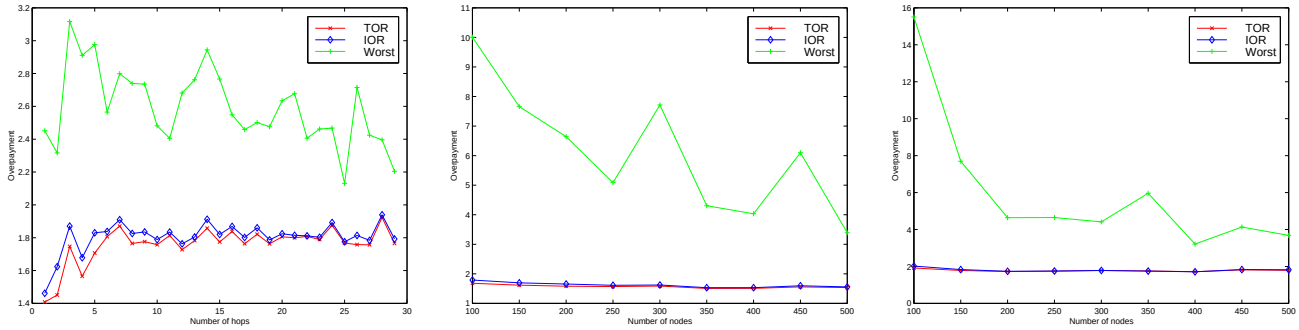
In our second simulations, we vary the transmission range of each wireless node from $100m$ to $500m$, and the cost $c_{i,j}$ of a node v_i to send a packet to another node v_j within its transmission range is $c_1 + c_2\|v_iv_j\|^\kappa$, where c_1 takes value from 300 to 500 and c_2 takes value from 10 to 50. The ranges of c_1 and c_2 we used here reflects the actual power cost in one second of a node to send data at $2Mbps$ rate. When node v_j is not within the transmission range of node v_i, cost $c_{i,j}$ is set to ∞. Figure 3 (e) and (f) illustrate the overpayment for random graph when $\kappa = 2$ and $\kappa = 2.5$ respectively.

H. Other Issues about the Pricing Mechanism

Other possible attacks: There are some other attacks possible to the scheme. A node may refuse to pay by claiming that he did not initiate some communication and thus should not pay for it. To count this attack, we require that each node sign the mes-



(a) IOR and TOR difference (b) UDG with $\kappa = 2$ (c) UDG with $\kappa = 2.5$



(d) the affect of hops (e) random graph with $\kappa = 2$ (f) random graph with $\kappa = 2.5$

Fig. 3. Overpayment ratios IOR, TOR and the worst ratio for UDG and random graphs.

sage 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 $\mathsf{P}(v_i, v_0, c)$ 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 count this attack, the initiator v_i pays the relay node v_k only when it receives a signed acknowledgment from node 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. When the access point receives a data from v_i, it verifies the truthfulness of the source and then pay each node v_k on the LCP p_i^k and charge that from node v_i. When a node v_i retrieves data from the node v_0, node v_k on the LCP will send a signed acknowledgment after receiving the data. Node v_0 then charges node v_i accordingly after receiving this signed acknowledgment.

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 $\mathsf{P}(v_i, v_0, c)$. Assume that $p_i > p_j + \max(p_i^j, c_j)$ for some neighbor v_j of v_i. Notice that $\max(p_i^j, c_j) = x_j(c, i, 0)p_i^j + (1 - x_j(c, i, 0))c_j$ since if v_j is on LCP $\mathsf{P}(v_i, v_0, c)$, then $p_i^j \geq c_j$ and $p_i^j = 0 < c_j$ otherwise. Here $x_j(c, i, 0)$ is the indicator function whether node v_j is on the LCP $\mathsf{P}(v_i, v_0, c)$. 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 $\mathsf{P}(v_i, v_0, c)$; (2) v_i pays v_j the cost $p_j + \max(p_i^j, c_j)$, which covers the payment of v_j and the actual payment v_j should get if v_i just sends the packets directly along the LCP $\mathsf{P}(v_i, v_0, c)$. (3) v_i and v_j splits the difference $p_i - (p_j + \max(p_i^j, c_j))$, which is the saving of node v_i from collusion with node v_j. Notice that it is possible that $p_i > p_j + \max(p_i^j, c_j)$ for some neighbor v_j of v_i. Figure 4 illustrates such an example of such possible collusion. It is easy

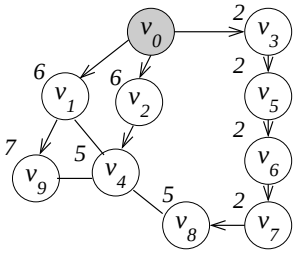


Fig. 4. An example of wireless network and the node cost. The directed links form the shortest path tree from v_0 to every v_i.

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.

IV. Conclusion

In this paper we give 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 optimal time algorithm to compute such payment

in a centralized manner. 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 VCG 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 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 exists that can prevent all pairs of nodes from colluding to improve their utilities. We designed a truthful payment scheme such that it can prevent nodes from colluding with its neighbors. Here we assume that the network is still connected by removing any node and all its neighbors. Our payment scheme is optimum in terms of the individual payment.

REFERENCES

[1] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao, "Cooperation in wireless ad hoc wireless networks," in *IEEE Infocom*, 2003.

[2] 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.

[3] Markus Jakobsson, Jean-Pierre Hubaux, and Levente Buttyan, "A micropayment scheme encouraging collaboration in multi-hop cellular networks," in *Proceedings of Financial Cryptography*, 2003.

[4] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. of MobiCom*, 2000.

[5] 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.

[6] 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.

[7] 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.

[8] Noam Nisan and Amir Ronen, "Algorithmic mechanism design," in *Proc. 31st Annual Symposium on Theory of Computing (STOC99)*, 1999, pp. 129–140.

[9] Joan Feigenbaum and Scott Shenker, "Distributed algorithmic mechanism design: Recent results and future directions," .

[10] 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.

[11] W. Vickrey, "Counterspeculation, auctions and competitive sealed tenders," *Journal of Finance*, pp. 8–37, 1961.

[12] E. H. Clarke, "Multipart pricing of public goods," *Public Choice*, pp. 17–33, 1971.

[13] T. Groves, "Incentives in teams," *Econometrica*, pp. 617–631, 1973.

[14] J. Green and J. J. Laffont, "Characterization of satisfactory mechanisms for the revelation of preferences for public goods," *Econometrica*, pp. 427–438, 1977.

[15] 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.

[16] 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.

[17] Naouel Ben Salem, Levente Buttyan, Jean-Pierre Hubaux, and Markus Kakobsson, "A charging and rewarding scheme for packet forwarding in multi-hop cellular networks," in *ACM MobiHoc*, 2003.

[18] 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.

[19] 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.

[20] Kamal Jain and Vajay V. Vazirani, "Group strategyproofness and no subsidy via lp-duality," 2002.

[21] 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`.