RESOURCE MANAGEMENT IN WIRELESS NETWORKING
M. Cardei, I. Cardei and D.-Z. Du (Eds.) pp. – – –
©2004 Kluwer Academic Publishers

Truthful Computing in Wireless Networks

Xiang-Yang Li
Department of Computer Science
Illinois Institute of Technology, Chicago, IL 60616
E-mail: `xli@cs.iit.edu`

WeiZhao Wang
Department of Computer Science
Illinois Institute of Technology, Chicago, IL 60616
E-mail: `wangwei4@iit.edu`

Contents

1 Introduction

1.1 Ad Hoc Wireless Networks

Wireless network has received significant attention over past few years due to its potential applications in various situations such as battlefield, emergency relief and environmental monitor, etc. Unlike wired networks and cellular networks which have fixed infrastructures, wireless *ad hoc* network enjoys a more flexible composition. Each mobile node has a transmission range and energy cost. A node v can receive the signal from another node u iff node v is within node u's transmission range. We assume that when u sends a packet, it consumes node u some energy and it does not cost node v any energy to receive it. If the receiving node is not within the sender's transmission range, then it must choose some intermediate nodes to repay the message. So unlike the wired networks, all nodes in the wireless ad hoc network should be able to act as a router. On the other aspect, the wireless node usually uses omni-directional antenna, which means that it can use a broadcasting-like manner to distribute the message to all nodes within its transmission range. We consider a wireless ad hoc network $G = (Q, E)$ consisting of a node set Q with $|Q| = n$ distributed in a two-dimensional plane, and directed edge $e = uv \in E$ if v can receive signal from u directly.

There are two different category of wireless ad hoc nodes: *fixed* transmission range and *adjustable* transmission range. For *fixed* transmission range nodes, their transmission range have been fixed and can't be adjusted afterward. So there is a directed arc from u to v if node v is in the transmission range of node u. Here the transmission cost depends on node u regardless of the distance between two nodes. Thus it can be considered as a *node weighted graph.* If all nodes' transmission range is the same, by properly scaling, we can assume all nodes have transmission range 1. Thus, wireless topology can be modelled by a *Unit Dish Graph(UDG).* The wireless nodes of second category have *adjustable* transmission range: they can adjust their transmission range when necessary. Thus the cost to send a packet from node u to v not only depends on u but also depends on the geometry distance of u and v. For example, under most power attenuation model, the power needed to support a link uv is $|uv|^\beta$, where $\beta \in [2, 5]$ depends on the transmission environment. We call this graph *edge weighted graph.* The weight of an edge uv is the power needed to support the communication between u and v.

1.2 Why Truthful Computing

Many existing works in wireless ad hoc networking assume that each individual wireless node (possibly owned by individual selfish users) will follow prescribed protocols without deviation. However, each user may modify the behavior of an algorithm for self-interested reasons.

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, his laptop might run out of energy prematurely. Therefore, to extend his laptop's 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. For some extreme cases, those students who needn't access the network even wouldn't care about the existence of the whole wireless *ad hoc* network. Clearly, they won't relay any messages at all. Thus, a stimulation mechanism is required to encourage users to provide service to other users.

Throughout this chapter, we address these stimulation mechanism that stimulates every individual node following prescribed protocols without deviation, which also known as *truthful computing.*

1.3 Approaches and challenges

There are generally two ways to implement the truthful computing: *credit based method* and *incentive based method.* The first category used various non-monetary approaches including auditing, system-wide optimal point analysis and special hardware. The basic idea of credit based method is that all nodes will cooperate in order to achieve the system optimal performance, and the overall system optimum will in turn benefit the individual node. Some methods falling in this category can be found in [1, 2, 3, 4, 5, 6, 7]. The drawback of this method is that overall system optimum doesn't necessarily guarantee the individual optimality. Thus the nodes still have the incentive to deviate from their normal activity. The second method borrowed some ideas from the micro-economic and game-theoretic world, which involves the monetary transfer. The key result of this category is that all nodes won't deviate from their normal activities because they will benefit most when they reveal their true cost, even knowing all other nodes' true costs. We can thus achieve the optimal system performance.

In wireless ad hoc network environment, it is very expensive for one node to run the centralized algorithm for all nodes. Thus we often need to design some distributed algorithms or even localized algorithms. But one difficult problem has risen in distributed truthful computing environment: the algorithm is running on the selfish-node, is it a paradox asking the node itself to truthfully compute its own payment? On the other hand, there are some questions in the wireless ad hoc networks that are computationally intractable, so can we design some approximation method without losing the truthfulness of the mechanism?

The rest of the chapter is organized as follows. In Section 2, we review the credit based methods proposed in the literature used for truthful computing in wireless ad hoc networks. In Section 3, we discuss in detail the incentive based methods used for unicast and multicast in wireless ad hoc networks. In Section 4, we discuss the truthful computing at other layers of the wireless ad hoc networks such as MAC, TCP, and application layer. We conclude this Chapter in Section 5.

2 Credit Based Methods

Credit based methods have been widely proposed to solve the selfishness in wireless ad hoc networks for several years. Most of them are based on the simulation and are heuristics. They usually lack of formal analysis and theoretical proof, but several of them work well in the real world.

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. *Watchdog* runs on every node keeping track of how the other nodes behave; *Pathrater* 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 traffics 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. It also does not provide any incentives to encourage nodes to relay the message for other nodes.

In [2], Buttyan *et al.* focused on the problem how to stimulate selfish nodes to forward the packets for other nodes. Their approach is based on a so called *nuglet counter* in each node. A node's counter is decreased when sending its own packet, and is increased when forwarding other nodes' packet. All counters should always remain positive. In order to protect the proposed mechanism against misuse, they presented a scheme based on a trusted and tamper resistant hardware module in each node, which generates cryptographically protected security headers for packets and maintains the nuglet counters of the nodes. They also studied the behavior of the proposed mechanism analytically and by means of simulations, and showed that it indeed stimulates the nodes for packet forwarding.

In [6], they still use a nugget counter to store the nuglets and besides that they use a fine which decreases the nugget counter to prevent the node from not relaying the packet. They use Packet Purse Model to discourage the user to send useless traffic and overload the network. The basic idea presented in [6] is similar to [2] but different in the implementation.

In [7], two acceptance algorithms are proposed. These algorithms are used by the network nodes to decide whether to relay traffic on a per session basis. The goal of them is to balance [1] the energy consumed by a node in relaying traffics for others with energy consumed by other nodes to relay its 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 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

[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 the 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.

to analyze the optimal trade-off between lifetime and throughput. More importantly, they assumed that each path is h hops long and the h relay nodes are chosen with equal probability from the remaining $n-1$ nodes, which is unrealistic.

In [8], 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 precharges some nodes and then refunds them only if a proper acknowledgment is received. Their basic payment scheme is still based on nuglets.

In [3] Jakobsson *et al.* described an architecture for fostering collaboration between selfish nodes of multi-hop cellular networks. Based on this architecture, they provided mechanisms based on per packet charge to encourage honest behavior and to discourage dishonest behavior. In their approach, all packet originators attach a payment token to each packet, and all intermediaries on the packet's path to the base station verify whether this token corresponds to a special token called *winning ticket*. Winning tickets are reported to nearby base stations at regular intervals. The base stations, therefore, receive both reward claims (which are forwarded to some accounting center), and packets with payment tokens. After verifying the validity of the payment tokens, base stations send the packets to their desired destinations, over the backbone network. The base stations also send the payment tokens to an accounting center. Their method also involves some traditional security method including auditing, node abuse detection and encryption etc.

Generally speaking, these methods need some extra equipment, including special hardware, which is not very realistic under certain situation. In addition some methods assume that every node will enjoy better performance if the whole system's performance increases, but it is easy to construct some counter cases. One of these counter cases is the TCP/IP's congestion control scenery. Nodes using TCP/IP protocols will decrease their packet sending rate when they encounter some packet loss or timeout, so the overall system can survive the network congestion. Considering some malicious users, if they don't decrease their sending rate even they meet packet loss or time-out, they will enjoy a much faster sending rate than other nodes which conform to the rule, in the meanwhile the overall system performance will decrease sometime. Thus, we will concentrate much on incentive based methods instead of credit based methods in the following sections, which has also been studied extensively in wired networks and economics recently.

3 Incentive Based Method

3.1 Mechanism Design

In designing efficient, centralized (with input from individual agents) or distributed algorithms and network protocols, the computational agents are typically assumed to be either *correct/obedient* or *faulty* (also called adversarial). Here agents are said to be *correct/obedient* if they follow the protocol correctly; agents are said to be *faulty* if (1) they stop working, or (2) they drop messages, or (3) they act arbitrarily, which is also called *Byzantine failure*, i.e., they may deviate from the protocol in arbitrary ways that harm other users, even if the deviant behavior does not bring them any obvious tangible benefits.

In contrast, economists design market mechanisms in which it is assumed that agents are *rational*. The rational agents respond to well-defined incentives and will deviate from the protocol only if it improves their gain. A rational agent is neither correct/obedient nor adversarial.

Notice that, besides correct/obedient, adversarial, and rational agents, there is another set of agents, called *irrational*, which behave strategically but do not follow a behavior modelled by the mechanism designer. They behave irrationally with respect to the mechanism, e.g., they may have utility functions depending on more than just their own preferences. Another example is that some agents may be unable to act rationally if the strategy calculation is too expensive.

In this chapter, we always assume that the agents are rational. In addition, the mechanism used in this chapter is not computationally expensive.

A standard economic model for analyzing scenarios in which the agents act according to their own self-interest is as follows.

1. 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 or its willing payment for a good in an auction environment. The type vector $t = (t^1, t^2, \cdots, t^n)$ of these agents is called a *profile*.

2. Each agent i has a set of strategies A^i that it can choose from. 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 $p = (p^1(a), \cdots, p^n(a))$. Here the payment $p^i(a)$ is the money given to each participating agent i under strategy vector a. If $p^i(a) <$

0, it means that the agent has to pay $-p^i(a)$ to participate in the action.

3. Agent i has preference given by a valuation function v^i that assigns a real number $v^i(t^i, o)$ 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 except the type t^i, which is known to agent i only.

4. For Agent i's *utility* is $u^i = v^i(t^i, o) + p^i$. By assumption of rationality, agent i always tries to maximize its utility u^i.

3.2 Truthful Mechanism Design

A mechanism is *strategy-proof* or *Truthful* if the types are part of the strategy space A^i and each agent maximizes its utility by reporting its type t^i as input *regardless* of what other agents do. We will focus our attention on the truthful mechanism in the rest of our chapter.

The following are some natural constraints which any truthful mechanism must satisfy, before that we introduce a notation that will be used very often in the following sections.

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 and the agent i uses strategy b.

1. **Incentive Compatibility (IC)**: For strategy-proof mechanism, the payment function should satisfy the incentive compatibility, i.e., for each agent i,

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

In other words, revealing the type t^i is the *dominating strategy.* If the payment were computed by a strategyproof mechanism, agent i would have no incentive to lie about its type because its overall utility would be no greater than it would have been if he had told the truth.

2. **Individual Rationality (IR)**: It is also called Voluntary Participation. Every participating agent must have non-negative utility, i.e.,

$$v^i(t^i, o(a|^i t^i)) + p^i(a|^i t^i) \geq 0.$$

Notice that here an agent is guaranteed to have non-negative utility if it reports its type truthfully no matter what other agents do.

3. **Polynomial Time Computability (PC)**: All computation, the computation of the output and the payment, is done in polynomial time.

3.3 VCG Based Mechanism

Arguably the most important positive result in mechanism design is what is usually called the generalized Vickrey-Clarke-Groves (VCG) mechanism by Vickrey [9], Clarke [10], and Groves [11]. The VCG mechanism applies to mechanism design maximization problems where the objective is to maximize the sum of all agents' valuations and the set of possible outputs is assumed to be finite.

A maximization mechanism design problem is called *utilitarian* if the function $g(o,t)$(also called *objective function*) to be maximized satisfies $g(o,a) = \sum_i v^i(a^i,o)$. A direct revelation mechanism $m = (o(a), p(a))$ belongs to the VCG family if (1) the output $o(a)$ computed based on the type vector a maximizes the objective function $g(o,a) = \sum_i v^i(a^i,o)$, and (2) the payment to agent i is

$$p^i(a) = \sum_{j \neq i} v^j(a^j, o(a)) + h^i(a^{-i}).$$

Here $h^i()$ is an arbitrary function of a^{-i} and different agent could have different function $h^i()$ as long as it is defined on a^{-i}. It is proved by Groves [11] that a VCG mechanism is truthful. Green and Laffont [12] proved that, under mild assumptions, VCG mechanisms are the only truthful implementations for utilitarian problems.

An output function of a VCG mechanism is required to maximize the objective function. This makes the mechanism computationally intractable in many cases. Notice that replacing the optimal algorithm with non-optimal approximation usually leads to untruthful mechanisms. In their seminal paper on algorithmic mechanism design, Nisan and Ronen [13] add computational efficiency to the set of concerns that must be addressed in the study of how privately known preferences of a large group of selfish agents can be aggregated into a "social choice" that results in optimal allocation of resources.

Similar to the *utilitarian* mechanism design problem, a maximization mechanism design problem is called *weighted utilitarian* if there exists positive real numbers $\beta_1, \cdots, \beta_n$ such that the objective function is $g(o,a) = \sum_i \beta_i \cdot v^i(a^i,o)$. A direct revelation mechanism $m = (o(a), p(a))$ belongs to the *weighted VCG family* if (1) the output $o(a)$ computed based on the type vector a maximizes the objective function $g(o,a)$, and (2) the payment to agent i is $p^i(a) = \frac{1}{\beta_i}\sum_{j\neq i}\beta_j \cdot v^j(a^j, o(a)) + h^i(a^{-i})$. Here $h^i()$ is an arbitrary function of a^{-i}. It is proved by Roberts [14] that a weighted VCG mechanism is truthful.

3.4 Network Model

We consider a set $Q = \{q_0, q_1, \cdots, q_{n-1}\}$ of n wireless nodes. Here q_0 is used to represent the access point (AP) of the wireless network to the wired network if it presents. Let $G = (Q, E)$ be the directed communication graph defined by Q, where E is the set of links (q_i, q_j) such that the node q_i can communicate directly to the node q_j. We assume that the graph G is node bi-connected. In other words, we assume that, the remaining graph, by removing any node q_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 q_j can receive the signal from another node q_i if node q_j is within the transmission range of the sender q_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 q_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 q_i. In the terminology of economic theory, c_i is the type of node q_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. Based on this network model, we will address two important routing problems– Unicast and Multicast in the following two subsections.

3.5 Unicast

3.5.1 Statement of Problem

If a node q_i wants to send data to the access point q_0, typically, the path with minimum total relaying cost from node q_i to node q_0 under profile c is used to route the packets. We call this path Least Cost Path (LCP) and denote it as $\mathbf{L}CP(c, i, 0)$. Consider a (directed) path $\Pi(i, 0) = q_{r_s}, q_{r_{s-1}}, \cdots q_{r_1}, q_{r_0}$ connecting node q_i and node q_0, i.e., $q_{r_s} = q_i$ and $q_{r_0} = q_0$, and node q_{r_j} can send signal directly to node $q_{r_{j-1}}$. 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 q_i pays some nodes of the network to forward the data for node q_i to the access point. Thus, each node q_j 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 q_i computes the least cost path $\mathbf{L}CP(d,i,0)$ according to the declared cost vector $d = (d_0, d_1, \cdots, d_{n-1})$. For each node q_j, node q_i computes a payment $p_i^j(d)$ according to the declared cost vector d. The *utility*, in standard economic model, of node q_j is $u_j = p_i^j(d) - c_j$. We always assume that the wireless nodes are rational: it always tries to maximize its utility $u_j = p_i^j(d) - c_j$.

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 q_i wants to send s packets to the access point q_0 in one session, then the actual payment of q_i to a node q_k will be $s \cdot p_i^k$.

If the payment scheme is not well-designed, a node q_j may improve its utility by lying its cost, i.e., declares a cost d_j such that $d_j \neq c_j$. Our objective is then to design a payment scheme such that each node q_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 with each other to improve their utility.

3.5.2 Pricing for Unicasting

For unicast problem, the output function $o(c)$ is just the LCP connecting q_i and q_0. The valuation $v^j(c_j, o(c))$ of a node q_j on the output $o(c)$ is $-c_j$ if node q_j is on the path and 0 otherwise. In other words, if node q_j is on the path, then node q_j will incur a cost c_j to carry the transit traffic for node q_i. We require that the pricing mechanism be strategyproof and nodes carrying no transit traffic receive no payment. Node q_i always prefers to find a path that maximizes the total valuation of all nodes, i.e., to find a path with the minimum total cost. In other words, given a path $\Pi(c,i,0)$, the objective function is $\sum_{j\in\Pi(c,i,0)} c_j = \sum x_j(c,i,0)\cdot c_j$, where $x_j(c,i,0) = 1$ if node j belongs to the path $\Pi(c,i,0)$ and $x_j(c,i,0) = 0$ otherwise. The payment $p_i^k(c)$ to a node q_k on the LCP from q_i to node q_0 by node q_i is

$$p_i^k(c) = x_k(c,i,0)c_k + [\sum_{j=1}^{n-1} x_j(c|^k\infty, i, 0)c_j - \sum_{1}^{n-1} x_j(c,i,0)c_j]. \qquad (1)$$

Here $\sum_{j=1}^{n-1} x_j(c|^k\infty, i, 0)c_j$ is the cost of LCP without q_k and $\sum_{j=1}^{n-1} x_j(c, i, 0)c_j$ is the cost of LCP using q_k.

This payment $p_i^k(c)$ can then be interpreted as follows: the payment to a node q_k in the LCP equals to c_k, plus the improvement of the least cost path from q_i to q_0 due to the existence of node q_k. Notice that if node q_k does not belong to the least cost path, clearly, its presence does not improve the cost of LCP, thus its payment is 0. From now on, we use the term q_k*-avoiding-path* to refer to a path that does not pass through node q_k, and denote the least cost such path by $\mathbf{L}CP^{-k}(c, i, 0)$. Let $c(i, 0)$ be the cost of $\mathbf{L}CP(c, i, 0)$ and $c^{-k}(i, 0)$ be the cost of the least cost q_k*-avoiding-path* $\mathbf{L}CP^{-k}(c, i, 0)$. Notice this payment falls into the VCG mechanism, so it is strategy-proof.

3.5.3 The Distributed Algorithm

In the previous subsection, we presented a strategyproof pricing mechanism for unicast routing. Now we focus our attention on how to compute this price p_i^k in a distributed manner.

The algorithm has two stages. First, all nodes together find the Shortest Path Tree T rooted at q_0. Second, every node q_i computes its payment p_i^k in a distributed manner which is based on the algorithm in Feigenbaum *et. al* [15].

In the first stage, the shortest path tree T rooted at q_0 can be easily implemented using Dijkstra's algorithm, so we omit the details of the implementation here. In the second stage, based on the tree T found in the first stage, every node knows its parent and children in tree T. Initially at node q_i, entry p_i^k is set to ∞, if q_k is on $\mathbf{L}CP(c, j, 0)$; otherwise, p_i^k is set to 0. Every node now broadcasts its p_i^k to its neighbors. When a node q_i receives an updated price from its neighbor q_j, it updates the price entries as follows:

1. If q_j is the parent of q_i, node q_i updates
$$p_i^k = \min(p_i^k, p_j^k) \text{ if } q_k \in \mathbf{L}CP(c, i, 0).$$
2. If q_i is the parent of q_j, node q_i updates
$$p_i^k = \min(p_i^k, p_j^k + c_i + c_j) \text{ if } q_k \in \mathbf{L}CP(c, i, 0).$$
3. If nodes q_i and q_j are not adjacent in tree T, for every $q_k \in \mathbf{L}CP(c, i, 0)$, node q_i updates
$$p_i^k = \min(p_i^k, p_j^k + c_j + c(j, 0) - c(i, 0)) \text{ if } q_k \in \mathbf{L}CP(c, i, 0),$$

$$p_i^k = \min(p_i^k, c_k + c_j + c(j,0) - c(i,0)) \text{ if } q_k \notin \mathbf{L}CP(c,i,0).$$

Whenever any entry p_i^k changes, the entry p_i^k is sent to all neighbors of q_i by node q_i. When the network is static, the price entries decrease monotonically and converge to stable values after finite number of rounds. Notice that, here we assume that all nodes will forward these control messages, used to calculate the payment later, for free.

3.5.4 Truthful Implementation

The algorithm presented in the previous section is simple and efficient, but notice that this algorithm relies on the selfish node q_i to calculate the payment p_i^k for itself, which cannot prevent node q_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. Further more, their method only applies to the *edge weighted graph*. Thus, in this section, we focus our attention on how to design a new distributed algorithm based on the previous algorithm to guarantee truthful price calculation. The following method is a review of approach [16].

While it is quite obvious to conceive that the node q_i has the incentive to not correctly calculate its payment p_i^k in the second stage, it is not so straightforward to notice that the node q_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 the node q_i calculates his payment correctly in the second stage, it is still necessary for us to worry about nodes' lying in the first stage.

In Figure 1, the shortest path between q_0 and q_1 should be $q_0q_2q_3q_4q_1$, it is easy to compute the payments of node q_1 to nodes q_2, q_3, and q_4 are both 2. Thus, its overall payment to send a packet to q_0 is 6. However, if node q_1 lies that it is not a neighbor of q_4, then the shortest path becomes $q_0q_5q_1$. Now node q_1 only needs to pay q_5 to send the packet and the payment is 5. Consequently, node q_1 benefits from lying about the connection of the network. This rises from the fact that the least cost path doesn't necessary to be the path that you pay the least. This example also shows that *there is no truthful mechanism for directed edge weighted graph* when we assume that the nodes are agents since each node can choose which links to report to minimize its total payment.

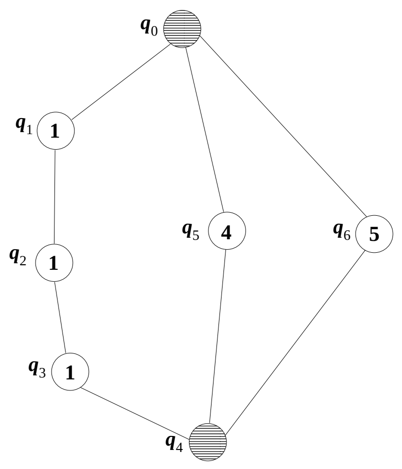


Figure 1: The node has the incentive to lie about its shortest path

We then modify the first stage of the algorithm as follows.

Algorithm 1 ***Modified Distributed Algorithm***

First Stage:

1. For every node q_i, it has two entries: $Dis(q_i)$ which stores the shortest distance to q_0 and its corresponding first hop neighbor $FH(q_i)$ on the least cost path. Initially, if q_0 is q_i's neighbor then set $Dis(q_i)$ to 0 and $FH(q_i)$ to q_0; else set $Dis(q_i)$ to ∞ and $FH(q_i)$ to NULL. Every node broadcasts its information to its neighbors.

2. For every node q_i, when it receives a broadcasting information from its neighbor q_j, first it compares $Dis(q_i)$ with $Dis(q_j) + c_j$: if $Dis(q_i) > Dis(q_j) + c_j$ then sets $Dis(q_i)$ to $Dis(q_j) + c_j$ and $FH(q_i)$ to q_j. After that it compares q_i and $FH(q_j)$:

   (a) Case 1: q_i is not $FH(q_j)$. If $Dis(q_i) + c_i < Dis(q_j)$ then node q_i contacts q_j directly using a reliable connection, asking q_j to update $Dis(q_j) = Dis(q_i) + c_i$ and $FH(q_j) = q_i$. After updating, node q_j should rebroadcast this information.

   (b) Case 2: q_i is $FH(q_j)$. If $Dis(q_i) + c_i \neq Dis(q_j)$ then node q_i contacts q_j directly using reliable connection, asking q_j to update

$Dis(q_j) = Dis(q_i) + c_i$ and $FH(q_j) = q_i$, after that q_j should rebroadcast this information.

Second Stage:

1. When q_i receives a broadcasting information p_j^k from its neighbor q_j, it updates the p_i^k using the payment updating algorithm presented in subsection 3.5.3. Additionally, if q_i triggers the change for p_j^k, it should recalculate p_j^k for node q_j using the payment updating algorithm in previous section to verify it. If it is not correct, then node q_i notifies node q_j and other nodes.

2. For every node q_i, when its entry for p_i^k changes, it not only broadcasts the value of p_i^k, but also broadcasts the information of the node that triggers this change.

It has been shown in [16] that the above approach can prevent nodes from misreporting its link information and its cost, and miscalculating the payment.

3.5.5 Collusion

Using the standard assumption from economic model, we assumed that the wireless nodes do *not* collude to improve their utility. But in practical situation, the collusion could happen very often and much disaster than the single node lying case. For example, if two nodes q_{k_1} and q_{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). So the collusion of nodes discussed here is different from the traditional *group strategyproof* concept studied in [17, 18]. A pricing mechanism is said to be group strategyproof in [17, 18] 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 we described before. We say that a mechanism is *k-agents strategyproof* if, when any subset of k agents 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 unicasting routing problem studied here: if all nodes but node q_i collude and declare arbitrarily high cost, then node q_i has to pay a payment arbitrarily higher than the actual payment it needs to pay if these nodes do not collude. Thus, it is interesting to design some mechanism that is *k-agents strategyproof* for some small integer k. Clearly, a *k-agents strategyproof* mechanism exists only if the underlying network topology is at least $k+1$ node connected.

For *k-agents strategyproof* problem, it is usually divided into two general categories: anonymous *k-agents strategyproof* problem and specific *k-agents strategyproof* problem. For the first category, what we only know is that k agents will collude, but we don't know exactly which the k agents are. It was conjectured that ever for $k = 2$, there are no strategy-proof mechanism, which means that two nodes can collude together and ask for arbitrary high price. So usually, we focus our attention on the second category. A simple case is that if we know exact which k agents probably will collude. Similar to finding the q_k-avoiding path, we can find a path that avoid these k nodes. It is easy to verify this method is strategy-proof. More sophisticated cases are these collusion nodes have some special property, for example, they should be neighbors in the wireless ad hoc. It is an open problem whether can design a *k-agents strategyproof* when knowing that possible colluding nodes are neighbors of each other.

3.5.6 Over Payment

Remember that the payment of a node q_i to a node q_k on the $\mathbf{L}CP(c, i, 0)$ is

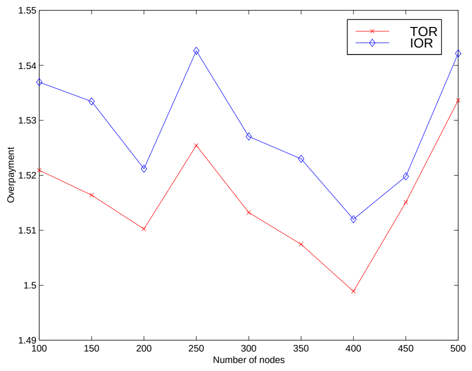
$$c_k + [\sum_{j=1}^{n-1} x_j(c|^k\infty, i, 0)c_j - \sum_{j=1}^{n-1} x_j(c, i, 0)c_j].$$

Clearly, node q_i overpays the nodes on the $\mathbf{L}CP(c, i, 0)$ to make sure that they will not lie about their costs. The overpaid value is the value of $\sum_{j=1}^{n-1} x_j(c|^k\infty, i, 0)c_j - \sum_{j=1}^{n-1} x_j(c, i, 0)c_j$. In theory, it is not difficult to construct a network example such that the over-payment of a node q_i could be arbitrarily large. But in practice, after conducting extensive simulations when the cost of each node is chosen independently and uniformly from a range and the network topology is a random graph, we find out that the over-payment is small compared to the cost of LCP.

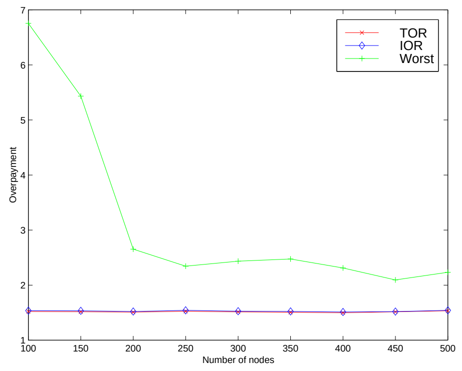
The metrics of the overpayment used in our simulations are *Total Overpayment Ratio* (TOR), *Individual Overpayment Ratio* (IOR), and *Worst Overpayment Ratio* (WOR). The TOR of a graph is defined as $\sum_i p_i / \sum_i c(i, 0)$, i.e., the total payment of all nodes over the total cost of all LCPs. The IOR of a graph is defined as $\frac{1}{n}\sum_i p_i/c(i,0)$, i.e., the average overpayment ratio over all n nodes. The worst overpayment ratio is defined as $\max_i p_i/c(i,0)$, i.e., the maximum overpayment ratio over all n nodes. Remember that here p_i is the total payment of node q_i to all nodes on the LCP from q_i to q_0 and $c(i,0)$ is the total cost of nodes on the LCP from q_i to q_0. We found that the IOR and TOR are almost the same in all our simulations and they take values around 1.5. In all of 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 q_i to forward a packet to another node q_j is $\|q_iq_j\|^\kappa$ where κ varies between 2 and 2.5. The number of nodes in our simulations varies among 100, 150, 200, $\cdots$, 500. Figure 2 (a) illustrates the difference between IOR and TOR when graph model is UDG and $\kappa = 2$. We found that the values of IOR and TOR are almost the same and both of them are stable when the number of nodes increases. Figure 2 (d) illustrates the overpayment with respect 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 IOR and TOR indeed increase when the hop distance to the source increases. Figure 2 (b) and (c) illustrate the overpayment for UDG graph when $\kappa = 2$ and $\kappa = 2.5$ respectively.

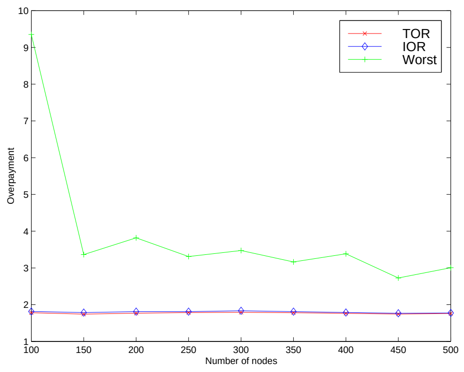
In our second set of simulations, we vary the transmission range of each wireless node from $100m$ to $500m$, and the cost $c_{i,j}$ of a node q_i to send a packet to another node q_j within its transmission range is $c_1 + c_2\|q_iq_j\|^\kappa$, where c_1 takes value from 300 to 500 and c_2 takes value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



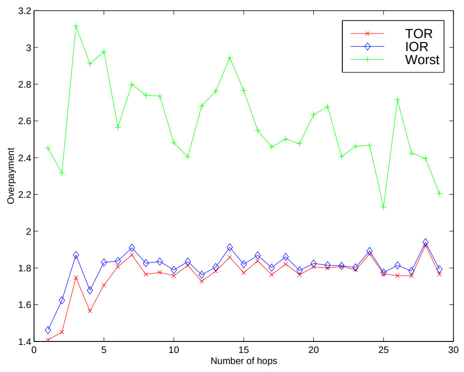
(a) IOR and TOR's difference for UDG with $\kappa = 2$



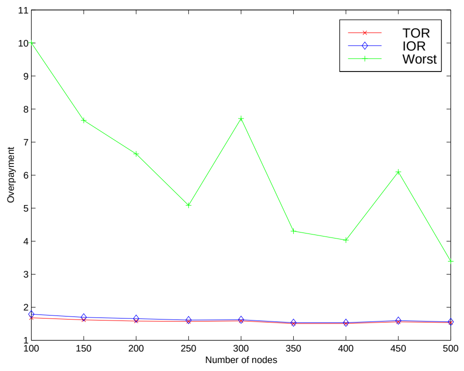
(b) IOR, TOR and WOR for UDG with $\kappa = 2$



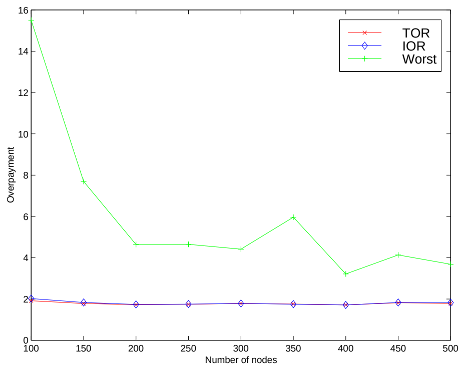
(c) IOR, TOR and WOR for UDG with $\kappa = 2.5$



(d) the affect of hops on IOR, TOR and WOR



(e) IOR, TOR and WOR for random graph with $\kappa = 2$



(f) IOR, TOR and WOR for random graph with $\kappa = 2.5$

Figure 2: Overpayment ratios IOR, TOR and WOR for UDG and random graphs.

3.6 Multicast

3.6.1 Statement of problem and related works

Assume that there is a set of users $R \subset Q$ that wants to receive information from the access point q_0. Each receiver node q_i from R has a valuation $v_i \geq 0$ of receiving the information, and which is the actual payment the node is willing to pay to receive the information. In addition, each node $q_i \in Q$ in the network has a cost c_i to forward data packets for other node.

Assume that each node q_i declares a valuation $w_i \geq 0$ if it receives the information and a cost $d_i \geq 0$ for forwarding the data to the access point q_0. The access point q_0 will then decide a subset of nodes R' to receive the information, compute a multicast tree spanning this set of nodes R', and compute a payment p_i for each node q_i. Notice that, if $p_i > 0$, then we say that the access point *pays* the node q_i for forwarding the data packets; if $p_i < 0$, then we say that the access point *charges* the node q_i for receiving the data packets. Clearly, we should have non-receiver relay nodes get positive payments. Notice that a receiver node may misreport its valuation of the data also.

The above multicast question is different from the question studied by Feigenbaum *et. al* [19]. They assumed that there is a multicast infrastructure, given any set of receivers $R \subset Q$, connects the source node to the receivers. Additionally, for each user $q_i \in R$, they assumed a *fixed* path from the source to it, determined by the multicast routing infrastructure. Then for every set R of receivers, the delivery tree is merely the union of the fixed paths from the source to the receivers R. They also assumed that there is a link cost associated with each communication link in the network and the link cost is *known* to everyone. For each receiver q_i, there is a valuation v_i that this node values the reception of the data from the source. This information v_i is only known to q_i. Node q_i will report a number v_i', which is the amount of money he/she is willing to pay to receive the data. The source node then selects a subset $R' \subset R$ of receivers to maximize the difference $\sum_{i \in R'} v_i' - C(R')$, where $C(R')$ is the cost of the multicast tree $T(R')$ to send data to all nodes in R'. The approach of fixing the multicast tree is relatively simple to implement but could not model the greedy nature of all wireless nodes in the network since it requires that the link costs of the tree are known to every node.

3.6.2 Computational Hardness and Strategy-proof Hardness

The Prize Collecting Steiner Tree problem (PCST) is closely related to the problem of finding a maximum efficiency multicast tree. Given a graph $G = (Q, E)$, a cost vector c for all nodes, a valuation function v for all nodes, and a subset of receiver nodes r, the objective of PCST is to find a tree T, which minimizes $PC(T) = c(T) + v(\overline{T})$, where $v(\overline{T})$ is the sum of valuations of nodes not in the tree T. Let $V = \sum_{i \in N} v(i)$. For any tree T, we have $PC(T) = V + c(T) - v(T)$. Since $v(T) - c(T)$ is the revenue of the network by performing the multicast using the tree T, minimizing $PC(T)$ is equivalent to maximizing the total revenue of the multicast. It is well known that although the PCST problem can be approximated within $2 - 1/n$, but the revenue maximizing multicast problem is NP-hard, and it cannot be approximated within any constant factor.

For some specific problems, there exist some constant approximation algorithms. If q_0 should send data to all nodes in R regardless of the value of w_i, the problem become the Minimum Steiner Tree Problem: finding a minimum cost tree spanning all receivers. It is well-known that finding the minimum cost Steiner tree the both general node weighted and edge weighted graphs are NP-Hard, and even in the Euclidean or rectilinear metrics [20]. There are several polynomial time approximation algorithm presented in [21, 22, 23, 24].

In [25], they gave an approximation algorithm with the best known approximation ratio approaching $1 + \frac{ln3}{2} \approx 1.55$. However, this heuristic may be not practical for ad hoc wireless networks due to its implementation complexity. Takahashi and Matsuyama [22] gave a simple 2-approximation algorithm for Steiner minimum tree in edge weighted graphs. This algorithm maintains a tree T which initially contains only the source node. At each iterative step, the tree T is grown by one path from T of least cost can reach a destination not yet in T. Such path can be found by collapsing the entire tree T into one artificial node and then applying the single-source shortest-path algorithm. This procedure is repeated until all required nodes are included in T. This algorithm can be regarded as an adaptation of the Prim's algorithm for MST.

Even we can find the polynomial time constant approximation algorithm, it is still difficult to find a strategy-proof mechanism based on this constant approximation algorithm easily. In [13], they have already pointed out that *Replacing the optimal algorithm with a non-optimal approximation usually leads to untruthful mechanisms.* For example, if we using Takahashi and Matsuyama's 2 approximation algorithm and VCG mechanism to calculate the payment: denote T^{-p_k} as the tree without the node p_k and $W(T)$ as the sum of the node weight of this tree, then p_k's payment is $W(T)-W(T^{-p_k})+c_k$. Figure 3 gives an example that a node may lie its cost to improve its utility. Here, q_0 is the access point and r_1, r_2 are receiving nodes. It is easy to calculate that $W(T)=8$ and $W(T^{-p_k})=5$. Thus, the payment to node q^k is $5-8+4=1$, which is less than q^k's true cost 4. This violates the individual rationality (IR).

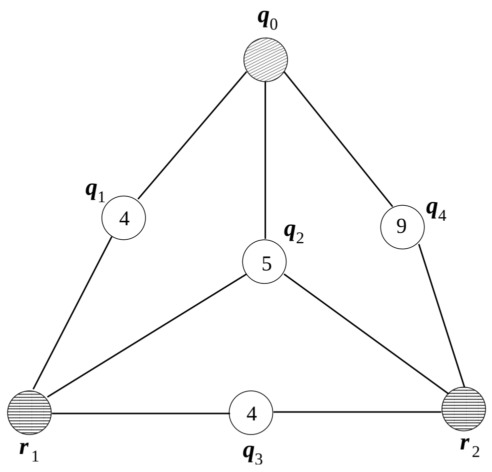


Figure 3: Non-optimal approximation usually leads to untruthfulness

Given any algorithm that approximates the minimum cost spanning tree with a factor α, we may design a payment function

$$\alpha \cdot W(T) - W(T^{-p_k}) + c_k.$$

However, it is unknown whether this payment function will satisfy the IR property. We suspect so and believe that counter-examples can be constructed such that $W(T^{-p_k}) = \alpha \cdot W(OPT^{-p_k})$ and $W(OPT^{-p_k}) > W(T)$. Notice that the first condition can be satisfied since we only have an α-approximation algorithm, and the second condition can be satisfied easily if $W(OPT)=W(T)$ since $W(OPT^{-p_k}) \geq W(OPT)$. In other words, we need design an example such that the *alpha*-approximation algorithm produces the best solution with node q_k and produces the worst solution without node q_k.

3.6.3 Node Weighted receiving relay free UDG graph cases

In this subsection, we study a special case of multicast routing and propose an optimal computable truthful method. We assume that (1) it is node weighted and with $c_i = 0$ if $p_i \in Q$, i.e., all receiver nodes will relay the message for free; (2) all receiver nodes must receive the data; (3) the graph is a UDG graph. The truthfulness of our mechanism actually does not depend on the third assumption. The third assumption only guarantees that the spanning tree (discussed later) found by our method approximates the minimum cost spanning tree with a constant factor. To achieve constant approximation of the minimum cost spanning tree, the last assumption can actually be relaxed to: the underlying communication graph G has a degree bounded spanning tree. We [26] showed that if graph G has a spanning tree with bounded degree Δ_T, then the spanning tree constructed for multicast is Δ_T-approximation of the minimum cost spanning tree. Given a graph G, there is a polynomial time algorithm [27] to find a spanning tree whose degree is at most $\Delta_{OPT} + 1$, where Δ_{OPT} is the minimum degree bound such that graph G has a spanning tree with degree bounded by Δ_{OPT}.

Consider a weighted graph $G = (V, E, c)$, where c represents the cost of a node relaying message for other nodes. There is a set of receivers $Q \subset V$ that want to receive a data from a fixed source node q_0. For the simplicity of notation, we also assume that Q also includes the fixed source node. We then present the algorithm to construct a tree spanning all receivers and its cost is no more than 5 times of the minimum cost.

Algorithm 2 ***Reduction MST Algorithm***

1. First, we calculate the pairwise shortest path $LCP(q_i, q_j, c)$ between any two nodes in $q_i, q_j \in Q$ when the node costs vector is c. Construct a complete graph $K(Q, E')$ using Q as its vertices, and edge $q_i q_j$ corresponding to $LCP(q_i, q_j, c)$, and its weight is the cost of $LCP(q_i, q_j, c)$ in G.

2. Calculate the minimum spanning tree on $K(Q, E')$. The resulting tree is denoted as $RMST(G)$.

For convenience of our analysis, we assume that no two nodes in $G(V, E, c)$ have the same cost, and also there are no two paths in $G(V, E, c)$ with the same length. Dropping this assumption doesn't change the result of our analysis.

Theorem 3.1 *[26] The $RMST(G)$ is a Δ_T-approximation of the minimum cost tree spanning all receivers if G has a spanning tree of degree bound Δ_T.*

Corollary 3.2 *[26] The $RMST(G)$ is a 5-approximation of the minimum cost tree spanning all receivers if G is a unit disk graph.*

Based on the $RMST(G)$ constructed by Algorithm 2, we [26] designed a truthful mechanism for calculating the payment. Before presenting the payment definition, we define some terms first.

If we change the cost of a node $v_k \in V$ to d_k, we denote the new graph as $G|^k d_k$. If we remove one vertex v_k from G, we denote the resulting graph as $G \backslash v_k$. If we apply Algorithm 2 on a graph G after removing a node v_k, we denote the resulting MST as $RMST(G \backslash v_k)$. Obviously, $RMST(G \backslash v_k) = RMST(G|^k \infty)$.

Given a spanning tree T, and a pair of nodes p and q on T, clearly there is a unique path connecting them. We denote such path as $\Pi_T(p, q)$, and the edge with the maximum length on this path as $LE(p, q, T)$. For simplicity, we use $LE(p, q, c)$ to denote $LE(p, q, RMST(G))$ and use $LE(p, q, c|^k d_k)$ to denote $LE(p, q, RMST(G|^k d_k))$.

Now we present the truthful mechanism to calculate the payment.

1. First each node $v_k \in V$ is required to report a cost, say d_k.

2. For every node $q_k \in V \backslash Q$ in G, first calculate $RMST(G)$ and $RMST(G|^k \infty)$ according to the nodes' declared costs vector d.

3. For any edge $e = q_i q_j \in RMST(G)$ and any node $v_k \in LCP(q_i, q_j, c)$, we define the payment to node v_k based on the virtual link $q_i q_j$ as

$$p^k(q_i q_j) = |LE(q_i, q_j, d|^k \infty)| - |LCP(q_i, q_j, d)| + d_k.$$

   Here $|\Pi|$ denotes the total cost of a path Π. If a node v_k is not on $LCP(q_i, q_j, c)$, then the payment $p^k(q_i q_j)$ to node v_k based on the virtual link $q_i q_j$ is 0. If the path $LCP(q_i, q_j, c)$ is not used in $RMST(G)$, then the payment to any node on path $LCP(q_i, q_j, c)$ based on edge $q_i q_j$ is also 0. The final payment to node v_k based on RMST is

$$p^k(d) = \max p^k(q_i q_j).$$

It is proved in [26] that this algorithm is not only truthful, but also it is optimal regarding the individual payment among all these truthful mechanisms based on the spanning tree $RMST$. For details of this algorithm please refer to [26].

3.6.4 Sharing Cost and Payment

Under some circumstance, if we have fixed the architecture of the multicast tree, it is also not trivial to design a *reasonable* cost-sharing mechanism to determine which users receive the transmission and how much they are charged. Here *reasonable* means at least:

1. Receivers cannot be charged more than what they are willing to pay.
2. The transmission costs of shared network links cannot be attributed to any single receiver.
3. The source node would not broadcast if the total payment received from the receiving node is less than what it should pay the relaying nodes (or links).

This problem has been formalized as the multicast cost-sharing problem(MCSP): For a graph $G = (V, E)$ and a tree T spanning the receiving nodes R, each receiving node has a utility u_i for receiving the information and known only to itself, so it can declare his utility as $u_i' \neq u_i$. Every internal node in the tree has a cost c_i to relay the data, so the access point q_0 should pay these nodes for relaying transit traffic. We let $x_i \geq 0$ denote how much user i is charged and σ_i denote whether user i receives the transmission; $\sigma_i = 1$ if the user receives the multicast transmission, and $\sigma = 0$ otherwise. We use u to denote the input vector $(u_1, u_2, \cdots, u_{|R|})$. The mechanism M is then a pair of functions $M(u) = (x(u); \sigma(u))$. The receiver set for a given input vector is $R(u) = \{i|\sigma_i = 1\}$. A user's individual welfare is given by $w_i = \sigma_i u_i - x_i$, The cost of the tree $T(R(u))$ reaching a set of receivers $R(u)$ is $W(T(R(u))) = \sum_{v_i \in T(R(u))} c_i$, and the overall welfare, or net worth, is $NW(R(u)) = \sum_{i \in R(u)} u_i - w(T(R(u)))$.

The goal of MCSP problem is to find a strategy-proof mechanism $M(u)$ subject to:

1. **No Positive Transfers (NPT)**, which means that the mechanism cannot pay receivers to accept the transmission,
2. **Voluntary Participation (VP)**, which means that no receiver can be forced to pay more than what it is willing to pay.
3. **Efficiency**: a configuration that will maximize $NW(R(u))$.

It may also has some additional desirable properties like:

1. **Consumer Sovereignty:** A receiver is always able to guarantee acceptance of the information if his price is increased to a sufficiently large value.

2. **Budget Balance:** the amount paid by the receiver exactly equals the cost of transmission.

The multicast cost-sharing problem has been studied extensively in recent years, first from a networking perspective in [28], then from a mechanism-design perspective [29], and most recently from an algorithmic perspective [19], [30], [31], [32].

Two mechanisms can be used to solve this problem: marginal cost (MC) and Shapley value (SH). MC mechanism satisfies strategy-proof, NPT, VP, CS and can be computed by a simple, distributed algorithm that uses only two modest-sized messages per link of the multicast tree in [19], [30], but one drawback is that it is not budget balanced, which means that sometimes it will have a budget surplus and sometimes it will have a budget deficiency. Shapley Value is to share the node q_i's cost within all its downstream receiving nodes. It is budget balanced and group-strategyproof and, among all mechanisms with these two properties, minimizes the worst-case welfare loss. But the SH method has a bad network complexity, computing the SH mechanism requires, in the worst case, that $O(|P|)$ bits be sent over $O(|N|)$ links, where P is the set of potential receivers, and N is the set of tree nodes.

The other interesting question is when the multicast infrastructure is not fixed, as the questions studied in previous subsection, and the receivers have to pay some other nodes to get data from the source. The mechanism described in the previous subsection provides a payment scheme to relay nodes such that they will not lie about their relay costs, but did not specify how the payment will be shared by all the receivers. We would like to design a payment sharing method such that it is better for each individual receivers to use multicast than to use unicast individually. In other words, the payment shared by a receiver node q_k should be no more than the total payment of receiver q_k when it uses unicast to connect with the source node.

4 Other problems

Besides wireless ad hoc network, game theory has been used extensively in computer science, we briefly discuss some of the applications in this section.

4.1 Non-cooperation of Topology Control

In wireless ad hoc networks, usually the nodes can adjust their transmission ranges to achieve some desired properties, which is known as the topology control. Several topology control issues in a non-cooperative environment have been addressed before in [33]. In order to meet the connectivity requirement, we are given node pairs $(s_1, t_1), \cdots, (s_k, t_k)$, and each s_i needs to connect to t_i. Each node s_i has to choose a radius so that it gets t_i while keeping the radius as small as possible. If the radius of s_i cannot reach the target t_i, it relies on some other nodes s_j to relay the message and it is assumed that these nodes on the chosen path will relay. Notice that node s_i's only purpose is to connect to t_i, so it wouldn't care about whether other nodes can connect to their destinations or not. But the complication of the problem comes from the fact that the path connecting the source and target may contain several intermediate nodes. If a node enlarges its transmission range to connect more nodes, it is possible that it will have more choices for the intermediate nodes, which will in turn result in a smaller overall energy expenditure. Modelling the cost of a radius vector $\overline{r}$ for all nodes as $C(\overline{r}) = \sum_v r_v^\alpha$ where α is a constant between 2 and 5, they define the utility as $U(v) = f_{\overline{r}}(v) - r_v^\alpha$ where $f_{\overline{r}}(v)$ denoted the number of vertices w that v can reach. Their goal is to find a Nash Equilibria in this game, but unfortunately the existence of the Nash Equilibria and even approximate one is not guaranteed, the figure below is a graph falling into this category.

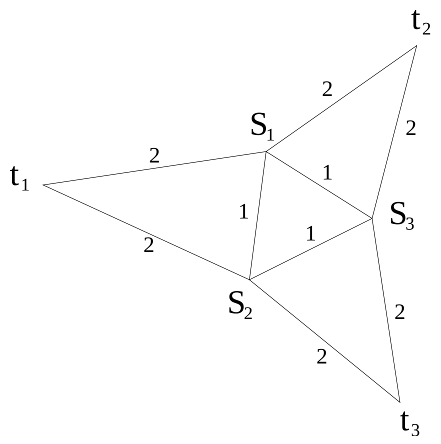


Figure 4: No Nash Equilibria

There are lots of other issues left untouched in this category, including min-power assignment problem, k-connectivity (node or edge) problem, undirected path problem and connected dominate set problem.

4.2 Incentives for Cooperation in Peer-to-Peer Networks

Peer-to-peer (P2P) file-sharing systems combine sophisticated searching techniques with decentralized file storage to allow users to download files directly from one another. The first mainstream P2P system, Napster, attracted public attention for the P2P paradigm as well as tens of millions of users for itself. Now P2P networks become a new platform for distributed applications, allowing users to share their computational, storage, and networking resources with their peers to the benefit of every participant. Most p2p system designs focus on traditional computer science problems including scalability, load-balancing, fault-tolerance, and efficient routing. While many peer-to-peer systems have implicitly assumed that peers will altruistically contribute resources to the global pool and assist others, recent empirical studies have shown that a large fraction of the participants engage in *freeriding* [34], [35]: 20% to 40% of Napster and almost 70% of Gnutella peers share little or no files [1, 2]. So it has been wildly acknowledged that the P2P file systems should take the user incentives and rationalities into consideration. So some recent literatures have been focus on how to solve this issue.

In [36], they define a game that models the file sharing scenario of P2P networks: n agents $a_1, ..., a_n$ participate in the system. Each agent a_i's strategy, denoted $S_i = (\sigma, \delta)$, consists of two independent actions: σ describes what proportion to share with other users, with σ_0(none), σ_1 (moderate) or σ_2 (heavy); δ determine how much to download from the network in each period. Each user can choosing between three levels: δ_0 (none), δ_1 (moderate) or δ_2 (heavy). Using this model, they propose several payment schemes: Micro-Payment Mechanisms, Quantized Micro-Payment Mechanism. Both schemes use *Nash Equilibrium* to find the agents' rational strategies and involve monetary transfer.

In [37], they use a rating scheme whereby a user is given a level in the P2P system to alleviate the free-rider problem. They discuss several issues of how the rating system should be, and based on the idea that when a user a receives a request from a user b, it uses user a's reputation to decide whether he will provide the service or not to. They give two distributed rating scheme to incentivize cooperation: *Structured Verification Scheme* (SVS) and *Lightweight Unstructured Verification Scheme* (LUVS). In SVS Scheme, every user i should have a supervisor and the supervisor should be chosen so there won't be easy collusion. (They use the ring structure *Chord* [38] to achieve that). User b gets user a's information and updates user a's reputation involving user a's supervisor. In LUVS scheme, every user will keep a list of the customers she has served and a list of servers from whom she has been served, along with the details of the transactions. Now, when a user a wants some service from a user b, a sends the list of its customers along with the request. If user b decides to verify a's rating, it samples a subset of a's customer list to confirm if they have received the claimed service from a or not. If most members in this sample say a yes, b trusts a and provides service depending on the rating of a. But they also pointed out that these schemes may suffer from the collusion problem.

4.3 Resource allocation

With the advances in computer and networking, especially the Internet, thousands of heterogeneous computers have been interconnected to provide a great resource including computing, storage, etc. Because of the heterogeneity of these resources, it is challenging to design an architecture to accommodate all. Further more, in many cases, the resources are owned by multiple organizations and it is often required to allocate the limited resource to maximize the total user satisfaction (called *Social Choice*).

Resource allocation problem has been studied in human economies for a long time. In economic model of a computer system, the consumers are applications such as web clients, computational tasks, multimedia entertainment consumers, and ISP users. The suppliers are these computer systems who control the resources like CPU time, memory, cache, disks, network bandwidth, etc. Suppliers control access to their resource via prices, and consumers buy resources from the suppliers to satisfy their needs. The price is decided by a way similar to that in economic world: the demand and supply curve.

4.4 Cooperation in MAC Layer

Wireless MAC protocols such as IEEE 802.11 use cooperative contention resolution mechanisms for sharing the channel, it is usually based on a fully distributed mechanism to control the access to the network. For example, in the CSMA protocol, a wireless node senses whether the channel idle every DIFS (Distributed Inter Frame Space) seconds. If the channel is idle, then the node transmits the frame. Otherwise, it does a binary back-off with NAV (Network Allocation Vector) seconds which specifies the minimal time of deferral. This mechanism guarantees the fair use of the bandwidth if all nodes conform to this protocol.

But some nodes can modify their behavior in order to gain some advantage over other nodes. In stead of doing a binary back-off with NAV when collision is detected, some nodes can keep the DISF a constant in order to gain more chances to transmit a package. If one node or a few nodes play this trick, other nodes will suffer from unfair bandwidth share. The more disastrous scenery happens when the majority of the nodes or all nodes play this trick, which would result in a situation that every node (including these nodes playing this trick) share a much lower bandwidth than when no nodes play this trick. Thus, some truthful methods must be designed to deal with cooperation issue in the MAC layer also.

4.5 Cooperation for TCP/IP on end-node

For years, the conventional wisdom has been that the continued stability of the Internet depends on the widespread deployment of "socially responsible" congestion control. But what about if network end-points behaved in a selfish manner? Under this assumption, each flow attempts to maximize the throughput it achieves by modifying its congestion control behavior. In [39], they use a combination of analysis and simulation to determine the Nash Equilibrium of this game. They also addressed the efficiency of the network operating at these Nash equilibria.

Unlike the MAC layer scenery where the protocol contained in some hardware, which are very hard to manipulate, TCP/IP stacks in some operating systems like Linux, FreeBSD, is very easy to modify. So it is more desirable to design some truthful methods to prevent the misbehavior from happening in the TCP/IP layer.

5 Conclusion

In this Chapter, we assume that all wireless nodes are possibly owned by individual users and the users are able to modify the algorithms deployed on them for the sake of their own interests. We also assume that each wireless node has a cost to forward the data for other nodes and a node will only relay the data if it got a payment to cover its relay cost. We studied how the source node can design a payment scheme to all relay nodes such that the relay nodes have to report its cost truthfully to maximize their profits both for the case of unicast and the case of multicast. We also discussed the selfishness of wireless nodes in other layers including MAC layer, TCP/IP layer, and application layer.

References

[1] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao, "Cooperation in wireless ad hoc 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(MONET)*, vol. 8, no. 5, October 2003.

[3] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," in *Proceedings of Seventh International Financial Cryptography Conference (FC)*, 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 *Proc. of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2000.

[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] N. B. Salem, L. Buttyan, J. Hubaux, and M. Kakobsson, "A charging and rewarding scheme for packet forwarding in multi-hop cellular networks," in *Proc. of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2003.

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

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

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

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

[13] N. Nisan and A. Ronen, "Algorithmic mechanism design," in *ACM Symposium on Theory of Computing*, 1999.

[14] K. Roberts, *Aggregation and Revelation of Preferences, edited by J. J. Laffont*, chapter The characterization of implementable choice rules, pp. 321–349, North-Holland, 1979, Papers presented at the 1st European Summer Workshop of the Econometric Society.

[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.

[16] Y. Wang X.Y. Li, W.Z. Wang and O. Frieder, "Truthful low-cost routing in selfish ad hoc networks," 2003, Working paper, submitted for publication.

[17] K. Jain and V. V. Vazirani, "Group strategyproofness and no subsidy via lp-duality," 2002.

[18] H. Moulin and S. 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.

[19] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker, "Sharing the cost of multicast transmissions," *Journal of Computer and System Sciences*, vol. 63, no. 1, pp. 21–41, 2001.

[20] M. R. Garey and D. S. Johnson, "The rectilinear steiner problem is np-complete," *SIAM Journal of Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[21] P. Berman and V. Ramaiyer, "Improved approximations for the steiner tree problem," *J. of Algorithms*, vol. 17, pp. 381–408, 1994.

[22] H. Takahashi and A. Matsuyama, "An approximate solution for the steiner problem in graphs," *Math .Jap.*, vol. 24, pp. 573–577, 1980.

[23] A. Zelikovsky, "An 11/6-approximation algorithm for the network steiner problem," *Algorithmica*, vol. 9, pp. 463–470, 1993.

[24] A. Zelikovsky, "Better approximation bounds for the network and euclidean steiner tree problems," Technical report cs-96-06, University of Virginia, 1996.

[25] G. Robins and A. Zelikovsky, "Improved steiner tree approximation in graphs," in *Proceedings of ACM/SIAM Symposium on Discrete Algorithms*, 2000, pp. 770–779.

[26] W.Z. Wang and X.-Y. Li, "Truthful low-cost multicast in selfish networks," 2003, Working paper submitted for publication.

[27] B. Raghavachari, "Algorithms for finding low degree structures," in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed., pp. 266–295. PWS Publishers Inc., 1997.

[28] S. Herzog, S. Shenker, and D. Estrin, "Sharing the "cost" of multicast trees: an axiomatic analysis," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 847–860, 1997.

[29] H. Moulin and S. Shenker, "Strategyproof sharing of submodular costs: Budget balance versus efficiency," *Economic Theory*, vol. 18, pp. 511–533, 2001.

[30] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, "Hardness results for multicast cost sharing," Tech. Rep. YALEU/DCS/TR1232, June 2002, http://ftp.cs.yale.edu/pub/TR/tr1232.ps.

[31] M. Adler and D. Rubenstein, "Pricing multicast in more practical network models," in *Proceedings of the 13th Symposium on Discrete Algorithms.* 2002, pp. 981–990, ACM Press/SIAM , New York/Philadelphia.

[32] K. Jain and V. Vazirani, "Applications of approximation to cooperative games," in *Proceedings of the 33rd Symposium on the Theory of Computing*, New York ACM Press, Ed., 2002, pp. 364–372.

[33] L. Anderegg and S. Eidenbenz, "Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *Proceedings of the 9th annual international conference on Mobile computing and networking.* 2003, pp. 245–259, ACM Press.

[34] J. Sweeny, "An experimental investigation of the free-rider problem," *Social Science Research*, vol. 2, 1973.

[35] G. Marwell and R. Ames, "Experiments in the provision of public goods: I. resources, interest, group size, and the free-rider problem," *American J. of Sociology*, vol. 84, 1979.

[36] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge, "Incentives for sharing in peer-to-peer networks," *Lecture Notes in Computer Science*, vol. 2232, 2001.

[37] D. Dutta, A. Goel, R. Govindan, and H. Zhang, "The design of a distributed rating scheme for peer-to-peer systems," in *Proceedings of the Workshop on the Economics of Peer-to-Peer System*, 2003.

[38] E. Adar and B. Huberman, "Free riding on gnutella," 2000.

[39] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou, "Selfish behavior and stability of the internet:: a game-theoretic analysis of tcp," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications.* 2002, pp. 117–130, ACM Press.