Autonomous Nodes and Distributed Mechanisms *

John C. Mitchell[1] and Vanessa Teague[2]

[1] Stanford University, Stanford CA 94305, USA,
mitchell@cs.stanford.edu
[2] Stanford University, Stanford CA 94305, USA,
vteague@cs.stanford.edu

Abstract. We extend distributed algorithmic mechanism design by considering a new model that allows autonomous nodes executing a distributed mechanism to strategically deviate from the prescribed protocol. Our goal is to motivate agents to contribute to a global objective and resist disruption by a limited number of malicious irrational agents, augmenting market incentives with cryptographic primitives to make certain forms of behavior computationally infeasible. Several techniques for distributing market computation among autonomous agents are illustrated using a marginal cost mechanism for multicast cost sharing from [3].

1 Introduction

Standard distributed algorithmic mechanism design [1–3] uses a model that separates the computational entities implementing the algorithm from the strategic entities that provide its inputs. In the standard model, there may be several strategic agents who reside at each computational node in the network. The agents provide some input to the node, possibly lying, and the node then faithfully executes the algorithmic mechanism. This model reflects the most important characteristics of market situations, such as satellite television with tamper-resistant receivers in customer homes, in which the mechanism designer has complete control over the hardware and software used to implement the mechanism. However, the standard model omits an important aspect of user behavior in systems such as network routing in which autonomously administered routers may be configured in complex ways to serve the business objectives of their owners. In Internet routing, a system administrator may choose to modify router software if this improves local performance, regardless of whether the modification deviates from published Internet routing standards. Further, malicious or careless administrators may do so in ways that are not even beneficial to themselves (by standard measures). In this paper, we consider the consequences of employing hardware or software that is controlled by strategic agents. We assume that

* This work was supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795 and also ONR MURI grant N00014-97-1-0505, Semantic Consistency in Information Exchange.

strategic agents control the computation at each local node that implements part of a distributed algorithmic mechanism. For simplicity, and to aid comparison between our “autonomous nodes ” model and standard distributed algorithmic mechanism design, we investigate multicast cost sharing, a traditional problem with known “tamper-proof nodes ” distributed solutions [3].

We use a network model that includes mostly selfish agents with some completely honest and a few malicious ones. We consider this a reasonable model for many Internet applications, possibly closer to reality than either the trusting distributed algorithmic mechanism design view or the security view that emphasizes worst-case scenarios. The vast majority of nodes on the Internet today are corporations that make rational decisions designed to maximize their profits. There are a few benevolent nodes (such as universities or government-subsidized sites) and a small number of actively malicious ones.

We focus on the example of the marginal cost mechanism for multicast cost sharing described by Feigenbaum *et al.* [3]. This mechanism shares the cost of a multicast transmission, such as a movie, among a tree of participating nodes. We may think of their distributed algorithm as being run by tamper-proof routers or set-top boxes. We will refer to this protocol as “the FPS protocol.”

If the FPS protocol is implemented naively in the new model where the nodes can implement the algorithm of their choice, then selfish agents can benefit from cheats that are not possible in the model where the nodes must execute the algorithm correctly. They can improve their welfare by telling different lies at different steps in the protocol, lying about what they have received from others or paying the incorrect amount at the end.

The multicast model includes a content provider that initiates the transmission and receives payments from each agent. This provides a convenient central point for providing further economic motivations to the agents. One method we use adds extra authentication to the protocol to allow each agent to “prove” that it behaved honestly. The content provider audits every agent with some probability and fines those who cannot produce a “proof”. The content provider’s computational burden and communication cost can be made constant by increasing the size of the fines with the size of the network. It must do a small number of local checks of one agent at a time. In other applications, where payments may not all reach a central point, we expect the same idea to apply in a more distributed fashion. We present two slightly different authenticated protocols, each with different messages and incentives. The first one is a lighter protocol whose main property is that honesty is an equilibrium – no agent is motivated to deviate from it if it believes that all the others will execute it correctly. Furthermore, as long as agents are selfish and keep to the protocol when there isn’t a strictly better option, all agents execute the protocol correctly. The second protocol contains an additional signed message and is much stronger: we show that keeping to it is strictly more profitable than any alternative.

We examine security by introducing a malicious agent into the system and considering how much it can cause the mechanism to fall below the social optimum. We are also interested in how much such attacks cost, but do not expect the malicious agent to be rational. We show that the FPS scheme in its original model is quite secure against attack by a single agent, but that an unauthenticated implementation is not at all secure in the model where nodes can deviate from the protocol at will. We then show that our strongest authenticated scheme is almost as secure as forbidding the agents to deviate from the protocol, and that the presence of malicious nodes does not cause selfish ones to wish to deviate from the protocol. The only requirement is that all the malicious node's neighbors are honest.

In the next section we describe related work, then give some brief game theory background and an overview of the distributed algorithmic mechanism of [3]. In section 3.1 we describe our model which dispenses with the assumption that the nodes executing the protocol can be trusted to do so correctly. In the following section we give some examples that show that the new model allows new cheating opportunities that can be exploited if we implement the mechanism naively. In section 4 we present and analyze the two authenticated versions of the protocol. In the final section we investigate the security properties of these schemes.

2 Background and Related Work

Algorithmic mechanism design was introduced by Nisan and Ronen in [8]. This involves applying ideas from Game Theory (specifically, mechanism design) to solve algorithmic and network problems. They assume a central mechanism administrator that can communicate freely with the nodes.

A different computational implementation of mechanism design, based on secure function evaluation, is presented in [7]. The mechanism is administered by two agents who are assumed not to collude.

Distributed algorithmic mechanisms are described in [1], [2] and [3]. The mechanism is administered by the network nodes themselves. Several agents inhabit each network node, and may lie to the node, but the network nodes always implement their part of the distributed algorithm correctly. We aim to extend their ideas to the case where the nodes are strategic (aiming to maximize their resident user's profit) and may implement any computationally feasible algorithm.

Feigenbaum *et al.* ([3]) provide and analyze two distributed mechanisms for multicast cost sharing. We will concentrate on the Marginal Cost Pricing mechanism, first presented in [6]. It is one of a class of mechanisms known as VCG mechanisms that have many desirable properties, including that an agent maximizes its welfare by telling the truth. We first review some standard definitions and results from game theory, then describe the centralized algorithm and distributed scheme from [3].

2.1 Game Theory background

Briefly, an agent has a *dominant strategy* if it has one strategy that is a best response to all possible behaviors of others. A mechanism is *strategyproof* if it is a dominant strategy to tell the truth. More detailed background is contained in [8] and [9].

2.2 Feigenbaum et al.'s scheme

This section gives a much-simplified description of the distributed mechanism of [3], in the special case that will be relevant for the rest of this paper.

We are interested in distributing a multicast transmission from a source node to the agents on a network. The network is a tree consisting of a set of nodes and a set of links, in which one agent resides at each node. Each link has a cost known to the nodes at each end. For any set of receivers define the *multicast tree* as the minimal subtree required to reach all those receivers. The cost of sending the transmission to a set of receivers is the sum of the costs of the links in the multicast tree.

Each agent derives some utility from receiving the transmission. The global benefit gained is the sum of the utilities of the receiving agents. We define the *net worth* to be the difference between global benefit and incurred link cost. This is the net benefit of sending the transmission along the multicast tree.

A *mechanism* takes as input a description of the network (including link costs) and a stated utility from each agent. The mechanism's outcome is a set of recipients and a payment vector stating how much each agent should pay the content provider. We assume that each agent may lie to the mechanism about its utility and that each agent seeks to maximize its utility minus its payment.

The mechanism design problem is to find a mechanism with the following two properties:

- Every agent's dominant strategy is to report its true utility to the mechanism (*i.e.* it maximizes its true utility minus payment by doing so).
- The recipient set selected by the mechanism maximizes net worth.

The Marginal Cost Pricing Mechanism is a mechanism for solving this problem. The centralized scheme for implementing it is described below. The distributed scheme (described afterwards) is a distributed algorithm for implementing the centralized scheme.

The centralized scheme The centralized scheme consists of the following steps:

1. Each agent reports its utility to the mechanism.
2. The mechanism computes which agents receive the transmission.
3. The mechanism computes a payment for each agent.

Call a recipient set *efficient* if it maximizes net worth. In step 2, the mechanism selects the largest efficient set, based on stated utilities, and sends the transmission to those agents. The largest efficient set exists because the union of any two efficient sets is itself an efficient set.

To compute payments in step 3, the mechanism first computes for each subtree how much the total agent utility derived exceeds the link cost incurred. This quantity is called the welfare of the subtree. Let u^i be agent i's utility and c^i be the cost of the link joining the node where i resides to its parent in the tree.

Definition 1. *The* welfare W^i *of the subtree rooted at the node where i resides is:*

$$W^i = u^i + \left(\sum_{\substack{j \text{ resides at a} \\ \text{child of } i\text{'s node} \\ \text{and } W^j \geq 0}} W^j \right) - c^i \tag{1}$$

Let the minimum welfare on any path between i's node and the root (inclusive) be A^i. If $A^i < 0$ then agent i does not receive the transmission. Otherwise, it receives it and pays:

$$pay^i = \max(0, u^i - A^i) \tag{2}$$

It is proved in [3] that agent i's net benefit is exactly the marginal contribution to the overall welfare provided by its having a nonzero valuation for the transmission. For a proof that the mechanism is strategyproof, see [6].

The distributed scheme In [3], there is a distributed algorithm for implementing the mechanism described above. First, each agent reports its utility to its node. The nodes then execute a two-pass algorithm at the end of which every node knows whether its resident agent receives the transmission and how much the agent has to pay for it. The payments and recipient set are the same as in the centralized scheme.

The first pass computes all of the W^i bottom-up. Every node sends its welfare to its parent. The parent calculates its welfare using its children's welfares and equation 1.

The second pass is for computing, for each node, the minimum welfare between that node and the root. This can easily be computed top-down: suppose a node has resident agent i and it knows the minimum welfare $A^{\text{parent}(i)}$ of any node between its parent and the root. Then the minimum welfare on any path between i's node and the root is given by:

$$A^i = \min(A^{\text{parent}(i)}, W^i) \tag{3}$$

Payment is a function of A^i and utility u^i, given by equation 2.

This is an efficient solution when the nodes are honest. In the next section we allow the nodes themselves to be strategic, their interests aligned with those of their resident agent. Our proofs of correctness rely on checking equations 1, 2and 3.

3 Cheating in a Different Network Model

3.1 The network model

In many settings where distributed algorithms are run, the strategic agents have control over the computers that are implementing the algorithm. Television viewers may try to modify the set-top boxes in their homes, or the administrators of autonomous computers on a network may alter the networking software for their own ends. Then it is no longer reasonable to assume that each agent reports its utility to its node which then runs the correct algorithm—an agent might alter its node's behavior to improve its own welfare beyond what it could achieve by simply reporting a utility to it.

For the rest of this paper we will identify the game-theoretic agents with their nodes. That is, each node is an agent that has complete control over the messages it sends and receives. It can elect to implement the algorithm as it is instructed by the mechanism designer, or it can implement any other computationally feasible algorithm and send whatever messages it chooses. We need to design an algorithm so that each strategic node's best strategy is to implement the algorithm as instructed.

We will refer to the centralized game-theoretic model used in the first part of section 2.2 as the game theory model, the distributed model where nodes always implement the specified algorithm as the tamper-proof nodes model (this is the model employed in [3] and the second part of section 2.2), and our new model of selfish nodes as the autonomous nodes model.

In this paper we consider just the example of multicast cost sharing presented in [3]. Each node derives some utility from receiving the transmission, which it has to pay for, and its aim is to maximize utility minus price. We will show that in this model a naive implementation of the FPS protocol is susceptible to cheating, and propose authenticated versions that produce the right outcome among selfish agents.

The content provider is also the administrator of the mechanism. This entity receives payments and provides the transmission. It also enforces some of the incentives of the mechanism, including levying fines and providing rewards. Our aim is to design a scheme with minimal communication between the content provider and the nodes.

The next section contains one scenario in which a user can benefit by lying under the autonomous nodes model. This does not contradict the result in [3] that their distributed game is strategyproof. It is based on the observation that the autonomous nodes model allows new ways of cheating that are not possible the tamper-proof nodes model. Two other examples are contained in Appendix A.

3.2 Some examples of agents behaving badly

In this example an agent sends one welfare value to its parent node, then sends a *different* value to its child node. It succeeds in paying one unit less than it should for the transmission and tricking its child into paying one extra.

In order for a deception to be successful, every other agent in the tree must receive a consistent set of messages. For the cheating example shown, we also provide the honest agents' perspective, showing that in each case for each honest agent there is a set of utilities that produces exactly the set of messages received by that agent during the cheat.

3.3 Wrongfully getting the transmission while others pay extra

The true state of the network is shown in Figure 1. Node 0 is the root of the distribution tree and we don't consider its payments in this example. If all agents behave truthfully, then all agents receive the transmission, agent 1 pays 1 and agent 2 pays 6 (see equation 2).

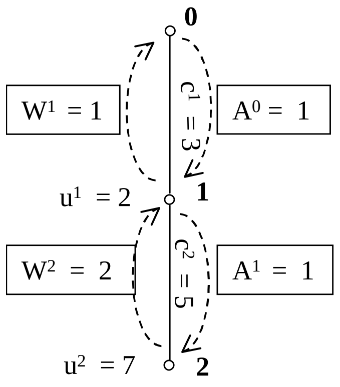

Fig. 1. The truth. $pay^1 = 1$ and $pay^2 = 6$.

Figure 2 shows a successful cheat by agent 1, which effectively lies to node 0 about its child's utility. Again both agents receive the transmission, but now honest agent 2 pays 7 (believing that this is what it owes) while agent 1 pays nothing. We assume that agent 2 does not communicate directly with node 0, and that agent 2 does not see how much agent 1 pays. The cheat succeeds because each of the honest agents has a consistent view of the network, shown in figures 3 (node 0's view) and 4 (node 2's view).

This attack is possible because agent 1 can lie to the source node about both its utility and the utility of agent 2. In the centralized game, it could lie about its own utility but not about another's.

This examples shows that if agents are not forced to implement the FPS algorithm then it is not a dominant strategy for them to do so faithfully with their true utility as input. A rational agent can sometimes gain a greater payoff by some other strategy not based on the FPS algorithm at all. The main ways

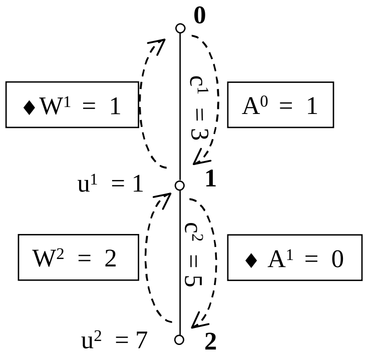

Fig. 2. The cheat. Agent 1 pays 0 and agent 2 pays 7.

to cheat are to send inconsistent messages up and down the tree and to pay an incorrect amount.

Appendix A contains two other examples of cheating. In the first, an agent receives the transmission for free even though its subtree's welfare is negative. In the second, an agent pays less than it ought to. These examples demonstrate that in order to check that an agent isn't cheating, it is necessary to check at least the agent's received messages, the welfare it sent and its payment.

Proposition 1. *In order to check whether a given agent has paid the correct amount, it is necessary to check the consistency of that agent's payment, the welfare value that it sent to its parent, and all the messages that it received from its children and parent.*

4 Improvements

We introduce the threat of auditing by the content provider to motivate rational agents to execute the protocol. In both the protocols described below, each agent collects enough signed messages during the protocol to "prove" that it didn't cheat (in the second protocol, some of this "proof" is stored at the agent's children). Proposition 1 shows the minimum that the content provider must check for each agent. At the end of the protocol, the content provider audits every agent with some probability which was common knowledge from the beginning. It imposes a large fine on those who can't prove that they were honest. We make the fine large enough that even a small probability of incurring it motivates risk-neutral rational agents to execute the algorithm correctly, including carrying out some checks on others' messages.

The first protocol (protocol A) is designed for agents that would deviate from the protocol only if it strictly benefitted them. We prove that honesty is an equilibrium, *i.e.* that if each agent assumes that all the others will be honest,

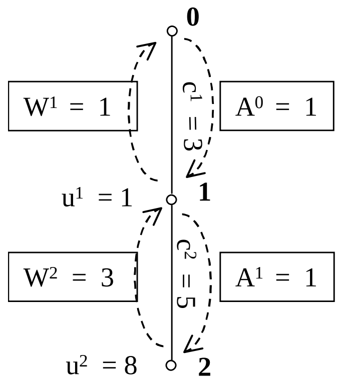

Fig. 3. View of agent 0. $pay^1 = 0$ and $pay^2 = 7$

it maximizes its welfare by implementing the protocol correctly. We then show the stronger result that if each agent assumes that all the others only deviate from the protocol in ways that strictly improve their welfare, it maximizes its welfare by implementing the protocol correctly. We also prove that an agent is only fined if it has actually cheated, so honest agents never pay more than their utility.

The second protocol (protocol B) adds an extra message and some extra incentives to the first one. This protocol works correctly for any welfare-maximizing agents, because following it is strictly better than diverging from it in any computationally feasible way.

Both protocols have the property that when all the agents are rational there should be no cheating. In both cases an agent may cause the protocol to abort if it detects another cheating, in which case no transmission is sent and the net worth is zero. This may seem to be an unduly harsh response, but we assume that the protocol could be restarted, possibly after excluding the (irrational) node that had caused the problem. However, all the following analysis is for one run of the protocol. A session number or timestamp would be needed for multiple runs.

The issue of how to guarantee that the transmission actually goes to the correct set of recipients is also beyond the scope of our discussion. We also don't consider collusion among agents, because the underlying Marginal Cost Pricing mechanism does not prevent collusion. See [1] for more on the issue of collusion.

4.1 Model of the agents and authentication

We model four different types of agents. Reporting of other agents' cheats is considered part of executing the protocol. The agents are:

honest Always follows the protocol correctly.

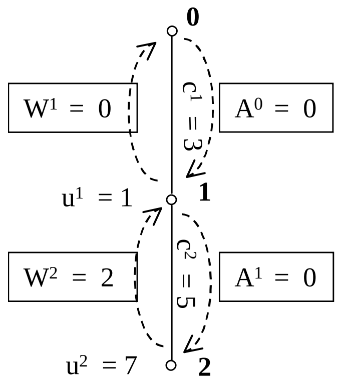

Fig. 4. View of agent 2. $pay^1 = 1$ and $pay^2 = 7$.

selfish-but-agreeable Maximizes its own welfare. Deviates from the protocol only if it strictly improves its welfare.
selfish-and-annoying Maximizes its own welfare. Has no preference for following the protocol unless that is the only action that maximizes its welfare.
malicious May deviate arbitrarily from the protocol. Isn't deterred by fines.

Selfish-but-agreeable is a reasonable model for many applications, because most users of a device (or software) that implemented the desired algorithm would bother meddling with the device only if doing so was beneficial.

In this section we will include only honest and selfish agents. We consider security issues in section 5 by adding malicious agents.

We assume an idealized version of digital authentication. Informally, we assume a public key infrastructure in which the content provider knows everyone's public key and everyone knows their children's, parent's and grandparent's public keys. The idealized signatures have the following properties (based on those in [5])

- Signed messages sent by an honest agent to a parent, child or grandchild cannot be altered or manufactured by others.
- Any receiver can identify the agent that originated a signed message, provided the originator is honest and is a parent, child or grandparent of the recipient.
- Signed messages sent by an honest agent to the content provider cannot be altered or manufactured by others.
- The content provider can identify the agent that originated a signed message, provided the originator is honest.

The assumptions about agents are expressed only in terms of honest agents' properties because a non-honest agent could release its private signing key to others. We will show that both types of selfish agent will keep their keys secret given our incentives, so the two conditions stated above will apply to their signatures also.

We write $sig_i(m)$ to mean the message m signed with the private key of agent i, and $sig_i((m_1, m_2))$ to mean the messages m_1 and m_2 concatenated and signed by i.

The content provider knows the network topology, all the network costs and what payments were received from what nodes. It has the authority to impose large fines on the participants and enough money to offer them significant rewards. More specifically, we assume there is a value C greater than the amount an agent can gain by cheating. This could be any amount greater than the maximum utility that any agent derives from receiving the transmission. The provider has some way to charge this amount, or multiples of this amount, as a fine. For example it could insist that every participant deposits some multiple of C as a bond before joining the network (as in [4]).

4.2 An authenticated protocol (protocol A)

Protocol Description Assume every node knows its parent, all of its children and the costs of the network links adjacent to itself. We have already assumed that each node can recognize the signatures of its children, parent and grandparent. If some child fails to respond it will notice. All other information must be gained by receiving messages.

See Figure 5 for a diagram of protocol A. The idea is that at the end of this protocol each agent should be able to "prove" to the content provider that it paid the correct amount. The bottom-up pass is identical to the one in section 2.2, except that agents sign the welfare that they send up. In the top down pass, each agent j sends to each child k:

$$sig_j(A^j, W^k)$$

The first part of this signed message is just the message sent in the top-down pass of the protocol in 2.2. The second part provides k with a verification of the welfare value that k itself sent up the tree in the first pass.

At the end of the first pass each node checks that it has received a correctly signed message of the correct form from all of its children. In the second pass, it similarly checks its parent's message. If any agent detects a protocol violation by another, such as a signature that doesn't verify or a failure to send a message of the correct form, it immediately notifies the content provider and the protocol is stopped.

Recall that C is a quantity greater than the amount an agent could gain by cheating. If an agent can provide two conflicting messages both signed by another (such as two different signed welfares) and it reports this, then the reporting agent receives a reward C and the agent that signed two conflicting messages receives a penalty C. If an agent releases its private signing key (which is a protocol violation) then an honest agent is allowed to use that key to sign

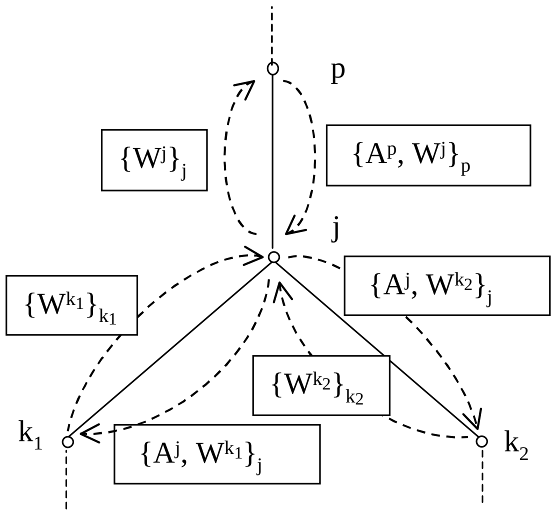

Fig. 5. Protocol A, messages to and from j

two conflicting messages, claim the reward and cause the agent that released its key to be fined.

At the end of this protocol each agent has a "proof" that it paid the correct amount, consisting of the messages it received during the protocol (see Definition 2). The content provider may audit each proof with some probability q (which is common knowledge) at the end of the protocol and if an agent fails to produce the appropriate proof that agent is fined C/q.

Definition 2. *A* proof of paying $Proof_j$ *for node* j *with parent* p *is a pair consisting of the signed messages that* j *received during the protocol.*

$$Proof_j = \langle sig_p(A^p, W^j), \{sig_{k_1}(W^{k_1}), \ldots, sig_{k_l}(W^{k_l})\}\rangle$$

where $k_1, \ldots, k_l$ *are all* j*'s children.*

A proof of paying $Proof_j$ determines what j should pay, because it allows the content provider to compute j's utility u^j and contains a record of the minimum welfare on the path from j to the root. Hence it makes sense to check whether j did actually pay the amount computed from the proof. A proof of paying is correct if the agent chose a valid welfare (one that corresponded to a non-negative utility) and it actually paid the amount it should have paid.

Definition 3. *The* utility derived from a proof of paying $Proof_j$ *is* $u^j = c^j + W^j - \sum_{h=1}^{l} W^{k_h}$. *The proof* $Proof_j$ *is* correct *if* $u^j \geq 0$ *and* j *paid*

$$pay^j = \begin{cases} 0 & \text{if } \min(W^j, A^p) < 0 \\ \max(0, u^j - \min(W^j, A^p)) & \text{otherwise} \end{cases}$$

This corresponds to satisfying equations 1 and 2.

Our content provider is not like the game-theoretic central authority that inputs everyone's utility and computes the outcome. Each audit of each agent uses only local information from one agent's proof of paying. If the content provider is able to impose large fines then it can set its probability of auditing very low, meaning that it has to audit very few if any agents in each run of the protocol. Hence there can be very little communication between the content provider and the agents.

Why selfish agents' signatures are reliable An agent's expected welfare decreases if it releases its private signing key, since another agent might use the key to sign conflicting messages, then report these conflicting messages to the content distributor, collect the reward C and cause the agent that released its private signing key to be fined C. Since receiving a reward C is greater than any benefit gained by cheating, any agent that learnt another's signing key would report it, and the agent whose key was revealed would lose more than it could gain by cheating.

Why honesty is an equilibrium We will show that if every other agent is honest, a single selfish agent maximizes its welfare by executing the protocol correctly also.

Suppose initially that the content provider's checking probability is 1, *i.e.* all nodes must always send in their proof of paying at the end of the protocol. In this case it is easy to show by case analysis that a node with honest ancestors and children cannot gain by cheating.

Lemma 1. *Let j be a node with parent p and children $k_1, \ldots, k_l$. Suppose all nodes other than j are honest. Assume that the content provider checks every node with probability 1 and that its fines are more than j could gain by cheating. Then j's welfare is maximized by implementing the protocol correctly with its true utility.*

Proof. Since j's neighbors are honest we can assume that j will not receive any contradictory signed messages, forged signatures, or other protocol violations that it would be obliged to report to the content provider. It cannot benefit by falsely reporting anyone to the content provider because it cannot fabricate a pair of contradictory signed messages and any other kind of violation stops the protocol and gives j a welfare of zero. If j sends two contradictory signed messages to a neighbor, then the neighbor will report it and j will receive a fine greater than the money it could have made by cheating. If it fails to send any message for some part of the protocol, the protocol will time out or fail to terminate and j will receive a welfare of zero. Hence j maximizes its payoff by sending exactly one message for each of the three classes of message it is supposed to send in the protocol, namely W^j to its parent and $W^{k_1}, \ldots, W^{k_l}$ and A^j to its children. It also chooses a payment pay^j at the end of the communication steps. We consider each of these messages in turn and show that j cannot benefit by lying about them.

A^j: This message is sent down the tree and no other message is received from j's children afterwards. It consequently doesn't affect $Proof_j$ or whether j receives the transmission, so j does not benefit by lying about it.

$W^{k_1}, \ldots, W^{k_l}$: By the same argument, j has no motive to lie about these either.

W^j and pay^j: Choosing a value of W^j fixes j's utility given its children's welfares and its uplink cost (by the formula in definition 3). Since all the other agents in the tree are honest, the value of A that j receives from its parent is the same as the value it would have received in a correct execution of the protocol in which it sent the same value of W^j. Likewise, since j's parent is honest, $Proof_j$ contains exactly the value of W^j that j sent. The content provider will ensure that $Proof_j$ is correct, namely that the signed values in it are consistent with j's payment. Therefore j's payment is fixed by $Proof_j$ (via the formula in definition 3) to be exactly the amount it would have paid in a correct execution of the protocol with utility as given in definition 3.

This shows that j maximizes its welfare by executing the protocol correctly with some utility. Since truth-telling is a dominant strategy when the protocol is executed correctly by everyone, j maximizes its welfare by using its true utility. $\diamondsuit$

Lemma 2. *Let j be a node with parent p and children $k_1, \ldots, k_l$. Suppose all nodes other than j are honest. Assume that the content provider checks every node with probability q and that its fines are more than $1/q$ times what j could gain by cheating. Then j's expected welfare is maximized by implementing the protocol correctly with its true utility.*

Proof. By an argument similar to the proof of lemma 1, if j has a correct proof of checking $Proof_j$ at the end of the protocol then its welfare is no greater than it would have been by correctly executing the protocol with its true utility. If it doesn't have a correct proof of checking at the end then its expected welfare is negative because the auditor's fine is so high. Hence its best strategy is to execute the protocol correctly and ensure a correct proof of checking. By the strategyproofness of the underlying mechanism, it should choose its true utility. $\diamondsuit$

Equilibrium is an important and often-studied concept in game theory. However, we would like show the stronger condition that truth telling is still a dominant strategy. This is not quite the case, but if it is common knowledge that all agents are restricted to being selfish-but-lazy then it is a best strategy to follow the protocol truthfully.

Why agents follow the protocol if others are only selfishly dishonest We show that an agent maximizes its payoff by being truthful, assuming that all the others are honest or selfish-but-agreeable.

The protocol is designed so that if some agent deviates from the protocol then any other agent that detects this will be motivated to report it, either to receive the reward or to avoid being fined for the lack of a correct proof of paying. Every cheat not detectable by others risks a fine from the content provider. Hence no agent is motivated to deviate from the protocol.

We will show first that honest agents do not incur fines, so the protocol still satisfies the voluntary participation constraint.

Lemma 3. *If an agent j has a chance of being fined at the end of the protocol, then j failed to follow the protocol.*

Proof. The agent is fined either for not having a correct proof of paying or because another agent a could produce two contradictory messages signed by j. In the first case, j has clearly not followed the protocol since it was supposed to report any incorrect signatures or missing or badly formed messages as soon as it received them. If it received a complete set of well-formed, correctly signed messages, then this always constitutes a correct proof of paying for some payment pay^j given by definition 3. In the second case, j either signed two contradictory messages or released its private key (which are certainly protocol violations), or another agent succeeded in forging j's signature, which we assume is impossible. ♢

Corollary 1. *An agent cannot gain a reward when no other agent cheated.*

Lemma 4. *If the protocol completes without any agent reporting a failure, then for each agent j there is exactly one correct proof of paying that j can produce.*

Proof. An agent with no correct proof risks being fined, so as long as the expected value of the fine is greater than the maximum profit it could make by cheating, the node will ensure that it has at least one correct proof.

An agent with two or more different proofs all consistent with its payment must have at least two contradictory signatures from some other agent. Therefore, as long as the reward for reporting contradictory signed messages is greater than the maximum profit it could make by cheating, the agent will report it and take the reward rather than continuing with the protocol. ♢

Lemma 5. *Agents are not motivated to send false messages down the tree.*

Proof. The messages that an agent sends down the tree to not affect its proof of paying and consequently don't alter how much it has to pay. Nor do they affect whether the agent gets the transmission, since no information from these messages is transmitted back to the source. ♢

An agent also has no motive for sending the correct values down the tree, which is why we allow only selfish-but-agreeable agents.

Lemma 6. *If the other agents deviate are selfish-but-agreeable, and if the protocol completes without any agent reporting a failure, then an agent j with one correct proof of checking $Proof_j$ has the same welfare at the end of the protocol as it would have had if it had executed the protocol honestly with utility u^j as derived from $Proof_j$.*

Proof. By lemma 5 (and induction down the tree) we can assume that the signed values of A and W^j that j receives are the true ones that were correctly calculated from the welfares sent up the tree. By lemma 4 and the definition of correctness, each welfare value sent up the tree must have been consistent with some valid (*i.e.* non-negative) utility. Therefore the value that the auditor computes for j to pay based on $Proof_j$ is exactly what it would have paid had it been honest, given the utilities of the agents above and below it as derived from their proofs. $\diamondsuit$

Theorem 1. *Assuming it is common knowledge that all agents are honest or selfish-but-agreeable, each agent implements Protocol A correctly with its true utility.*

Proof. If the protocol does not complete, all agents who have not reported another's cheating receive a welfare of zero. Reporting another agent for cheating is always more profitable than cheating, but only if the report is true (by Corollary 1). Hence agents truthfully report a protocol failure. By lemma 4, if there is no protocol failure then each agent has only one proof of checking and by lemma 6, each agent has the same welfare as it would have if it had executed the protocol honestly with whatever utility the proof implies. Since it's a best strategy to use the true utility in the underlying mechanism, it is a best strategy to use the true utility in this scheme also. $\diamondsuit$

This result would not be true if we included malicious or selfish-and-annoying nodes. If an agent cheats without benefiting itself, such as by sending the wrong messages down the tree, then another agent (even a selfish-but-agreeable one) can still profit by lying.

4.3 A stronger authenticated protocol (protocol B)

In the previous section we had to make the assumption that agents would deviate from the protocol only if it strictly benefited them to do so. In this section we add an extra message and an extra check to the protocol to ensure that implementing it correctly (with some utility) is strictly better than any kind of cheating. This means the protocol can withstand selfish-but-annoying agents. The main difference is that the protocol ensures that agents send the correct messages down the tree, rather than relying on the fact that an agent's downward messages don't affect its own welfare.

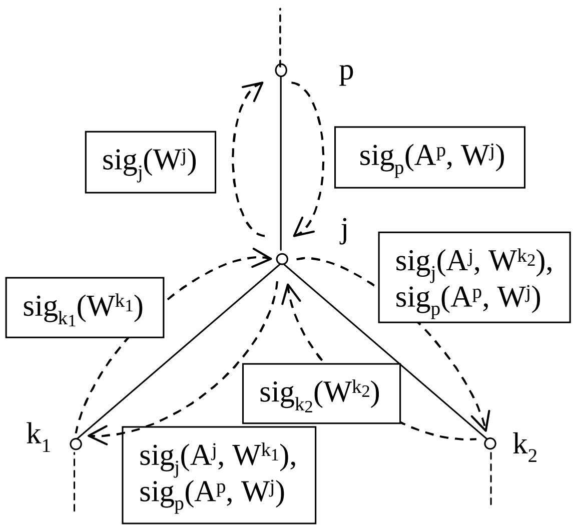

Fig. 6. Protocol B, messages to and from j

Protocol Description Figure 6 shows a diagram of protocol B. This protocol is very similar to protocol A, except that the top-down pass has one additional message: each node j with parent p sends to each child k not only $sig_j(A^j, W^k)$ but also $sig_p(A^p, W^j)$, which is the message that it would have received from its parent in protocol A. This is enough to "prove" to j's child (and to the auditor) that j computed A^j correctly (satisfying equation 3). We will call this pair of messages k's *proof of parent* and denote it by $ParentProof_k$. Child node k checks the signatures on this proof and checks that the values satisfy equation 3.

We retain the same checking, fines and rewards as for protocol A (including the fine for signing contradictory messages) and add one: when the content provider checks $Proof_j$ it also chooses one of j's children k at random and requests $ParentProof_k$. It checks that the signatures and value of A^j are correct and that the value of W^k is the same as that in $Proof_j$. If all of these checks succeed, j receives a bonus of 1. If not, k is fined an amount Cl/q, where l is the number of children.

Why rational agents follow the protocol

Lemma 7. *If an agent j has a chance of being fined at the end of the protocol, then j failed to follow the protocol.*

Proof. Same as Lemma 3 ♢

Corollary 2. *No agent can gain a reward when no other agent has cheated.*

Lemma 8. *For each step of the protocol, each agent sends at most one message.*

Proof. Direct from the fine for signing contradictory messages and the reward for the agent who reports it. ♢

Lemma 9. *If the protocol completes without any agent reporting a failure, then for each agent i there is exactly one correct proof of paying and exactly one correct proof of parent that i can produce.*

Proof. Similar to Lemma 4. ♢

Lemma 10. *Sending the correct messages down the tree is strictly more profitable than sending incorrect ones.*

"Correct" means consistent with the values received from the agent's parent and children.

Proof. Consider agent j with child k and parent p. By lemma 9 either the protocol will be interrupted by someone reporting cheating, or k will receive a correct proof of parent. If the protocol is interrupted by someone other than j then j gets a welfare of at most zero. So suppose k receives a correct proof of parent. By lemma 8, j sent each part of this proof at most once. It is impossible for j or k to alter the message received from p, hence j must have calculated A^j correctly. By lemma 8 again, k sent only one welfare value up the tree. Since j cannot forge k's signature, and since the definition of correctness for proofs of parent includes consistency with the parent's proof of paying, j must have sent the correct value of W^k down the tree. In this case j has positive expected welfare because there is some chance that it will be audited and receive the small bonus for having given its child a correct proof of parent. Hence it is strictly better for j to send the correct messages down the tree than any incorrect ones. ♢

Lemma 11. *Suppose the protocol completes without any agent reporting a failure and that agent j with child k has one correct proof of checking $Proof_j$ which implies a utility of u^j. Then j has the same welfare at the end of the protocol as it would have had if it had executed the protocol honestly with utility u^j, except possibly for the extra bonus for having provided a correct proof of parent to k, which is independent of u^j.*

Proof. Similar to lemma 6, using lemma 10. ♢

Theorem 2. *Assuming it is common knowledge that all agents are honest, selfish-but-agreeable or selfish-and-annoying, each agent implements Protocol B correctly with its true utility.*

Proof. Similar to Theorem 1. ♢

5 Security Perspective

We are interested in designing systems which assume that most agents are selfish but are still robust against occasional malicious nodes. Useful properties to consider include reducing the extent to which the malicious node(s) can reduce the group welfare, increasing the cost of being malicious and making sure that other nodes continue to behave correctly in the presence of malicious ones.

In this section we consider what can be achieved by an adversary trying to reduce the group welfare while unconcerned about its own. In a traditional security model we would assume that this adversary could compromise several nodes at once, but since this scheme is susceptible to collusion anyway that seems to be too strong a model. Hence for this section we consider one adversary that controls only one node.

We consider first the security of the original FPS scheme in the tamper-proof nodes model as a basis for comparison. We then consider the unauthenticated scheme in the autonomous nodes model. Finally we compare these to the behavior of protocol B and find that, except for denial of service attacks, its security is almost as good as that in the tamper-proof nodes model, as long as all malicious nodes have honest neighbors. The only shortcoming is that protocol B does not prevent a malicious node from stating a negative utility, then executing the protocol correctly. We show that this possibility does not encourage selfish nodes to change their behavior. Such a malicious node would eventually be audited by the content provider and its cheating would be detected.

When evaluating the success of the adversary, we consider the difference between the net worth if it had behaved truthfully and the net worth after it cheated. We exclude the utility of the compromised node from the calculation. We will also consider the cost incurred by the adversary.

5.1 Security in the tamper-proof nodes model

The tamper-proof nodes model is susceptible to collusion among nodes (see [1]) but is quite secure against single malicious nodes. The security in this model is our basis for comparison of protocol B's security. We show that a single adversary can significantly reduce the group welfare but only by incurring a large cost to itself.

Lemma 12. *Let the adversary be node i. Suppose it wasn't going to receive the transmission by bidding zero. Then it cannot reduce the group welfare by more than it pays.*

Proof. Let A_0^i be the minimum welfare of any node from i to the root (inclusive) when i bids 0. Since i does not receive the transmission, A_0^i must be negative. If i bids less than $-A_0^i$ then it does not receive the transmission and the set of receivers is the same as if it had bid zero. Hence the group welfare is also unchanged. If it bids $u^i \geq -A_0^i$ then the minimum welfare A^i of any node from i to the root is $u^i + A_0^i$ so the agent pays $pay^i = u^i - A^i = -A_0^i$ (See equation 2).

The difference in group welfares between bidding zero and bidding u^i, excluding the welfare of agent i, is the cost of including the subtree tree whose true welfare is $-A_0^i$. This is exactly what i pays. $\diamondsuit$

Lemma 13. *Let the adversary be i. Suppose it was going to receive the transmission by bidding zero. Then it cannot reduce the group welfare.*

Proof. Overbidding does not alter where the transmission is sent and consequently doesn't affect the group welfare. Underbidding is impossible. $\diamondsuit$

5.2 Security of the original protocol in the autonomous nodes model

A naive implementation of the FPS protocol in the autonomous nodes model is not secure. The method of attack and the amount of damage an agent can do depends on whether it would have received the transmission by truthfully bidding zero or not.

If adversary i would have received the transmission by executing the protocol correctly with a utility of zero, then it cannot reduce the group welfare by more than W^i. It can easily prevent its descendants from receiving the transmission. Its interference in other parts of the tree is achieved only by altering the value of W^i that it sends up. In the worst case it can set this to zero, causing a tree of total welfare at most W^i not to receive the transmission. This attack costs i zero.

If a would not have received the transmission by executing the protocol correctly with a utility of zero, then it can reduce the group welfare by a large amount (minus the lowest node welfare on its path to the root) by overbidding. Furthermore, it can avoid paying anything by transferring the cost of its overbidding to its descendants.

5.3 Security of protocol B

We will show that, even if an agent doesn't care about maximizing its welfare, it can't undetectably harm the system much more than it could in the tamper-proof nodes model, as long as all its neighbors are all honest. It can still pretend it has a negative utility in some cases by declaring that the welfare of its node is 0, but we will show that this possibility doesn't change the behavior of the selfish nodes in the tree and can eventually be detected by the content provider.

A malicious node can perform denial of service attacks by not sending messages when it is its turn, or by falsely reporting that another agent has cheated. Cheating in a detectable fashion could also be regarded as a denial of service attack since it stops the protocol. It could also release its private key to other (selfish) agents, allowing them to print contradictory messages that appear to have been signed by the malicious node and collect the reward for reporting this. This would be expensive for the malicious node. We assume that if the protocol is aborted in this way then it is restarted, with some clever way of ensuring that messages from different protocol runs can't be confused. Hence a denial-of-service attack could be annoying but is unlikely to last very long because the content provider could eventually exclude the malicious node from the network.

We will show that apart from these denial-of-service attacks, a malicious node that isn't reported for cheating must have executed the protocol correctly with some utility, which may be negative.

Lemma 14. *Let j be a malicious agent with children $k_1, \ldots, k_l$ and parent p. If $k_1, \ldots, k_l$ and p are honest and the transmission is sent then j must have sent consistent messages.*

Proof. There is always some utility consistent with the welfare j sent, according to equation 1. The honest children check that the values of A^j sent down were consistent with that welfare, *i.e.* satisfying equation 3. ♢

Note that j needn't have paid the correct amount, relying on the chance of not being checked. It also may have sent messages consistent only with a negative utility. We show next that the possibility of negative utilities doesn't cause the selfish agents to change their (honest) behavior.

Lemma 15. *Assume the game theory model with exactly one agent at each node, but let there be some (malicious) agents that are allowed to state negative utilities. Then it is still a dominant strategy for the other agents to state their true utility.*

Proof. Without loss of generality consider the strategy of agent 1 and suppose that agents $2 \ldots m$, with uplink costs $c^2, \ldots, c^m$ respectively, make negative bids $u^2, \ldots, u^m$ respectively (and there are no other negative bids). Then the welfares of each subtree and the value of A^1 are the same as they would be in the tree where the uplink costs of agents $2 \ldots m$ were $c^2 + u^2, \ldots, c^m + u^m$ and they stated utilities of zero. Therefore how much agent 1 pays and whether it receives the transmission is the same as in that case, so its best strategy is the same, *i.e.* to state its true utility. ♢

Lemma 16. *If there may be honest, selfish and malicious agents in the tree, and all malicious ones have only honest neighbors, then every selfish agent's best strategy is to follow the protocol correctly.*

Proof. Similar argument to lemma 11. ♢

Theorem 3. *If there may be honest, selfish and malicious agents in the tree, and all malicious ones have only honest neighbors, then every selfish agent's best strategy is to follow the protocol correctly with its true utility.*

Proof. Direct from lemmas 15 and 16. ♢

A malicious agent can still reduce the group welfare significantly without being detected, but only by executing the protocol correctly (with a possibly false and/or negative utility). Even the negative utilities are detectable by the auditor if it happens to audit the malicious node, so that node could eventually be caught and excluded from the network. The important point is that the possible presence of some malicious nodes doesn't cause the rest of the nodes to deviate from their honest strategy.

6 Conclusions and Further Work

We illustrate some general techniques by transforming a distributed mechanism from one that assumes all nodes faithfully implement a specified algorithm into one that assumes nodes are strategic in all of their actions. The transformation involves some cryptography and some extra economic incentives. Our two authenticated protocols are designed for two slightly different agent models. Protocol A works correctly if we assume that it is common knowledge that agents deviate from the protocol only if it strictly improves their welfare. Protocol B is appropriate for agents that are selfish but will not follow the protocol unless they have some reason to do so. We prove that it is strictly better to implement the correct algorithm than to try to cheat.

We also evaluate the robustness of protocol B when malicious agents with honest neighbors are present, showing that these malicious agents can only reduce global welfare within limits and cannot give other agents an incentive to deviate from the protocol. We compare the security of protocol B to that of the FPS protocol in the tamper-proof nodes model and show that it is almost as strong, as long as malicious nodes have all honest neighbors. This is a significant improvement over a naive implementation of the tamper-proof nodes algorithm in the autonomous nodes model.

We hope to extend our ideas to other Internet applications such as BGP ([2]) by distributing the auditing functions. A node expecting to be paid by another could probabilistically demand a proof that it is being paid correctly. The challenge is to try to distribute the punishment and reward ideas without giving agents too many new ways to cheat the system. If misbehaving nodes are punished by excluding them from the network and routing around them, then this not only provides an incentive for rational agents but also quarantines malicious nodes and limits their impact on the global system.

7 Acknowledgements

We thank Joan Feigenbaum, Amir Ronen, Rahul Sami, and Andre Scedrov for helpful discussions about this work.

References

1. J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and collusion in multicast cost sharing. In *Proceedings of the 3rd Conference on Electronic Commerce*, pages 253–255, New York, 2001. ACM Press.
2. J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A bgp-based mechanism for lowest-cost routing. In *Proceedings of the 21st Symposium on Principles of Distributed Computing*, pages 173–182, New York, 2002. ACM Press.
3. J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001. Special issue on Internet Algorithms.

4. P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of the ACM Conference on Electronic Commerce*, 2001.
5. L. Gong, P. Lincoln, and J. Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults. In R. K. Iyer, M. Morganti, W. K. Fuchs, and V. Gligor, editors, *Dependable Computing for Critical Applications—5*, volume 10 of *Dependable Computing and Fault Tolerant Systems*, pages 139–157, Champaign, IL, sep 1995. IEEE Computer Society.
6. H. Moulin and S. Shenker. Strategyproof sharing of submodular costs:budget balance versus efficiency. *Economic Theory*, 18(3):511–533, 2001.
7. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conf. on Electronic Commerce*, 1999.
8. N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
9. M. J. Osborne and A. Rubinstein. *A course in game theory.* MIT Press, Cambridge, Mass., 1994.

A Other cheating examples

A.1 Second cheat: Getting the transmission for less while others pay extra

This attack relies on agents' confusion about the costs of links they are not adjacent to.

We assume:

- agent 2 does not communicate directly with agent 0 (it does communicate indirectly via node 1),
- each agent knows the link costs of only those edges it is adjacent to.

We dispense with the assumption that payments are secret — this attack succeeds even if every agent can see every other's payment.

The true state of the network is shown in Figure 7. Here the tree's net welfare is negative, so no transmission is sent. Figure 8 shows the attack, where Agent 1 lies so as to receive the transmission for free, while making agent 2 pay for it and causing the content provider to lose revenue. Agent 1 gains by one unit because it receives for free a transmission it values at one.

Figures 9 and 10 demonstrate that each of the honest agents receives a consistent view of the network. Agents 0 and 2 could not detect the cheat even if they could communicate about their supposed values for u^1 and u^2 (but not if they could compare link costs). Figure 10 shows that agent 2 can be allowed to observe that agent 1 doesn't pay. In that network, $A^1 = A^2 = 1$ but $u^1 = 0 < A^1$ so agent 1 pays nothing. Also, agent 1 lies consistently in the sense that the W^1 it sends up the tree is equal to the value of A^1 that it sends down. That is, watching agent 1's incoming and outgoing messages to check for consistency would also not reveal that it was cheating.

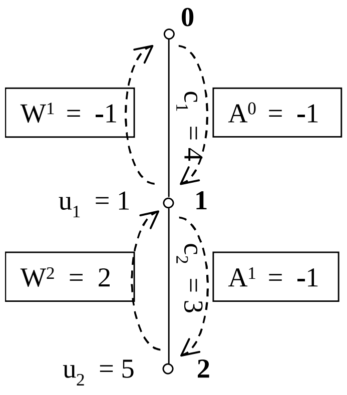

Fig. 7. 2nd cheat: The truth. No transmission is sent.

A.2 Third cheat: Short changing the content provider

In this example a cheater succeeds in paying less than it ought to for a transmission. It first lies consistently about its utility, then pays an amount inconsistent with that lie. This is interpreted by the other agents as implying that one of the children of the cheater had a higher utility than it really did.

The cheat works even if we assume all of the following:

1. link costs are not secret,
2. payments are not secret,
3. each agent i is magically forced to lie consistently, *i.e.* to send down a value for A^i that is truly $\min\{A^{\text{parent}(i)}, W^i\}$ where W^i was the welfare it sent.

The point here is that the node can cheat just by paying a wrong amount, so any solution must include scrutiny of each node's inputs, welfare and payment.

Figure 11 shows the true state of the network. Every agent receives the transmission, the two leaf agents pay 1 each and agent 1 pays 2. Figure 12 shows a successful cheat by agent 1: it first lies consistently about its utility, effectively inflating it by two, so every other agent in the tree believes the total welfare to be two greater than it really is. It then pays zero, leading every other agent to believe that one of the leaf agents bid two more than it truly did. The network states assumed by agent 3 (and possibly agent 0) are shown in Figure 13. The perspective of agent 2 is like Figure 13 with the roles of agents 2 and 3 reversed.

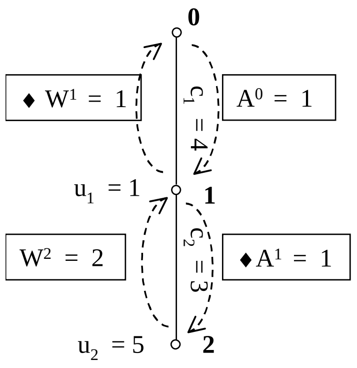

Fig. 8. 2nd Cheat: The cheat. Agent 1 pays 0, agent 2 pays 4.

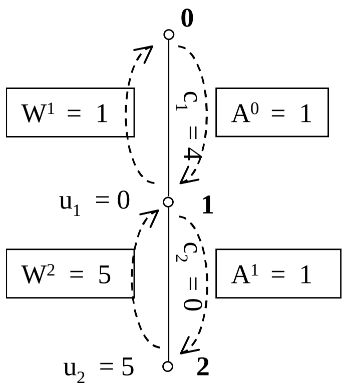

Fig. 9. 2nd Cheat: View of agent 0. $pay^1 = 0, pay^2 = 4$.

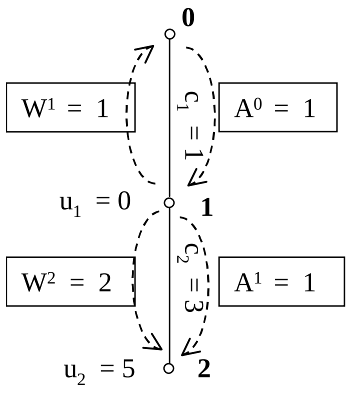

Fig. 10. 2nd Cheat: View of agent 2. $pay^1 = 0, pay^2 = 4$.

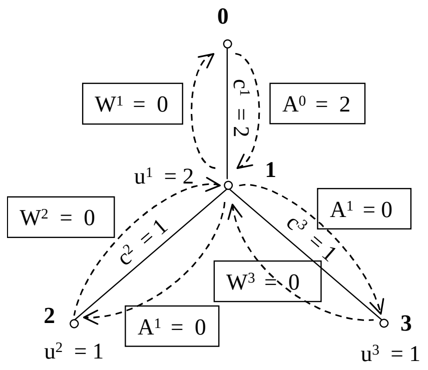

Fig. 11. 3rd Cheat: The truth. $pay^1 = 2, pay^2 = pay^3 = 1$.

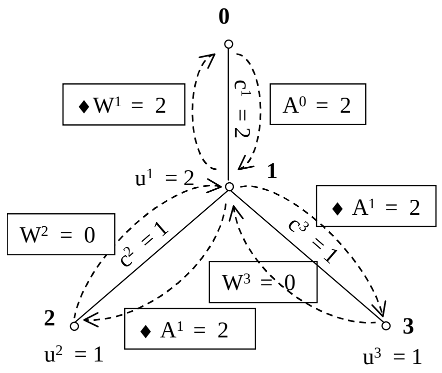

Fig. 12. 3rd Cheat: The cheat. Agent 1 pays 0, agents 2 and 3 pay 1.

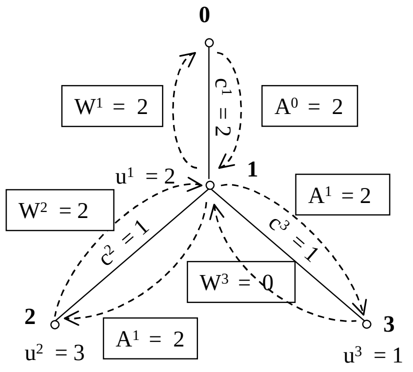

Fig. 13. 3rd Cheat: view of agents 3 and 0. $pay^1 = 0, pay^2 = pay^3 = 1$.