From: AAAI-99 Proceedings. Copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Distributed Games: From Mechanisms to Protocols

Dov Monderer and Moshe Tennenholtz
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa 32000, Israel

Abstract

The theory of mechanism design in economics/game theory deals with a center who wishes to maximize an objective function which depends on a vector of information variables. The value of each variable is known only to a selfish agent, which is not controlled by the center. In order to obtain its objective the center constructs a game, in which every agent participates and reveals its information, because these actions maximize its utility. However, several crucial new issues arise when one tries to transform existing economic mechanisms into protocols to be used in computational environments. In this paper we deal with two such issues: 1. The communication structure, and 2. the representation (syntax) of the agents' information. The existing literature on mechanism design implicitly assumes that these two features are not relevant. In particular, it assumes a communication structure in which every agent is directly connected to the center. We present new protocols that can be implemented in a large variety of communication structures, and discuss the sensitivity of these protocols to the way in which information is presented.

Introduction

Work in mechanisms design deals with the design of multi-agent interactions of rational agents. This work is fundamental to Economics (Kreps 1990), as well as to the attempt to design protocols for non-cooperative computational environments (Boutilier, Shoham, & Wellman 1997; Rosenschein & Zlotkin 1994; Bond & Gasser 1988; Durfee 1992). The importance of the latter has increased dramatically due to the desire to design economic interactions in the Internet setup. In such a context, the fact we have different human users in a distributed system, suggests we can not rely on standard protocol design; we have to design protocols that are incentive-compatible. In order to handle the above problem the theory of mechanism design has to be adapted to computational environments, i.e. to the context of distributed computing.

In a typical mechanism design setup (Fudenberg & Tirole 1991; Kreps 1990; Monderer & Tennenholtz 1998), a center attempts to perform some task, or to arrive at some decision, based on information available to a set of agents. The major problem is that the agents might not supply the desired information to the center, and might cheat the center in order to increase their individual payoffs. The central problem is to design a mechanism that when the agents interact rationally through that mechanism, the center will be able to obtain its objective. For example, the center may wish to sell a particular good while obtaining maximal profit, or to distribute goods among the agents in a way that maximizes social welfare. In order to do so, a mechanism has to be designed; typically, this mechanism determines actions to be taken by the center as a function of messages sent to it by the agents; the agents' strategies correspond to the messages they send to the center as a function of the private information they have. Such interaction is modeled by a Bayesian game. The aim is to design a mechanism that when the agents behave rationally (i.e. according to an equilibrium of the corresponding game), a desired behavior is obtained.

The design of mechanisms is the part of economics that is the most relevant to Artificial Intelligence and computer science. At first glance it may seem that mechanism design is in fact a different terminology for protocol design, when it is applied to non-cooperative environments. Although this claim is basically correct, researchers have noticed that in order to design protocols for non-cooperative computational environments, various modifications to the mechanism design theory are needed. In particular, researchers have been concerned with the fact that agents are resource bounded (Linial 1992; Sandholm & Lesser 1997; Kraus & Wilkenfeld 1991). However, except for the important issue of bounded resources there are several other issues that may make a difference when one tries to adapt the game theoretical approach to mechanism design in computational settings. These issues arise from the fact that the players in such a mechanism are part of a distributed system. So, what we need is a model for dealing with the classical and newer problems in distributed systems within the framework of game theory. Such a model is naturally titled a *distributed game*. Hence, a distributed game is a model for dealing with the most general multi-agent interactions in distributed systems. As such, its definition is quite complex and requires the description of many features which are crucial in some of these interactions. A particular type of a distributed game

Copyright ©1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

is presented in (Monderer & Tennenholtz 1999). The complete definition of distributed games is given in the full paper. In the current report we concentrate on two aspects of distributed games that are essential to the design of protocols for non-cooperative computational environments, and have been ignored by the mechanism design literature:

1. In a computational environment protocols are run by agents/processors that are connected by a *communication network*. However, the theory of mechanism design ignores this aspect, and in fact implicitly assumes a very concrete communication network, where each agent is directly connected to the center. This assumption, which is not realistic in most computational settings, is crucial for the mechanism design literature; as it turns out, its adaptation to general computational settings is non-trivial.
2. The theory of mechanism design ignores the actual way the information available to the agents is represented, as well as the (bit) structure of the messages. As it turns out, and as we later show, the design of the representation language takes a significant role in the adaptation of economic mechanisms to computational environments.

This paper presents several basic results with regard to the transition from mechanisms to protocols in communication networks. We take each node in the network to be associated with a different player (i.e. each node can be viewed as an agent of a different player). A basic observation is that when we try to implement a mechanism in a communication network, a new game-theoretic feature arises: an agent may have the ability to modify the messages sent by other agents to the center. We need therefore to make such malicious behavior irrational for the agents.

We start by presenting a general model for *distributed protocols* in a game-theoretic setting. Distributed protocols extend the concept of standard (economic) mechanisms to the context of computational environments. The two added features of distributed protocols is that they run in a communication network (which is in general different from the one that is implicitly assumed in the game-theory literature), and that they have to be explicit about the syntactic representation of the agents' information and messages. The main problem we face is the need to transform a mechanism that implements some desired behavior into a distributed protocol that obtains that behavior.

Our first result deals with distributed protocols, such that a unilateral deviation of an agent from them is not beneficial for it, when we assume that the agents' private information is selected from a given set with a uniform distribution. We show that given any 2-connected communication network L, any desired behavior which is implementable by a standard mechanism is also implementable by a distributed protocol executed in L. As it turns out, the corresponding implementation technique is not satisfactory for non-uniform distribution functions. In order to handle the more general case, we present improved protocols, which make use of a more complicated representation (syntax) of the agents' information and messages. This enables to implement any desired behavior, that is implementable by a standard mechanism, by a distributed protocol executed in the communication graph L. Together, these results show that economic mechanisms can be constructively transformed into working protocols, and emphasize the crucial role played by the communication network and the syntactic representation of the agents' information and messages. Finally, we introduce the notion of strong implementation by mechanisms and distributed protocols. This notion refers to the concept of strong equilibrium, and requires the protocols to be stable against deviations by any subset of the agents. We show that any desired behavior that is strongly implementable by a standard mechanism is also strongly implementable by a distributed protocol in a communication network with a ring topology.

Distributed Protocols

Consider a set of agents, $N = \{1, \ldots, n\}$, and a *center* denoted by 0. Each agent i is associated with a type $v_i \in W$, where W is a finite set. The interpretation of a type is context dependent. For example, v_i may be the maximal willingness of agent i to pay for a good or for a service. The type of an agent is known only to this agent. It is assumed that for each agent i there exists a random variable $\tilde{v}_i$ that determines its type. We assume that these random variables are independent and identically distributed, with the distribution function f. That is, $Prob(\tilde{v}_i = x) = f(x)$ for every $x \in W$. The center is associated with a set of possible actions A. Each $a \in A$ is also called an *outcome*[1]. We assume that A contains a distinguished action, Λ, which stands for *no action*. In addition there is a utility function $u : A \times W \rightarrow R_+$, where R_+ denotes the set of nonnegative real numbers. The utility of agent i with type v_i from the action $a \in A$ is $u(a, v_i)$. We assume the normalization $u(\Lambda, v_i) = 0$ for every $v_i \in W$. The tuple $E = (N, A, u, W, f)$ is called an *environment*.

The center has some objective function which depends on the types of the agents and on the action it chooses. For example, the center may wish to maximize social welfare or to maximize its own profit. The exact nature of this objective function is irrelevant to this paper. In order to achieve its goal, and since it does not know the types of the players, the center uses a mechanism. A *mechanism* is a pair $H = (M, h)$, where M is a set of messages and $h : M^n \rightarrow A$. Each agent is asked to send a message, and the center commits itself to the action $h(m) = h(m_1, m_2, \ldots, m_n)$, if it receives the message m_i from agent i for every $i \in N$. We assume that M contains a distinguished null message e which stands for "no message", that $h(\bar{e}) = \Lambda$, where $\bar{e}$ is the vector of null messages. The environment E and the mechanism H generates a Bayesian game $B = (E, H)$. In this game , agent i sends a message m_i, and receives the utility $u(h(m), v_i)$[2]. In the mechanism

[1] In some applications it is useful to distinguish between the notions of action and outcome. In such cases there is also a set of outcomes and a mapping which assign outcomes to actions.

[2] In some applications it is useful to allow probabilistic mecha-

design theory it has been assumed that agent i has a strategy $b_i : W \to M$, where $b_i(v_i)$ is the message sent to the center by i when its type is v_i[3]. Let $b = (b_1, b_2, \ldots, b_n)$ be a vector of strategies that are used by the agents. If the vector of types drawn by Nature is $v = (v_1, v_2, \ldots, v_n)$, then the vector of messages that are sent to the center is $m = b(v) = (b_1(v_1), b_2(v_2), \ldots, b_n(v_n))$, and the center chooses the action $h(b(v))$. Therefore the utility of agent i is $u(h(b(v)), v_i)$. Note that the utility of agent i depends on the messages sent by the other agents as well as on its own message. However, agent i does not know the types or the strategies of the other agents.

The basic question in game theory and economics is how do agents choose their strategies given their ignorance about the other agents' strategies and types? The economics literature assumes that the vector of strategies $b = (b_1, b_2, \ldots, b_n)$ chosen by rational agents is in equilibrium. That is, for every player i and for every type v_i, the maximal expected utility of i given $\tilde{v}_i = v_i$, and given that each agent j, $j \neq i$ is using b_j, is obtained in choosing $m_i = b_i(v_i)$. This definition of rationality in multi-agent interactions is the most common in economics in the last four decades. It was introduced by (Nash 1950). It is based on the idea that every agent is a utility maximizer, and that every agent believes that all other agents use their equilibrium strategies. However, there is no good explanation to the question of how do agents form their beliefs about the strategies played by the other agents. One of the stories that usually comes in economics as a justification for this equilibrium assumption (i.e., that people behavior is in equilibrium) refers to an implicit dynamics, that after a while converges to a stable behavior. The equilibrium assumption is in particular problematic in a mechanism that has more than one equilibrium. To slightly resolve this difficulty we assume in this paper that the center, except for publishing the rules of the mechanism also recommends a particular behavior to the agents, which is an equilibrium vector of strategies b. Hence, agent i uses b_i because it knows that every other agent j is recommended to use b_j, and for every possible type v_i, sending the message $b_i(v_i)$ maximizes $i's$ utility under the assumption that every agent j uses b_j. The fact that the center recommends b makes b a correlated equilibrium in the sense of (Aumann 1974). The concept of correlated equilibrium is more general, but we do not discuss this novel concept in this paper.

Suppose that the environment E is fixed. Given a pair (H, b) of a mechanism H and an equilibrium vector of strategies b, we define $g : W^n \to A$ by $g(v) = h(b(v))$. That is, g is an outcome-valued random variable that determines the outcome as a function of Nature's draw. g is called the outcome function of (H, b), and we say that (H, b) implements g. When the mechanism H has a unique equilibrium, or when the equilibrium under discussion is clear from the context, we say that H implements g.

A mechanism $H = (M, h)$ is a *direct mechanism* if $M = W \cup \{e\}$. A direct mechanism is *truth revealing* if the vector of strategies t in which every agent reveals its true type is in equilibrium, that is $t_i(v_i) = v_i$ for every $i \in N$ and for every $v_i \in W$. The revelation principle (see e.g., (Myerson 1991)) implies that if (H, b) implements g, then there exists a direct mechanism H' such that (H', t) implements g. Hence, we we can restrict our attention to direct mechanisms where agents send their actual types as their messages.

Assume now that a certain outcome function g can be implemented in an environment E. That is, there exists a mechanism that implements g. However, we assume the existence of a communication network in which the center and the agents are operating. This network is described by a *communication graph* L as follows: Let $L = (V, E, \psi)$ be an undirected graph, where $V = N \cup \{0\}$ is the set of nodes, E is the set of edges, and ψ is the incidence function, which associates with every edge e an unordered pair of nodes. For every $i \in V$ let $D(i)$ be the set of all nodes j for which there exists an edge e that joins i and j, that is $\psi(e) = ij$. We assume that there are no loops. That is, for every i, $i \notin D(i)$. So, $D(i)$ denotes the set of agents (including the center) with which i has a direct communication line. We assume that for every agent i there exists a path that connects it to the center. We call the pair (E, L) a *distributed environment*.

Let (H, b) be a mechanism that implements g in the environment E. If the center attempts to use this mechanism in the distributed environment, it may fail to implement g because the incorporation of the new environment enriches the set of strategies of the agents. For example, if the path that joins the center to agent j goes through agent i, then i may read the message sent by j, and it may also change this message or destroy it. For example, in auction mechanisms (Milgrom 1987; Wilson 1992), if agent j sends its bid via a path that goes through i, then i can change the bid of j to the minimal possible bid. In fact, it may be able to do so even if the message is encrypted by some standard cryptographic technique. If the communication graph L has the property that every agent is directly connected to the center, that is, $D(0) = N$, then the distributed environment (E, L) is equivalent to the environment E, and every outcome function which is implementable in E is also implementable in (E, L) by the same mechanism. However, implementation in non-degenerate distributed environments requires a new definition of mechanisms (and strategies).

A mechanism in a distributed environment is called a *center's protocol*. Every such protocol induces a game which we refer to as a *distributed game*. A strategy of an agent in a distributed game is called an *agent's protocol*, and an $(n+1)^{th}$ tuple of a center's protocol and an equilibrium vector of agents' protocols, together with some additional necessary parameters is called a *distributed protocol*. We now describe more specifically the notion of distributed protocols: Let (E, L) be a distributed environment. Every message m is assumed to be a string of bits, that is $m \in \{0, 1\}^*$.

nisms in which $h(m)$ is a probability distribution over outcomes. In such a case $u(h(m), v_i)$ is the expected utility of agent i.

[3] In more general applications it is assumed that an agent can use a *behavioral strategy*, which is a function that maps types to probability distributions on M.

The first parameter in a distributed protocol is a positive integer k determining the length of a legal message. That is, the set of messages is $M_k = \{0,1\}^k \cup \{e\}$. The second parameter describes the language. It is given by an interpretation function $I : \{0,1\}^k \to W$. We assume that for every $w \in W$ there exists $x \in \{0,1\}^k$ with $I(x) = w$. Actually, every message has two parts, the actual content, and the header with the sender name, the name of the agent to whom the message is addressed, and ordered location stamps, from the locations at which this message passed. An agent may change the header, the content and both. For the ease of exposition we assume in this paper that the agents cannot interfere with the headers of the messages. The distributed protocols that we give in this paper work also in an environment in which this assumption does not hold.

We assume a synchronous system: time is discrete, and at each round (i.e. time unit) each agent recalls all messages received by it in the previous rounds, as well as all messages sent by it, and can send messages to other agents through its neighbors (Tanenbaum 1988). These messages are to be received by the neighbors after at most q rounds, where q is some given constant. Hence, the system is fair, in the sense that any message is received after no more than q rounds. In general, there also exists a probability distribution from which the arrival time of messages is drawn, but this probability distribution will not play a role in our analysis, and therefore it will be ignored. We further assume that there is a (big) number Q such that at each round an agent can send at most Q messages. Let H_i^t, $0 \leq i \leq n$, be the set of all possible histories that may be observed by agent i at stage t, $t \geq 1$. So, $H_1^i = \{\phi\}$– the empty history, and for $t \geq 1$, a typical member of H_i^t has the form $h_i^t = (d^{s,1}, d^{s,2})_{s=1}^{t-1}$, where $d^{s,1}$ is the set of all messages received by i at stage s, and $d^{s,2}$ is the set of all messages sent by i at stage s. A protocol for agent $i \in N$ is a sequence of functions $((f_{ijq}^t)_{j \in (N-\{i\}) \cup \{0\}, q \in D(i)})_{t \geq 1}$, where $f_{ijq}^t : H_i^t \to M_k$ determines the message sent to j at round t, through q. That is $f_{ijq}^t(h_i^t)$ is the message sent by i to q, intended to be received by j. The center protocol is similar to an agent protocol, alongside two sequence of functions $(\delta_1^t)_{t \geq 1}$ and $(\delta_2^t)_{t \geq 1}$, where $\delta_1^t : H_0^t \to \{HALT, COTINUE\}$, and $\delta_2^t : H_o^t \to A$. These functions determine the action selected by the center at each round, and the termination of the process. It is assumed that $\delta_2^t = \Lambda$ if $\delta_1^t = CONTINUE$. We assume that there exists a time T_e such that for every sequence of histories $(h_0^t)_{t=1}^{\infty}$ that can be generated by the agents, there exists $1 \leq T \leq T_e$ such that $\delta(h_0^t) = CONTINUE$ for every $t < T$ and $\delta(h_o^T) = HALT$. In particular, the distributed protocol terminates after a finite number of stages, whether the agents obey it or not (it is obviously assumed that the center obeys its protocol).

As we said, the basic question we tackle is whether an outcome function that is implementable by a standard mechanism in an environment E, is also implementable by a distributed protocol in a given distributed environment (E, L).

In the sequel, we will restrict our attention to 2-connected graphs. This assumption is appropriate for most classical communication networks (e.g. ring, hypercube), and it is essential if we wish the system to be immuned at least against one machine failure (Dwork & Moses 1990).

Implementation by distributed protocols: the uniform distribution case

Consider a fixed environment E, an outcome function g and a truth revealing direct mechanism $H = (W \cup \{e\}, h)$ that implements g. Without loss of generality we assume that the number of types (i.e., the cardinality of W) is 2^k for some positive integer k, and that the distribution function f is the uniform distribution. That is, $f(x) = \frac{1}{2^k}$ for every $x \in W$. The latter assumption will be relaxed in the next section. Let L be a 2-connected communication graph L as described in the previous section. We present a distributed protocol that implements g in the distributed environment (E, L). The length of a legal message content in this protocol is k, and the interpretation function is any 1-1 function $I : M_k \to W$. To simplify the notations we can assume without loss of generality that $W = M_k$. We give a shortened description of the distributed protocol. The full description can be easily derived from this shorter version. In order to describe the protocol, we specify for every agent i an ordered pair of node-disjoint paths, (Γ_i^1, Γ_i^2) to the center. This is possible since the graph is 2-connected.

The protocol of agent i with type v_i (sketch):

1. Use the uniform distribution on W to generate a random bit string, y_1 of k bits.
2. Let y_2 be the bit-by-bit exclusive or (XOR) of y_1 and v_i.
3. Send y_1 and y_2 to the center through your neighbors determined by Γ_i^1 and Γ_i^2, respectively.
4. If you receive a message with a header in which the original sender is j, send it without any change to the next node in the designated path of j.

If every agent uses the above protocol then for every vector of types $v \in W^n$ there exists a stage $T(v)$ such that at least one of the agents may be active (in particular, one of its messages may not arrive yet) at every stage $t < T(v)$, and all agents are not active from this stage on. Let T^* be the maximal value of $T(v)$ over $v \in W^n$.

The description of the center's protocol:

1. The center receives messages and execute "CONTINUE" until stage T^*.
2. If the sequence of messages received by the center up to stage T^* can be generated by the agents' protocols, then it does as follows:
   - It takes the XOR of the y_1 and y_2 it received from each agent and treats it as this agent's type. Let $v = (v_1, v_2, \ldots, v_n)$ be the vector of types obtained in this way.
   - It runs the truth revealing mechanism that implements g. That is, it executes the action $h(v) \in A$ and halts.

3. If the sequence of messages received by the center is not consistent with any vector of types, then it executes Λ and halts.

Note that we do not deal with efficiency issues in this paper. E.g., it is obvious that in many cases the center can detect deviations of the agents from their protocol, and halt before stage T^*.

The above distributed protocol is the main tool in proving the following theorem:

Theorem 1 *Let E be an environment, in which the type of each agent is selected according to the uniform probability function on the set of types, and let L be a 2-connected graph. If an outcome function is implementable by a mechanism in E, then it is implementable by a distributed protocol executed in (E, L).*

The proof of the above theorem, as well as the proofs of the other theorems presented in this paper are omitted from this report. The first step of the proof is to use the revelation principle in order to get an implementation of g with a truth revealing mechanism. We then define a distributed protocol as described above and show that the agents' protocols are in equilibrium. That is, an agent does not gain by deviating from its protocol assuming the others stick to their protocols. Consider agent i with type v_i. Let a_i be the expected payoff of i, if all agents (including itself) obey their designated protocols. Recall that $a_i \geq 0$. If this agent deviates and generates an history which is not consistent with any vector of types, it receives 0, because $u(\Lambda, v_i) = 0$. Agent i can also deviate in a way that generates a consistent history. It can do it by not sending its true type and/or by changing the content of some of the messages of the other agents. However, it can be shown that all such changes do not change the distribution of types faced by it (here the uniform distribution assumption is used), and keep its expected payoff at the level of a_i. Hence, such deviations are not profitable.

Implementation by distributed protocols: the general case

In the previous section we proved that if the types of each agent are uniformly distributed, then every outcome function that can be implemented by a mechanism can also be implemented by a distributed protocol executed in a 2-connected communication network. In this section we generalize this result to the case of arbitrary probability functions on the agents' types.

In order to see the problematic issue which one faces in this case, assume an environment in which the set of types is $W = \{x_0, x_1\}$, where $0 < x_0 < x_1$, and $f(x_0) = \frac{1}{4}$ and $f(x_1) = \frac{3}{4}$.

Now, assume that the outcome function we wish to implement in a given communication graph L, is the one which is implemented in a standard second-price auction (Vickrey 1961). In a second-price auction each agent submits a bid for a good held by the center. The good is sold to the agent who has made the highest bid in a price that equals the second-highest bid; in a case of a tie we use a random unbiased coin flipping. It is well known that a second-price auction is a truth revealing mechanism. We take x_0 and x_1 to be the valuations (i.e. maximal willingness to pay) of an agent. Let $M_1 = \{0, 1\}$ and let $I : M_1 \to W$ be the natural interpretation function, that is $I(0) = x_0$ and $I(1) = x_1$. If the center uses the distributed protocol described in the previous section, it becomes worthwhile for an agent to change the content of another agent's message by replacing it with a random bit, where each value of this bit is chosen with probability 0.5. Such a change will decrease the probability that the bid of the other agent is high. However, by using another language, i.e., by using a set of messages with a bigger cardinality and an appropriate interpretation function, we can show the following generalization of Theorem 1.

Theorem 2 *Let E be an environment, and let L be a 2-connected graph. If an outcome function is implementable by a mechanism in E, then it is implementable by a distributed protocol executed in (E, L).*

We now illustrate the ideas of the proof in the framework of the above example. We consider encodings of the two possible types by strings of two bits. There are 4 such strings, and we associate three of them (e.g., 11, 01, 10) with x_1 and only one string (e.g., 00) is associated with x_0. That is, the set of messages is M_2, $I(11) = I(01) = I(10) = x_1$, and $I(00) = x_0$. We modify the protocols given in the proof of Theorem 1 in the following way: The agent selects y_1 as a random 2-bit string (with the uniform probability), and y_2 is defined to be the XOR of y_1 with a random representation of x_i (i.e. 00 if the type is x_0, and either 11,01, or 10, with equal probability to each one of them, if the type is x_1). It follows that both y_1 and y_2 are selected uniformly from the set of 2-bit strings. The rest of the proof is similar to the analogous proof in Theorem 1. In general, we represent each type with a set of bit strings such that the sizes of these representing sets are proportional to the probabilities of the types. The novel idea of this representation technique is that it enables us to view the agents' types *as if* they are selected from a uniform distribution. This in turn enables us to apply the techniques of proof used in Theorem 1.

Strong Implementation

The general techniques we have introduced in the previous sections enable to implement outcome functions that are implementable by standard mechanisms, by protocols in communication networks. Our work relies on the game theoretic notion of equilibrium; that is, we are interested in protocols, in which it is irrational for an agent to deviate from its protocol, assuming the other agents stick to their protocol. More generally one may replace the concept of equilibrium with the concept of *strong equilibrium* (Aumann 1959). In a strong equilibrium, there does not exist a group (coalition) of agents such that a joint deviation of this group can benefit each of its members. If an outcome function g is implemented by a mechanism and by an equilibrium profile b,

which is also a strong equilibrium, we say that the mechanism *strongly implements* g. We similarly define strong implementation by a distributed protocol in a communication graph. It can be easily shown that the revelation principle holds with "strong equilibrium" replacing "equilibrium" . We prove:

Theorem 3 *Let E be an environment, and let L be a ring. If an outcome function is strongly implementable by a mechanism in E, then it is strongly implementable by a distributed protocol executed in (E, L).*

For ease of exposition, we sketch the proof of this result for the case in which the types of an agent are uniformly distributed. The proof of the general case can be derived from this proof analogously to the way in which the proof of Theorem 2 is derived from the proof of Theorem 1, that is by an appropriate change of language. We describe the dynamics generated by the distributed protocol when all agents obey their protocols. In game theoretic language, we describe the equilibrium path. If the center observes a deviation from the equilibrium path of any group of agents, it punishes all agents by an appropriate chosen payments (in the full paper we present elaborated protocols, where there are no monetary punishments). If an agent observes a deviation which will not be necessarily observed by the center, it deviates from the equilibrium path in a way that is observed by the center. We assume that the order of the agents in the clockwise direction of the ring is $1, 2, \ldots, n$. The types of the agents are represented by k bit strings, where 2^k is the cardinality of W. Without loss of generality we assume that $W = M_k$. At stage 0 the center makes n independent uniformly distributed draws $z_1, z_2, \ldots, z_n$ of k bit strings , which will be used as keys. At stage $3j-2, 1 \leq j \leq n$, the center sends the key z_j to agent j through the path starting at agent n. Agent j computes the XOR,y_j of z_j with the representation of its type, $v_j \in \{0,1\}^k$. It also makes a uniformly distributed draw of a k bit string, q_j, and computes the XOR, r_j, of q_j and z_j. At stages $3j-1$, agent j sends y_j to the center through the path that goes through agent 1. At stage $3j$, agent j sends q_j to the center through the path that goes through agent 1, and sends r_j to the center through the pass that goes through agent n. The center computes the type v_j of j by XORing y_j with z_j. It the XOR of q_j and r_j is indeed z_j for all j, as expected when the agents follow the protocols, then it applies the mechanism that strongly implements g to the resulting vector of types.

Given the above protocols it is easy to see that if all agents obey them then the true types will arrive at the center. In order to get the strong implementation result we should show that agents in $N-\{i\}$ can do nothing about agent i-th or their submissions that will increase their payoff. The full proof is omitted from this report. However, the reader should notice that the technique which is used here differs from the techniques we introduced previously. The idea here is that the center distributes a random key, z_i, with which agent i's type announcement will be encrypted. By receiving the key back at end, the center will know that no one has modified it.

References

Aumann, R. 1959. Acceptable points in general cooperative n-person games. In Tucker, A., and Luce, R., eds., *Contribution to the Thoery of Games, Vol. IV (Annals of Mathematics Studies, 40)*. 287–324.

Aumann, R. 1974. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics* 1:67–96.

Bond, A. H., and Gasser, L. 1988. *Readings in Distributed Artificial Intelligence*. Ablex Publishing Corporation.

Boutilier, C.; Shoham, Y.; and Wellman, M. 1997. Special issue on economic principles of multi-agent systems. *Artificial Intelligence* 94.

Durfee, E. 1992. What your computer really needs to know, you learned in kindergarten. In *10th National Conference on Artificial Intelligence*, 858–864.

Dwork, C., and Moses, Y. 1990. Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures. *Information and Computation* 88(2):156–186.

Fudenberg, D., and Tirole, J. 1991. *Game Theory*. MIT Press.

Kraus, S., and Wilkenfeld, J. 1991. The Function of Time in Cooperative Negotiations. In *Proc. of AAAI-91*, 179–184.

Kreps, D. 1990. *A Course in Microeconomic Theory*. Princeton University Press.

Linial, N. 1992. Games Computers Play: Game-Theoretic Aspects of Computing. Technical Report 92–5, CS Department, Hebrew University, Israel.

Milgrom, P. 1987. Auction theory. In Bewly, T., ed., *Advances in Economic Theory: Fifth World Congress*. Cambridge University Press.

Monderer, D., and Tennenholtz, M. 1998. Optimal Auctions Revisited. In *Proceedings of AAAI-98*.

Monderer, D., and Tennenholtz, M. 1999. Distributed Games. to apear in Games and Economic Behavior.

Myerson, R. B. 1991. *Game Theory*. Harvard University Press.

Nash, J. 1950. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America* 36:48–49.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter*. MIT Press.

Sandholm, T. W., and Lesser, V. R. 1997. Coalitions Among Rationally Bounded Agents. *Artificial Intelligence* 94(1):99–137.

Tanenbaum, A. 1988. *Computer Networks*. Prentice Hall.

Vickrey, W. 1961. Counterspeculations, auctions, and competitive sealed tenders. *Journal of Finance* 16:15–27.

Wilson, R. 1992. Strategic analysis of auctions. In Aumann, R., and Hart, S., eds., *Handbook of Game Theory*, volume 1. Elsevier Science Publisher.