# Auto-adaptation of Open MAS through On-line Modifications of the Environment*

Roberto Centeno[1] and Holger Billhardt[2]
rcenteno@lsi.uned.es holger.billhardt@urjc.es

[1] Departamento de Lenguajes y Sistemas Informáticos
Universidad Nacional de Educación a Distancia, Madrid, Spain
[2] Centre for Intelligent Information Technology (CETINIA)
University Rey Juan Carlos, Madrid, Spain

**Abstract.** In this paper we propose a mechanism that is able to encourage agents, participating in an open multiagent system, to follow a desirable behaviour, by introducing modifications in the environment. This mechanism is deployed by using an infrastructure based on institutional agents called *incentivators*. Each external agent is assigned to an incentivator that is able to discover its preferences, and to learn the suitable modifications in the environment, in order to encourage its agent to perform actions, such that, the system's preferences are fulfilled. Finally, we evaluate the mechanism in a p2p scenario.

**Keywords:** Incentives, Regulation, Adaptation, Organisation

## 1 Introduction

Open MultiAgent Systems (OMAS) are systems designed with a general purpose in mind but with an unknown population of autonomous agents at design time. The agents that populate such systems may be heterogeneous (e.g., they may have different unknown preferences and behaviour) and their number may vary dynamically at runtime. Based on their open nature, the general problem when designing OMAS consists in assuring that agents will behave according to the system's global objectives and preferences.

The research community has tackled this problem by defining organisational models that structure and regulate the agent's action space. Some approaches (e.g., the one adopted in Electronic Institutions [6]), define the actions that agents can take in each state of the system (performative structure) and rely on a certain infrastructure that assures that agents cannot violate the defined rules. In these proposals, the implemented rules – defined at design time – are fixed. They can be seen as heuristics that help to meet the system's global objectives. This approach has two disadvantages. First, the agents may still have a certain degree of freedom and this may still imply a more or

* The present work has been partially funded by the Spanish Ministry of Education and Science under projects TIN2009-13839-C03-02 and "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010)

less efficient completion of the systems overall objective. Second, the fixed nature of predefined rules may imply less flexibility in certain unforeseen situations. This may occur especially in highly dynamic and complex systems.

Alternative approaches (e.g., OMNI [4]) deal with this problem. They also define the valid actions in terms of norms, but agents are able to violate such norms [4]. In order to avoid violations those approaches rely on penalties/rewards and implement violation detection mechanisms. However, a new problem may emerge, what happens if the current population of the system is not sensitive to the defined penalties and rewards?

In our opinion, in (norm based) OMAS it is hard to specify a good set of norms at design time. It may not be clear whether the proposed punishments/incentives have the desired influence on the agents nor whether the specified norms may actually effect the global utility in a positive way. Addressing the aforementioned problems, we propose to endow OMAS with a mechanism that tries to induce agents at each moment to act in a way that is appropriate from the point of view of the global utility of the system. The work is inspired by the theory "Economic Analysis of Law", proposed by R.A. Posner in [13]. In this work the author analyses normative systems from an economic point of view. He focuses on the effects of norms in terms of outcomes on both, the behaviour of individuals and the society as a whole. Assuming that individuals are rational, norms are actually incentives that may induce agents to act in a certain way and this, in turn, may have a certain effect on the society. Following these ideas, we propose an adaptive incentive mechanism that: *i)* is able to identify the appropriate agents' actions from the systems point of view, *ii)* estimates agents' preferences, and *iii)* induce agents to act in the desired way by modifying the consequences of their actions.

The paper is organized as follows, Section 2 provides basic definitions and assumptions. Our approach is presented in Section 3. Section 4 describes the experimental validation we have carried out in a p2p scenario. Section 5 puts forward some related work. Finally, Section 6 gives some conclusions and points out lines of future work.

## 2 Definitions and Assumptions

We model an agent participating in a open multiagent system (OMAS) as a rational utility maximizer defined as a tuple $\langle \mathcal{S}, g, \mathcal{U}, t, s_0 \rangle$; where $\mathcal{S}$ is the set of internal states of the agent ($s_0$ is the initial state); $g : \mathcal{X}' \times \mathcal{S} \rightarrow \mathcal{S}$ is the agent's state transition function[3] that assigns a new internal state to the current state and a partial observation of an environmental state $x' \in \mathcal{X}'$; $\mathcal{U} : \mathcal{S} \rightarrow \mathbb{R}$ is the utility function that assigns a value to each possible internal state; and $t$ is the agent's decision function such that $t : \mathcal{S} \rightarrow \mathcal{A}$ follows the principle of maximising the expected utility (MEU). That is, $t(s) = argmax_{a \in \mathcal{A}} eu(a, s) = argmax_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{U}(s') \cdot \overline{P_s}(s'|s, a)$, where $\mathcal{A}$ is the set of possible actions; $eu(a, s)$ is the expected utility of performing the action $a$ in the state $s$; $\mathcal{U}(s')$ is the utility of the state $s'$ estimated by the agent; and $\overline{P_s}(s'|s, a)$ is the agents' estimate, at state $s$, of the probability that state $s'$ will occur when executing action $a$ in state $s$.

[3] In this work we assume that agents have a partial but perfect observation of the environment.

Agents participate in an OMAS that specifies the environment in which they carry out their activities. An OMAS is modelled as a tuple $\langle \mathcal{A}g, \mathcal{A}, \mathcal{X}, \Phi, \varphi, \mathcal{U}, x_0 \rangle$; where $\mathcal{A}g$ is a set of agents, $|\mathcal{A}g|$ denotes the number of agents in the system; $\mathcal{A}$ is a possibly infinite action space that includes all possible actions that can be performed in the system; $\mathcal{X}$ is the environmental state space; $\Phi : \mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \rightarrow [0..1]$ is the OMAS transition probability distribution, describing how the environment evolves as a result of agents' actions; $\varphi : \mathcal{A}g \times \mathcal{X} \times \mathcal{A} \rightarrow \{0, 1\}$ is the agents' capability function describing the actions agents are able to perform in a given state of the environment (*physical* restrictions); $\mathcal{U} : \mathcal{X} \rightarrow \mathbb{R}$ is the utility function of the system that assigns a value to each environmental state; and, finally, $x_0 \in \mathcal{X}$ stands for the initial state.

OMAS are usually designed with a general purpose in mind – the global objective of the system. This objective is represented by means of preferences which are captured by the utility function $\mathcal{U}$ of the system. From the point of view of the designer, the problem consists of how to optimise the global utility of the system assuming that agents will try to optimise their own individual utilities. In order to do this, the agents' behaviour may be influenced by endowing the system with some kind of organisational mechanisms[2]. In particular, in this work we focus on incentive mechanisms.

Our notion of "incentive" is slightly different to the usual consideration of incentives to be something positive. In our work, we consider that incentives are modifications of the environment that have the aim to make a particular action $a$ more attractive than other alternatives $\{b, ...\}$ and such that a rational agent would decide to take $a$. This can be done either by making action $a$ more attractive or by making the alternatives less attractive. In the framework of the given definitions above, an incentive mechanism is a function $\Upsilon_{inc} : \mathcal{X}' \rightarrow [\mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \rightarrow [0..1]]$ that taking into account its partial view of the environmental state, changes the transition probability distribution $\Phi$ of an OMAS. The rationale behind this definition is that changing the consequences of actions may produce variations in the expected utility of agents. Therefore, they would change their decisions accordingly to the new consequences. We say that an incentive mechanism is *effective* if its implementation implies an improvement of the utility of the system $U$.

We make the following assumptions:

**Assumption 1** *The action space in the system is finite.*

**Assumption 2** *the environment of a system can be discretized by a finite set of attributes: $\mathcal{X} = \{X_1, \ldots, X_n\}$. An environmental state $x_i \in \mathcal{X}$ can be modelled as a set of tuples $x_{i,j}$ = $\langle attribute, value \rangle$ assigning a value to each attribute. The incentive mechanism has permission to modify the values of at least some of those attributes.*

**Assumption 3** *agents have multi-attribute utility functions ([10]) defined over some attributes of the environment. The attributes are additively independent. That is, the utility function can be expressed as: $\mathcal{U}(s) = \sum_{j=1}^{n} w_j \cdot u_{i,j}$ where $u_{i,j}$ is the utility of the attribute $x_{i,j}$ perceived in the state $s$ and $w_j$ is the weight of attribute $X_j$. Their preferences do not have to be aligned with the preferences of the system.*

**Assumption 4** *The global utility function of the OMAS is a multi-attribute utility function over the attributes that define an environmental state and where attributes are additively independent.*

## 3 Effective Incentive Mechanism

The proposed incentive mechanism accomplishes two basic tasks: *i)* selecting the actions to be promoted in order to improve the global utility of the system; and *ii)* performing the required changes in the environment in order to make the desired actions more attractive for the agents.

Whereas in standard normative approaches both tasks are solved at design time (by specifying norms), we propose to solve them at runtime. We use learning algorithms that allow to adapt the incentive mechanism to the current agent population as well as to possible changes in the environment.

Similar to the use of an infrastructure in other organisation-based multi-agent systems for regulating the interactions between external agents and the system (e.g., in AMELI in Electronic Institutions [7]), we propose to deploy the incentive mechanism as an infrastructure. We propose to endow an OMAS with institutional agents, called *incentivators*, in charge of implementing the functionality of the incentive mechanism for a particular agent. That is, each external agent is assigned to an incentivator that aims to discover agent's preferences, as well as, to learn how to stimulate it by modifying the consequences of its actions. Furthermore, incentivators can communicate with each other allowing them to coordinate their actions.

### 3.1 Discovering Agents' Preferences

The action selection mechanism of rational agents is based on an ordering over actions, in terms of expected utility. In order to induce such agents to perform a certain action, an incentive mechanism could either modify the consequences of that action in such a way that the resulting state is more attractive, or it could modify the consequences of the rest of the actions in order to make them less attractive. If the agent's preferences are known, this becomes a relatively easy task. However, in open systems, agents' preferences are unknown and, thus, have to be discovered.

One approach to preferences elicitation is either to ask agents a set of questions regarding their preferences. Based on the identified preferences, the utility functions can be estimated [1]. However, this approach has obvious disadvantages, especially in non-collaborative domains, e.g. agents could lie giving their answers. Alternatively, we propose to use a non-intrusive approach where each incentivator discovers its agent's preferences by observing its behaviour in response to given incentives. The characteristics of the discovering process are: *i)* it is a learning process; *ii)* it is independent, i. e., the incentivator does not require to coordinate with any other incentivator; and *iii)* the incentivator receives an immediate local reward, i. e., whether the agent reacts to the modifications in the environment or not. With these characteristics in mind, Q-learning with immediate rewards and $\epsilon$-greedy action selection [15] has been chosen as mechanism to carry out this task. In each step, each incentivator selects the most promising

attribute to modify and a value for this attribute, applies the changes, observes its agents reaction and modifies the q-values for attributes and values accordingly.

For the sake of simplicity, the process has been split up into two different but related learning process. On one hand, the incentivator has to discover the attributes that affect an agent's utility function, and, on the other hand, it has to identify the required values of these attributes in order to make an agent change its mind.

**Learning the attributes that influence agents' behaviour** In the scope of Q-learning, the action space $Z_i$ of the incentivator for agent $ag_i$ is composed of the attributes the incentivator is authorised to modify in the system. More formally: $Z_i \subseteq \{X_1, \ldots, X_n\}$, where $X_j$ are attributes belonging to the environmental state of the system. Thus, when the incentivator takes the action $z_{i,j}$ this means that the attribute $X_j$ is modified. After that, it receives a reward that rates that action, and the action-value function estimation is updated as follows:

$$\mathcal{Q}_{t+1}(z_{i,j}) = \mathcal{Q}_t(z_{i,j}) + \alpha \cdot [\mathcal{R}_t(z_{i,j}) - \mathcal{Q}_t(z_{i,j})] \tag{1}$$

where $\alpha$ is the learning rate and $\mathcal{R}_t(z_{i,j})$ the reward. As we said before, the idea is to discover an agent's preferences by observing how it reacts to the modification proposed. Thus, an action is rated positively if the external agent performs the action the incentivator wanted to, and negatively if not. This is captured by using the following reward function:

$$\mathcal{R}_t(z_{i,j}) = \begin{cases} \text{+1 if agent performed the action} \\ \text{-1 i.o.c.} \end{cases} \tag{2}$$

Besides, in order to explore new attributes, a random selection is made with small probability $\epsilon$, and the highest q-value attribute (*greedy* action) is exploited (chosen in the next step) with probability $(1 - \epsilon)$.

**Learning the values of attributes** The next step in the learning process is to learn the most effective value of the attribute selected. The characteristics of this problem are the same as the attribute learning process. Again, Q-learning with immediate rewards and $\epsilon$-greedy action selection is used. In this case, the action space $Y_{i,j}$ of the incentivator for agent $ag_i$ depends on the attribute $X_j$ selected previously by the attribute learning algorithm. It is composed of the different values that $X_j$ may take. Formally: $Y_{i,j} = \{value \in \left[value_{X_j}^{min}, value_{X_j}^{max}\right]\}$, where $value$ stands for the set of different values the attribute $X_j$ may take[4]. As update and reward functions we use the same formulae as before (equations 1 and 2).

Combining both learning phases, in each step an incentivator proposes a modification – a new value for a pair $x_i^*$=$\langle attribute, value\rangle$. Over time, $x_i^*$ eventually converges towards a pair that influences the behaviour of agent $ag_i$.

[4] This approach requires the set of values to be discrete. In case it is continuous, it should be discretized previously.

### 3.2 Identifying desirable actions

As we have introduced previously, the incentive mechanism has to decide which actions should be incentivized in order to improve the system's utility. In scenarios where the outcome of the action performed by an agent does not depend on the actions taken by others, these actions could be determined locally. However, in many common situations the outcome of an action depends on the joint action of all participating agents. In order to account for this fact, all incentivators should work as a team so as to coordinate the actions to be promoted. The main problem in order to carry out such a task is that incentivators have just a local view of the system – the result of the action performed by "their agents".

Therefore, this task should have the following capabilities: *i)* learning a joint action in a cooperative way; *ii)* dealing with the lack of information about the actions taken by other members of the team; and *iii)* dealing with immediate local rewards. With this in mind, incentivators are endowed with a reinforcement multiagent cooperative learning algorithm that updates the action-value function estimation with the typical Q-learning function (equation 1). As reward function we use the global utility, that is calculated by aggregating the local rewards through a gossip-based algorithm.

**Incentivator's action space** The first issue that needs to be addressed, is determining the action set of an incentivator in this learning task. It is composed of the set of actions its agent ($ag_i$) can take in the current situation, combined with the attribute modification selected by the attribute learning algorithm described earlier. Formally: $V_i \subseteq \{\emptyset, \beta_{i,1}, \ldots, \beta_{i,n}\}$; where $\emptyset$ is the $skip$ action; and $\beta_{i,j} = \langle a_{i,j}, x^*_{i,j} \rangle$, where $a_{i,j}$ stands for an action agent $ag_i$ can perform in the current situation; and $x^*_{i,j}$ is the attribute modification selected as result of the learning process explained in 3.1. When an incentivator takes the action $\beta_{i,j}$, this means that the action $a_{i,j}$ should be made more attractive by modifying its consequences through a change of the $\langle attribute, value \rangle$ pair $x^*_{i,j}$. The action $\emptyset$ means that none of the actions will be promoted (e.g. no parameters of the environment will be changes).

**Calculating the reward** After taking an action ($v_{i,j} \in V_i$), the incentivator $i$ receives a reward that rates such an action, and the action-value function estimation is updated by using the corresponding formula (equation 1). This reward is calculated as follows: $\mathcal{R}_t(v_{i,j}) = \bar{\mathcal{U}}_i(x)$; where $\bar{\mathcal{U}}_i(x)$ is the incentivator's estimation of the global utility in the environmental state $x$ reached after the last step. Since an incentivator has only a local view of the system, it can calculate $\bar{\mathcal{U}}_i(x)$ only based on its local perception of the environment[5]. In order to take into account the actions taken by other incentivators, it should transform its local estimation into an estimation of the global utility. Based on the assumption of additive independence of the attributes in the system's utility function, the global utility can be estimated by aggregating the local utility estimations of all incentivators. In order to perform this task, each incentivator is endowed with the gossip-based aggregation algorithm presented in [9].

[5] We assume that incentivators know the system's utility function.

| **a)** active thread | **b)** passive thread |
|---|---|
| **do** once each $\delta$ time steps | **loop** |
| $j \leftarrow getIncentivator()$ | $\bar{\mathcal{U}}_j(x) \leftarrow receive(*)$ |
| $send\, \bar{\mathcal{U}}_i(x)$ to $j$ | $send\, \bar{\mathcal{U}}_i(x)$ to $sender(\bar{\mathcal{U}}_j(x))$ |
| $\bar{\mathcal{U}}_j(x) \leftarrow receive(j)$ | $\bar{\mathcal{U}}_i(x) \leftarrow update(\bar{\mathcal{U}}_i(x), \bar{\mathcal{U}}_j(x))$ |
| $\bar{\mathcal{U}}_i(x) \leftarrow update(\bar{\mathcal{U}}_i(x), \bar{\mathcal{U}}_j(x))$ | **end loop** |

**Table 1.** Gossip-based algorithm executed by an incentivator

The idea is that each incentivator holds a local value, and by exchanging messages with its neighbours the local values are aggregated by using some aggregation function. Two different threads are executed (see table 1). The active thread periodically initiates an information exchange with a random incentivator $j$ by sending a message containing the local utility estimation $\bar{\mathcal{U}}_i(x)$ and waits for a response with the utility estimation $\bar{\mathcal{U}}_j(x)$ from incentivator $j$. On the other hand, the passive thread waits for messages sent by other incentivators and replies with the local utility estimate. The $update$ method updates the local utility estimation by aggregating the current value and the received value. In our particular use case (see Section 4), we have chosen the average. Therefore, $update(\bar{\mathcal{U}}_i(x), \bar{\mathcal{U}}_j(x))$ returns $(\bar{\mathcal{U}}_i(x) + \bar{\mathcal{U}}_j(x))/2$. This function decreases the variance over the set of all local estimates of the global utility.

### 3.3 Interaction between agents and incentivators

Two different types of interactions could be performed between an incentivator and its agent. On one hand, in order to enable agents to reason about incentives, the incentivator informs its agents about the consequences of the desirable actions. Before an agent selects a new action, it will query the incentivator asking for the possible new consequences of the actions the agent considers[6].

On the other hand, each incentivator observes the reaction of "its" agents to the proposed incentives. This information provides feedback for the preference learning algorithm described earlier. It should be noted that it is possible that an agent may perform an action because of its own interests (and not because of the proposed incentive). We do not have any mean to distinguish such situations. We assume that the exploration/exploitation process will detect such situations and converge to an estimation of the agent's preferences.

## 4 Case Study: a P2P System

We have chosen a peer-to-peer (p2p) file sharing scenario for evaluating our approach. Such scenarios are clear examples of open systems where the objectives of individuals may not coincide with the objectives of the system.

In such systems normally only few peers (*seeders*) have the whole information; and the rest of peers (*leechers*) download pieces by using a particular protocol. We focus

[6] We suppose rational agents will always use this "service" since it is in their own interest.

on peers sharing a file with the BitTorrent protocol [3]. Following this protocol, a file is split in pieces of $256KB$ each and every piece is split in 16 sub-pieces called *blocks*. Peers exchange blocks and each block can be downloaded from different peers. For the sake of simplicity we leave out the phases in which peers and data are identified and peers get a list of neighbours to communicate with (view figure 1). We just focus on the phase carried out to get each block. In this phase, each peer sends a *bitfield* message to its neighbours asking for their file status. After that, the peer has to decide which neighbours will be asked for the next block to download. A *request* message is sent to the selected peers. When a peer receives a request message it has to decide whether the requested block will be uploaded or not. Once a peer accepts the request, it sends a *piece* message containing the requested block. Immediately, the receiver of the piece sends a *cancel* message to the other neighbours it asked for the same block. When the download is finished, a *have* message is sent to the agent's neighbours in order to update their file status.

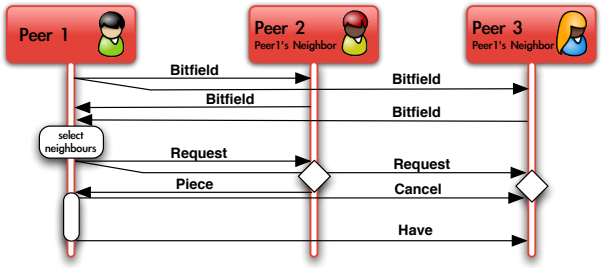



**Fig. 1.** Simplified BitTorrent Protocol

### 4.1 P2P system model

Regarding the most common problems in p2p systems (e.g. non-cooperation of peers, flooding of the network, etc. [8]), the objectives of the system could be specified as follows: *i)* peers should download/upload as many files/blocks as possible in order to increase the number of available files; *ii)* the usage of the network should be as low as possible; and *iii)* the time spent on downloading files should be as short as possible in order to avoid an overload of the network. These objectives might be captured by a multi-attribute utility function as follows:

$$\mathcal{U}(x) = \mathcal{U}_{files} \cdot w_0 + \mathcal{U}_{blocks}(x) \cdot w_1 + \mathcal{U}_{C_n}(x) \cdot w_2 + \mathcal{U}_{C_t}(x) \cdot w_3$$

where, $\mathcal{U}_{files}(x)$, is the utility of the number of already downloaded files, $\mathcal{U}_{blocks}(x)$ represents the utility of the number of downloading/uploading blocks in the state $x$, the greater the number of downloading/uploading blocks, the greater the utility is; $\mathcal{U}_{C_n}(x)$ is the utility of the usage of the network in the state $x$, following the work presented in [11], we have defined this parameter as *network cost* and it represents the sum of the network usage of each message ($c_{m_i}$) sent among agents. It is calculated as follows:

$$C_n = \sum_{i=0}^{\#msgs} c_{m_i} \text{ such that } c_{m_i} = m_{length} \cdot Lat(m_{org}, m_{dst})$$

where $m_{length}$ is the length of a message, and $Lat(m_{org}, m_{dst})$ is the latency between the origin and destiny of the message[7]. The lower the network cost, the greater the utility is. Finally, $\mathcal{U}_{C_t}(x_j)$ is the utility of the time spent on downloading a file (*time cost*). The shorter the time cost, the greater the utility of the system. The partial utilities $\mathcal{U}_x$ are calculated, in case of maximization, as the ratio between the actual value of the parameter $x$ divided by the maximum possible value of the current state (1 minus the ratio in case of minimization). $w_0, w_1, w_2, w_3$ allow us to weight each attribute.

### 4.2 Peer agent model

Peers are modelled as rational agents that follow their own preferences. We focus on two main decisions peers have to make: *i)* to decide to how many neighbours it will send a request message asking for the next block to download; and *ii)* to decide how many requests received from other peers are accepted (i.e.., how many piece messages are send). Accordingly, the action space of a peer is: $\mathcal{A} = \{send_{Piece}(n), send_{Request}(n), skip\}$, where $n$ is the number of neighbours it sends a piece or request message[8].

Peers that join the system obtain a certain bandwidth. Furthermore, we assume that peers have to pay a regular fee in order to connect to the network. Besides, a peer has a file it is sharing and whose status can be partially/completed downloaded. When a peer joins the system it receives a list of peers (neighbours) it can contact. We assume that a peer knows the latency with all its neighbours[9]; and their file status is updated when *have* messages are received.

The attributes that may have some influence on a peer's preferences are bandwidth, fee, number of downloading/uploading blocks of a file and time spent on downloading a file. We capture such preferences through the following general multi-attribute utility function:

$$\mathcal{U}(x) = \mathcal{U}_{bw}(x) \cdot w_4 + \mathcal{U}_{fee}(x) \cdot w_5 + \mathcal{U}_{down}(x) \cdot w_6 + \mathcal{U}_{up}(x) \cdot w_7 + \mathcal{U}_t(x) \cdot w_8$$

where $\mathcal{U}_{bw}(x)$ is the utility of the bandwidth rate - the bandwidth regarding to the available bandwidth; $\mathcal{U}_{fee}(x)$ represents the utility of the fee that has to be paid to connect to the network; $\mathcal{U}_{down}(x)$ stands for the utility of the number of downloading blocks; $\mathcal{U}_{up}(x)$ is the utility of the number of uploading blocks to other peers; $\mathcal{U}_t(x)$ represents the utility of the time spent on downloading a file; and $w_4, w_5, w_6, w_7, w_8$ are the weights assigned to each attribute.

Different kind of peers can be modelled by instantiating the weights of this utility function. For instance, we can model agents interested in having the greatest number of downloading blocks, or agents interested in having the lowest available bandwidth possible, etc.

---

[7] Latency is assumed to be constant and symmetric.

[8] Note that the action that, for instance, sends two requests to neighbours 1 and 2 is different to the one that sends two requests to neighbours 1 and 3.

[9] It can be easily calculated by using ping messages.

### 4.3 Regulating the system

With the aim of improving the system's global utility, we have endowed the system with two different types of regulation mechanisms: a standard normative system, and our incentive mechanism.

The normative system is based on a set of norms coupled with penalties that are applied when norms are violated. In particular, three norms have been designed at design time, that is, before knowing the population of the system: **N1**: "It is prohibited to use more bandwidth than $x\%$"; **N2**: "A peer is obliged to upload a block when at least $y\%$ of the bandwidth is available"; and **N3**: "It is prohibited to request a block to more than the $z\%$ of neighbours"[10]. The set of norms is designed according to the global objective of the system. Norm violations are penalised with an increase of the fee in 5 units. Such violations are detected with a $100\%$ efficiency.

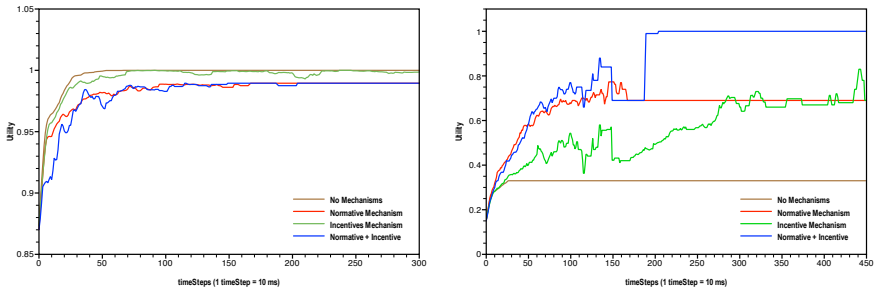
Regarding the incentive mechanism, it is deployed by taking advantages of the own nature of p2p systems. That is, incentivators are located at network service providers. Thus, the communication among them will be fast. Incentivators are authorized to modify the bandwidth assigned to peers and the fees peers pay to connect to the network. So, the action spaces agent $ag_i$'s incentivator are $Z_i = \{fee_{ag_i}, bw_{ag_i}\}$ and $Y_{i,fee}$ = $\{value_{fee_{ag_i}} \in [value^{min}_{fee_{ag_i}}, value^{max}_{fee_{ag_i}}]\}$ in case the attribute selected is the fee; or $Y_{i,bw}$ = $\{value_{bw_{ag_i}} \in [value^{min}_{bw_{ag_i}}, value^{max}_{bw_{ag_i}}]\}$ in case of bandwidth. The actions that can be incentivized by incentivators are $send_{Piece}(n)$, $send_{Request}(n)$ and $skip$, where $n$ can be instantiated by a number of neighbours.

### 4.4 Experimental results

We have conducted some experiments to evaluate our proposal and to compare it to other approaches. We have used the following setup: we have instantiated the p2p system, where peers only have to exchange a single file of 15 blocks. The latencies among all peers are randomly distributed in the range $[10, 400]$ $ms$ and the bandwidths in $[640, 1024, 2048, 4096]$ Mb/s. The fee an agent has to pay to connect the system is randomly selected in the range $[10, 50]$ with steps of $5$ units. In order to calculate the network cost measure of the system a message length for each kind of message is required. The following values have been assigned: piece message = 128Bytes, request and cancel message = 1Byte, and incentivators communication = 1Byte. The simulation environment has been developed at a synchronous way, that is, the execution is performed by time steps. Each time step is equivalent to $10ms$, it means that a request/cancel message sent between two peers connected with a minimum latency of $10ms$ takes $1$ time step to arrive. Moreover, each message has a ttl of $6$ time steps since it arrives to the destiny, that is, a request, not answered in less than $6$ time steps, will be discarded. The utility function of the system is weighted with the following parameters: $w_0 = 0.5$, $w_1 = 0.2$, $w_2 = 0.125$ and $w_3 = 0.175$. Regarding the normative system, norms were initiated with $x = 85\%$, $y = 25\%$ and $z = 85\%$. The learning algorithms in the incentive mechanism are initialised with $\alpha = 0.9$ and $\epsilon = 0.1$. Besides, the q-values are initialised optimistically with $1$ (the maximum value) in order to assure more exploration at the beginning.

[10] $x, y, z$ are percentages that can be defined by the designer

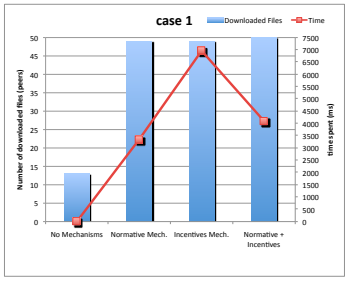
**Experiment 1: Non collaborative agents that are sensitive to the fee** In this experiment the p2p system is populated with $50$ peers. They are modelled as non collaborative agents. That is, they are interested in downloading blocks, but not in uploading. In addition, they are sensitive to modification in the fee they are paying for connecting to the network. This is captured by the following weights in their utility function: $w_4 = 0.2$, $w_5 = 0.399$, $w_6 = 0.3$, $w_7 = 0.001$ and $w_8 = 0.1$. Each peer receives a list of 2-4 neighbours that is randomly generated among all possible peers. Moreover, a $25\%$, as maximum, of peers are *seeders*, and the rest of them are *leechers*. All peers have, at least, 1 seeder as neighbour. Leechers have already downloaded between 0-14 blocks, that is, they aim to download between 1 and 15 blocks. In the normative system, norm violations are penalised with an increase of the fee in 5 units.




(a) Agents utility




(b) System utility




(c) Downloaded files and time spent

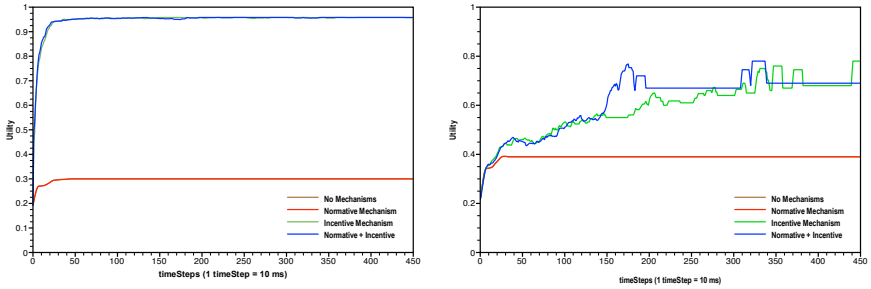
**Fig. 2.** Experimental results experiment 1

The experiment has been executed with four different configurations: *i)* without any regulation, *ii)* with the normative system, *iii)* with our proposal, and *iv)* with both, normative and incentive mechanisms at the same time. Figure 2(a) plots the average utility obtained by all peers participating in the system. As we can see, agents obtain the highest utility when the system is executed without any regulation mechanisms. That is obviously, because nothing restricts the freedom of agents to do what they want. The second best performance is of the incentive mechanism. The reason is that agents are regulated by giving incentives, that is, they are incentivized to perform in a cer-

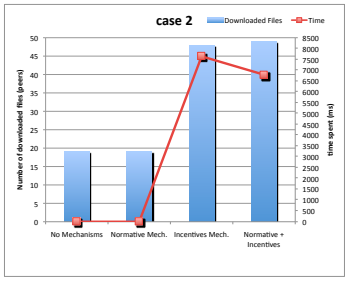
tain manner by paying a lower fee, in this particular case. The normative system and the combination of both mechanisms work similarly "bad" regarding the agents utility. That happens because norms restrict agents' behaviour, so, their utility is lower because of the punishments. Regarding the system, figure 2(b) plots the evolution of the its utility for each configuration. It shows clearly that the systems utility is low if no regulation takes place. This was expected because the analysed population of peers does not behave according to the system's preferences. On the other hand, the normative and incentive mechanisms work similarly well. The incentive mechanism is a bit "slower" at the beginning due to the learning algorithm, that is, incentivators have to discover that the money is the attribute that has influence on the peers. Finally, the combination of incentive and normative is the one that performs best. It could happen because norms restrict the valid actions and the incentive mechanism induces agents to perform the desirable actions, so utility is always improved. This is also shown in figure 2(c), which compares the four mechanisms regarding to the number of peers that are able to download the whole file and the time spent on it. By using the combination of incentive and normative, all peers (50) are able to download the whole file. On the other hand, 49 peers are able to download the file when using the incentive and the normative mechanisms respectively. However, the time spent is higher in case of the incentive mechanism, due to the learning process, as it has been pointed out previously.

Concluding, in the case of previous knowledge about the agents preferences and where norms have been designed by taking into account such information, the incentive mechanism works similar to the normative system.

**Experiment 2: Non collaborative agents that are not sensitive to the fee** In this case the system is populated by 50 peers that are not collaborative as well. This is captured by the following weights in their utility function: $w_4 = 0.2$, $w_5 = 0.001$, $w_6 = 0.399$, $w_7 = 0.001$ and $w_8 = 0.399$. We have specifically chosen a peer population that is not sensitive to changes in the fee they are paying, e.g., they do not care about increasing fees. This can be seen from the weights in the agents' utility function (in particular $w_5 = 0.001$). This describes a case where the designed norms (all based on an increase in the fee as a punishment) will not be very effective for the given population of agents. Figure 3(a) plots the average utility obtained by all peers. The results are clearly better when the incentive mechanism is employed, either alone or coupled with the normative system. The results are clearly better when the incentive mechanism is employed. Implicitly, this mechanisms is able to identify that instead of the fee, the bandwidth has an influence on peers' utility. It uses changes in the available bandwidth to make the upload of blocks to other peers attractive. Regarding the system, figure 3(b) plots the utility of the system when it is regulated by the different mechanisms. As it was expected, the system improves its performance when it is regulated by the incentive mechanism because it is able to influence agents' behaviour regarding the sending of their blocks. Finally, in figure 3(c) we can see the number of peers that are able to download the whole file. In the case of normative and no mechanisms, none of the peers are able to download the files, only the initial seeders have the whole file, that is why the time spent on downloading the files is zero. On the other hand, with the incentive mechanism 48 out of 50 peers download the whole file, by spending 7640ms.




(a) Agents utility (b) System utility




(c) Downloaded files and time spent

**Fig. 3.** Experimental results experiment 2

In contrast, when the incentive mechanism is coupled with the normative system the results are improved and 49 out of 50 peers download the whole file spending less time (6770ms). This may be due to the fact that the influence of the fee on the utility of agents is not zero, and thus, the punishments together with the incentives may have a better behaviour.

It is important to note that the overhead introduced as a consequence of the gossip algorithm is taken into consideration in the *network cost* attribute calculated in the system utility function. That means, the incentive mechanism is beneficial even if the overhead is taken into account. The experiment makes clear that a normative system does not fulfil its goal if the agents are not sensitive to the applied punishments. In contrast, the proposed incentive mechanism is able to adapt to such situations.

## 5 Related Work

Many approaches have been proposed, in the field of multi-agent organisations, to regulate the activities of agents in a MAS [6, 4]. In most of them, the concept of *norm* appears as a main piece. Some approaches, like [6], focus on defining the set of allowed actions in each possible state of the system and assure that agents are just allowed to perform the valid actions. We think our approach is compatible with such approaches. Our work can be used for inducing a particular action, among all possible valid actions, optimising in that way the system's efficiency. Other approaches, like [4], propose to

couple norms with penalties/rewards which give agents the possibility to violate a norm. In these approaches norms are usually defined at designed time, what implies that some assumptions are made with regard to the agent population (e.g. the attributes that have an influence on agent's preferences). In our work, we do not assume a priori knowledge about agents' preferences; our approach tries to learn such preferences for each agent and, thus, provides a basis for an individualised incentivation/punishment mechanism.

Our work is quite related to mechanism design [12], as it aims at influencing the behaviour of rational agents in a desired direction. However, instead of directly defining the (interactions) rules of the system, we try to achieve desired behaviour of agents by modifying their environment at runtime (i.e. agents interact only through environment). Furthermore, we do not assume that the agents' pay-off functions are known, but we intend to learn which of the environment attributes are relevant for an agent's pay-off, so as to dynamically adjust incentives for it at runtime.

Recently, some papers have been published with a similar focus as ours [14, 5, 16, 17]. All of them address the same problem, how to influence agents' behaviour in order to induce some desirable behaviour. As solution they focus on formal approaches based on environment design techniques, incentive based approaches and behaviour cultivation. They prove by experimental results that their approaches effectively induce desirable behaviour. However, they do not deal with scalability issues. Thus, the main contribution of our work regarding to them, is that we propose to deploy the mechanism by using an infrastructure where the control and computation is distributed.

Regarding p2p systems, in [11] the authors propose to regulate and adapt a p2p system by learning and establishing new social conventions with the aim of improving the system's performance. The main difference to our work is that they assume a certain control over agents, because when a new convention is learnt, it is established and assumed that agents will follow it. Other works have proposed incentive-based solutions for p2p related problems (e.g., [8]). However, it is usually assumed that some attribute have an influence on peers' preferences and punishments and incentives are defined based on this assumption.

## 6 Conclusion

In this paper we have put forward an incentive mechanism that is able to induce desirable behaviour by modifying the consequences of actions in open MAS. The mechanism tries to discover which attributes have some influence on agents' preferences and learns how agents can be incentivized. It is deployed by using an infrastructure based on institutional agents called *incentivators*. The incentivators use the Q-learning algorithm to discover agents' preferences by observing how they react to modifications in the environment. Moreover, they learn – in a cooperative way, using Q-learning and a gossip-based reward calculation algorithm – which joint actions should be incentivized in order to increase the utility of the system. Finally, the proposed mechanism has been tested in a p2p file sharing scenario, showing that it is a valid alternative to standard normative systems. In particular it outperforms regulation mechanisms based on fixed norms if the design assumptions of such norms are not fulfilled.

As future work, we plan to explore other learning techniques that may be more suitable in scenarios where agents preferences may vary over time. In principle, Q-learning can deal with such situations, but it may be too slow to obtain the desired adaptation of the system. Furthermore, the mechanism is currently able to perform the modification of just one attribute at the same time. In this sense, we will explore the possibility of modify a set of attributes jointly.

## References


1. Boutilier, C., Patrascu, R., Poupart, P., Schuurmans, D.: Regret-based utility elicitation in constraint-based decision problems. In: IJCAI'09. pp. 929–934 (2005)
2. Centeno, R., Billhardt, H., Hermoso, R., Ossowski, S.: Organising mas: A formal model based on organisational mechanisms. In: SAC: 24th Annual ACM Symposium on Applied Computing. pp. 740–746 (2009)
3. Cohen, B.: The bittorrent protocol specification. $http : //www.bittorrent.org/beps/bep\_0003.html$
4. Dignum, V., Vazquez-Salceda, J., Dignum, F.: OMNI: Introducing social structure, norms and ontologies into agent organizations. In: PROMAS'04. vol. LNCS 3346, pp. 181–198 (2004)
5. Dufton, L., Larson, K.: Multiagent policy teaching. In: AAMAS'09 (2009)
6. Esteva, M., Rodriguez, J., Sierra, C., Garcia, P., Arcos, J.: On the formal specification of electronic institutions. Agent Mediated Electronic Commerce LNAI 1991, 126–147 (2001)
7. Esteva, M., Rosell, B., Rodríguez-Aguilar, J., Arcos, J.: AMELI: An agent-based middleware for electronic institutions. In: AAMAS'04. vol. 1, pp. 236–243 (2004)
8. Hales, D.: From selfish nodes to cooperative networks - emergent link-based incentives in peer-to-peer networks. In: Proc. of the 4th Int. Conf. on Peer-to-Peer Computing. pp. 151–158. Washington, DC, USA (2004)
9. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(3), 219–252 (2005)
10. Keeney, R., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge University Press (1993)
11. Miralles, J.C., López-Sánchez, M., Esteva, M.: Multi-agent system adaptation in a peer-to-peer scenario. In: SAC'09. pp. 735–739. ACM, New York, NY, USA (2009)
12. Parsons, S., Wooldridge, M.: Game theory and decision theory in multi-agent systems. Autonomous Agents and Multi-Agent Systems 5, 243–254 (2002)
13. Posner, R.A.: Economic analysis of law. Little Brown (1977)
14. Rabinovich, Z., Dufton, L., Larson, K., Jennings, N.: Cultivating desired behaviour: Policy teaching via environment-dynamics tweaks. In: AAMAS'10. pp. 1097–1104 (May 2010)
15. Watkins, C.: Learning from Delayed Rewards. Ph.D. thesis, King's College, Cambridge, UK (1989)
16. Zhang, H., Parkes, D.: Value-based policy teaching with active indirect elicitation. In: AAAI'08. pp. 208–214. AAAI Press (2008)
17. Zhang, H., Parkes, D.C.: Enabling environment design via active indirect elicitation. In: Proc. Workshop on Preference Handling. Chicago, IL (July 2008)