Multi-Agent System adaptation in a Peer-to-Peer scenario

Jordi Campos Miralles
Universitat de Barcelona
585 Gran Via
08007 Barcelona, Spain
jcampos@maia.ub.es

Maite López-Sánchez
Universitat de Barcelona
585 Gran Via
08007 Barcelona, Spain
maite@maia.ub.es

Marc Esteva
Artificial Intelligence Research
Institute (IIIA, CSIC)
Campus UAB
08193 Bellaterra, Spain
marc@iiia.csic.es

ABSTRACT

From a system's perspective —as opposed to an individual agent perspective—, MAS adaptation is now becoming an important topic, since it can help to obtain expected outcomes under changing circumstances. In this paper, we propose a MAS architecture (2-LAMA) to adapt social conventions in dynamic systems. The proposed architecture consist of two layers: the conventional MAS system (we call *domain-level*) and an additional layer or *meta-level* in charge of adaptation. A Peer-to-Peer scenario helps us to illustrate our approach. The resulting model changes participant organisation depending on environment and agent changes. Finally, we present some preliminary results of the empirical evaluation of our approach.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*MultiAgent Systems*

General Terms

Design, Algorithm.

Keywords

Multiagent Systems, Assistance, Coordination, P2P.

1. INTRODUCTION

A Multi-Agent System (MAS) consists of a set of agents that interact among them within an environment. At individual level, agents adapt their behaviour to better accomplish their goals upon changes in their environment. On the other hand, from a system's perspective, MAS adaptation is now becoming an important topic, since it can help to obtain the expected outcomes under changing circumstances. We approach MAS adaptation through the modification of *social conventions*.

MAS are distributed by nature, and so it should also be its adaptation mechanism. This would avoid centralisation limitations such as fault-tolerance or global information unavailability. Accordingly, we propose adaptation to be done by means of an additional distributed layer (we call *meta-level*) on top of a regular MAS. We suggest that this *meta-level* updates *social conventions* when societal or environmental changes occur. As a motivating scenario, we work on a Peer-to-Peer (P2P) sharing network. In such network, computers contact among them to share some data. The relationships they establish change over time depending on network status. Our vision is that these relationships define the system's organisation (i.e. how computers organise themselves to interact), whereas changes in network status constitute its dynamic environment.

Generally, related work on MAS adaptation uses as basis a system description. It is used by the adaptation mechanism when deciding how to update the system. Most of them assume it is feasible to identify which tasks are necessary to achieve system goals. For instance, in [6] once they have identified required tasks, they can assign them to available agents —they know their capabilities— and establish their organisation depending on task dependencies. In [7] they go a step further because they can derive new required tasks related to coordination issues.

However, we are interested in contexts where it is not possible to identify which tasks achieve system goals. For example, in a traffic scenario we want to decrease the number of accidents and save control resources [2], but we cannot identify which tasks are necessary to achieve it. Moreover, such contexts usually have norms, and their relationship with global goals can become even more complex. Other works, such as [1] or [5], share our interests. In [1], agents update social norms by agreement without dealing with goals. On the other hand, agents in [5] change their local *conventions* in a P2P scenario but keep global norms static. [5] also has an additional layer, but with supervision purposes only. Similarly, *organisational agents* of [4] have an extra layer, although they assume the mapping between tasks and goals previously mentioned.

The rest of the paper is structured in seven sections. Section 2 depicts our scenario. Sections 3 and 4 present our model and its application to this scenario. Next, sections 5 and 6 show the results obtained with the current implementation. Finally, section 7 presents the conclusions and future work.

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

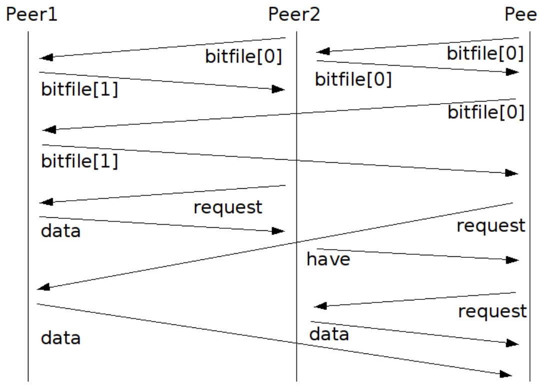

Figure 1: P2P simplified protocol. "`bitfile[1/0]`"≡peer does/doesn't have data.

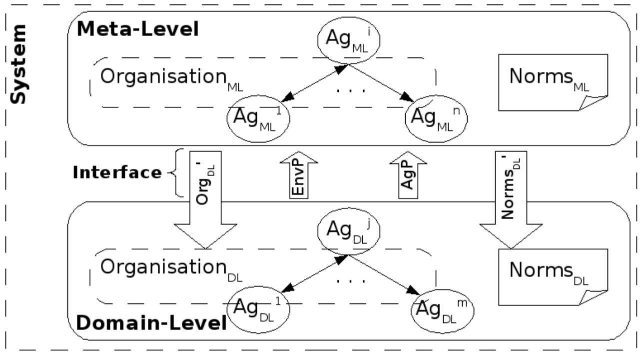

Figure 2: General Model.

2. PEER-TO-PEER SCENARIO

Our case of study is a P2P scenario, where a set of computers connected to Internet (*peers*) share some data. Initially, not all of them have such data but they interchange pieces of it to collect the whole information.

In this scenario, we consider *peers* as software agents that act on behalf of human users that request this information. Therefore, agents need to contact other agents in the P2P network. Time is a valuable resource, and so, the faster the data is obtained the better for the user. Similarly, it is a requirement to do it using as little network bandwidth[1] as possible. Thus, although a *peer* could potentially contact all other *peers*, it usually contacts only a subset of them to save its network resources. The performance of a system will be computed in terms of time and network consumptions. The actually used net of connections among *peers* is called *overlay network*. We see this *overlay network* as the social organisation of these agents.

Peers interact through Internet, an open network. This means it is a dynamic environment, as connection quality can change over time. Thus, *peers* tend to re-organise when there are changes in connection quality or population. In fact, all users would potentially benefit from low network usage because it reduces Internet overload. Hence we could even think of some general norms that agents should follow in order to minimise network overload —like controlling its bandwidth consumption.

Overall, this P2P case study seems to be representative of scenarios that require MAS adaptation research: it can be modelled as a dynamic MAS (in terms of environment and population) that requires changes on its social conventions (social structure and norms). However, real P2P networks are highly complex. So we try to reduce complexity by assuming some simplifications. The rest of this section provides the details of our actual scenario.

Before sharing data, real P2P networks require to identify this data and the peers sharing it. In our scenario, we obviate these initial phases and just focus on peer communication to obtain the data. We also assume shared data has a single piece, so *peers* change their state to *completed* when they get it. Besides, we express communication quality as latency: time required for a message of size 1 to be transmitted among two peers (we assume it is symmetric and constant).

Regarding peer communication protocol, we use a simplified[2] version of BitTorrent [3] protocol shown in Figure 1. It has an initial handshake phase in which *peers* indicate if they have the datum, and a second phase in which they request the datum to those *peers* having it. In the example of previous figure, communication among `Peer1`-`Peer2` and `Peer2`-`Peer3` is assumed to be faster than `Peer1`-`Peer3`. Consequently, `Peer3` receives the datum from `Peer2` before than from `Peer1`.

3. GENERAL MODEL: 2-LAMA

We propose a Two Layer Assisted MAS Architecture (2-Lama) that is able to adapt the social conventions of a regular MAS to changes in its environment and/or participants. With this aim, we add an additional layer *meta-level* (ML) on top of the previous system we call *domain-level* (DL). This new layer may also have its own social conventions which could, in turn, be adapted by a new *meta-level*. The model can thus have as many layers as required. Nevertheless, since extra layers would not require new specifications, we focus on describing first two levels. In addition, we define a communication interface (Int) among both levels. Thus, our model can be expressed as: $M = \langle ML, DL, Int \rangle$.

Each level has a set of agents (Ag_{xL})[3] and its social conventions defined by a *social structure* or organisation (Org_{xL}) and a set of *norms* (Nor_{xL}). Hence, each level can be defined as: $xL = \langle Ag_{xL}, Org_{xL}, Nor_{xL} \rangle$ (see Figure 2). On the one hand, we see the social structure as a set of roles (Rol_{xL}) and the relationships (Rel_{xL}) among agents playing them: $Org_{xL} = \langle Rol_{xL}, Rel_{xL} \rangle$. Although *roles* among levels differ, a single agent may play different roles at different levels if it is authorised. On the other hand, norms limit agent's behaviour and are expressed as first-order deontic logic formulae to define agents permissions, prohibitions and obligations

Furthermore, communication among levels covers bottom-

---

[1] This bandwidth is the capacity to transfer data over user's network connection. The less is used by the *peer*, the more is left for other purposes.

[2] The actual BitTorrent protocol [3], has an extended handshaking, a queue management and a cancel message to avoid retrieving data once it is already received from another *peer*.

[3] The suffix xL is a generalisation of ML and DL.

up (Up) and top-down (Dn) information exchanges: $Int = \langle Up, Dn \rangle$. The *meta-level* perceives *domain-level* observable properties —through the Up channel—, evaluates them, and adapts *domain-level* social conventions —through the Dn channel. Perceived properties are those that can be observed in the environment ($EnvP$, e.g. date, temperature...) and those that can be observed in agents (AgP, e.g. colour, position...) —i.e. $Up = \langle EnvP, AgP \rangle$. While adapted social conventions correspond to new organisation (Org'_{DL}) and norms (Nor'_{DL}) of the *domain-level* —i.e. $Dn = \langle Org'_{DL}, Nor'_{DL} \rangle$.

In summary, we suggest to add an abstraction layer (*meta-level*) in charge of adapting existing social conventions (Org_{DL} and Nor_{DL}) depending on environment and participant properties ($EnvP$ and AgP). We assume each *meta-level* agent ($a_{ML} \in Ag_{ML}$) has partial information about such properties, so it only perceives a subset of $EnvP$ and AgP. This assumption relies on the fact that in many scenarios global information is not available due to information spread costs or privacy issues, for example. Thus, an a_{ML} has aggregated information about a subset of *domain-level* agents that can partially share with other *meta-level* agents. The decisions to update the *domain-level* social conventions may be made by a single a_{ML} or may require an agreement or consensus among a set of them.

4. PEER-TO-PEER MODEL

In this section we apply the general model to the P2P scenario. We define the two layers: the *domain-level* (DL), which corresponds to peers sharing data; and the *meta-level* (ML), in charge of updating social conventions to improve system's performance.

Participant agents in the *domain-level* (Ag_{DL}) play a single role called "`peer`", so $Rol_{DL} = \{peer\}$. Their network connections are represented as arcs connecting nodes in a weighted graph (costs correspond to latencies). This is, a complete graph, since each agent can potentially contact any other agent through Internet (see Figure 3). Nevertheless, peers usually contact just a subset of neighbours, defining an *overlay network*. In our model, we define it as the relationships among agents (Rel_{DL}) that form a sub-graph of the network graph. These relationships, which belong to the agents' organisation, will be updated by the *meta-level* taking into account the system status. To keep the model simple, we assume latencies do not change over time. However, we add a norm to Nor_{DL} that can limit the bandwidth *peers* are allowed to use. More specifically, it limits the number of message units a *peer* can send at each time step. This way, *peers* could not use the network as an infinite resource. We assume agents follow current social conventions.

Regarding our *meta-level*, it also has a single role, called "`assistant`", so $Rol_{ML} = \{assistant\}$. Each agent in Ag_{ML} playing this role collects information about a set of *peers* ($cluster \subset Ag_{DL}$, we assume *clusters* are disjoint) and adapts their local organisation (Org_{DL}). Its decisions are based in local information about its associated *cluster* —such as latencies ($EnvP$) or *peers* having the data (AgP)— and information about other *clusters* they get from their neighbours in the *meta-level* organisation (Org_{ML}). Furthermore, norms (Nor_{ML}) also exist at this level[4], so that they define, together with Org_{ML}, the *meta-level* social conventions.

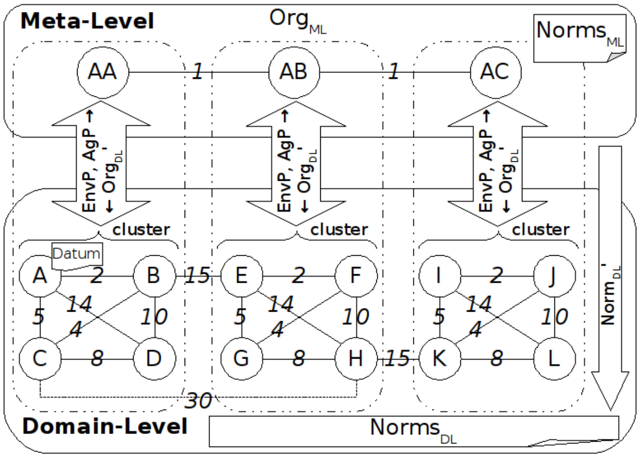

Figure 3: P2P model example. Peer potential interconnection can be expressed as a complete graph (missing arcs have latencies of 30). Note that each *assistant* only perceives information about a subset of this graph. Interface's latencies are 1.

Finally, we assume communications at *meta-level* and among levels are faster than communications at *domain-level*. This is because, in the P2P scenario, *assistants* could be located at Internet Service Providers (ISP) which have better communication among them than their clients (i.e. *peers*). Similarly, as any communication of a client requires to pass trough its ISP, communication between a *peer* and its *assistant* would be faster than among *peers*.

4.1 Extended protocol

The simplified P2P protocol —previously introduced at Figure 1— deals with communication at the *domain-level* but requires an extension to include both communications at *meta-level* and among levels.

Our extended protocol starts with a *peer* handshaking its *assistant* with a "`join <hasDatum>`" message ("`hasDatum`" is a Boolean variable that specifies if the *peer* has the datum that is being shared). Then, the *assistant* asks the *peer* to measure its latencies with all other *peers* in its *cluster*. This is done by sending "`get_latency <peer>`" messages. As a consequence, the *peer* measures latencies by means of PING[5] messages, and informs back the *assistant* with a "`latency <amount>`" message. Once an *assistant* has all latencies among their *peers* ($EnvP$) and knows which ones have the datum (AgD), it estimates which would be the best re-organisation. Then it adapts the agent *relationships* ($Rel'_{DL} \in Org'_{DL}$) by sending "`contact <peer> [,<peer>]`" messages to all the *peers* in its *cluster*.

From this point on, the simplified P2P protocol —again see Figure 1— is performed among the *peer* and its neighbours. This time, however, when a *peer* receives the datum it also informs its *assistant* with a "`completed`" message. Then, at *meta-level* this *assistant* informs its neighbour *as-*

---

[4] For instance, a limit in the number of *peers* and *assistant* can ask to contact other *peers*.

[5] A Packet InterNet Groper (PING) can be used to estimate the round-trip time of a data package among network nodes.

sistants with a "`completed_peer <peer>`" message. For instance in Figure 3, when B receives the datum, it informs AA, which will inform AB but not AC. Next, contacted *assistants* spread this information towards their *peers* with a "`has_datum <peer>`" message (i.e. AB informs E, F, G and H that B has the datum). In that moment, agents measure their latencies to the new *peer* and request the datum if it is better than any previous source (i.e. F will request the datum to B, but when E gets it, F will also request it to E; afterwards, even if D gets the datum, F won't request it to D because E is a faster source). Finally, when an *assistant* detects that all its *peers* are completed, it sends an "`all_completed`" message to its neighbour *assistants* to avoid receiving more "`completed_peer`" notifications.

4.2 Assistant decisions

Our P2P model does not restrict how *assistants* make their decisions to re-organise and adapt norms. It only specifies the information they have available. Nevertheless, we briefly describe which algorithm they use in our current implementation. Mainly, an *assistant* faces two different situations: (a) some *peers* in its *cluster* already have the datum, or (b) no *peer* in its *cluster* has it yet.

In the first case (a), the *assistant* computes the shortest paths —using Dijkstra's algorithm over arc latencies— from each *peer* having data to the rest of *peers* in the cluster (in case there are several source nodes, the minimum shortest path is considered instead). Then, it re-organises its *cluster* by telling each *peer* to contact with its predecessor in its shortest path to a data source. For example, in Figure 3, AA will tell D to contact B. This way, the graph of new *relationships* (Rel'_{DL}) may have different arcs than the old relationships (Rel_{DL}), although both are sub-graphs of the under-laying possible communications.

In the second case (b), the *assistant* organises its *cluster* to be prepared for data entering through any *peer*. Accordingly, it assumes any *peer* can become a data source and computes all possible shortest paths. Next, it provides to each *peer* its predecessors in all its corresponding shortest paths. This way, all *peers* are in contact with the neighbours that could provide rapidly the data when it enters through any node in the *cluster*. In the example, AB would tell H to contact F if the datum is in E, but it would tell H to contact G if the datum is in G. Thus, AB will tell H to contact F and G. The resulting *relationship* graph (Rel_{DL}) is larger than in previous case (a) but considering the information available, it still smaller than all possible *relationships* ($EnvP$).

In both cases (a, b) the contact among *peers* that would not be used to transfer data is avoided, so that the corresponding network usage is saved. For now, these changes are organisational adaptations (Org'_{DL}). However, we've being experimenting with different bandwidth norm values ($Norm'_{DL}$) to study norm adaptation algorithms in future.

4.3 Evaluation

We can evaluate the resulting system performance in two dimensions: time and network usage. Accordingly, we define their corresponding measures. On the one hand, we define the *time cost* (c_t) as the number of time steps from the start of simulation up to when all nodes have the datum. On the other hand, we define the *network cost* (c_n) as the network usage of each message (c_{m_i}) sent among agents: $c_n = \sum_{i=0}^{\#msgs} c_{m_i}$. This usage depends on the message's length (m_{length}) and the latency among its origin (m_{org}) and destination (m_{dst}) agents, expressed as: $c_{m_i} = m_{length} \cdot Lat(m_{org}, m_{dst})$, being $Lat : \mathbb{A}g_{DL} \times \mathbb{A}g_{DL} \rightarrow \mathbb{Z}$.

5. EXPERIMENTS

We have tested our proposed Two Layer Adaptive MAS Architecture (2-LAMA) on the P2P scenario depicted in Figure 3. Its implementation —in Repast Simphony— uses the evaluation criteria described in previous subsection 4.3. This criteria requires message lengths (m_{length}). We assign their values depending on message types and layers: at DL, ping = 1, data = 10 and control[6] = 2; at ML, messages among *assistants* = 2 ; at Int, messages among *peers* and *assistants* = 2.

Since we propose to add a distributed *meta-layer*, we name our implementation *Distributed*. Besides, in order to have reference performance values, we also present two alternative implementations: *All4All* and *Centralised*. Firstly, in *All4All*, all *peers* contact each other at the beginning, and then request data from sources along all possible paths. *All4All*'s parallelism guarantees minimum execution time (c_t), but its lack of *meta-layer* does not prevent maximum network cost (c_n), since all *peers* exploit all their communication alternatives simultaneously. Secondly, *Centralised* implements a *meta-layer* composed by a single *assistant* agent. This agent has global information so that it can make fully informed decisions when computing shortest paths (see subsection 4.2). As a consequence, it recommends the optimal neighbour to each *peer*, and thus, guarantees the minimum network cost (c_n). *Centralised*'s execution time is slightly longer than *All4All*'s, though. This is because all *peers* in *All4All* send the handshaking (`bitfile`) simultaneously, whereas, in *Centralised*, handshaking is a dialogue: answers are sent once `bitfile`s are received (see Fig. 1).

Finally, it is worth mentioning that current simulations start once all *peers* have contacted their *assistants*. In future work, we plan to simulate the initial phase, in which *peers* join gradually and *assistants* collect information about latencies and data sources. We estimate it will have almost no impact on *All4All*, slight impact on *Distributed* —only local information is collected— and a larger impact on *Centralised*.

6. RESULTS

The results correspond to the execution of the three alternatives with the *peers* and network latencies depicted in Figure 3. Alternatives are simulated with different bandwidth limits (Nor_{DL} ="max BW message units per time step") as explained in section 4. In *Centralised* and *Distributed*, there are also different simulations for various network latencies between *domain-level* and *meta-level*, and among *assistants* –we assume both have the same value (L_{x2a}). Each combination has been executed once with the datum in each *peer*. Results in table 1 show the round average of these executions in time (c_t) and network (c_n) costs.

Results confirm our minimum and maximum costs assumptions. Generally, *All4All* requires the minimum time[7]

---

[6] Control messages at DL are: `bitfile`, `request` and `have`.

[7] Only when limiting bandwidth to 1, *All4All* takes a little

		All4All		*Centralised*		*Distributed*	
BW	L_{x2a}	c_t	c_n	c_t	c_n	c_t	c_n
1	1	525	25480	512	2896	648	5926
1	5	-	-	”	2984	688	6348
4	1	476	25053	488	2896	618	5971
4	5	-	-	”	2984	653	6394
∞	1	464	24976	481	2896	610	5979
∞	5	-	-	”	2984	645	6402
∞	30	-	-	”	3534	861	8939

Table 1: Resulting costs in simulations

but uses the maximum network, whereas *Centralised* consumes the minimum network. Indeed, they show our proposal of adding a *meta-level* is worth, since the cost derived from adding it is less than its benefit. Specifically, the results of the *Distributed* approach show that adding the *meta-level* provides more savings in network usage than expenses in time. For instance, being $BW = \infty$ and $L_{x2a} = 1$ our *Distributed* approach requires 31% more time than *All4All* but saves 76% network costs. In fact, the *Distributed* is an intermediate point in network consumption among *All4All* —its *peers* need to discover its shortest path to data sources— and the *Centralised* —its *assistant* already has all the information. In *Distributed*, *assistants* already have knowledge about its cluster, but *peers* are required to discover the shortest path with data sources outside its cluster. Currently, *assistants* tell all their *peers* to discover these shortest paths. But in future work, we plan to add a norm to *meta-level* (Nor_{ML}) to limit the number of *peers* an *assistant* can send "`has_datum <peer>`" messages. Even if we increase communication latencies with the new layer (L_{x2a}) up to the maximum of *domain-level* —30 among *peers*— it still uses less network than having no *meta-level* at all (*All4All*). For example, being $BW = \infty$ and $L_{x2a} = 30$, *Distributed* consumes 64,2% less network than *All4All*. We test up to this latency to prove that the advantages of our solution were not related to having faster communication channels, but they derive from doing a better use of these channels. This means, that having some agents playing both roles —*peer* and *assistant*— without having fast *assistant* agents located at ISPs, our approach still saves network resources. In *Centralised*, network usage increases with L_{x2a}because *peers* also inform the *assistant* when they are completed. However, as they send their neighbours the "have" message before contacting the *assistant*, the execution continues regardless of L_{x2a}, so the required time is constant.

In addition to prove the benefits of organisational adaptations, we wanted to check the effects that could have adapting our proposed norm. Results show that limiting bandwidth influences over network consumption. However, with current configuration it has more impact on execution time than on network usage. For instance, being $BW = 1$ and $L_{x2a} = 1$, *Distributed* saves 0.8% network but requires 6.2% more time than without bandwidth limitation ($BW = \infty$ and $L_{x2a} = 1$). In future work, we plan to simulate network traffic jams to increase the influence when changing the norm. This way, we could study its adaptation.

7. CONCLUSION

In this paper we presented the Two Layer Assisted MAS Architecture (2-LAMA) that adds a *meta-layer* in charge of a system adaptation to dynamic changes. The proposed adaptation is distributed requiring no global information. As a case study we introduced a P2P scenario to which we apply our model obtaining an adaptive P2P MAS. We provided means to evaluate it and we designed some alternatives and experiments to contrast its benefits.

The experiments showed interesting results, notably the fact that the cost of adding the *meta-layer* is lower than the obtained benefit. We conclude it is feasible and worth to add our proposed *meta-layer*. In future works, we plan to experiment different configurations with our current norm, and work on its adaptation by the *meta-layer*. Even more, we plan to experiment with a norm at *meta-layer* level to bound its weight over the rest of the system (i.e. limit the number of *peers* and *assistant* can tell that another *peer* has data). Besides, we want to update latencies depending on network traffic and study how our approach adapts to these environmental changes. In the medium term, we would like to deal with open MAS, where agents can join and leave and transgress social conventions. Currently, *meta-layer* provides adaptation directions that agents follow, but we think about providing advices to agents. We envision an open MAS with an *assistant layer* that improves the *coordination support* the infrastructure provides to its agents.

Acknowledgements. This work is partially funded by IEA (TIN2006-15662-C02-01), AT (CONSOLIDER CSD2007-0022) projects, by EU-FEDER funds, by the Catalan Gov. (Grant 2005-SGR-00093) and by Marc Esteva's Ramon y Cajal contract.

8. REFERENCES

[1] Artikis A., Kaponis D., and Pitt J. *Multi-Agent Systems: Semantics and Dynamics of Organisational Models*, chapter Dynamic Specifications of Norm-Governed Systems. 2009.

[2] Jordi Campos, Maite López-Sánchez, Juan A. Rodríguez-Aguilar, and Marc Esteva. Formalising situatedness and adaptation in electronic institutions. In *COIN 2007 Workshops*, 2008.

[3] Bram Cohen. The BitTorrent Protocol Spec. http://www.bittorrent.org/beps/bep_0003.html.

[4] Scott A. Deloach, Walamitien H. Oyenan, and Eric T. Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16(1):13–56, 2008.

[5] A. Grizard, L. Vercouter, T. Stratulat, and G. Muller. A peer-to-peer normative system to achieve social order. *LNCS*, 4386:274, 2007.

[6] Kota R., Gibbins N., and Jennings N. Decentralised structural adaptation in agent organisations. In *AAMAS Workshop on Organised Adaptation in MAS*, 2008.

[7] Mark Sims, Daniel Corkill, and Victor Lesser. Automated Organization Design for Multi-agent Systems. *Autonomous Agents and Multi-Agent Systems*, 16(2):151–185, 2008.

---

bit more time than *Centralised* because *All4All peers* start contacting all nodes at once, and *Centralised peers* have a single neighbour to contact —the one in the shortest path.