Norms in 2-LAMA

Jordi Campos*, Maite López-Sánchez† and Marc Esteva‡

Research Report RR-IIIA-2008-05

Abstract

As a MAS can experience a lot of changes about its participants, or even about system's external context, its original organisation may not lead to its design goals. Thus, adapting such organisation is now becoming an important topic, since it can help to obtain the expected outcomes under changing circumstances. This work presents 2-LAMA: an architecture in two layers that brings system-wide adaptation to an open MAS organisation. It has a distributed *Assistance* layer, the so-called *meta-level*, that perceives information about agents and their environment, and is able to decide how to adapt the system's organisation. Therefore, we endow the system with adaptation capabilities through an additional layer instead of expecting the agents to increase their behaviour complexity. Specifically, we deal with organisations that include norms to regulate participants' activity. In this paper we describe our general model and our case study: a P2P sharing network. We apply the model to the case study, detailing the norms it contains. Furthermore, we evaluate empirically our approach by means of a series of experiments. These experiments provide positive results on norm effects and our adaptation process.

1 Introduction

A Multi Agent System (MAS) is conceptualised in terms of distributed autonomous pieces of software (agents) that interact among them. Developing MAS entails the problems of designing a distributed concurrent system plus the difficulties of having flexible and complex interactions among autonomous entities [17]. Organising such systems to regulate agent interactions is a practise that helps to face their complexity [15]. Open MAS go a step further in complexity, since they consider that agents are unknown beforehand, can be developed by third parties and may enter or leave the system at any moment. Thus, there are no guarantees about their behaviour, and so, openness without a coordination model may lead to chaotic behaviours. From a system's point of view, we define this model as an *organisation* that includes a *social structure*, interaction

*MAiA Dept. Universitat de Barcelona; jcampos@maia.ub.es
†MAiA Dept. Universitat de Barcelona; maite@maia.ub.es
‡Artificial Intelligence Research Institute (IIIA) CSIC; marc@iiia.csic.es

protocols, *norms*, and global *goals*. The *social structure* defines agents' roles and their relations. Interaction *protocols* and *norms* specify what agents should conform to and can expect others to conform, and thus, they limit agents' actions. Finally, global *goals* can be explicit at design time and they define the system *goals*. Notice that they may differ from individual participants' goals. However, as an open MAS can experience a lot of changes about its participants, or even about system's external context, its original organisation may not lead to its design goals. Thus, adapting such organisation is now becoming an important topic [10, 20, 22, 25], since it can help to obtain the expected outcomes under changing circumstances.

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 (*meta-level*) on top of a regular MAS (*domain-level*). We suggest that this *meta-level* modifies the *organisation* when population or environmental changes occur. The resulting MAS has its own *organisation*, which includes the way *meta-level* agents are arranged. In other words, we are interested on adapting an organised MAS that contains *norms*, using a set of agents that are also regulated by *norms*. We propose goal fulfilment —in efficacy and efficiency— as the driving force for adaptation within the context of a rational world assumption. This implies (1) to observe system's evolution, (2) to compare it with the organisational *goals* and (3) to alter the *organisation* trying to improve *goal* fulfilment [6]. Since we construe that the *organisation* defines a coordination model for MAS participants, we see its adaptation as a *Coordination Support* [4] facility. Specifically, as next subsection details, we envision this adaptation as an organisational assistance functionality. Therefore, we call our approach: Two Level Assisted MAS Architecture (2-LAMA [5]).

As a motivating scenario to explore this approach, we are interested in domains situated in dynamic environments and whose *organisations* can be dynamically changed. This way, we focus on systems where adaptation is both necessary and feasible. Moreover, we are interested in domains where there is no direct mapping between goals and those tasks required to achieve them. For such cases, it becomes complex to determine how to adapt their *organisation* to achieve certain *goals*. For instance, in [6] we work on a traffic scenario in which it was not possible to directly identify which tasks are necessary to decrease the number of accidents and save control resources. As opposed to [20][22], in which once they have identified required tasks, they can assign them to available agents and establish their organisation depending on task dependencies. Taking into account these requirements, we use a Peer-to-Peer (P2P) sharing network as case study. In such network, computers –i.e. *peers*– contact among them to share some data and their relationships change over time depending on network status. Thus, we model its *organisation* as consisting of a *social structure* defining *peer* actual connections, the *protocol* they use to share the data, some *norms* to limit communications, and the global *goal* of using as few time and network resources as possible. Consequently, this *organisation* changes depending on network

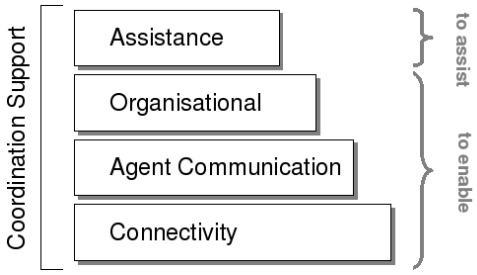

Figure 1: Coordination Support layers.

status and *peer* population, which constitute its dynamic environment.

1.1 Coordination Support

Since we construe that the *organisation* denotes a coordination model for participants, we see it as being part of a Coordination Support [4] facility. In fact, we identify different abstraction layers in this Coordination Support vision (see Figure 1), each of them providing features used by higher level layers. These abstractions simplify agents' engineering and reduce the overall engineering complexity by isolating each functionality. In our approach, the first three layers are devoted to enable agents coordination at different levels and last layer is devoted to enhance it by assisting agents and/or adapting its *organisation*. First layer (*Connectivity* layer) enables information exchange –e.g. defining a physical connection, a protocol on this connection, etc.– whereas second one (*Agent Communication* layer) determines the structure and content of this information —e.g. defining a message structure or an ontology. Third layer (*Organisational* layer) provides the organisational dimension to structure agent interactions so that, among other features, it is in charge of supporting the coordination model. Last layer (*Assistance* layer) aids agents to use more effectively and efficiently previous coordination mechanisms. We see this layer as an step forward in MAS development, specially in an open MAS, because it facilitates the engineering and enrolment of heterogeneous agents designed by different parties.[1] It may even include pro-active capabilities that let the MAS infrastructure take the initiative and act intelligently.

Assistance layer features have two main fronts: agent assistance and organisational assistance. In the former case, the *Assistance* layer can provide agents with information to participate in the MAS, justify consequences or constraints, give advice to agents and/or estimate action outcomes. In the latter case, the *Assistance* layer can adapt the *organisation* to achieve its *goals* under changing circumstances. More specifically, this can be the case when participant agents' behaviours differ notably from expected ones or when MAS external context –such as resource availability or technological updates– changes. In fact, it can be seen as a reconfiguration

[1] They may simplify agent development by providing new system facilities that agents can use instead of implementing them individually.

aspect of autonomic computing in which the MAS is able to reconfigure itself without human intervention [18]. In this paper, we suggest the 2-LAMA approach to perform such adaptation.

The rest of the paper is structured in seven sections. Section 2 describes our general model called 2-LAMA whereas section 3 illustrates its application to our P2P case study. Furthermore, section 4 details the norms used in this case study scenario and adaptation mechanisms. Our approach is evaluated through different experiments in section 5 and it is compared with related work in 6. Finally, section 7 presents the conclusions and future work.

2 General model: 2-LAMA

As previously introduced, we propose a Two Level Assisted MAS Architecture (2-LAMA [5]) that provides the described adaptation functionality. It is able to adapt the *organisation* of a regular open MAS to changes in its environment and/or in participant agents. Since we consider open MAS, we assume agents (Ag) do not belong to the *organisation* (Org). We define this *organisation* (Org) in terms of a *social structure* ($SocStr$), interaction *protocols* ($Prot$), *norms* (Nor) and *goals* ($Goals$): $Org = \langle SocStr, Prot, Nor, Goals \rangle$, where:

- The *social structure* ($SocStr$) consists of a set of roles (Rol) and the relationships (Rel) among agents playing them.
- *Protocols* ($Prot$) structure participant interactions and are expressed as sequences of messages.
- *Norms* limit participant's behaviour and are expressed using first-order deontic logic formulae to define agent permissions, prohibitions and obligations.
- *Goals* specify desired values for certain *observable properties*.

In order to perform the desired adaptation, we propose to extend an existing MAS with an additional *meta-level* on top of it. As Figure 2 shows, the resulting extended system is characterised by a two level architecture composed of this *meta-level* (ML) and the previous system we refer to as *domain-level* (DL). For notation convenience, we use level acronyms as suffixes of corresponding elements in the overall system *organisation*. Thus, for example, Nor_{DL} denotes the norms at domain level. In fact, the new system could, in turn, be adapted by a new *meta-level*. Therefore, the model can have as many levels as required. Nevertheless, since extra levels would not require new specifications, we focus on describing first two levels. Furthermore, we define a communication interface (Int) among levels. Thus, our model can be expressed as: $M = \langle ML, DL, Int \rangle$, where each level is composed of the set of participant agents and its *organisation* $xL = \langle Ag_{xL}, Org_{xL} \rangle$[2]. Note that a single agent may play different roles at different levels if authorised.

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

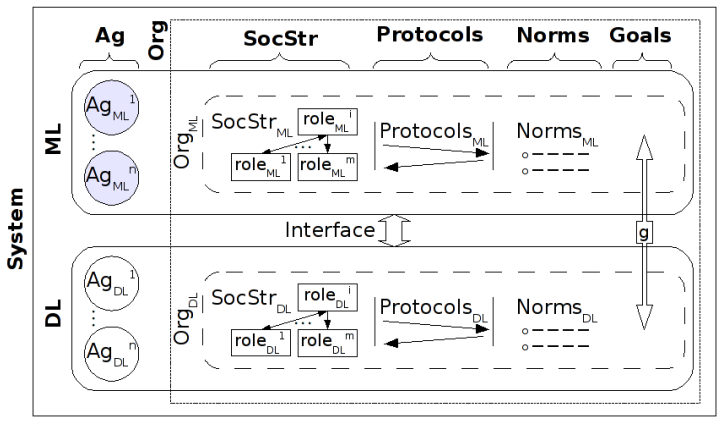

Figure 2: 2-LAMA

Communication among levels covers bottom-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 organisation* accordingly —through the Dn channel. These properties are those that can be observed from agents (AgP) and those that can be observed from the environment ($EnvP$) —i.e. $Up = \langle AgP, EnvP \rangle$. On the one hand, AgP includes *domain-level* agent's observable properties –e.g. colour or position– and may contain the actions they perform —depending on the system, the *meta-level* may perceive some of the actions performed by Ag_{DL}. On the other hand, $EnvP$ includes observable environment properties that are just affected by system's activity –e.g. the amount of an initial resource consumed by the system–, properties that are independent from it –e.g. date–, or properties affected by both the system and external factors –e.g. network traffic. Finally, the adapted *organisation* in the top-down interface channel (Dn) corresponds to a new *social structure* ($SocStr'_{DL}$), new *protocols* ($Prot'_{DL}$) and new *norms* (Nor'_{DL}) for the *domain-level* —i.e. $Dn = \langle SocStr'_{DL}, Prot'_{DL}, Nor'_{DL} \rangle$.

In summary, we suggest to add an abstraction level (*meta-level*) in charge of adapting existing *organisation* ($SocStr_{DL}$ and Nor_{DL}) depending on participant and environment properties (AgP and $EnvP$). 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, for example, information spread costs or privacy issues. Thus, an a_{ML} has aggregated information about a subset of *domain-level* agents and it can share this –or part of this– information with other *meta-level* agents. The decisions to update the *domain-level organisation* may be made by a single a_{ML} or may require an agreement or consensus among a set of them. In general, the decisions to update the *domain-level organisation* will require an agreement or consensus among *meta-level* agents.

3 Case study: P2P

Our case study is a Peer-to-Peer sharing network, where a set of computers connected to the Internet (*peers*) share some data. It is worth to apply our model because it has a dynamic environment —due to the very nature of the Internet— and its *organisation* can be changed, since *peers* can contact different *peers* to share the data. Moreover, it allows the addition of some *norms* to regulate communication. Overall, it lets us apply our organisational and adaptive approach. This section is devoted to explain how this is done.

3.1 P2P Networks

In a P2P network, some data is spread among *peers*. They exchange pieces of it in order to collect the whole information –as if they were collecting trading cards, but cards are replicated. In this scenario, we consider *peers* as software agents that act on behalf of human users that request this data. Therefore, agents need to contact other agents through the Internet, an open network. This means it is a dynamic environment, as connection quality and population can change over time. Thus, *peers* tend to re-organise to achieve their *goals* under new conditions.

Time is a valuable resource, and so, the faster the data is obtained the better for the user. Similarly, it is also a requirement to do it using as little network *bandwidth*[3] as possible. 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. Thus, although a *peer* could potentially contact all other *peers*, it usually contacts only a subset of them to save its network consumption. The actually used net of connections among *peers* is called *overlay network*. We see this *overlay network* as the organisation of these agents. Consequently, we see this scenario as a MAS where agents have a *social structure* –the *overlay network*– that evolves over time and some global *goals* —time and network consumptions.

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 *organisation* —*social structure* and *norms*. However, real P2P networks are highly complex. So we try to reduce complexity by assuming some simplifications about the protocol and the underlying network. The rest of this section provides the details of our actual scenario.

Simplified protocol

Before sharing data, real P2P networks require to locate this data and the peers sharing it. In our scenario, we obviate these initial phases and just focus on *peer* communication to obtain the

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

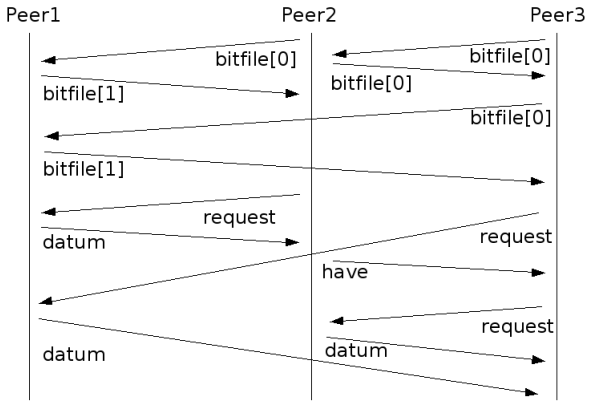

Figure 3: Simplified protocol

data. We also assume shared data has a single piece, so we say a *peer* is completed if it already has it —or uncompleted if otherwise.

Accordingly, we use the standard BitTorrent [9] protocol, but simplified[4] as shown in Figure 3. At the beginning, a *peer* initiates a handshake phase with another *peer* by sending it a "`bitfile [0/1]`" message. In this message it indicates if it has (1) or not (0) the datum. In turn, the other *peer* finishes this handshake phase by replaying with another "`bitfile [0/1]`" message to indicate its status. In case one of these *peers* have the datum and the other lacks it –e.g. `Peer1` and `Peer2` in the figure–, the later sends a "`request`" message to the former. Then, the former replies with a message containing the datum. On the contrary, if none of the *peers* have the datum –e.g. `Peer2` and `Peer3`– they will not exchange further messages. However, as soon as one of these a *peers* receives the datum, it sends a "`have`" message to these other *peers* to let them know that its status has changed. In such cases, if they still lack of the datum, they will request it. Thus, following our example, `Peer3` sends a "`request`" message to `Peer2`.

Network abstraction

We also simplify the network that *peers* use to communicate, which will influence all their communications. First, we are interested in having a different communication capacity for each *peer*. Thus, we define a single communication link among each *peer* and its Internet Service Provider (ISP) —see *individual* links in Figure 4. We also want to model simultaneous network usage by different *peers*, thus we define an *aggregated* link[5] among each group of *peers* –those connected to the same ISP– and the Internet. Finally, we abstract the Internet as a single message exchange point among all ISPs.

For each link, we define one channel per direction –upload/download. Each channel has its

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

[5]We call it *aggregated* link since it transmits all messages from *individual* links in the same group.

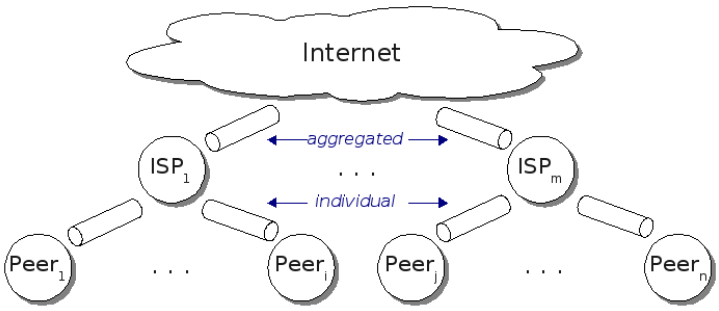

Figure 4: Network abstraction. Each cylinder represents a communication link. The Internet is abstracted as a single exchange point.

own communication capacity, which is determined by its *bandwidth* —for simplicity, we assume both directions have equal bandwidth. Therefore, the usage of a link in one direction does not affect the other. We define this *bandwidth* ($bw\left(c_i\right)$) as the number of data units that can traverse the channel (c_i) in a time unit. Hence, we define the *usage* (usg) of a channel as the ratio of *bandwidth* that is used in a given time unit (t): Equation 1 describes it using $\#msg_i^t$ to denote the number of messages traversing the channel in that moment, $msg_{j,i}^t$ to designate the jth message in the channel, and $trans$ as the effective data units transmitted during that period. For instance, if a channel has a *bandwidth* of 6 and there is a message of 6 data units, this message traverses the link in one time unit and the channel has *usage* equal to 1 during this time. Accordingly, if there are two messages of 6 data units each, both spend two time units to traverse the channel –channels transmit the same portion of every pending message–, which has *usage* equal to 1 during this time. However, if there is only one message of 3 data units, it still traverses the channel in one time unit but its *usage* is 0.5 during this period.

$$usg\left(c_i, t\right) = \frac{\sum_{j=0}^{\#msgs_i^t} trans\left(msg_{j,i}^t\right)}{bw(c_i)} \tag{1}$$

In other words, the time required by a message to travel from a *peer* to another one depends on the *bandwidth* and the *usage* of the channels it traverses. To denote this time, we define *latency* as the time required for a message of one data unit to be transmitted among two peers. Thus, if a *peer* is receiving a lot of messages at the same time, the *usage* of its *individual* download channel is high and their *latencies* increase —it takes longer to deliver all messages to it. Analogously, if each *peer* of the same ISP is receiving a few messages, the *usage* of their download channels are low and their *latency* is smaller —messages are delivered fast. However, as all these messages traverse the *aggregated* download channel of the same ISP, the *usage* of this channel may be high and messages may be slowly delivered. Consequently, it may not be a good strategy that all *peers* use all their communication *bandwidth*, because network channels may be full and it would delay the complete sharing process.

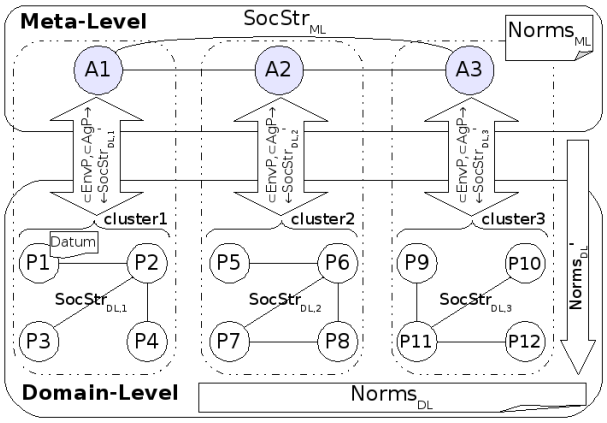

Figure 5: 2-LAMA applied to P2P scenario. Shown $SocStr_{DL}$ is just an example, and does not correspond to actual *social structures* in our experiments.

The defined *usage* metric does not distinguish between a fully occupied channel –that operates normally– and a channel with delayed messages due to excessive traffic. In the latter situation, message *latencies* increase boundlessly whilst *usage* cannot increase any further than 1. Therefore, we introduce another metric to identify and quantify this situation. We define channel *saturation* (sat) as the amount of data waiting to be transmitted over the data the channel can actually transmit in a unit time, see Equation 2. In such equation $portion_{to\,trans}$ denotes the remaining data units of a message to transmit in a given channel. For instance, if a channel has a *bandwidth* of 6 and there is a message of 6 data units, this message traverses the link in one time unit and we say its *saturation* is 1 during this time. But if there is one message of 6 data units and another message of 12 data units, the *saturation* is 3 the first time unit ($= \frac{6+12}{6}$), 2 the second time unit ($= \frac{3+9}{6}$) and 1 the third last unit ($= \frac{0+6}{6}$). The average of this *saturation* is 2 ($= \frac{3+2+1}{3}$) meaning the channel had, in average, the double of data than it can transmit.

$$sat\,(c_i, t) = \frac{\sum_{j=0}^{\#msgs_i^t} portion_{to\,trans}\left(msg_{j,i}^t\right)}{bw(c_i)} \tag{2}$$

Finally, to keep the model simple, in this paper we assume that our sharing process is the only one generating all network traffic. It implies that the observable environment properties ($EnvP$) are only affected by system's activity.

3.2 2-LAMA approach

In this section we apply the general model described in Section 2 to our P2P scenario. Accordingly, we define two layers: the *domain-level* (DL), which corresponds to *peers* sharing data; and the *meta-level* (ML), in charge of updating its *organisation* to improve system's performance.

Participant agents in the *domain-level* (Ag_{DL}) play a single role called *peer*, so $Rol_{DL} = \{\texttt{peer}\}$. Their network connections form a complete graph, since each agent can potentially contact any other agent through Internet. Nevertheless, peers usually contact just a subset of neighbours, defining an *overlay network* (see subsection 3.1). In our model, we define it as the relationships among agents (Rel_{DL}, which belong to the *social structure*) that form a sub-graph of the network connections (see Figure 5). These relationships are updated by the *meta-level* taking into account the system status ($EnvP$ and AgP) —see subsection 4.1. On the other hand, the set of *norms* at *domain-level* (Nor_{DL}) contains a single *norm* that limits the amount of upload *bandwidth* that *peers* can use ($norBW_{DL} \in Nor_{DL}$). This way, *peers* cannot use the network as an infinite resource. As we will see in subsection 4.2, this *norm* is updated by the *meta-level*. Additionally, it is worth mentioning that by now, we assume agents follow *norms*.

Regarding our *meta-level*, it also has a single role called *assistant* and so $Rol_{ML} = \{\texttt{assistant}\}$. Agents in Ag_{ML} collect local information about a group of *peers* they are associated to. We call these groups *clusters* (i.e., $cluster_i \subset Ag_{DL}$), and we assume they are disjoint. Moreover, *assistant* agents obtain information about other *clusters* provided by their neighbours in the *meta-level social structure* ($SocStr_{ML}$). The local information an *assistant* acquires consists of agent and environment properties (AgP and $EnvP$). *Assistants* obtain individual agent properties (AgP) initially, when *peers* join the system and inform whether they have the datum or not, and update them afterwards, during the sharing process, as *peers* receive the data. Whereas $EnvP$ is indirectly obtained through its *cluster's peers* since it is a measure of the communication *latency* among each pair of them. The *assistant* builds a model of the network as a graph, where nodes represent *peers* and arc's weights are the measured *latencies* (see left graphs in Figure 8). Using all this information, an *assistant* adapts the *social structure* of its *cluster* ($SocStr'_{DL}$) and agrees with other *assistants* how to update the *norms* of the *domain-level* (Nor'_{DL}). Also, the *meta-level* has a *norm* ($norHAS_{DL} \in Nor_{ML}$), that limits the number of *peers* that can be informed about data sources out of their *clusters*. Thus, this *norm* regulates how *assistants* can update *domain-level's social structure* since it limits the relationships with *peers* of other *clusters* that an *assistant* can suggest.

Finally, the organisational *goals* ($Goals$) are to spread the data among all *peers* using the minimum time and network. Thus, given some time cost (c_t) and network cost (c_n) metrics, we can define a global goal function that minimises a weighted combination of them: $Goals = min(w_t \cdot c_t + w_n \cdot c_n)$, where ($w_t$,$w_n$) are the corresponding weights that represent the relative importance of each measure —see subsection 5.1.

Extended Protocol

Our simplified P2P protocol —previously introduced in Figure 3— deals with communication at *domain-level* but requires an extension to include communication at *meta-level* and among

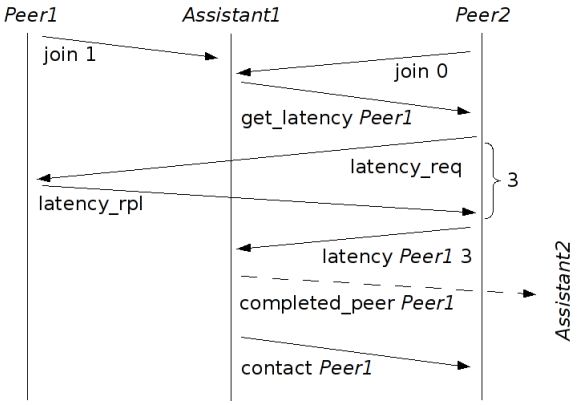

Figure 6: Extended protocol. Initial phase.

levels. Figure 6 depicts this extended protocol. Initially, a *peer* handshakes its *assistant* with a "`join <hasDatum>`" message (where "`hasDatum`" is a Boolean variable that specifies if the *peer* has the datum that is being shared). The corresponding *assistant* answers it back with a "`get_latency <peer> [,<peer>]*`" message, providing a list of *peers* whose *latencies* should be measured. The list of *peers* happens to be all other *peers* in its *cluster* in order to generate enough traffic[6]As a result, the *peer* contacts these other *peers* with a "`latency_req`" message. The other *peers* answer with a "`latency_rpl`" message. The *peer* measures the time elapsed between sending the "`latency_req`" message and receiving its "`latency_rpl`" response. Then, it estimates the *latency* by computing the division of this time over the size of both messages —following the example in Figure 6, `Peer2` measures a latency of 3 time units against `Peer1`. Afterwards, it informs back the *assistant* with a "`latency <peer> <amount>`" message.

Once an *assistant* has all latencies among its *peers* and knows which ones have the datum, it estimates which would be the best re-organisation. Accordingly, it alters the agent *relationships* ($Rel'_{DL} \in SocStr'_{DL}$) by sending "`contact <peer> [,<peer>]*`" messages to all *peer*s. Then, the simplified protocol —see Figure 3— is followed among the *peer* and its neighbours to obtain the datum. However, in current extended protocol, as the *peers* have a *latency* measure of their communication with other *peer*s, they can estimate the expected datum arrival time from other *peers*. Thus, if they have already requested the datum from a *peer*, they will only request it from another *peer* if they estimate that it is going to arrive before. For example, in Figure 3, `Peer3` would send a request to `Peer2` only if it estimates that datum will arrive before the one that was requested to `Peer1`.

Additionally, when a *peer* receives the datum, it also informs its *assistant* with a "`completed`" message. Then, at *meta-level* this *assistant* informs its neighbour *assistants* with a "`completed_peer`

[6]If traffic saturates channels, *latency* measures show differences among channels with distinct *bandwidths*. Moreover, in current implementation *latencies* are measured in both directions —so in Figure 6 `Assistant1` would send a "`get_latency` Peer2" message to `Peer1`.

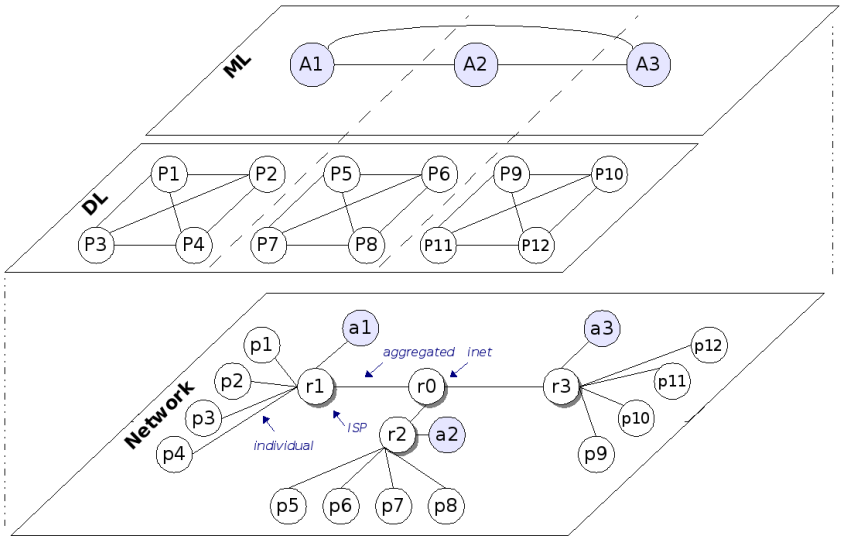

Figure 7: 2-LAMA model and the underlying network.

`<peer>`" message. For instance in Figure 5, when P2 receives the datum, it informs A1, which will inform A2 and A3. Next, contacted *assistants* spread this information towards their *peers* –may be a subset of them[7]– with a "`has_datum <peer>`" message —e.g. A2 informs P5, P6, P7 and P8 that P2 has the datum. In that moment, agents measure their latencies to the new *peer* and request it if it is better than any previous 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.

Last, the *norm* adaptation process requires two more messages —see Section 4.2. When an *assistant* wants to update the *domain-level norm* that restricts the *bandwidth* ($norBW_{DL}$), it sends a "`suggested_bw <value>`" message to the rest of *assistants*. Then, when a new value is finally agreed, each *assistant* informs its *peers* with a "`norm_updated <norm_id> <new_definition>`" message.

Underlying network

The network model used to transport messages was described in subsection 3.1. This model defines an *individual* link for each *peer* and some *aggregated* links among the ISPs. On the one hand, we define *clusters* as groups of *peers* that have a good communication among them. Therefore, we consider all *cluster's peers* have an *individual* link to the same ISP. Also, those *peers* have a good communication with their *assistant*. Thus, we assume their *assistant* is connected to the same ISP as depicted in Figure 7. The upper part of this figure represents the conceptual

[7] Actually, current norm at Nor_{ML} limits the number of *peers* to which an *assistant* can send a "`has_datum <peer>`" message —see subsection 4.2.

model defined by the 2-LAMA, whereas the lower part outlines the actual network connection. For instance, at conceptual level *peers* `P1` and `P2` are connected, which at network level is achieved by connecting network terminations[8] `p1` and `p2` to a common network element representing their ISP (`r1`) —we assume that *peer* `P1` uses the network termination `p1`, and so on. Both *peers* belong to the first *cluster* just as their *assistant* `A1` does, thus `A1`'s network termination (`a1`) is also connected to `r1`. On the other hand, as at conceptual level the communication graph is complete at network level all agents are also connected. Specifically, each *cluster* is connected with the others through the *aggregated* links. For instance, `r1` and `r2` have *aggregated* links connected to `r0`, which acts as an abstraction of the interconnection through the Internet.

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 ISPs which have better communication among them than their clients. Alternatively, it could also be the best connected *peer*. Accordingly, 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* —since the *assistant* has the best connection.

4 Norms and adaptation

So far in this paper we have proposed a coordination model for open MAS. In this model we consider an *organisation* structured in a two-level architecture (2-LAMA) that adds a *meta-level* on top of a *domain-level*. Coordination key components in the *organisation* are the *social structure* and the set of *norms* at both levels. Previous subsection 3.2 described the *social structure*. In this section, we detail which *norms* are added to the P2P scenario and how *meta-level* adapts *domain-level*'s *social structure* and *norms*.

4.1 Social structure adaptation

Assistants adapt the *domain-Level social structure* ($SocStr_{DL}$) based on local information they gather from their clusters *peers*. This information corresponds to network status (represented by means of *latency* measures, an environmental property belonging to $EnvP$) and *peer* information regarding data possession (an agent property in AgP). *Social structure* adaptation is also performed locally, since each *assistant* just contacts its *cluster peers* to suggest their new corresponding *social structure* —i.e. the subset of $SocStr_{DL}$ that involves them.

Furthermore, *assistants* share some of their local information. More specifically, they communicate which *peers* in their clusters have the data. Consequently, each *assistant* has detailed information about its *cluster* and an overview about the rest of the *domain-level*. With this

[8] A network termination is a device that connects a customer's communication device to its local data exchange carrier.

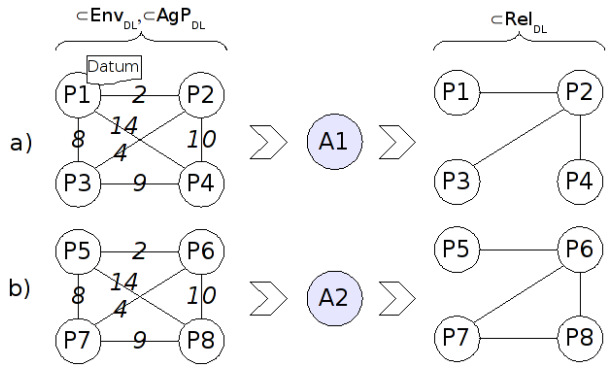

Figure 8: *Social structure* adaptation examples. Left graphs show arc *latencies* ($EnvP$) and datum possession (AgP). Right graphs represent the resulting relationships among *peers* (Rel_{DL}) that *assistants* compute. *Latency* values are just illustrative.

information, *assistants* can estimate which are the shortest paths among data sources and destinations. Accordingly, they ask *peers* to contact those other *peers* that are in these shortest paths by sending "`contact`" messages.

Specifically, in our current implementation *assistants* face two different situations depicted in Figure 8: (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 [11] 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 path among all 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 8, A1 tells P4 to contact P2. This way, the new relationship graph (Rel'_{DL}) may have different arcs than the ones in the old relationship graph (Rel_{DL}).

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, A2 would tell P8 to contact P6 if the datum enters by P5, but it would tell P8 to contact P7 if the datum enters by P7. Consequently, A2 tells P8 to contact P6 and P7. In general, we can expect the resulting *relationship* graph (Rel'_{DL}) to be larger than the one in previous case (a). Nevertheless, it uses current available information to reduce the complete graph —i.e., the one considering all possible relationships. Shortest path computation avoids the usage of non-optimal paths, thus saving communication –i.e. network usage– among unnecessarily involved *peers*.

4.2 Norm adaptation

In our case study, we endow the *organisation* with two *norms*: one at *domain-level* ($norBW_{DL} \in Nor_{DL}$) and one at *meta-level* ($norHAS_{ML} \in Nor_{ML}$). We describe these norms as *regulative norms* [2, 21] using a first-order deontic logic [14]. Thus, we have the operators obligation (O), prohibition (F) and permission (P). Our language of first-order logic includes:

- the symbols ag denoting an agent in Ag, m a type of message and p a parameter of a message.
- the property $\mathtt{R}_{ag}$, which refers to the roles played by agent ag.
- the predicate $M(ag_i, ag_j, m, p)$, which means that a message of type m with a parameter p was sent from agent ag_i to agent ag_j.
- $Pace : Ag \times t \rightarrow \mathbb{Z}$, a function that specifies the amount of data units that an agent sends –i.e. uploads– at a given time. Transmitted data units are part of messages that the agent is sending at that time —all protocol message types are included. For instance, if an agent sends a message of 20 data units by its upload channel of *bandwidth* 5 without any restriction, its $Pace$ is 5 and will require 4 time units to be transmitted. However, if the agent decides to send it at a $Pace$ of 2 to fulfil the *norm* that restricts the *bandwidth*, it will require 10 time units.

Norm at *domain-level*

The norm at *domain-level* is targeted to limit the amount of network resources that *peers* can use. It specifies the maximum *bandwidth* a *peer* can use in its upload *individual* channel. We express it with a forbidden (F) formula (φ) that is true if a *domain-level* agent (ag_i) that plays the role *peer* ($\mathtt{peer} \in Rol_{DL} \subset Rol$) is already uploading to the network ($Pace$) more than $\mathtt{max}_{\mathtt{BW}}$ data units in any moment (t):

$$norBW_{DL} = \mathsf{F}\varphi, \quad \varphi = (\mathtt{peer} \in \mathtt{R}_{ag_i} \wedge Pace\,(ag_i, t) > \mathtt{max}_{\mathtt{BW}}))$$

Thus, *peers* cannot use the network as an infinite resource. Limiting the network usage should increase the execution time –since messages will be sent at a lower pace– but using it as an infinite resource can also increase the execution time because network channels become saturated and message latencies increase.

Norm at *meta-level*

The norm at *meta-level* is targeted to control how *assistants* adapt the *domain-level's social structure*. It specifies the maximum number of *peers* to which an *assistant* can send a "`has_datum`"

message. We express it with a forbidden formula (ψ) that is true if an *assistant* (ag_{ai}) is informed by another one (ag_{aj}) that a peer (ag_{px}) is completed (m_{comp_peer}), and it pretends to send more than $\texttt{max}_{\texttt{HAS}}$ "`has_datum`" messages (m_{has}) to its peers (ag_{py}):

$$norHAS_{DL} = \mathsf{F}\psi, \quad \psi = \begin{array}{l}(\texttt{assistant} \in \mathtt{R}_{ag_{ai}} \wedge M(ag_{aj}, ag_{ai}, m_{comp\_peer}, ag_{px}) \wedge \\ \|M(ag_{ai}, ag_{py}, m_{has}, ag_{px})\| > \texttt{max}_{\texttt{HAS}} \ )\end{array}$$

Thus, *assistants* cannot simply suggest all its *peers* –the ones in its *cluster*– to contact the one having the datum. Instead, they may have to choose only some of them. Therefore, it regulates the number of *peer* neighbours among different *clusters*. A small $\texttt{max}_{\texttt{HAS}}$ reduces *aggregated* channels usage but may increase the sharing time since some *peers* in a *cluster* have to wait until the other ones receive the datum.

In our current implementation, all *meta-level* agents follow the *norms*. This is a reasonable simplification, since the whole layer is added at design time to enhance an open system. Accordingly, it makes sense to build collaborative and benevolent agents. In fact, an analogous assumption can be made for the *domain-level*, because ISPs could be involved on this P2P extension and enforce rules to avoid abusive *bandwidth* consumption.

Norm adaptation

Meta-level adapts $norBW_{DL}$ depending on network status by updating its $\texttt{max}_{\texttt{BW}}$ parameter. In fact, each *assistant* chooses its desired $\texttt{max}_{\texttt{BW}}$, and altogether agree on its actual value.

To choose the desired $\texttt{max}_{\texttt{BW}}$, an *assistant* takes into account an estimation of the network status in its *cluster*. To perform such estimation, it can obtain data from different sources: (a) network status information, (b) *peer latency* measures, (c) *assistant latency* measures. In the former case (a), an *assistant* has access to some information from its related ISP, which knows about the actual network *usage* of its *aggregated* link —i.e. about inter-*cluster* network usage. In the second case (b), an *assistant* receives updated information about the *latencies* among *peers*. This information is basically related to *individual* links —i.e. intra-*cluster* network usage. In order to update this information, an *assistant* can actively query a *peer* by sending a "`get_latency`" message, or this *peer* can pro-actively send a "`latency`" message when the measure changes —e.g. when the difference of a *latency* with its previous measure is greater than a given threshold. This information update is feasible because, during system execution, *peers* can extract *latency* information from the cadence of their *protocol* messages instead of sending "`latency_req`" messages. Finally, in the latter case (c), an *assistant* obtains information from measuring its *latencies* against other *assistants* by sending "`latency_req`" messages to them. As we assumed that *assistants* have the best *individual* link, the difference about these measured *latencies* is related to the *cluster's aggregated* link: the larger the *assistant*'s link *bandwidth* is, the less it contributes to increase *latencies*, so they mostly depend on its *aggregated* link

saturation.

In our current implementation, *assistants* use the first alternative (a) to estimate network status. In fact, they use an heuristic to estimate the network status from the *aggregated* link *usage*. It assumes that an usage of 75% of an *aggregated* link entails a worth utilisation of the network resources without saturating them. Accordingly, an *assistant* chooses to increase the $\texttt{max}_{\texttt{BW}}$ parameter if the network status presents low usage. Then *peers* can use more network resources, which may lead to a smaller execution time. Otherwise, it chooses to decrease the $\texttt{max}_{\texttt{BW}}$ value, so *peers* use less network resources. This decrement helps to reduce network usage but it may increase execution time. However, a lower network usage can lead to less saturation in links, which may decrease latencies and as a result the system may require an smaller execution time. The mentioned heuristic is a compromise value among the time and network consumptions present in the organisational *goals*.

Finally, *assistants* have to agree on the actual value of $\texttt{max}_{\texttt{BW}}$ depending on their *meta-level organisation*. Such agreement can be reached in several ways, as for instance negotiation, argumentation or voting the preferred value. In current implementation, a voting approach is used by computing an average of $\texttt{max}_{\texttt{BW}}$ values suggested by *assistants*. Specifically, each *assistant* receives "`suggested_bw`" messages from its neighbours and computes their average[9]. Next, *assistants* inform their *peers* about the new $\texttt{max}_{\texttt{BW}}$ value by sending them a "`norm_updated`" message. This process is repeated at certain time intervals, which is a parameter of the adaptation process. Each time, *assistants* start counting the new time interval when they receive the last "`suggested_bw`" message —thus at each cycle, their notion of time is re-aligned. If the graph of relationships among *assistants* is not complete –e.g. because the system scales up, *it increases the number of assistants* and contacting all of them requires too much network– the notion of time and *domain-level norms* may not be exactly the same around the system. However, time and *domain-level norms* may keep their coherence among contiguous *clusters*.

5 Experiments

We have tested our approach on the P2P scenario depicted in Figure 5. It has 12 *peers* equally distributed in 3 *clusters* Figure 9 shows its underlying network. Note that this network topology conforms our assumptions: communications at *meta-level* and among levels are faster than communications at *domain-level*. We consider that *assistants* are located at ISPs[10] since they have an infinite *bandwidth* link with the node that aggregates external –i.e. inter-cluster– communication.

In order to test our proposed approach, we have implemented a simplified P2P simulator in Repast Simphony [19]. It simulates the agents –*peers* and *assistants*– as well as the transport of

[9] Replicating a computation may lead to subtle different values. However, as *assistants* interchange an absolute value, the possible precision errors are not accumulated.

[10] In a previous work [5] we showed that even if *assistants* are plain clients the system still presents an advantage by having the *meta-level*.

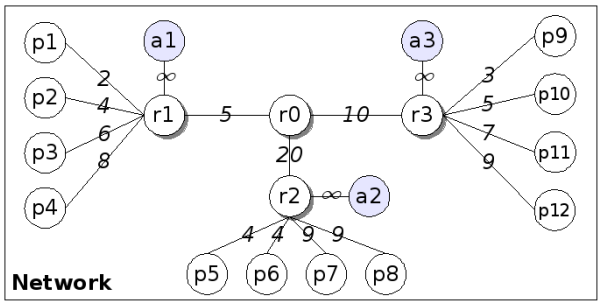

Figure 9: Network topology. Arc weights correspond to link's *bandwidths*, which are symmetric for upload and download channels.

their messages through the network. We used this simulator to perform various executions with different configurations —including our proposed 2-LAMA. Each execution has been evaluated in terms of time and network consumption.

We have tested three alternative implementations: *All4All*, *Centralised* and Distributed. In the former alternative (*All4All*) all *peers* obtain a complete list of *peers* in the system[11]. They start by contacting all of them before proceeding with the simplified protocol —see subsection 3.1. Nevertheless, this generates a large amount of messages travelling along the same links, and thus, they may get longer to reach their destination. However, if we assume messages do not share links, we obtain a lower bound for c_t. We call this implementation variant *All4All-ideal*.

In the second alternative (*Centralised*), all *peers* contact a single *assistant* that collects –i.e. centralises– the whole information (AgP and $EnvP$) about them. This assistant is attached to `r0` and suggests a single *social structure* to coordinate all *peers* in the system. This structure results from computing individual shortest paths over the whole network. Thus we can argue that it makes global decisions because of its global knowledge.

In the latter alternative (*Distributed*), *peers* contact their corresponding *assistants* as we suggest in our 2-LAMA approach. Thus, the *meta-layer* is distributed and each *assistant* has local knowledge. *Assistants* share some of their local information among them. We executed this alternative with different *norm* parameters ($\texttt{max}_{\texttt{BW}}$, $\texttt{max}_{\texttt{HAS}}$) to study their effects on system performance. In addition, we executed this alternative with the ability to adapt the *domain-level's norm* to show it is able to change it depending on network status.

5.1 Evaluation

We evaluate the executions in terms of time and network consumption. This evaluation criteria is aligned with the configurations that have explicit *goals* –i.e. *Centralised* and *Distributed*– so it is also a measure of *goal* fulfilment. On the one hand, we define the *time cost* (c_t) as the number of time steps elapsed in the simulation (which lasts until all *peers* are completed). On

[11] As in BitTorrent [9], they contact a central *tracker* –attached to `r0`– that returns the list of existing *peers*.

Level	Type	Message	Size
DL	*latency*	`latency_req`, `latency_rpl`	15
	control	`bitfile`, `request`, `have`	20
	data	`piece`	200
$interface$	all	`join`, `get_latency`, `latency`, `contact`, `completed`, `has_datum`, `norm_updated`	20
ML	all	`completed_peer`, `all_completed`, `suggested_bw`	20

Table 1: Message sizes used in current implementation.

the other hand, we define the *network cost* (c_n) as the network consumed by all sent messages (c_{m_i}): $c_n = \sum_{i=1}^{\#msgs} c_{m_i}$. This consumption depends on the size of the message ($||m_i||$) and its path's length ($||path\,(m_i)\,||$, the number of links it traverses): $c_{m_i} = ||m_i|| \cdot ||path(m_i)||$. In our executions, we used different message sizes depending on message types and levels. Control messages are one order of magnitude shorter than data messages, and slightly greater than the messages used to measure the *latencies* —see Table 1.

In order to get a further insight about what is going on at network level during simulations, we have also computed three additional performance measures: *network hops* (see Equation 3), *network usage* (Eq. 4) and *network saturation* (Eq. 5).

- *Network hops* (net_{hop}) is the average number of of links traversed by all messages. Thus, a large net_{hop} value means there where a lot of messages among *clusters* (i.e, inter-cluster), and the opposite means most communications where local (intra-cluster).

$$net_{hop} = \frac{\sum_{i=0}^{\#msgs} ||path(m_i)||}{\#msgs} \tag{3}$$

- *Network usage* (net_{usg}) corresponds to the average of channels' *usage* (usg, see Equation 1) given as a fraction of unity. Therefore, a net_{usg} close to 0 means links are empty in average, whereas a value close to 1 means links are mostly occupied.

$$net_{usg} = \frac{\sum_{t=0}^{time} \frac{\sum_{i=0}^{\#channels} usg(c_i,t)}{\#channels}}{time} \tag{4}$$

- *Network saturation* (net_{sat}) is the average of the channels' *saturation* (sat, see Equation 2) as a positive real number. Hence, a net_{sat} value among 0 and 1 means that links are not saturated in average, whereas values greater than 1 indicate that there is *saturation* in the network. Accordingly, it lets identify and quantify the situation in which *latencies* are increasing because the network has not enough capacity to transmit all generated messages. The greater net_{sat} is, the larger message *latencies* have increased because of *saturation*.

$$net_{sat} = \frac{\sum_{t=0}^{time} \frac{\sum_{i=0}^{\#channels} sat(c_i,t)}{\#channels}}{time} \tag{5}$$

Style	c_t	c_n	net_{hop}	net_{usg}	net_{sat}
All4All	1325.97	69884.3	3.39	0.186	16.64
All4All-ideal	275.43	74028.7	3.42	0.946	14.77
Centralised	1356.52	34157.8	3.28	0.103	3.89
Distributed	1146.38	25089.2	2.67	0.092	4.43
Distributed-norms1 ($\texttt{max}_{\texttt{HAS}}$=1, $\texttt{max}_{\texttt{BW}}$=1)	1216.57	18408.1	2.38	0.066	0.73
Distributed-norms2 ($\texttt{max}_{\texttt{HAS}}$=1, $\texttt{max}_{\texttt{BW}}$=4)	629.64	17536.3	2.35	0.115	1.74
Distributed-adaptation ($\texttt{max}_{\texttt{HAS}}$=1)	876.43	18854.5	2.34	0.088	1.47

Table 2: Global results using different execution styles. Evaluation metrics –introduced in previous sections–: total *time cost* (c_t), total *network cost* (c_n), average number of hops (net_{hop}), average *network usage* (net_{usg}), and average *network saturation* (net_{sat}).

5.2 Results

We have tested the alternative implementations introduced before by varying the *peer* that initially has the datum. Thus, tests include twelve different settings so that they cover all possible initial data positions in a single *peer*. In addition, we executed each test 10 times in order to compute the average of its evaluation parameters. In this subsection, we first provide a general perspective of the overall results. We then continue with closer details about norm effects on our approach and, finally, we present results on its adaptation process.

General view

Table 2 shows the evaluation metrics in different configurations: two configurations of the *All4All* implementation –the regular one *(All4All)* plus an ideal one (*All4All-ideal*)–; one corresponding to the *Centralised* implementation; and four different distributed configurations. In particular, distributed configurations correspond, in this order, to: one without norms nor adaptation (*Distributed*); two with two different norm configurations but without adaptation (*Distributed-norms*); and one with both norms and adaption (*Distributed-adaptation).* As previously stated, presented metric values have been averaged from $12 \times 10 = 120$ tests.

First observation that rises from the table is that the *All4All* alternative presents a very high *time cost* (c_t). This is because it implies the maximum *saturation* in channels (net_{sat}). This *saturation* occurs because all *peers* contact simultaneously all other *peers*, and so, they try to obtain the datum from all possible sources. This behaviour also implies that its communications traverse the maximum number of links (net_{hop}) since all *peers* in a *cluster* request the datum to other *clusters* —i.e. they perform inter-*cluster* requests. Accordingly, it also presents extremely high *network cost* (c_n) and high *network usage* (net_{usg}).

On the contrary, the *All4All* alternative with *ideal* channel *bandwidths* (*All4All-ideal*) has the minimum *time cost*. This is a theoretical alternative in which messages always traverse the channels as if they were alone. In other words, when more than one message is traversing a

channel, their *latencies* depend directly on the channels *bandwidth* but not on the number of messages. This is an *ideal* alternative we use only to estimate the minimum time required to exchange the data in a certain network topology so it provides us a sort of lower boundary. However, it presents the maximum *network cost* because these theoretical low *latencies* allow *peers* to exchange the maximum number of messages. Its *network usage* reflects this situation since the average is almost a 100% usage during the whole execution.

If we add a *Centralised meta-level*, the *network cost* is reduced by 50% –versus *All4All*– since *peers* only contact with the ones suggested by the *assistant*. This *social structure* reduces the number of messages, so network *saturation* and *usage* are lower than in *All4All*. However, *time cost* is almost the same because now we have an initial phase in which the *assistant* asks *peers* to measure all their *latencies* —as it requires global knowledge, *latencies* are measured from every *peer* against all the rest in the system.

Furthermore, if we distribute the *meta-level* (*Distributed*), the results present a 14% reduction in *time cost* and a 64% reduction in *network cost* versus the *All4All* alternative. Now, the initial phase devoted to collect information is shorter because *latency* measurements are just local to *clusters* –notice that the reduced net_{hop} value reflects this locality.

The system improves further when we add some *norms* that limit the *social structure* –mainly Nor_{ML}– and the amount of communications —Nor_{DL}. In such cases, the performance depends on both *norm* parameters: $\texttt{max}_{\texttt{HAS}}$ and $\texttt{max}_{\texttt{BW}}$. For instance, the worst combination (*Distributed-norms1*) provides a further improvement in *network cost* –74% less than *All4All*– but a worse *time cost* —8% less than *All4All*. It is due to two main reasons: first, the *bandwidth* limitation ($\texttt{max}_{\texttt{BW}}$) has been set to 1 and it is too restrictive and, second, the network is underused — only a 6.6% of net_{usg} along the execution. In contrast, the best *norm* parameter combination (*Distributed-norms2*) has a *bandwidth* limit of 4, which increments the *network usage*, and obtains the minimum time and network costs: 52% less c_t and 75% less c_n than *All4All*. The details about different parameters are explored later in this subsection.

Nevertheless, choosing the best values to *norm* parameters at design time is not a straightforward decision in open MAS. Furthermore, the best norm values may vary under different circumstances. Thus, we endow the *meta-level* with an automatic adaptation capability able to tune the values at run time, and adapt them to different conditions. The last row in the table shows our distributed adaptation approach. Despite the fact that it starts with the worst value of the *domain-level*'s *norm* ($\texttt{max}_{\texttt{BW}} = 1$), it obtains desirable reductions both on time and network costs —34% and 73% less than *All4All* respectively. Further details about adaptation results can be found at the end of this subsection.

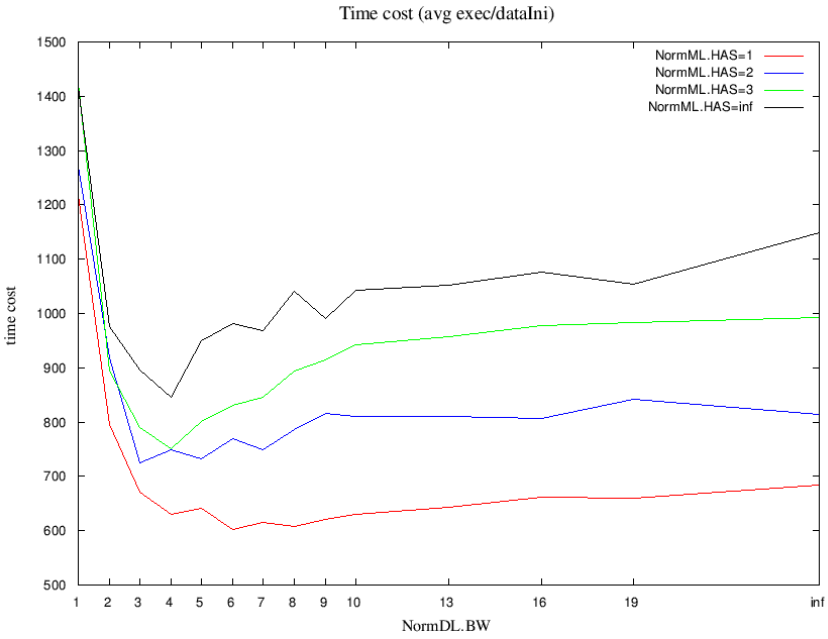

Figure 10: Time cost (c_t) depending on norm parameter values. An inf value means that the *norm* does not apply any restriction to agents' behaviour.

Norms

As previously mentioned, different *norm* parameter values –$\texttt{max}_{\texttt{HAS}}$ and $\texttt{max}_{\texttt{BW}}$– lead to diverse results. Thus, we have studied their influence by executing the same configurations with different values.

Figure 10 shows a plot of the *time cost* –ordinate– given different values of $\texttt{max}_{\texttt{HAS}}$–series– and $\texttt{max}_{\texttt{BW}}$—abscissa. From it, we can observe that when communication is too restricted –being $\texttt{max}_{\texttt{BW}} = 1$ the maximum restriction possible– *time cost* becomes higher than for any other configuration. Obviously, if *peers* transmit messages at a pace of a single data unit per unit of time, these transmissions will take longer than if it is done at a larger pace. In this way, for less restrictive *bandwidth* limits –e.g. $\texttt{max}_{\texttt{BW}} = 4$– *time cost* becomes lower. This is because *peers* are better exploiting their network resources: they use as much network *bandwidth* as possible but without saturating the channels in an excessive manner —see Figure 12 for saturation results. However, less restrictive limits –e.g. $\texttt{max}_{\texttt{BW}} = 16$– again entail high *time costs*, because *network saturation* increases too much. Regarding the *meta-level norm*, it produces a clear effect on time: the smaller number of *peers* it allows *assistants* to contact, the smaller amount of time it is required to distribute data among all *peers*. And this can be stated so clearly because the series in Figure 10 do not cross. The main reason for time reduction is the limitation in the number of inter-cluster communications, which, in turn, reduces *saturation* at *aggregated* links [12].

[12] Nevertheless, although lower values –e.g. $\texttt{max}_{\texttt{HAS}} = 1$– provide good *time costs*, they also increase the risk of

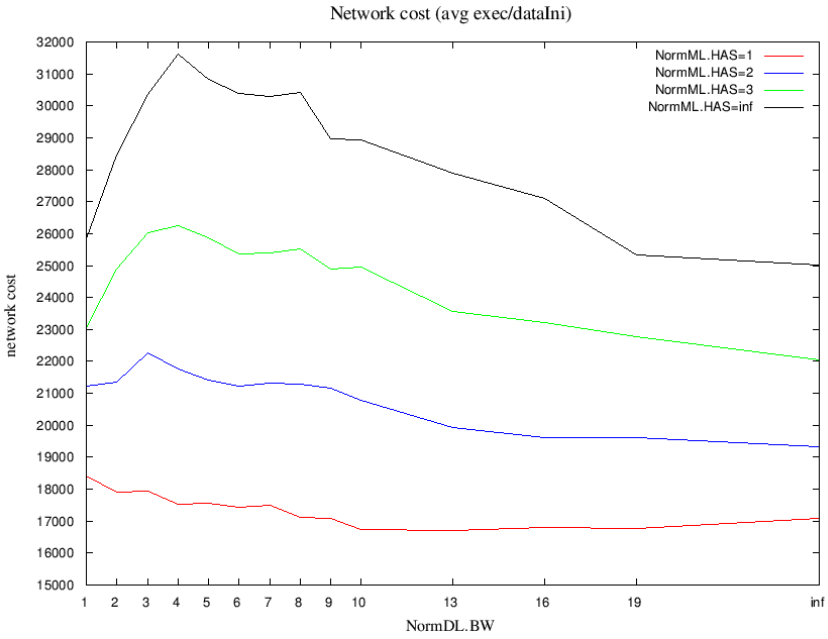

Figure 11: Network cost (c_n) depending on norm parameter values. An *inf* value means that the *norm* does not apply any restriction to agents' behaviour.

Additionally, Figure 11 shows how *network cost* depends on *norm* parameters. In general, a restrictive limit on communications –e.g. $\texttt{max}_{\texttt{BW}} = 1$– leads to a low *network cost*, whereas a less restrictive limit –e.g. $\texttt{max}_{\texttt{BW}} = 4$– leads to a higher *network cost*. Nevertheless, if the restriction is still weaker –e.g. $\texttt{max}_{\texttt{BW}} = 16$–, then the *network cost* decreases again. This is because the resulting *saturation* delays the communications, and thus, less messages are exchanged among *peers*. On the other hand, the *meta-level norm* decreases the *network cost* in the same consistent way as for the *time cost* —i.e, series do not cross. Nevertheless, it also has a softening effect on the shape of its curve. Whereas a weak restriction –e.g. $\texttt{max}_{\texttt{HAS}} = 3$– presents different results depending on the *domain-level norm*, a strict restriction –e.g. $\texttt{max}_{\texttt{HAS}} = 1$– presents a quasi flat curve. This is particularly remarkable because, despite the fact that reducing network and time costs could be expected to represent antagonistic goals, small $\texttt{max}_{\texttt{HAS}}$ values allow us to choose the $\texttt{max}_{\texttt{BW}}$ that minimises the *time cost* while still consuming network in a really moderated way —specially if compared with other configurations.

Last, Figure 12 shows *network saturation* results, which are necessary to understand previous time and network costs. As it has been previously stated, intermediate values for *domain-level norm* parameter –e.g. $\texttt{max}_{\texttt{BW}} = 4$– keep a reasonable balance between both costs. In fact, these intermediate values correspond to what could be interpreted as a reasonable resource exploitation: to make use of the available network *bandwidth* but without blocking the communication

cluster isolation if certain nodes fail.

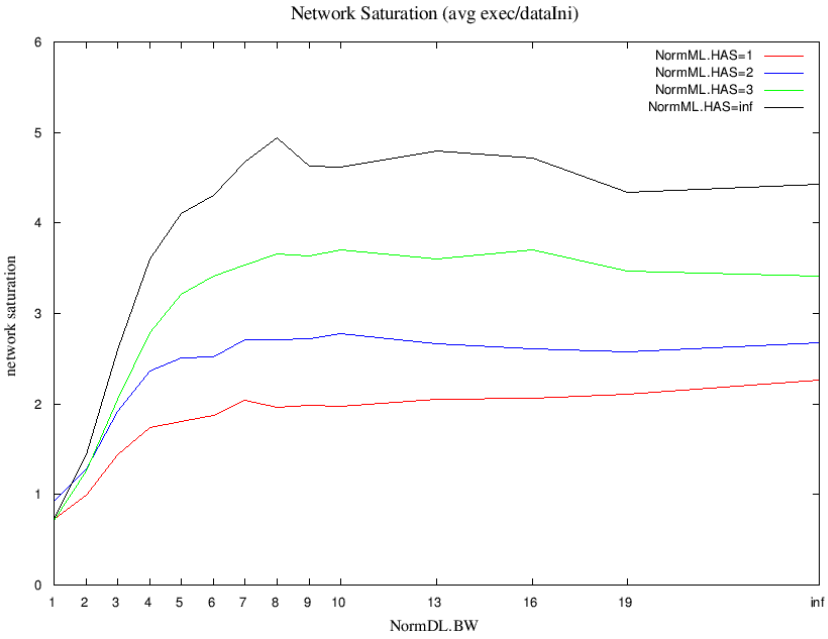

Figure 12: Network saturation (net_{sat}) depending on norm parameter values. An inf value means that the *norm* does not apply any restriction to agents' behaviour.

channels.

Adaptation

Finally, we present some results about the adaptation process. As previously mentioned, it is performed by the *meta-level* and consists in changes of the the $\texttt{max}_{\texttt{BW}}$ parameter of the *domain-level*'s *norm*.

In order to get a better insight of the process, we focus on a single execution instead of showing averaged results. Specifically, we analyse a setting where the data is initially located at *peer* P3, the *meta-level norm* has a restrictive parameter value $\texttt{max}_{\texttt{HAS}} = \texttt{1}$ and the *domain-level norm* is updated at time intervals of 100 units. Figure 13 illustrates its execution along time. The bold orange line represents the evolution of *network saturation* –left ordinate axis– along time —abscissa. As we can observe, it increases quite steeply. Nevertheless, *assistants* cannot perceive *saturation*. They are just able to measure local *network usage*: the one in the *aggregated* link in its *cluster*. Red, green, and blue lines in the figure correspond to the *usage* measures –left ordinate axis– perceived by *assistants* 1, 2, and 3 respectively. Thus, for example, at time step 300, *assistant* A1 perceives an *usage* of 1 at the *aggregated* link `r1-r0`. Additionally, the bold black line in the figure plots the $\texttt{max}_{\texttt{BW}}$ value –right ordinate axis– that is being adapted along the execution.

During the adaptation process, *assistants* use an heuristic which assumes that performance

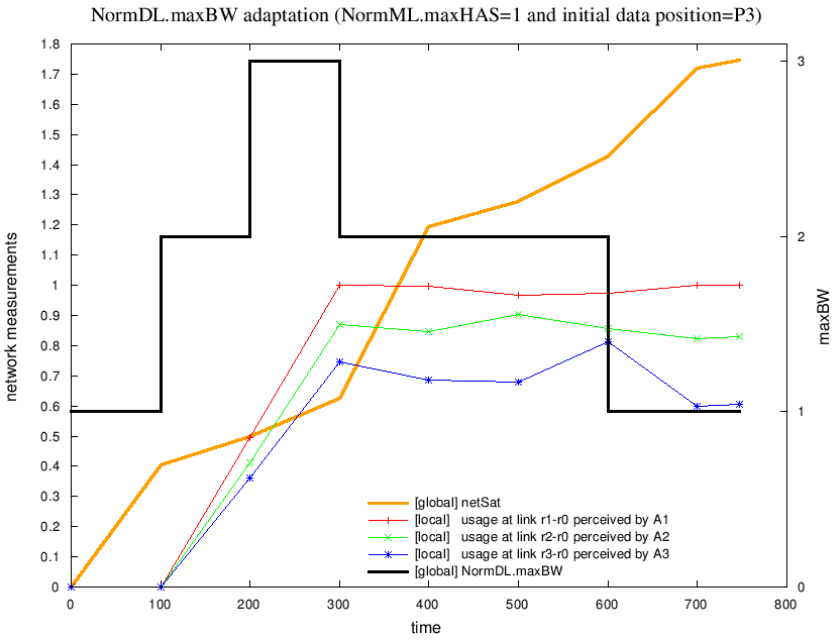

Figure 13: Norm adaptation ($\texttt{max}_{\texttt{BW}} \in Nor_{DL}$) depending on network status.

is optimal if the *network usage* is kept around 0.75. Besides, *network usage* is proportional to $\texttt{max}_{\texttt{BW}}$ values: it decreases if *peers* send messages at a lower pace. As a consequence, assistants apply this heuristic to suggest changes on $\texttt{max}_{\texttt{BW}}$ values depending on their currently perceived usage. Thus, for instance, Figure 13 shows that at time step 300, *assistants* A1 and A2 perceive a value that is greater than 0.75 whereas A3 already perceives the target value. In such a case, A1 and A2 suggest to decrease $\texttt{max}_{\texttt{BW}}$ –from 3 to 2– whilst A3 votes for keeping it at 3.

Assistants reach an agreement by voting for their preferences over changing $\texttt{max}_{\texttt{BW}}$. Therefore, the resulting value is computed as a round average. In previous example, the resulting $\texttt{max}_{\texttt{BW}}$ value is 2 ($2.33 = \frac{2+2+3}{3}$), and consequently, the black line in the figure drops from 3 to 2[13]. As we have said before, Figure 13 shows a complete execution. Its initial $\texttt{max}_{\texttt{BW}}$ value was one, which happens to be the worst. Nevertheless, *assistants* at *meta-level* adapt this norm parameter based on their network status perception. Specifically, they start by increasing it since the network is initially underused, and afterwards, whenever the *saturation* becomes a problem, they decide to decrease this limit to reduce the number of communications.

All test configurations summarised in the last case of the general view had an initial $\texttt{max}_{\texttt{BW}}$ value of one. All of them were able to adapt it so to obtain, in average, the *time* and *network costs* shown in last row of table 2. Therefore, we can conclude that adaptation was successfully performed.

[13] At time steps 400 and 500, A1 and A2 vote for decreasing $\texttt{max}_{\texttt{BW}}$ to 1 –because they perceive an $usg > 0.75$– whereas A3 vote for increasing it to 3 —it perceives an $usg < 0.75$. As a result, $\texttt{max}_{\texttt{BW}}$ keeps its value of 2 due to rounding effects on average result ($1.66 = \frac{1+1+3}{3}$).

6 Related work

This paper focuses on the adaptation of open MAS depending on changes on the environment as well as on participant agents. This adaptation can be seen as a reconfiguration aspect of autonomic computing [18]. Specifically, we aim on updating systems' organisation. This is motivated by the computational organisational theory, which claims that the best organisation designs are domain and context dependent [7]. In fact, we conceive this organisation as a coordination model having explicit components with *computational reflection*[14] capabilities [23]. Existing approaches like Electronic Institutions [12] or Moise [16] also define coordination models by specifying the agents's *social structure* and the *protocols* that participants use to interact. Moreover, we are interested in *organisations* that include *norms* to regulate agent behaviours. There exist works that add *norms* to previous approaches, such as Synai [3], which adds a *normative organisation* on top of an existing one. In such an approach, the agents belonging to this new *normative organisation* are in charge of supervising that others fulfil the *norms*. It also uses *regulative norms* [2] to bound agent actions depending on their role, as we do, but they include sanctions. However, Synai does not update *norms* as 2-Lama does. Nevertheless, approaches have been made to update *norms*. For example, in the work by Artikis [1], agents follow an argumentation protocol to change the rules of order. Specifically, agents use an argumentation protocol to discuss and update *norms*. Moreover, they can even discuss about the argumentation protocol itself to update it. In our case, *meta-level* agents could use their argumentation technique to discuss about changes in *domain-level*.

We envision MAS adaptation as a process to achieve system's global goals —such as, for example, performance improvement. Thus, we add *goals* as an explicit component of the *organisation* specification. This way, they can be used to guide the *computational reflection* process. Most work on MAS adaptation assumes it is feasible to identify which tasks are necessary to achieve such goals. Consequently, they map goals to tasks and look for agents with capabilities to perform them in order to achieve the organisational goals. For instance, in [20], once they have identified required tasks, they assign them to available agents and establish their *organisation* depending on task dependencies. Nevertheless, in [22], they go a step further and derive new required tasks related to coordination issues. Thus, when an original task requires more than one agent, a new task is created in charge of coordinating such agents. Recursively, the assignation process is repeated to find an agent that can perform the new task. Moreover, in [10], they even use an extra layer with *organisational agents* to perform this assignation process based on the mapping between tasks and organisational goals. On the contrary, as we already mentioned, we assume our system lacks of this mapping between goals and those tasks required to achieve them.

The closest proposal to our 2-Lama approach is MASPA [25]. It also consists on a distributed adaptation mechanism, where agents have a partial view of the whole system. As our (recursive)

[14] The word reflection refers to the possibility of a software to observe and modify its own structure.

meta-level, its *multi-level supervision organisation* has agents in charge of adapting the *organisation* of *worker clusters*, which is equivalent to our Org_{DL}. These agents provide *rules* and *suggestions* to agents in their previous layer —*suggestions* are optional and *rules* are mandatory. Both of them specify a *condition* and some *actions*. In this way, when the *condition* is satisfied, agents perform the specified *actions*. Thus, it assumes agents are implemented to check such *conditions* and perform its corresponding *actions*. In other words, their aim is to create adaptive MAS developing all its components, whereas our purpose is to deal with open MAS —where we have no control over agent development.

Regarding our P2P case study, there are some approaches that follow a MAS viewpoint and others that take a network management perspective. From a MAS angle, Grizard et al. [13] focus on *norm* adaptation, like our proposal. It defines two types of norms: *rules* and *conventions*. The former ones are global mandatory norms whereas the latter are local conventions. Their approach consists of a layer of *controller* agents on top of *application* agents —having one *controller* per each *application* agent. *Controllers* keep reputation values about their *application* agents summarising their *rule* fulfilment. Every *controller* can observe all its agent actions to detect *rule* violations/fulfilments and decrease/increase its reputation value accordingly. Thus, other agents can ask to their *controllers* about this reputation to decide whether they interact with it. This way, *rules* are enforced by social control. In respect of local norms, an agent indicates its *conventions* to its *controller*, which informs others about them. Additionally, *controllers* share information about agents' actions and report their agents about *convention* violations. In their approach: agents can adapt their *conventions* but not the *rules*; *social structure* is derived from norm violation; and *controllers* are just supervisors. On the contrary, our *assistants* can adapt both *social structure* and *norms* taking into account information about more than one *peer*.

Finally, from a network management perspective, there are several works that enhance P2P systems based on communication costs. On the one hand, some of them try to achieve it without ISPs involvement, like Ono [8]. It suggests that *peers* use information collected by Content Distribution Networks (CDN) to select their neighbours. CDNs use dynamic DNS redirection to send clients to low-latency replica servers. Thus, if two clients are sent to the same replica server, they are likely to be close to each other. Accordingly, *peers* query CDN servers and store the corresponding replica identifiers. Then, when a *peer* contacts another, it estimates their distance based on the set of replica identifiers. If it is closer than some of its neighbours, it starts the sharing process with the other *peer*. In our approach, *assistants* could also collect this kind of information to estimate the *distance* among *peers*. On the other hand, there are approaches that involve ISPs. For example, P4P [24] adds an additional layer with elements called iTrackers. These elements can access network information —e.g. topology or traffic measurements. Thus, they use it to estimate the *distance* among *peers* and inform to P2P ordinary *trackers*. Then, *trackers* may suggest different neighbours to a *peer* depending on this information. As it can be observed, both approaches only adapt the *social structure*, and they cannot update network

consumption to balance net capacity and traffic.

7 Conclusions

This work presents 2-LAMA, an architecture in two layers that brings system-wide adaptation to MAS organisation. In 2-LAMA, an additional distributed *Assistance* layer –we name *meta-level*– perceives information about agents and their environment, and is able to decide how to adapt the system's organisation. This *Assistance* layer corresponds to the top abstraction layer in our *Coordination support* model of MAS engineering. Our vision is that the system infrastructure should simplify agent development. Thus, we endow the system with adaptation capabilities through an additional layer instead of expecting the agents to increase their behaviour complexity. In fact, this approach opens new research challenges in the MAS area, specially in open MAS, where agents and infrastructure are developed by different parties.

The paper defines the 2-LAMA general model, whose organisation includes norms to regulate participants' activity. We illustrate and analyse our approach by means of a case study that corresponds to a simplification of a P2P sharing network. Its dynamic environment and organisation make it specially suitable for applying our norm adaptation proposal. Hence, we present an adaptive MAS that can update its organisation depending on underlying network status.

In order to prove our proposal feasibility empirically, we have defined several experiments. First, we have shown the general improvements of the 2-level architecture versus other brute force and centralised approaches. Then we have studied the averaged effects of introducing norms to the organisation. From this analysis, we have concluded that norm configuration is a complex issue –specially if done at design time rather than at run-time– that can lead to dissimilar performance results. Then, we have got into the detail of analysing the adaptation process in an specific system configuration. Overall, we have obtained results that improve previous ones significantly. Therefore, we can conclude that Assistance is possible and that the cost of adding the *Assistance* layer is lower than the obtained benefit.

Finally, as future work, we plan to confront further issues in open MAS such as how the system should react to agents joining or leaving the MAS anytime, or transgressing its organisational restrictions.

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.

References

[1] Alexander Artikis, Dimosthenis Kaponis, and Jeremy Pitt. *Multi-Agent Systems: Semantics and Dynamics of Organisational Models*, chapter Dynamic Specifications of Norm-Governed

Systems. 2009.

[2] G. Boella and L. van der Torre. Regulative and constitutive norms in normative multiagent systems. *Proceedings of KR'04*, pages 255–265, 2004.

[3] Olivier Boissier and Benjamin Gâteau. Normative multi-agent organizations: Modeling, support and control. In *Normative Multi-agent Systems*, 2007.

[4] Jordi Campos, Maite López-Sánchez, and Marc Esteva. Assistance layer, a step forward in Multi-Agent Systems Coordination Support. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, to appear 2009.

[5] Jordi Campos, Maite López-Sánchez, and Marc Esteva. Multi-Agent System adaptation in a Peer-to-Peer scenario. In *ACM SAC09 - Agreement Technologies*, to appear 2009.

[6] Jordi Campos, Maite López-Sánchez, Juan A. Rodríguez-Aguilar, and Marc Esteva. Formalising situatedness and adaptation in electronic institutions. In *to appear in Coordination, Organizations, Institutions, and Norms in Agent Systems IV. Lecture Notes in Computer Science (LNAI)*. Springer, 2009.

[7] Kathleen Carley. Computational and mathematical organization theory: Perspective and directions. *Computational & Mathematical Organization Theory*, 1(1):39–56, 1995.

[8] David Choffnes and Fabián Bustamante. Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems. *SIGCOMM Comput. Commun. Rev.*, 38(4):363–374, 2008.

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

[10] 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.

[11] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[12] Marc Esteva. *Electronic Institutions: from specification to development.* IIIA PhD Monography. Vol. 19, 2003.

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

[14] R. Hilpinen, S. Kanger, K. Segerberg, and B. Hansson. *Deontic Logic: Introductory and Systematic Readings.* Reidel, 1971.

[15] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *Knowl. Eng. Rev.*, 19(4):281–316, 2004.

[16] Jomi Fred Hübner, Jaime Simao Sichman, and Olivier Boissier. Using the $\mathcal{M}$oise+ for a cooperative framework of mas reorganisation. In *LNAI - Proc. of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, volume 3171, pages 506–515. Springer, 2004.

[17] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems*, 1:275–306, 1998.

[18] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[19] M.J. North, T.R. Howe, N.T. Collier, and J.R. Vos. Repast Simphony Runtime System. In *Agent Conference on Generative Social Processes, Models, and Mechanisms*, 2005.

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

[21] John Searle. *The Construction of Social Reality.* The Free Press, New York, 1995.

[22] 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.

[23] Brian C. Smith. Reflection and semantics in a procedural language. Technical Report MIT/LCS/TR-272, 1982.

[24] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Liu, and Avi Silberschatz. P4P: provider portal for applications. 2008.

[25] Chongjie Zhang, Sherief Abdallah, and Victor Lesser. MASPA: Multi-Agent Automated Supervisory Policy Adaptation. Technical Report 03, 2008.