Computing (2011) 91:169–215
DOI 10.1007/s00607-010-0141-9

Organisational adaptation of multi-agent systems in a peer-to-peer scenario

Jordi Campos · Marc Esteva · Maite López-Sánchez · Javier Morales · Maria Salamó

Received: 25 January 2010 / Accepted: 4 October 2010 / Published online: 3 December 2010

Abstract Organisations in multi-agent systems (MAS) have proven to be successful in regulating agent societies. Nevertheless, changes in agents' behaviour or in the dynamics of the environment may lead to a poor fulfilment of the system's purposes, and so the entire organisation needs to be adapted. In this paper we focus on endowing the organisation with adaptation capabilities, instead of expecting agents to be capable of adapting the organisation by themselves. We regard this organisational adaptation as an assisting service provided by what we call the *Assistance Layer*. Our generic Two Level Assisted MAS Architecture (2-LAMA) incorporates such a layer. We empirically evaluate this approach by means of an agent-based simulator we have developed for the P2P sharing network domain. This simulator implements 2-LAMA architecture and supports the comparison between different adaptation methods, as well as, with the standard BitTorrent protocol. In particular, we present two alternatives to perform norm

Communicated by T. Eiter.

J. Campos (✉) · M. López-Sánchez · J. Morales · M. Salamó
MAIA Department, Universitat de Barcelona,
Gran Via de les Corts Catalanes 585, 08007 Barcelona, Spain
e-mail: jcampos@maia.ub.es

M. López-Sánchez
e-mail: maite@maia.ub.es

J. Morales
e-mail: jmorales@maia.ub.es

M. Salamó
e-mail: maria@maia.ub.es

M. Esteva
Artificial Intelligence Research Institute (IIIA),
Spanish National Research Council (CSIC), Campus Universitat Autònoma de Barcelona,
08193 Bellaterra, Spain
e-mail: marc@iiia.csic.es

adaptation and one method to adapt agents' relationships. The results show improved performance and demonstrate that the cost of introducing an additional layer in charge of the system's adaptation is lower than its benefits.

Keywords Adaptation · Organisation · Coordination · Norms · MAS · CBR

Mathematics Subject Classification (2000) 68T42 · 68T05

1 Introduction

Briefly, multi-agent systems (MAS) can be defined as computational systems where a set of autonomous agents interact within an environment [31]. In such systems, in addition to the system design purpose, agents have individual goals and a certain autonomy to pursue them by/while interacting with other agents. Depending on the nature of the MAS, agent interaction can be collaborative, competitive or both at the same time. We refer to it henceforth as *coordination* and assume that it is structured by means of a *coordination model*. This coordination model may be explicitly designed or may emerge implicitly as a result of agent interactions [43]. In particular, Organisation Centred MAS (OCMAS [22]) approaches[1] make coordination models of this kind explicit through regulative structures called organisations [22]. An organisation constrains the system evolution and allows agents to construe other participants' behaviour by considering organisational components such as social conventions or enacted roles. Thus, they help to face the inherent complexity of MAS [28]. In fact, agent organizations are inspired by human real-world organisations, which have also proved useful in structuring human societies.

In addition to defining the coordination model, most organisational approaches provide an operational framework. This framework includes domain-independent infrastructure services in charge of enhancing both system development and deployment [20,30]. Therefore, we can define *Coordination Support* [7] as a mechanism that aids agent interaction—i.e. coordination—by encompassing services that enable the operability of the system. These services range from connectivity services that support data exchange to organisational services devoted to role enactment or rule enforcement. Moreover, our notion of *Coordination Support* can be extended by a new set of services located in an *Assistance layer* [7] which assists coordination rather than merely enabling it. Thus, instead of "making coordination happen", these services "help to coordinate seamlessly". A number of assistance services have been identified [8]. Nevertheless, our research focuses on the *Organisational Adaptation* service, which adapts the organisation in order to achieve its design objectives. This service is particularly useful when environmental or agent population changes distract the organisation from its goals. As a matter of fact, organisation adaptation has been studied for more than a decade now [15,25,27,29] and it still constitutes a major research topic [34,44], since it can help to obtain the system's expected performance under changing circumstances and/or in dynamic environments. Runtime adaptation

[1] OCMAS [22] as opposed to ACMAS (Agent Centred MAS), which follow the emergence approach.

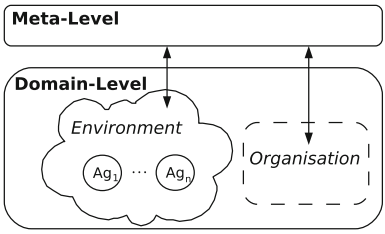

Fig. 1 Meta-Level perceiving Domain-Level's activity and adapting its organisation

is also in line with the computational organisational theory, which claims that the best organisation designs are domain- and context-dependent [13]. As a further remark, when regarding the MAS as a whole, the organisational adaptation can be related to reconfiguration in autonomic computing, since it allows the whole MAS to reconfigure itself without human intervention [33].

As far as organisational adaptation is concerned, most studies consider cooperative agents that happen to be designed in-house [18,49]. The control over the whole system in these approaches allows adaptation to be conducted by supervisor agents in charge of redistributing new tasks among agents. However, task assignment somewhat limits agents' autonomy and requires them to be cooperative, and therefore, it is an unsuitable approach when dealing with self-interested autonomous agents. Furthermore, task decomposition approaches cannot be applied for domains lacking a direct mapping between the system's goals and agents' tasks. Thus, for example, in a traffic scenario, a goal of collision minimisation cannot be directly mapped into car agents' basic driving tasks. Alternatively, rather than assigning tasks, *norms* can regulate agents' activity and still preserve much of their autonomy and individual interests. Afterwards, under changing circumstances, these norms and/or the relationships among agents can be adapted to better accomplish the system goals. As previously stated, we consider OCMAS and ACMAS as alternative approaches—see [22] for definitions. From an ACMAS perspective, organisational changes are expected to emerge from agents' activity. In contrast, OCMAS perspective supports to reason about the organisation and to adapt it so as to induce changes in agents' activity. In particular, when dealing with adaptive OCMAS systems, some research questions arise: how the organisation is specified; to what extend it influences agents' behaviour; who defines it; how it can be adapted; who is in charge of adaptation; when adaptation occurs; how to deal with transition periods; and last but not least, what is the cost of change adoption.

We can design the Assistance layer mentioned above by following an abstract architecture with a distributed *meta-level* providing different assistance services to a domain-specific MAS, which we call the *domain-level*. This paper focuses on—and formalises—the Organisational Adaptation service, which perceives *domain-level* activity and adapts its organisation to improve the system's performance—see Fig. 1. In particular, the meta-level is composed of several *assistant* agents which reason at a higher level of abstraction than regular MAS participants and provide them with assistance services in a distributed manner.[2] Thus, we call our approach "Two Level

[2] Assistants are internal agents considered as trusted third parties by external participant agents so that they can consider their directions to be trustworthy.

Assisted MAS Architecture (2-LAMA)". We evaluate an implementation of our overall approach in a Peer-to-Peer sharing network (P2P) as a representative case study of a dynamic MAS, with no direct mapping between goals and tasks, and where agent autonomy is preserved. In such a network, a set of third-party developed agents contact each other to share data. Their organisation requires adaptation, since changes in agent population and network load due to message transmission make it necessary to adjust their coordination model. Finally, we use a simulator that we have developed to perform an empiric evaluation of different alternatives to organisational adaptation, such as adapting norms by using heuristics or machine learning—currently Case-Based Reasoning (CBR). Moreover, as base-line, we also compare our adaptive approaches to a non-adaptive approach widely used in P2P: the BitTorrent protocol [4].

The rest of the paper is structured in seven sections. Our general model of the Assistance layer and its Organisational adaptation service is described in Sect. 2, which also includes the definition of our proposed architecture 2-LAMA. This architecture is applied to the P2P case study in Sect. 3 by modelling it as a MAS with our two-level perspective, and by specifying both the interaction protocol and the network model that support agent communications. Afterwards, Sect. 4 details how the Organisational adaptation is conducted in this case study, including both norms and relationships among agents according to their social structure. It focuses on norm adaptation and provides a detailed description of our heuristic and machine learning adaptation approaches. In Sect. 5, the empirical evaluation of different adaptation alternatives using our simulator is described and discussed. Section 6 discusses related work so as to further contextualise and justify our approach. Finally, Sect. 7 presents our conclusions and future work.

2 General model

This section is devoted to describe our *Coordination Support* model and its stratification by means of a two-layer abstract architecture. Although it provides different coordination support services, we further elaborate on the formalisation of the organisational adaptation service. Along this section, concepts are illustrated in a traffic scenario.

2.1 Notation

Before presenting the formalisation of our model, it may be useful to introduce the following notation conventions.

- Upper-case denotes types whereas lower-case denotes type instances—it is worth mentioning we consider types as sets. As an illustration, in order to define the function that returns the speed of a given agent, we define a relation between the set of agents (Ag) and the natural numbers as $speed : Ag \rightarrow \mathcal{N}$. Hence, $speed(ag)$ returns the speed of agent $ag \in Ag$.
- Superscripts are used to differentiate among different grouping levels. In particular, an empty superscript denotes individual level, a 'S' superscript stands for system

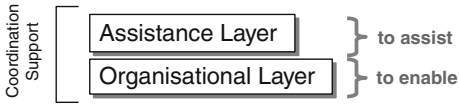

Fig. 2 Coordination Support layers

level, and a 'C' superscript denotes an intermediate level between individual and system levels. For instance, ag stands for an agent, ag^C denotes a cluster of agents (i.e. Ag^C is the type of such a set), and ag^S stands for all the agents in the system (i.e. Ag^S is the type of such a set).
- Subscripts are used to differentiate among elements in a set. For instance, $ag_j \in ag_i^C$ stands for the jth agent of ith cluster. Usually, we use superscript i to index clusters and superscript j to index agents.
- In general, n denotes the total number of clusters (i.e. usually i—the index of clusters—stands for a value among 1 and n), whereas m_i denotes the total number of agents within ith cluster (i.e. usually j—the index of agents—stands for a value among 1 and m_i).

2.2 Coordination Support

We use the term *Coordination Support* [7] to denote those services that are useful for agent coordination. Services offered by currently available infrastructures in OCMAS approaches range from elemental connectivity to organisational mechanisms. However, we propose to provide new services that assist coordination rather than just enabling it. We illustrate this concept in Fig. 2, where these services are classified in two layers: an *Organisational Layer* that provides coordination enabling services, and an *Assistance Layer* on top of it, which provides coordination assistance services —by counting on previous layer services to provide its functionalities. The latter (sub)subsections detail both layers and the services they comprise.

2.2.1 Organisational Layer

Within organisation-centred MAS (OCMAS) approaches, we define an *Organisational Layer* that provides a variety of domain-independent enabling services. These services are meant to enhance both system development and deployment. They can range from basic connectivity services that support data exchange to higher-level organisational services devoted, for instance, to role enactment or rule enforcement. In order to support agent coordination, organisational services require an organisation to be defined. In fact, its explicit specification will allow the layer on top (see 2.2.2) to perform its assisting tasks.[3] We assume an organisation is characterised by three main components:

[3] Having explicit components makes easier to perform *computational reflection*—that is to build software that observes and modifies its own structure [45].

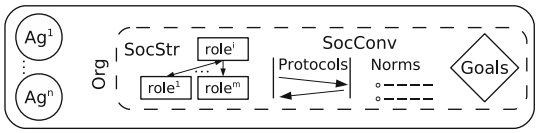

Fig. 3 Organisation model

Definition 1 We define an organisation as:

$$Org = SocStr \times SocConv \times Goals \tag{1}$$

where:

- $SocStr$ is a *social structure* consisting of a set of roles (Rol), groups ($Group$) and the relationships (Rel) among agents playing certain roles/belonging to certain groups. As an illustration, we detail them in a traffic scenario. In this scenario, there are two types of agents: car agents and police agents—i.e. $Rol = \{\texttt{Car}, \texttt{Police}\}$. We also define two groups of vehicles oil-powered/electric-powered that may include agents of both roles—i.e. $Group = \{\texttt{Oil}, \texttt{Electric}\}$. Besides, we define a relation of visibility between two cars and a relation of surveillance between a police agent and a car—i.e. $Rel = \{\texttt{Visibility}(car, car), \texttt{Surveillance}(police, car)\}$.
- $SocConv$ stands for the *social conventions* agents should conform and expect others to conform [36]. Social conventions are expressed as a set of interaction protocols ($Prot$) and a set of norms ($Norms$). Following our traffic scenario, we define a protocol ($Prot = \{\texttt{Turning}\}$) that describes the valid sequence of actions—concerning the use of blinkers—required to turn in a crossroads. Regarding norms, we can define one limiting the speed so that $Norms = \{\texttt{Speed\_limit}\}$.
- $Goals$ is a set of goals that describe the purpose of the organisation design in terms of desired values for certain *observable properties*. For instance, in our traffic example, the goals are fluid traffic flow and lack of collisions —i.e. $Goals = \{traffic\_flow = fluid, number\_collisions = 0\}$.

As an illustration, Fig. 3 depicts an organisation regulating the activities of a set of agents. The social structure defines roles, groups and their relationships. Based on that, other organisational components can refer to participant agents in a generic manner. Therefore, social conventions include protocols and norms that restrict the behaviour of specific roles. Protocols do that by defining legitimate sequences of actions whereas norms define permissions, prohibitions and obligations expressed as first-order deontic logic formulae. Notice that social conventions constrain possible actions but it is still an agent's decision to choose which actions to execute in each specific situation, and thus, agent autonomy is preserved. In addition, we assume the organisation has explicit *goals* that describe its design purpose—which may differ from a participant's individual goals. They are expressed as a function over the system's observable properties and may include reference values they should approach. In this way, system performance can be evaluated by using these *goals* to determine the extent to which the system is fulfilling its design objectives.

2.2.2 Assistance Layer

Our proposal consists on adding an *Assistance Layer*—on top of previous one—in charge of facilitating the enrolment of third-party agents and/or adapting their organisation. This layer provides two main types of services: assisting individual agents to achieve their goals under current organisation and context, (*Agent Assistance*); and adapting the organisation to varying circumstances (*Organisational Assistance*).

The former includes services to provide agents with useful information to participate in the MAS (*Information service*); to provide justifications of the consequences of their actions, for example, when an agent action is not allowed (*Justification service*); to suggest alternative plans that conform *social conventions* (*Advice service*) and to estimate the possible consequences of certain actions due to current conventions (*Estimation service*). As an illustration, in our traffic example, an information service notifies cars about updates in the speed limit norm. Afterwards, if a police agent fines a car for exceeding this speed limit, a justification service can detail the violated norm and the detection circumstances. Additionally, an advice service provides alternative routes, whose trip time can be approximated by the corresponding estimation service.

The latter, the *Organisational Assistance*, consists in adapting the existing organisation to improve the system's performance under varying circumstances. Within a rational world assumption, we propose adaptation to be driven by the goal fulfilment criteria. Thus, in our traffic example, the speed limit norm can be updated with the aim of minimising average trip times and number of collisions. In order to accomplish that, the *Assistance Layer* requires some way (i) observing system evolution, (ii) comparing it with the organisational goals and (iii) adapting the organisation accordingly. This paper focuses on the organisational assistance service, nevertheless, we refer the reader to [7] for further details of agent assistance enumerated services.

2.3 Proposed architecture: 2-LAMA

This section describes the abstract architecture we propose to distribute the assistance layer services previously mentioned.

2.3.1 Abstract Architecture

An abstract *Two Level Assisted MAS Architecture* (2-LAMA) was proposed [9] to implement a MAS with the assistance services described. Conceptually, it consists of a distributed *meta-level* (ML) that provides assistance to the part of the system performing domain activities (i.e. the *domain-level,* DL). As shown in Fig. 4, both levels communicate through an interface. Furthermore, if we assume a multi-agent system to be composed of a set of agents within an environment that participate in an organisation (see Eq. 2), then we can express this architectural stratification by means

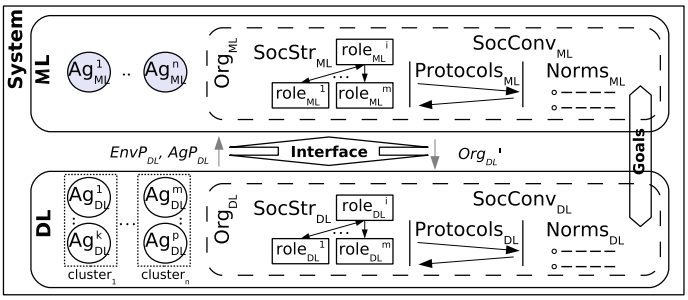

Fig. 4 Two Level Assisted MAS Architecture (2-LAMA)

of subsequent Eqs. 3, 4 and 5.[4] Since agents at meta-level provide assistance services, we henceforth refer to them as *assistants*. Notice that, it is possible to nest subsequent meta-levels which update the previous level's organisation. Regarding the system's design purposes ($Goals \in Org$), they are to an extend shared between both levels, since they are specified at domain level ($Goals_{DL} \in Org_{DL}$) and the meta-level's goals ($Goals_{ML} \in Org_{ML}$) come down to help the domain level to achieve theirs.

$$MAS = Ag \times Org \times Env \tag{2}$$

$$2LAMA = ML \times DL \tag{3}$$

$$ML = Ag_{ML} \times Org_{ML} \times Env_{ML} \tag{4}$$

$$DL = Ag_{DL} \times Org_{DL} \times Env_{DL} \tag{5}$$

Equation 6 illustrates the idea that, since the ML needs to perceive DL's activity, the whole DL could be considered as part of meta-level's environment. Obviously, observation—transmitted trough the interface—can only be done over agent/environment observable properties ($AgP^S_{DL}/EnvP^S_{DL}$) and the organisation specification (Org_{DL}).

$$Env_{ML} = EnvP_{ML} \times AgP^S_{DL} \times Org_{DL} \times EnvP^S_{DL} \tag{6}$$

Nevertheless, locality—a fundamental feature of any MAS—is also applied to this architecture. In this manner, assistants are just "in charge of" assisting a subset (a *cluster* hereafter) of domain-level agents ($cluster \subset Ag_{DL}$). This leads to a partial information assumption, where assistants only perceive the observable properties of both its assisted agents and the local environment where they are situated. Clusters are defined depending on domain-specific criteria. Considering our traffic example, the meta-level (ML) corresponds to a traffic regulatory authority that dictates traffic regulations over an (agent populated) road network, the domain-level (DL). In this setting, clusters are defined as the set of vehicle agents traversing a given road-network

[4] ML an DL subscripts are introduced here to distinguish system main components among those two levels. Nevertheless, in subsequent sections, there will be no ambiguities when referring to them, and so these subscripts will be omitted for the sake of notation simplicity.

region. Thus, an assistant can observe, among other properties, the speed of its assisted cars (AgP_{DL}) or the traffic flow density in its region ($EnvP_{DL}$).

In general, as Eq. 7 shows, agents can be characterised by a set of properties. All agents share the same properties ($AProp_l \in AgP$), but with different values (e.g. cars driving at different speeds $AgP = \{Speed\}$ in the traffic scenario). Thus, an assistant is able to observe the properties of all agents in its cluster (see Eqs. 8). Similarly, Eq. 9 computes the set of system-wide observed properties as the union of properties observed along clusters. Regarding environment properties, we use an analogous notation to denote the properties of the environment region where a cluster of agents are located ($EProp_l \in EnvP^C$), and how they are aggregated at system level ($EnvP^S$) (see Eqs. 10 and 11). Again, in the traffic example, an assistant can measure the average traffic density of the road-network region it is in charge of. Afterwards, if assistants share their data, the global average value can be computed.

$$AgP = AProp_1 \times \cdots \times AProp_{\#ag\_prop} \tag{7}$$

$$AgP^C = \bigcup_{j=1}^{m_i} AgP_j \tag{8}$$

$$AgP^S = \bigcup_{i=1}^{n} AgP_i^C \tag{9}$$

$$EnvP^C = EProp_1 \times \cdots \times EProp_{\#env\_prop} \tag{10}$$

$$EnvP^S = \bigcup_{i=1}^{n} EnvP_i^C \tag{11}$$

2.3.2 Discussion

So far we have presented an abstract architecture containing a separated assistance layer (the ML) with a distributed design. Separation of concerns and distribution are two design decisions that follow the MAS paradigm.[5] Therefore, they also benefit from the same advantages of robustness and the lack of global-information requirements. Separation of concerns allows assistants to reason at a higher level of abstraction than domain-level agents since they can be completely devoted to summarise and share local data. This is also beneficial for DL agents, since they do not need to increase their reasoning complexity. Additionally, having assistants separated, allows to grant certain information privileges such as having access to environmental properties (since it is not always the case that, for example, cars have access to the average traffic flow density). Similarly, their decision making involves system goals that may not be available to domain-level agents. Furthermore, they can individually specialize to provide specific services or to reorganise their society so to best fit the heterogeneity and dynamism of the needs of assisted agents. Finally, other agents should regard them as trusted third-parties when accepting their assistance or revealing information. Regarding distribution, it requires agent communication, our proposal minimizes its costs by

[5] AOSE: Agent Oriented Software Engineering [47].

keeping most of it local to clusters. In conclusion, the proposed architecture assumes assistants at ML to: have some information access rights; to be able to reason at a higher level of abstraction; and to be reliable. In order to fulfil these requirements, we have chosen an implementation for this abstract architecture that defines assistants as *staff* agents belonging to the organisation.

2.4 Organisational Assistance

As we stated above, this paper focuses on the Organisational Assistance as a service of the Assistance Layer. In particular, we propose a service that adapts the domain-level organisation to improve the system's performance when there are environmental or agent population changes. Next, we formalise the adaptation of the organisation and describe how it is distributed among meta-level agents.

2.4.1 The adaptation function

We define the *adaptation of an organisation* as a function α^O that provides an updated organisation (Org) depending on both the system's observable properties ($EnvP^S$ and AgP^S) and current organisation. Equation 12 below defines the domain and range of this function:

$$\alpha^O : EnvP^S \times AgP^S \times Org \rightarrow Org \tag{12}$$

Depending on the organisation design, the adaptation of its components will be totally dependent, partially related or completely independent. The more dependent they are, the more information is required when making adaptation decisions. Our driving force behind adaptation is goal accomplishment, and so $Goals$ will be considered in the adaptation functions of all organisational components. Therefore we are taking an assumption of partial-relation. If all components were dependent, then the whole organisation would be required to be considered when adapting. Obviously, such a case would also imply an increase of the adaptation function complexity.

$$\alpha^{SS} : EnvP^S \times AgP^S \times Goals \times SocStr \rightarrow SocStr \tag{13}$$
$$\alpha^{SC} : EnvP^S \times AgP^S \times Goals \times SocConv \rightarrow SocConv \tag{14}$$
$$\alpha^{P} : EnvP^S \times AgP^S \times Goals \times Prot \rightarrow Prot \tag{15}$$
$$\alpha^{N} : EnvP^S \times AgP^S \times Goals \times Norms \rightarrow Norms \tag{16}$$
$$\alpha^{G} : EnvP^S \times AgP^S \times Goals \rightarrow Goals \tag{17}$$

Accordingly, we define the *adaptation of the Social Structure* (α^{SS}, see Eq. 13) as a function that provides an updated social structure depending on system's status—i.e. its observable properties—-, the defined goals and current social structure. Analogously, we define the *adaptation of the Social Conventions* (α^{SC}, see Eq. 14) and a function to adapt each one of its components. Specifically, one function for the *adaptation of interaction Protocols* (α^P) and another one for the *adaptation of Norms*

(α^N) as defined in Eqs. 15–16. In the same manner, we define the *adaptation of Goals* (α^G) as the function expressed in Eq. 17. Overall, when having a specific organisation $org \in Org$, its adaptation is defined as follows:

Definition 2 Given an organisation $org = (socstr, socconv, goals) \in Org$ where $socconv = (prot, norms)$, current values of environment properties $envp^S$, and current values of agent properties agp^S, we define the organisation adaptation of org as:

$$\begin{aligned}
\alpha^O\left(envp^S, agp^S, org\right) &= org', where: \\
org' &= \left(socstr', (prot', norms'), goals'\right) \\
socstr' &= \alpha^{SS}\left(envp^S, agp^S, goals, socstr\right) \\
prot' &= \alpha^{P}\left(envp^S, agp^S, goals, prot\right) \\
norms' &= \alpha^{N}\left(envp^S, agp^S, goals, norms\right) \\
goals' &= \alpha^{G}\left(envp^S, agp^S, goals\right)
\end{aligned}$$

Basically, these adaptation functions evaluate the current system's status in order to modify specific organisational components. As mentioned above, these changes are driven by system goals, so that changes are introduced with the aim of inducing a higher accomplishment of current goals. Our proposal is that assistants in the meta-level apply these adaptation functions when agents at the domain-level fail to obtain the desired performance. In this sense, it can be interpreted as a top-down adaptation approach. Nevertheless, the meta-level could also be sensitive to changes in the agent population that may require fundamental changes—such as goal adaptation. This case is closer to a bottom-up approach, and thus, a hybrid adaptation approach may be more flexible. Notice that the goals adaptation function (α^G) requires special attention since its outcomes affect the rest of adaptation functions (see subsequent subsubsection on Frequency for a further discussion on that). More importantly, it may change fundamental design goals, and so, some basic system properties should be guaranteed by means of additional mechanisms such as specific goal updating policies. Although this discussion goes beyond the scope of this paper, we envision that measures related to the number of convention violations—which may be related to agents' degree of satisfaction—may motivate reconsidering some goals.

As an illustration, we provide some brief descriptions of these adaptation functions in our traffic scenario. This scenario has a social structure adaptation function (α^{SS}) that is able to add roles (e.g. $rol'_{DL} = rol_{DL} \cup \{\texttt{Ambulance}\}$) and their relationships (e.g. $rel'_{DL} = rel_{DL} \cup \{\texttt{PickInjured}\,(ambulance, car)\}$) to deal with collisions and help in restoring traffic flow. What is more, it has a protocol adaptation function (α^P) that can update current turning procedure to include safety distances (e.g. $prot'_{DL} = \{\texttt{TurningSafely}\}$) to prevent collisions. In addition, its norm adaptation function (α^N) is able to decrease the speed limit to avoid collisions when there

are a large number of cars. Finally, a goal adaptation function (α^G) may update goals to give more weight to collisions than to traffic flow.

2.4.2 Distributed adaptation in 2-LAMA

In our 2-LAMA approach, the adaptation function (α^O) defined in the above subsection is performed by the meta-level to adapt the domain-level's organisation. The distributed nature of meta-level raises some issues that we need to take into account. First, each assistant perceives partial information about system status and computes a summary that shares subsequently with other assistants. Second, organisation adaptation is distributed between assistants. In this manner, each assistant computes the desired adaptations for each component, and later on, their adaptation proposals have to be combined to end up with new organisational configurations. Finally, adaptation costs must be taken into account when deciding the adaptation frequency. The rest of this subsection is devoted to exploring each of these issues: information, decision making and frequency.

Information

As mentioned in Sect. 2.3, in 2-LAMA, each assistant perceives information about the cluster of agents it assists and partial information about the corresponding environment. We refer to this information as assistant 's *local information* (agp_i^C, $envp_i^C$, i being the assistant's index). Afterwards, it shares a summary of this information with other assistants. Thus, we define as *remote assistant information* all pieces of *summary information* received from other assistants ($sump_1, \ldots, sump_{i-1}, sump_{i+1}, \ldots, sump_n$). This way, each assistant has an abstraction of overall information when taking its decisions. Finally, we define the *knowledge* of an assistant ($knowp_i$) as the aggregation of its local and remote pieces of information.

This modelling requires us to define two processes: how local information is summarised and how an assistant aggregates its local and remote information. First, we define a *summary function* (σ, see Eq. 18) which constructs a summary ($sump_i \in SumP$,[6] see Eq. 19) out of an assistant's local information. Thus, statistical functions, such as mean or average, are good candidates for summary functions. Second, we define the *aggregation function* (λ, see Eq. 20) as the process that combines an assistant's local information with pieces of remote information (i.e. a set of summaries) to obtain an assistant's *knowledge* ($knowp_i$ in Eq. 21). This *knowledge* is of type $KnowP$ and will be used in subsequent adaptation functions.

$$\sigma : EnvP^C \times AgP^C \rightarrow SumP \tag{18}$$

$$sump_i = \sigma(envp_i^C, agp_i^C) \tag{19}$$

$$\lambda : EnvP^C \times AgP^C \times (SumP)^{n-1} \rightarrow KnowP \tag{20}$$

$$knowp_i = \lambda(envp_i^C, agp_i^C, \{sump_1, \ldots, sump_{i-1}, sump_{i+1}, \ldots, sump_n\}) \tag{21}$$

[6] Notice that $SumP$, the type of this information summary, is not qualified by the cluster (C) superscript because there is no need to differentiate it from $SumP^S$.

As an example, in the traffic scenario, where an assistant is in charge of a region of a road-network, the summary function (σ) consists in computing the average traffic flow density in this region as well as the average speed of the cars traversing it (i.e. $SumP = \mathbb{R} \times \mathbb{R}$, $sump_i = (avg(\{density_r | r \in Area_i\}), avg(\{speed_{i,j} | j = 1..m_i\}))$). Similarly, the aggregation function (λ) is an average of the summary of local information plus all remote information received (i.e. $KnowP = \mathbb{R} \times \mathbb{R}$, $know_i = (avg(sump_i^{density} \cup \{sump_x^{density} | x \neq i\}, avg(sump_i^{speed} \cup \{sump_x^{speed} | x \neq i\})))$.

Finally, it is worth noticing that, although previous formulae assume assistants receive remote information from all other assistants (i.e., $(otherSumP_i = \{sump_x | x \neq i\})$), it is not required to be the case. Actually, it depends on: whether or not the meta-level's social structure is fully connected, the reliability of communications, and the assistants' capacity to gather local information. Obviously, the lack of information may affect assistant's (and thus ML's) performance.

Decision making

Distribution at meta-level concerns both information and decision making. As mentioned above, assistants initially make their individual decisions based on the available information and the system's goals. Afterwards, they reach an agreement over the actual domain-level organisational changes. The equations below formalise this process for the norm adaptation case. First, as shown in Eq. 22, we define the decision making of a single assistant i (i.e. a *partial norm adaptation function* α_i^N), as a function with a similar domain and the same range as the meta-level adaptation function previously introduced (α^N, Eq. 16). In particular, when an assistant i applies its adaptation function α_i^N, it uses its knowledge, the system's goals and organisational norms to make its own decision about the definition of new norms. Afterwards, all assistants perform an *agreement process* by means of a function (β_{α^N}) which takes as many norm updates proposals as there are existing assistants and generates the actual norm update as described in Eq. 23.

$$\alpha_i^N : KnowP \times Goals \times Norms \rightarrow Norms \tag{22}$$

$$\beta_{\alpha^N} : (Norms)^n \rightarrow Norms \tag{23}$$

Previous equation aggregates n different decisions because it assumes norms are global—at domain-level—and relevant to all assistants. Nevertheless, it could be the case that certain norms apply only to certain contexts, and thus, just affected assistants should agree upon their update. Taking that to the limit, it may be the case that a single involved assistant does not need to agree with anyone else. Taking this into consideration, Eq. 24 shows the overall meta-level norm adaptation function (α^N) as the agreement of partial norm adaptation functions α_i^N individually computed by assistants. Following our traffic example, the adaptation function (α^N) that updates the speed limit (see 2.4.1) applies a voting mechanism to reach an agreement (i.e. $\beta_{\alpha^N}(\{norms_1, \ldots, norms_n\}) = \texttt{Voting}(\{norms_1, \ldots, norms_n\})$). Votes come from individual assistants' decisions that depend on current road density values (i.e. $\alpha_i^N(envp, agp, goals, norms) = \alpha_i^N(envp^{density}, norms)$).

$$\alpha^N(envp, agp, goals, norms) = \beta_{\alpha^N}(\alpha_1^N(knowp_1, goals, norms), \ldots, \alpha_n^N(knowp_n, goals, norms)) \quad (24)$$

Regarding the other organisational components' related functions—i.e. partial adaptations ($\alpha_i^{SS}, \alpha_i^P, \alpha_i^G$), agreement processes ($\beta_{\alpha^{SS}}, \beta_{\alpha^P}, \beta_{\alpha^G}$) and adaptation functions ($\alpha^{SS}, \alpha^P, \alpha^G$)—their domain/range and definitions are analogous to Eqs. 22–24. They use the knowledge derived from exchanged summaries and the organisational goals to compute the corresponding updated organisational component —notice though, that the domain of goal related functions is simpler, since it does not consider any other organisational components.

Frequency and costs

The process of organisational adaptation (α^O) involves some associated costs—in time and/or resources—that should be considered when defining the adaptation frequency. That is to say, the resulting frequency should keep the adaptation costs below the benefits it generates. In particular, as Eq. 25 shows, the organisational adaptation cost (c_{α^O}) comprises an information retrieval cost (c_{info^O}), a computation cost (c_{comp^O}), an adoption cost (c_{adopt^O}) and a transition cost (c_{trans^O}).

$$c_{\alpha^O} = c_{info^O} + c_{comp^O} + c_{adopt^O} + c_{trans^O} \quad (25)$$

The former, the *information retrieval cost* (c_{info^O}), is related to the cost of collecting the information required by the adaptation function. For example, collecting AgP may require some time and resources to exchange messages between assistants and participant agents. The second cost, the *computation cost* (c_{comp^O}), reflects the time and resources required to compute the adaptation function. That is to say, the time required to compute all α_i^N in parallel plus the cost of achieving an agreement (β_{α^N}) at meta-level. The third cost, the *adoption cost* (c_{adopt^O}), is related to the cost of transforming the previous organisation into the adapted one. As an illustration, when a norm is updated, messages must be sent to inform agents and they may need time to alter their activity in order to comply with the new norm. The last cost, the *transition cost* (c_{trans^O}), is the time and resources required for system's stabilisation. In this manner, the new organisation can be evaluated without interference from the previous one, since the effects of previous norms may persist longer in the environment than their actual activation period.

Furthermore, as the organisational adaptation function (α^O) is defined by the adaptation functions of all its components (see Definition 2), its costs depend on a combination (f) of the costs of each adaptation function. For example, as shown in Eq. 26, the information retrieval cost derives from the cost s of collecting information to perform: the social structure adaptation ($c_{info^{SS}}$), the adaptation of social conventions ($c_{info^{SC}}$) and the goal adaptation (c_{info^G}). Notice that in Eq. 26 there is not an addition but a function of those costs since there may be retrieved information that is useful to adapt more than one component. Thus, the cost of retrieving such shared information may

be present in different component associated costs (e.g. $c_{info^{SS}}$, $c_{info^{SC}}$, $c_{info^{G}}$), but only needs to be added once to final cost (e.g. $c_{info^{O}}$).

$$c_{info^{O}} = f(c_{info^{SS}}, c_{info^{SC}}, c_{info^{G}}) \tag{26}$$

What is more, the adaptation frequency ($freq$) of each of these organisational components can be chosen depending on its associated costs—and benefits. For instance, as Eq. 27 shows, the cost of retrieving information in order to perform the social structure adaptation ($c_{info^{SS}}$) is the sum of the cost of collecting information every time (x) the $SocStr$ is adapted. In this equation, the number of added terms ($\#\alpha^{SS}$) is the total number of performed $SocStr$ adaptations, which depends on the frequency[7] of adapting the social structure ($freq_{\alpha^{SS}}$) and the system's execution time (t). Thus, the higher the frequency, the higher the number of performed adaptations and the higher its associated cost.

$$c_{info^{SS}} = \sum_{x=1}^{\#\alpha^{SS}} c_{info_x^{SS}}, \quad \#\alpha^{SS} = freq_{\alpha^{SS}} \cdot t \tag{27}$$

Although organisational components may adapt at different frequencies, it is important to ensure that goals are adapted with the lowest frequency (see Eq. 28). In this way, the rest of the adaptation functions may have enough time to update their corresponding organisational components to current goals before they change. Moreover, in this manner the period between goal adaptations may be long enough to allow the consequences of other component adaptations to emerge. Thus, the other adaptation functions may have its feedback.

$$freq_{\alpha^{G}} < freq_{\alpha^{SS}} \wedge freq_{\alpha^{G}} < freq_{\alpha^{SC}} \tag{28}$$

Finally, it is worth mentioning that every cost in a single adaptation step can be computed as the sum of the costs endured by all assistants. For instance, the information retrieval cost associated with $SocStr$ in adaptation step x can be expressed as the sum of the costs of collecting the information for each assistant i (i.e. $c_{info_{x,i}^{SS}}$):

$$c_{info_x^{SS}} = \sum_{i=1}^{n} c_{info_{x,i}^{SS}} \tag{29}$$

Again, we can use the traffic scenario to illustrate previous concepts. Regarding information retrieval costs ($c_{info^{O}}$), there is a cost associated to observable properties, since assistants require compiling information from radar traps (used to detect vehicle speeds, i.e. AgP) and automatic traffic counters (used to estimate road densities, i.e. $EnvP$). There is also a cost of performing the computations ($c_{comp^{O}}$) of the adaptation functions mentioned in Sect. 2.4.1 and achieving agreement between the assistants (e.g. the voting scheme cited in *Decision making*). Moreover, upon organisational component updates, domain-level agents need to be informed, and this implies an an adoption cost ($c_{adopt^{O}}$). For example, when speed limit is updated, all electronic traffic signs must be updated accordingly. Furthermore, when this new speed limit is

[7] Notice that for the sake of simplicity, we express adaptation frequencies as fixed time intervals. However, they can be dynamic depending on system status.

already communicated, there is a time cost (c_{transo}) to allow vehicles to adapt their speeds, since they cannot change their speed abruptly. Finally, the traffic scenario also requires goals to be updated at a lower frequency. Thus, for instance, the frequency of the norm adaptation function is lower than that of the goals (i.e. ($freq_{\alpha^N} < freq_{\alpha^G}$). This allows assistants to monitor traffic flow and collisions to measure goal fulfilment after updating the speed limit. Afterwards, it will only be possible to compare current performance results with previous ones if both have been computed by considering the same reference goals.

3 2-LAMA in a P2P scenario

Our case study is a Peer-to-Peer sharing network (P2P), where a set of computers connected to the Internet (peers) share some data. We have chosen to apply our model in this scenario because it is a highly dynamic environment due to the nature of Internet communications. We regard the net of actually contacted peers[8] as an instance of its organisational social structure, which is dynamically updated. Finally, this scenario allows the addition of *norms* to regulate communications. Overall, it allows us to apply our organisational and adaptive approach.

Performance in this scenario is evaluated in terms of time and network consumptions during the sharing process. Thus, we can define system's goals as the minimisation of such measures so that the faster the data is obtained and the less network is consumed, the better for the users. Notice, though, that there is a trade-off between time and network usage. Therefore, although a *peer* can potentially contact any other peer to get the data as fast as possible, it usually contacts just a subset in order to consume less of the network.

Real P2P networks are highly complex, so we try to reduce complexity by assuming certain simplifications about the protocol and the underlying network. The rest of this section provides the details of the actual scenario and our 2-LAMA approach applied to it.

3.1 Architecture in P2P

As previously mentioned, we model the set of computers of the P2P scenario as a MAS. Specifically, we apply our generic 2-LAMA architecture to this scenario. Thus, the resulting system has a domain-level to perform the sharing activity, and meta-level to adapt its organisation. The next (sub)subsections detail both components and the interface between them for a given 2LAMA specification ($2lama_{P2P}$) based on our model (2LAMA):

$$\begin{aligned} 2lama_{P2P} &\in 2LAMA \\ 2lama_{P2P} &= (ml, dl) \end{aligned} \tag{30}$$

[8] In the P2P scenario, the net of actually contacted peers is called *overlay network* since it is a net over the underlaying physical network, i.e. the Internet.

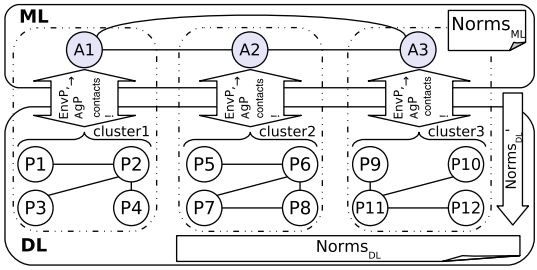

Fig. 5 2-LAMA in the P2P scenario ($P1 \ldots P12 \in ag_{dl}$, $A1 \ldots A3 \in ag_{ml}$)

3.1.1 Domain-level

We model the P2P scenario as a MAS where computers sharing data are participant agents within the domain-level (ag_{DL}). They play a single role $rol_{DL} = \{peer\}$ within an organisation (org_{DL})—see Fig. 5. We assume that the organisational goals ($goals$) is that all agents obtain the data consuming the minimal 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.

As peers usually contact a subset of neighbours, we regard these contacts as the net of relationships between agents. These relationships, which are an instance of their social structure specification ($socstr_{DL}$), are updated by the meta-level taking into account the system's status. Regarding social conventions, peers use the sharing protocol ($prot_{DL}$) specified below in Sect. 3.2 and two norms $norm_{DL} = \{normBW_{DL}, normFriends_{DL}\}$.

Both norms are prohibitions that can be expressed as:

- $normBW_{DL}$ = "a peer cannot use more than $\texttt{max}_{\texttt{BW}}$ percentage of its nominal bandwidth[9] to share data".
- $normFriends_{DL}$ = "a peer cannot simultaneously send the data to more than $\texttt{max}_{\texttt{Friends}}$ peers".

The former ($normBW_{DL}$) prevents peers from making massive use of their bandwidth to send/receive data to/from all other peers. The latter ($normFriends_{DL}$) limits the number of peers to whom a peer can simultaneously send the data.

3.1.2 Interface between levels

In 2-LAMA, the meta-level relates to the domain-level by accessing the observable properties of its agents/environment ($AgP^S/EnvP^S$), and its organisation (Org_{DL})—see Eq. 6. As we defined in Eqs. 7–11, the observable properties are compounds of information that are local to clusters. In the P2P scenario, as expressed in Eq. 31, the

[9] The *bandwidth* is the capacity to transfer data over user's network connection. It is expressed as the number of data units that can traverse a communication channel in a time unit. The less is used by the peer, the more is left for other purposes.

properties about the environment in a cluster ($EnvP^C$) represent *network bandwidths* ($NetBW^C$) and *latencies* between different peers ($NetLat^C$).

$$EnvP^C = NetBW^C \times NetLat^C \tag{31}$$

The network bandwidths ($NetBW^C$) are information about the network connection of each peer ($NetBW$)—see Eq. 32, where m denotes the number of peers in a given cluster. This information includes its *nominal bandwidth* ($NomBW$)—i.e. its maximum available—and *upload effective bandwidth* ($EffUpBW$) and *download effective bandwidth* ($EffDnBW$)—i.e. its current actual bandwidth consumption. For the sake of simplification, we assume that the nominal upload and download bandwidths are symmetrical, so that a single value describes both of them. In contrast, we consider two different effective bandwidth values, since the effective usage may in fact be different even though the nominal bandwidth is symmetrical.

$$\begin{aligned} NetBW^C &= \{NetBW_1 \dots NetBW_m\} \\ NetBW &= NomBW \times EffUpBW \times EffDnBW \end{aligned} \tag{32}$$

Besides, the *network latencies*($NetLat^C$) contain information about the round-trip time required by a unit of data to be transmitted between each pair of peer network connections—see Eq. 33.

$$NetLat^C = \left\{NetLat_{1,1} \dots NetLat_{1,m}, NetLat_{2,1} \dots NetLat_{m,m}\right\} \tag{33}$$

On the other hand, the information about agent observable properties (AgP^S, see Eq. 9) in the P2P scenario is related to datum possession and agent's activity. It is a compound of agent observable properties in each cluster (AgP^C, see Eq. 8). In fact, this information at cluster level consists of the observable properties for each agent (AgP, see Eq. 7). Specifically, the information about each agent includes information about which pieces of data it has ($Piec$). It also includes information about which action (Act) it is performing on each piece, such as serving, receiving or no action. Equation 34 contains these definitions—note that in our current P2P implementation, there is only possession/activity information about a single piece.

$$AgP = Piec \times Act \tag{34}$$

Each assistant obtains described environmental and agent information at cluster level ($EnvP^C$, AgP^C) by: collecting it from peers or accessing network inside information. In current implementation we use both, so assistants query peers and also obtain this information by themselves by observing the properties of the agents and the environment properties.

In addition, assistants exchange summaries ($SumP$) about these observable properties in order to build their knowledge ($KnowP$). Both types of information are detailed in Sect. 4.1.

Table 1 Protocol messages grouped into subsequent phases

Phase	Level	Protocol messages
Initial	Int	join<hasDatum>
Social structure	Int	get_lat<peers>, lat<peer><measure>, contact<peers>
	DL	lat_req, lat_rpl, bitfield<hasDatum>
Data sharing	DL	request, data, cancel, have, choke, unchoke
	Int	completed, has_datum<peer>
	ML	all_completed, completed_peer<peer>
Norms	ML	norm_bw<value>, norm_friends<value>, summary<sump>
	Int	norm_updated<norm_id><new_definition>

3.1.3 Meta-level

In the P2P scenario, the meta-level has a single role $rol_{ML} = \{assistant\}$. Each agent in ag_{ML} assists a disjoint subset of *peers* ($cluster_i \subset ag_{DL}$). It does so by collecting information about them and adapting their local organisation using its partial adaptation functions (specifically α_i^{SS} and α_i^{N}, see Sects. 4.2, 4.3). Its decisions are based on local information about its associated cluster aggregated to remote information about other clusters—see Sect. 4.1. This information about other clusters come s from other assistants in the meta-level. M ore specific details about organisational adaptation in P2P are shown in Sect. 4.

Besides, we assume that assistants are located at Internet Service Providers (ISP) and thus their communications are fast. Moreover, the cluster of peers of a given assistant is the set of peers connected to the same ISP. Hence, those cluster peers have lower latency communications with each other and with their assistant. As regards the meta-level *norm* ($norm_{ML}$), we consider one that limits the number of peers (in the cluster) an assistant can inform about a new peer (in another cluster) having the data. Thus, when an assistant receives the information that one peer in another cluster has completed, the number of peers it can decide to inform is limited. Therefore, the norm is a prohibition that can be expressed as:

– $normHas_{ML}$ = "Upon reception of a completed peer (peer $\notin cluster_i$) message, inform no more than $\texttt{max}_{\texttt{Has}}$ peers $\in cluster_i$ ".

3.2 Protocol

Our proposed protocol is an adapted version of the widely used BitTorrent [4] protocol. On the one hand, we assume that the information is composed of a single piece of data. In BitTorrent, the information is divided into several pieces of data, each one shared independently among peers. On the other hand, we have extended the protocol to include message between levels, belonging to the interface between them, as well as messages between ML agents. Table 1 presents the messages that are exchanged during protocol phases.

Initial phase

Initially, new peers join a cluster by contacting the assistant in charge of it (`join <hasDatum>`). In this message, they inform the assistant whether they have the datum (`<hasDatum> = 1`) or not (`<hasDatum> = 0`). In the current version, peers entering the system contact the closest assistant (the one they have a smallest latency with). Analogously, in other domains agents can use this locality criterion to choose their assistant.

It is worth to mention that in the original BitTorrent protocol, peers initially contact to a *directory service* (so called Tracker). In particular, they join this directory and obtain the references to other participants from it—i.e. agents use this service to be involved in their organisation. Accordingly, in our general model, such a service is formally provided by the Organisational Layer. Then, the Assistance Layer can access the list of organisation's participants and update their net of relationships by interacting with the Organisational Layer. However, for the sake of simplicity, in our implementation's protocol, we let peers directly provide joining information to assistants and we let these assistants directly update peers' relationships by informing peers about other participants—see next phases. Notice that, although DL is inspired in a pure P2P organisation (i.e. BitTorrent), 2-LAMA applied to current scenario is no longer following such an organisation.

Social structure phase

Afterwards, in order to compute the *net of relationships*, assistants need information about latencies between peers in their cluster. Hence, they start requesting peers to measure their latency with all other peers in their cluster (`get_lat <peers>`). After receiving that message peers measure their latencies with other peers in their cluster (`lat_req`, `lat_rpl`), and report them to the assistant (`lat <peer> <measure>`). Assistants use this information to compute a net of relationships among peers in their clusters (see Sect. 4.2), and tell each peer which other peers it has to contact (`contact <peers>`). The actual social structure—i.e. the net of relationships that fulfil social structure specification—defines which other peers each peer can contact in order to obtain the data. After receiving their contacts peers perform a handshake, introducing themselves to each one of their contacts (`bitfield <hasDatum>`), specifying whether they have the datum (`<hasDatum> = 1`) or not (`<hasDatum> = 0`).

Data sharing phase

Then, peers not having the datum start a sharing phase requesting the data from their contacts who do have it (`request`). Notice that peers can only serve a maximum number of peers (defined by the $\texttt{max}_{\texttt{Friends}}$ value). Hence, upon a request peers having the datum start sending the datum (`data`) if they are serving fewer peers than number allowed. Otherwise, if a peer is serving the maximum number of allowed peers, it replies that it can not serve the datum and will ignore any further messages (`choke`). However, as soon as one of the current data transmissions ends, the peer informs waiting peers to let them know that its status has changed (`unchoke`). Thus, peers still

lacking the datum can request it again. Peers lacking the datum are allowed to obtain data from two sources simultaneously for a short period of time. This is done in order to compare their effective *bandwidth*, and to choose the faster source and discard the other one (`cancel`).

As soon as one peer has the datum, it informs its handshake peers (`have`) so they can request the datum if they still lack it. Furthermore, upon data reception, a peer also informs its assistant (`completed`), which shares this information with other assistants (`completed_peer <peer>`). Thus, other assistants can inform some ($\texttt{max}_{\texttt{Has}}$) peers in their cluster about the new data source (`has_datum <peer>`).[10] An assistant also informs other assistants when all peers in its cluster are completed (`all_completed`), preventing further unnecessary communications.

Norms phase

During system execution each assistant perceives local information about system status ($envp_i^C$, agp_i^C) and shares some of this information with other assistants ($sump_i$, `summary <sump>`). They use this information to compute new desired norm values (see Sect. 4.3). Specifically, an assistant can propose to update the value of $normFriends_{DL}$ (`norm_friends <value>`) and $normBW_{DL}$ (`norm_bw <value>`). Then, when a new value is finally agreed among assistants, each assistant informs the peers in its cluster (`norm_updated <norm_id> <new_definition>`).

3.3 Network abstraction

We consider the network as the environment in which the sharing process takes place. Notice that the network topology and its saturation influences the communication time of messages between agents. In our network model we have individual and shared channels, which we describe next. Notice that they can be regarded as individual and shared resources, which is a common situation in many MAS scenarios. First, we are interested in having a different communication capacity for each peer. Therefore, we define a single communication link between each peer and its Internet Service Provider (ISP)—see *individual* links in Fig. 6. We also want to model simultaneous network usage by different peers, and so we define an *aggregate* link[11] between each group of peers (those connected to the same ISP) and the Internet. Finally, we abstract the Internet as a single exchange point among ISPs.

For each link, we define one channel per direction—upload/download. Each channel has its 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 as the number of data units that can traverse the channel in a time unit.

[10] Notice that `has_datum` message has more semantics than `contact` one since `has_datum` implies that the referred peer has the datum. Hence, `has_datum` receiver can start sending a `request` to the new source without exchanging `bitfield` messages to obtain other's datum possession. This saves the time and network resources consumed by the corresponding `bitfield` messages.

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

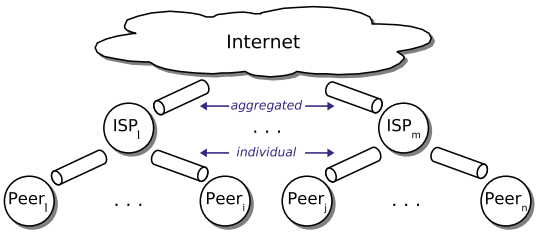

Fig. 6 Network abstraction. Each *cylinder* represents a communication link. The Internet is abstracted as a single exchange point

4 Organisational adaptation in the P2P scenario

While in the previous section, we applied 2-LAMA to the P2P scenario, in the present one we focus on how to apply the organisational adaption process described in Sect. 2.4 to this scenario. In particular, we focus on the adaptation of norms and the actual relationships among agents (the instantiation of the social structure specification). In other words, in this paper we formalise a general framework to adapt all organisational components and provide a detailed description about two of them as a proof of concept. Even more, in despite of the potential richness of these two adaptations, the scope of this paper is to present some straightforward versions that illustrate our general model's basic concepts. However, in order to show that the general model is able to deal with different implementations of these functions, we present two alternatives to perform the norm adaptation. One alternative is based on an heuristic coded by system designer whereas the other one uses learning to provide the adaptation mechanism. Specifically, we use Case-Based Reasoning (CBR) as learning technique. It is worth mentioning that we focus on norm adaptation since they are relevant in scenarios without a direct mapping between goals and tasks (e.g. current P2P sharing network scenario). In such scenarios, it is no possible to decompose a goal task into sub-tasks and assign them to agents. Hence, norms can be used to influence agent's behaviour as an alternative to task assignation.

4.1 Information required

As detailed in the general model, assistants take their demissions based on local information of their cluster and remote information received from other assistants—see 2.4.2. On the one hand, each assistant perceives local information $(envp_i^C, agp_i^C)$ directly from its cluster through the interface—see previous Sect. 3.1.2. On the other hand, each assistant receives remote information from other meta-level agents. This remote information $(SumP)$ is generated by other assistants by applying the summary function, and has to include the relevant information to compute the adaptation functions. In the P2P scenario, it includes information about completed peers which is used by the social structure adaptation function and about peers bandwidths relevant for the norm adaptation function.

Equation 35 defines the summary information $(SumP)$ in the P2P scenario. Each summary is computed by assistants from its cluster environment and agent properties—see Eqs. 36 and 37. Regarding the bandwidths, a summary contains information about

the *serving* ($SrvBW$), *receiving* ($RcvBW$) and *effective receiving* ($RcvEffBW$) bandwidths within a cluster. The first one ($SrvBW$) corresponds to the sum of nominal bandwidths of peers that are serving data—see Eq. 38, where $(i, j) \in S$ stands for peer j-th of cluster i-th. The second one ($RcvBW$) is the sum of nominal bandwidths of peers that are receiving data—see Eq. 39, while the third one ($RcvEffBW$) is similar to the previous one but adding the effective bandwidth instead—see Eq. 40. Notice that the effective bandwidth ($RcvEffBW$) may be smaller than the nominal one ($RcvBW$) when only a few data are served or if there is network saturation that delays message transport. Moreover, a summary also contains the amount of *waiting* ($Wait$) and *complete* ($Compl$) peers—see Eqs. 41 and 42 respectively. Notice that the waiting peers component corresponds to the number of peers that do not have the datum and are neither receiving it. Finally, a summary also contains the sum of all nominal bandwidths ($AllBW$) and the number of peers within a cluster ($NPeers$)—see Eqs. 43 and 44 respectively.

$$SumP = SrvBW \times RcvBW \times RcvEffBW \times Wait \times Compl \times AllBW \times NPeers \tag{35}$$

$$\sigma(envp_i^C, agp_i^C) = sump_i \tag{36}$$

$$sump_i = (srvbw_i,\ rcvbw_i,\ rcveffbw_i, wait_i, compl_i, allbw_i, npeers_i) \tag{37}$$

$$srvbw_i = \sum_{i,j \in S} (nombw_{i,j}),\ S = \left\{(i, j)\,|act_{i,j} = \texttt{serving}\right\} \tag{38}$$

$$rcvbw_i = \sum_{i,j \in R} (nombw_{i,j}),\ R = \left\{(i, j)\,|act_{i,j} = \texttt{receiving}\right\} \tag{39}$$

$$rcveffbw_i = \sum_{i,j \in R} (effbw_{i,j}),\ R = \left\{(i, j)\,|act_{i,j} = \texttt{receiving}\right\} \tag{40}$$

$$wait_i = \left|\left\{p_{i,j} \in peers_i \mid piec_{i,j} = \emptyset, act_{i,j} = \emptyset\right\}\right| \tag{41}$$

$$compl_i = \left\{p_j \in peers_i \mid piec_{i,j} \neq \emptyset\right\} \tag{42}$$

$$allbw_i = \sum_{j=1}^{m_i} \left(nombw_{i,j}\right) \tag{43}$$

$$npeers_i = m_i \tag{44}$$

On the other hand, the information used in partial adaptation functions, is an aggregation of local and remote information. In our general model, this is called knowledge information. Specifically, it includes local information ($envp_i^C, agp_i^C$) plus a weighted addition of summary information. This summary information includes all received remote information ($\{sump_j \mid j = 1..n \wedge j \neq i\}$) and the summary of local information ($sump_i = \sigma(envp_i^C, agp_i^C)$). In particular, *knowledge information* ($KnowP$) has $envp^C$, agp^C and one component related to each summary component as shown in Eq. 45—notice that we have added a k prefix to previous summary components to denote knowledge. Each of these summary-related components is the sum of a local summary component plus remote ones using different weights. For instance, the *knowledge serving bandwidth* of a given assistant ($ksrvbw_i$) is the local serving

bandwidth ($srvbw_i$) multiplied by a *local weight* (w_L) plus the sum of every remote serving bandwidth ($\{srvbw_j \mid j \neq i\}$) multiplied by remote weights ($w_{R,j}$) as defined in Eq. 46. The sum of all these weights is one, so the result has the same range as the original components. Moreover, we assume local information more relevant than remote one—see Eq. 47. Thus local information may be more relevant. For example, if local weight has its maximum value ($w_L = 1$), then each assistant takes into account only its cluster status. In contrast, if this weight is the minimum ($\forall_j w_{R,j} = w_L$), then each assistant gives the same importance to local information as to the remote information—this is the case in current tests. The mid-point is an imbalance importance among local and remote information that leads an assistant to take its decisions giving more importance to its local cluster, but taking into account the rest of the system. Furthermore, the *aggregation function* (λ, Eqs. 20 and 48) consists in computing local summary and performing weighted additions of summary components to obtain the knowledge components—see Eq. 49.

$$KnowP = \Big\langle EnvP^C, AgP^C, KsrvBW, KrcvBW, KrcvEffBW, Kwait, Kcompl, Kallbw, Knpeers \Big\rangle \tag{45}$$

$$ksrvbw_i = w_L \cdot srvbw_i + \sum_{j=1}^{n} \{w_{R,j} \cdot srvbw_j \mid j \neq i\} \tag{46}$$

$$\left(\left(w_L + \sum_{j=1}^{n} \{w_{R,j} \mid j \neq i\}\right) = 1\right) \wedge \left(\nexists w_{R,j} \mid w_{R,j} > w_L\right) \tag{47}$$

$$\lambda(envp_i^C, agp_i^C, \{sump_j \mid j = 1..n \wedge j \neq i\}) = knowp_i \tag{48}$$

$$knowp_i = \Big\langle envp_i^C, agp_i^C, ksrvbw_i, krcvbw_i, krcveffbw_i, kwait_i, kcompl_i, kallbw_i, knpeers_i \Big\rangle \tag{49}$$

In our general model, this knowledge information is available to all partial adaptation functions (see Sect. 2.4.2). However, as we mentioned above, these functions may use only part of it. In particular, the current partial social structure adaptation function (α_i^{SS}) just uses information about local latencies ($lat_i \in envp_i^C$) and about datum possession ($piec_i \in agp_i^C$ and $kcompl_i$). On the other hand, current partial norm adaptation function (α_i^N) uses information on bandwidth ($ksrvbw_i$, $krcvbw_i$, $krcveffbw_i$, $kallbw_i$) and waiting peers ($kwait_i$, $knpeers_i$).

4.2 Social structure adaptation

The social structure adaptation function (α^{SS}, Eq. 13) is performed by each assistant applying the corresponding partial adaptation function (α_i^{SS}, see Sect. 2.4.2). In fact, in this paper we adapt the actual net of relationships that fulfils the social structure specification, instead of the specification itself. Accordingly, each assistant is in charge

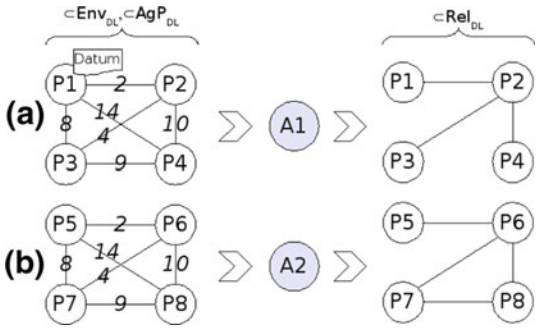

Fig. 7 Social structure adaptation examples. *Left-hand graphs* show arc latencies ($envp^C$) and datum possession (agp^C). *Right-hand graphs* represent the resulting relationships among peers (Rel_{DL}) that assistants compute. Latency values are just illustrative

of updating the net of relationships of its cluster. Specifically, an assistant can request any peer in its cluster to contact other peers in the same cluster or in another one. However, an assistant cannot provide contacts to peers in other clusters. It can only provide complete information ($compl_i \in sump_i$) to other assistants, who may send contacts to their own peers. The resulting domain-level's net of relationships, is the union of all intra-cluster and inter-cluster relationships—i.e. the agreement function (β_{SS}) is the union of partial relationships adaptation (α_i^{SS}) results.

In current implementation, assistants apply thei adaptation function during the protocol's social structure phase (see Sect. 3.2) and every time a peer is completed. In both cases, they take into account knowledge information about communication latencies ($NetLat^C$) and which peers have the datum ($Piec$ and $Kcompl$). This way, the net of relationships is created and updated depending on system's evolution.

First, during the *protocol's social structure phase*, an assistant faces the two different situations depicted in Fig. 7:

- Fig. 7a: some peers within the cluster have the datum: the assistant computes the shortest paths—using Dijkstra's algorithm [19] over arc latencies—from each peer having the datum to the rest of peers in the cluster. Then, it re-organises its cluster by telling each peer to contact with its predecessor in its shortest path to a data source—see `contact` message in Sect. 3.2.
- Fig. 7b: no peer has the datum: the assistant organises its cluster to be prepared for data entering through any peer. Accordingly, it assumes that any peer can become a data source and computes all possible shortest paths—using Dijkstra's algorithm too. Next, it provides each peer with its predecessors in all its corresponding shortest paths. This way, all peers are in contact with their neighbours which could rapidly provide the datum when entering through any node in the cluster.

Later on, every time a remote peer is completed, the partial adaptation function is invoked again. Specifically, when an assistant receives a `complete_peer` message, its complete information ($kcompl_i$) is updated. In such a case, the partial function (α_i^{SS}) determines a new net of relationships in which some local peers can contact the remote one. Then, the assistant sends `has_datum` messages to those peers to request that they contact it. Accordingly, the number of selected peers depends on norm $normHas_{ML}$—notice that it limits the amount of these messages, see Sect. 3.1.3.

4.3 Norm adaptation

In our general model, we detailed how the norm adaptation function (α^N, Eq. 24) is distributed between meta-level agents. Each assistant takes a decision using its partial adaptation function (α_i^N, Eq. 22), and the actual norm update is determined by an agreement function (β_{α^N}, Eq. 23)—see Sect. 2.4.2. In our P2P scenario, we propose two alternatives for computing the partial norm adaptation functions (α_i^N): one coded by the system designer (heuristic approach) and another one learnt by the system itself (machine learning approach). And we propose an agreement function (β_{α^N}) based on a voting approach.

Information required

The underlying rationale of the current adaptation functions is to align the amount of served data with the amount of received data. Thus, the information required consists of measures about peers serving the datum ($KsrvBW$), peers that lack it ($KrcvBW$, $KrcvEffBW$, $Kwait$) and all peers ($kallbw_i$, $knpeers_i$)—the last ones are only used by the learning approach in order to normalise the other metrics. In addition to this information included in knowledge ($KnowP$), both adaptation approaches use information derived from this knowledge and $Norm_{DL}$. This additional information is the *expected receiving bandwidth* ($RcvExpBW$), that re-scales receiving nominal bandwidth ($RcvBW$) according to current bandwidth limit ($\texttt{max}_{\texttt{BW}}$) as expressed in Eq. 50. This new information reflects that actual receiving bandwidth may be lower when there is a bandwidth limit applied to serving peers—since less data is being injected towards receiving peers.

$$rcvexpbw_i = rcvbw_i \cdot \frac{\texttt{max}_{\texttt{BW}}}{100} \tag{50}$$

Adaptation approaches

While the *heuristic approach* always uses the same coded algorithm to adapt norms—see Sect. 4.3.1—, in the *learning approach* the algorithm evolves as it learns—see Sect. 4.3.2. More specifically, the latter is based on Case-Based Reasoning (CBR), and so it uses a base of previous cases to decide on new ones.

On the other hand, as the current agreement function (β_{α^N}) follows a voting scheme, both approaches provide a result in the form of one vote about each norm. That is, one vote on the $normFriends_{DL}$ update (vFR) and another one on the $normBW_{DL}$ modification (vBW):

- $\texttt{vFR} = \{\texttt{DECR}, \texttt{SAME}, \texttt{INCR}, \texttt{BLANK}\}$: this indicates how to update $\texttt{max}_{\texttt{Friends}}$. It can be done by increasing one unit (INCR), decreasing one unit (DECR), keeping the same value (SAME) or avoiding influencing it (BLNK, i.e. a *blank ballot-paper*).
- $\texttt{vBW} = \{\texttt{DECR}, \texttt{SAME}, \texttt{MAX}\}$: this defines how to adapt $\texttt{max}_{\texttt{BW}}$. It can be done by setting it to 100% (MAX), keeping the same value (SAME) or dividing it by two (DECR).

In brief, as both approaches take the same information and provide the same sort of result, it is easy to compare them in our simulator—see Sect. 5.1.

Agreement

After each assistant computes its desired update for norm parameters all of them have to agree on their actual modification. As we said, we currently use a voting approach as the *agreement function* (β_{α^N}, see Eq. 24). In particular, we have a distributed implementation that replicates certain computations. First, each assistant sends its votes (vFR, vBW) to the rest of assistants—see `norm_bw` and `norm_friends` messages. Then, when it receives the votes of all the others, it computes the most frequent vote for each parameter (discarding *blank ballot-papers*). Afterwards, it applies the resulting solution to current norms—if there is a tie, parameters are not modified. Finally, if norms are actually updated, then each assistant sends them to its peers by means of the `norm_updated` message.

Norm adoption

Regarding norm adoption, once a peer receives new norms, it tries to fulfil them because peers do not violate norms in current implementation. Thus, when a peer receives a $normBW_{DL}$ with a $\mathtt{max}_{\mathtt{BW}}$ smaller than the one it is using, it decreases its sending ratio to fulfil this norm. Besides, when it receives a $normFriends_{DL}$, it also tries to fulfil it. This means that if a peer is serving fewer peers than $\mathtt{max}_{\mathtt{Friends}}$, it will send `unchoke` messages to those peers it had choked previously. In contrast, if it was serving to more friends than $\mathtt{max}_{\mathtt{Friends}}$, it will cancel some of those data transmissions and send a `choke` message. Nevertheless, in our current implementation, a peer does not need to cancel a data transmission if it has already sent more than 75% of the datum. This behaviour avoids cancelling data transmissions that will finish really soon. Moreover, as applying norm changes still has an associated cost—see Sect. 2.4.2—, the norm adaptation process is performed at an empirically tested time interval ($\mathtt{adapt}_{\mathtt{interv}}$) specified in Sect. 5.

4.3.1 Heuristic approach

In our *heuristic approach*, assistants use the process schematised in Algorithm 1 to implement their partial norm adaptation function (α_i^N). This algorithm receives the knowledge information ($knowp_i$) plus current norms ($norm_{DL}$) expressed by their parameter values. Notice that we use the following notation in algorithm: $\mathtt{srvBW} = ksrvbw_i$, $\mathtt{rcvBW} = krcvbw_i$, $\mathtt{rcveffbw} = krcveffbw_i$, $\mathtt{waiting} = kwait_i$, $\mathtt{rcvExpBW} = rcvexpbw_i$, $\mathtt{maxFR} = \mathtt{max}_{\mathtt{Friends}}$ and $\mathtt{maxBW} = \mathtt{max}_{\mathtt{BW}}$. In line 2, some constants are initialised to be used as thresholds in comparisons. Afterwards, the expected receiving bandwidth is computed from the nominal one re-scaled by current bandwidth limit (line 3).

The main decision regarding the choice of a $normFriends_{DL}$ is related to comparing the available bandwidth used to serve (srvBW) with the available bandwidth used to receive (rcvBW). If there is a lack of serving bandwidth (line 6), the suggestion is to

Algorithm 1 *Heuristic approach* of partial norm adaptation function (α_i^N).

```
00 def adapt( srvBW, rcvBW, rcvEffBW, waiting, maxFR, maxBW ):
01
02   τ = 0.1 ; ϵ = 0.2
03   rcvExpBW = rcvBW * (maxBW / 100)
04
05   // Adapt maxFriends --------
06   case  ( srvBW < (1-τ)*rcvBW )   : vFR = DECR
07
08   case  ( srvBW > (1+τ)*rcvBW
09                &&   waiting > ϵ ): vFR = INCR
10
11   case  ( srvBW > (1+τ)*rcvBW
12                &&   waiting < ϵ ): vFR = BLNK
13
14
15   other /*srvBW ≈  rcvBW */      : vFR = SAME
16
17
18   if( rcvEffBW <(1-τ)*rcvExpBW ): vFR = DECR
19
20   // Adapt maxBW ------------
21   case (vFR==DECR ∧ maxFR==1  : vBW= DECR
22   case (vFR==INCR ∧ maxBW<100): vBW= MAX
23   other                       : vBW= SAME
24
25   return [ vFR, vBW ]
```

decrease the number of friends. This way, server peers will be simultaneously serving data to fewer peers, and these transmissions will finish sooner. Afterwards, once these other peers have the datum, there will be more data sources in the system and it will take less time to finish the datum distribution. On the other hand, if there is an excess of serving bandwidth and there are still peers waiting for data (lines 8–9), then the assistant can increase the number of friends in order to serve more peers. There is another situation in which there is also an excess of serving bandwidth but there are no peers waiting for data (lines 11–12). This does not necessarily mean all peers have the datum, but at least the ones lacking it are receiving it from some source. In this case, the assistant uses a blank-ballot paper to let other assistants push for their own interests.[12]

Finally, if none of the previous cases holds, it means that the serving bandwidth is similar to the receiving one (line 15) then, the vote is for keeping the same norm. This is because if there is no excess of serving bandwidth, the assistant prefers to vote for the same norm instead of just leaving the decision to the rest of the assistants.

In spite of above cases, if there is network saturation in the intermediate channels, it is always better to decrease the number of friends. This will reduce the number of

[12] Notice, though, that the weighting method applied to measures may bring an *assistant* to this case when no *peers* in its *cluster* are waiting for data, but there are still waiting *peers* in other *clusters*. In such a case, if there is enough serving *bandwidth*, it is better to let other *assistants* choose by themselves the *norm* parameter values.

data transmissions. Hence, it will cut back network traffic and hopefully network saturation. In order to estimate whether there is network saturation, the assistant checks whether the effective receiving bandwidth (`rcvEffBW`) is smaller than the expected one (`rcvExpBW`). If so this suggests that data packets are delayed by the intermediate network because it is saturated. Consequently, as a solution to saturation, the assistant votes for decreasing $\texttt{max}_{\texttt{Friends}}$ (line 18).

As for the $normBW_{DL}$, it is only decreased if it is not possible to reduce the network usage further by decreasing the number of friends—since $\texttt{max}_{\texttt{Friends}}$ is already 1. In such a case, the assistant votes for dividing $\texttt{max}_{\texttt{BW}}$ by 2 (line 21). This way, server peers will use less bandwidth, which can help to diminish the network saturation. In contrast, if the bandwidth was previously limited but there is no network saturation—since the assistant chose to increase $\texttt{max}_{\texttt{Friends}}$—, then the bandwidth limit can be reset to 100% (line 22). For the remaining cases, $\texttt{max}_{\texttt{BW}}$ keeps its value (line 23).

4.3.2 Learning approach

In our *learning approach*, assistants use a Case-Based Reasoning methodology (CBR) to implement their partial norm adaptation function (α_i^N). CBR is inspired by human reasoning and memory organization. A person uses the lessons learned in similar situations to understand or solve new ones. CBR [41] is defined as the process of solving new problems by retrieving the most similar past problems from an existing knowledge-base and adapting them to fit the new situations. In CBR, problems are referred as *cases* (i.e. a case contains the problem description and its solution) and the previous experienced cases are stored in the *case base*. This CBR methodology requires the specification of a case description and the specification of certain processes used in its main cycle (e.g. how to compute the similarity between two cases).

Case description

A problem and its solution description make up a *case* that can be stored in the CBR's *case base* for subsequent usage, as a previous case. A problem ($Prob$) is described by a set attributes ($Attribs$) derived from knowledge information ($KnowP$). In our scenario, we use discrete attributes to describe a *problem*—we make their values discrete by taking a reference magnitude specified in each case. The discrete attributes are the following:

- `servingCapacity` = {`VERY_LOW`, `LOW`, `GOOD`, `HIGH`, `VERY_HIGH`}: this indicates whether there is enough serving capacity to serve all receiving peers. It is extracted from comparing the bandwidth of all serving peers ($ksrvbw$) with the bandwidth of all receiving peers ($krcvbw$). The reference magnitude of this attribute is the sum of the bandwidth of all individual channels in the system ($kallBW$).
- `netSaturation` = {`VERY_LOW`, `LOW`, `GOOD`, `HIGH`, `VERY_HIGH`}: this estimates the network saturation by comparing the actual receiving bandwidth of receiving peers ($krcveffbw$) and their expected bandwidth ($krcvexpbw$). A $krcveffbw$ smaller than $krcvexpbw$, when $ksrvbw$ is equal or greater than

krcvbw, may indicate that data packets are delayed because the network is saturated. This attribute's reference magnitude is also *kallBW*.
- `waiting` = {`NONE`, `FEW`, `A_LOT`}: this reflects the amount of peers waiting for data. l This attribute's reference magnitude is the total number of peers (*knpeers*).
- `maxShareRatio` = {`ONE`, `FEW`, `A_LOT`}: this indicates sources' maximum ratio to spread the datum. The higher it is, the more peers are receiving data at a lower bandwidth. It is the current $\texttt{max}_{\texttt{Friends}}$ normalised with *knpeers* as reference magnitude.
- `bandwidthUsage` = {`LOW`, `HIGH`, `MAX`}: this indicates the bandwidth used by peers in their communications. It is the current $\texttt{max}_{\texttt{BW}}$ normalised with 100 as reference magnitude (`MAX` = 100% means that a peer can use its full individual bandwidth).

In order to describe the *solution* of a given problem, we use two additional discrete attributes. They are two votes, one per each norm (`vFR`, `vBW`) as explained in Sect. 4.3.

CBR Cycle

The classical CBR [1] cycle has four main phases: *retrieve*, *reuse*, *revise* and *retain*. Initially, once a new problem is encountered, the first phase *retrieves* one (or several) *similar* cases from the case base. Then, the second phase *reuses* the retrieved cases to provide a solution to the new problem. Next, the third phase *revises* the results of applying this new solution. Finally, the fourth phase *retains* the new problem if it is representative enough. In addition, in our scenario, the mapping between the system's states—i.e. problems—, norms—i.e. solutions—and outcomes is highly complex. Unlike classical CBR approaches, our implementation starts with an empty case base and queries an *expert* when no similar previous case are found. Currently, we use the heuristic approach described above to emulate this expert and to feed the system with cases.

The first phase (*retrieve*) fetches the most similar cases (`retrCases`) from the case base (`caseBase`) as illustrated in Algorithm 2. It starts with an empty list of cases and a minimum reference similarity (`bestS`)—see line 2. Then, it traverses the case base—line 3—computing the similarity (Θ, see below) of the description of the problem in each previous case (`prevCase.prb`) with the new problem (`newCase.prb`)—line 4. If this similarity is greater than a *minimum trusted similarity* (`MIN_SIM`) the case is a candidate for retrieval—line 5. In particular, if this similarity is greater than any previous one—line 6—then the previous case is the one to be retrieved—line 7. Alternatively, if the similarity is equal to previous greatest one—line 9—then current previous case is recorded with the rest of similar ones —line 10. Finally, if no previous case has the minimum trusted similarity to consider it as representative enough to adapt its solution to the new problem—line 11—, the algorithm queries an expert—line 12—to solve this unknown problem. Finally, in both cases the cases are returned—line 14.

On the other hand, the *case similarity function* (Θ) among two problems ($p_x, p_y \in Prob$) consists in computing the *attribute similarity function* (θ_i) among corresponding values of the same attribute ($a_i^{p_x}, a_i^{p_y} \in Attrib_i$) to aggregate them in a weighted manner—see Eq. 51.

Algorithm 2 Retrieve phase of learning approach.

```
01 def retrieve( newCase, caseBase):
02  retrCases = {} ; bestS = 0
03  foreach prevCase in caseBase:
04    s = Θ( prevCase.prb, newCase.prb )
05    if ( s > MIN_SIM ):
06      case ( s > bestS ):
07        retrCases = { prevCase }
08        bestS     = s
09      case ( s ≈ bestS ):
10        retrCases=retrCasesU{prevCase}
11  if ( retrCases is empty ):
12    expertCase = Expert.solve(newCase)
13    retrCases  = { expertCase }
14  return retrCases
```

$$\begin{aligned} &\Theta : Prob \times Prob \rightarrow [0..1] \\ &\theta_i : Attrib_i \times Attrib_i \rightarrow [0..1] \\ &\Theta(p_x, p_y) = \sum_{i \in Attribs} \left(w_i^{\Theta} \cdot \theta_i(a_i^{p_x}, a_i^{p_y}) \right) \\ &\qquad\qquad | \sum w_i^{\Theta} = 1 \end{aligned} \tag{51}$$

In order to compute this attribute similarity function (θ_i), we define a *label distance function* (Δ_i) that provides a numeric distance among two discrete labels as shown in Eq. 52. In fact, we regard discrete labels as an ordered set of equidistant values—see Eq. 54.

$$\Delta_i : Attrib_i \times Attrib_i \rightarrow [0..\Delta_i^{MAX}] \tag{52}$$

Then, we define θ_i as an *inverse mapping* from labels's distance $[0..\Delta_i^{MAX}]$ to the [0..1] interval as shown in Eq. 53.

$$\begin{aligned} &\theta_i : Attrib_i \times Attrib_i \rightarrow [0..1] \\ &\theta_i(a_i^{p_x}, a_i^{p_y}) = 1 - \frac{\Delta_i(a_i^{p_x}, a_i^{p_y})}{\Delta_i^{MAX}} \end{aligned} \tag{53}$$

In sum, in both similarity functions (Θ, θ_i), 0 means no coincidence at all and 1 means that the items are equal—see the example in Eq. 54.

$$\begin{array}{cccc} a_{\texttt{waiting}}^{p_x} & a_{\texttt{waiting}}^{p_y} & \Delta_{\texttt{waiting}} & \theta_{\texttt{waiting}} \\ \texttt{NONE} & \texttt{NONE} & 0 & 1.0 \\ \texttt{NONE} & \texttt{FEW} & 1 & 0.5 \\ \texttt{NONE} & \texttt{A\_LOT} & 2 & 0.0 \\ \multicolumn{4}{c}{\ldots} \end{array} \tag{54}$$

Algorithm 3 Reuse phase of learning approach.

```
01 def reuse( retrCases, newCase ):
02  if ( δ(retrCases) > MAX_DIV )
03    expertCase = Expert.solve(newCase)
04    retrCases  = { expertCase }
05  sol = adapt( retrCases, newCase )
06  return Case( newCase.prb, sol )
```

From the retrieved cases, the second CBR phase (*reuse*) employs their solutions to build a new one for the current case as described in Algorithm 3. If there is more than one similar case, we use a *divergence function* (δ) to compute the divergence among them. Thus, the reuse phase starts by checking if the divergence of retrieved cases is greater than a *maximum trusted divergence* (`MAX_DIV`)—see line 2. In such a case, it considers that solutions to the previous cases are too contradictory to provide a good single solution. Then, as no representative previous case is found, the expert is consulted—lines 3–4. Once there is a set of slightly divergent previous cases—notice that a single previous case has no divergence—it adapts their solution to the current problem—line 5. This task can take into account of (i) all retrieved solutions but also (ii) the differences between the retrieved problems and the current one. In the current implementation, our *adapt* function uses only the former (i). In particular, it returns a solution composed by the most frequent `vFR` and the most frequent `vBW`. If there is a tie, the less conservative actions (i.e. `DECR`, `INCR` or `DECR`, `MAX`) have priority over the more conservative ones (i.e. `SAME`, `BLNK`).[13] These less conservative actions may make the system evolve in a different way, so there is less chance of there being a tie in a subsequent adaptation process.

The above-mentioned *divergence function among solutions* (δ) takes into account only the attribute of the `vFR` solution, since in our experiments `vBW` was correlated with it. Specifically, the divergence function that takes into account only `vFR` (δ') is the standard deviation of `vFR` discrete values converted into integers as shown in Eq. 55.

$$\begin{gathered}
\delta : (VFR \times VBW)^* \rightarrow [0..2] \\
\delta' : VFR^* \rightarrow [0..2] \\
\chi : VFR \rightarrow [-1..1] \\
\chi(\texttt{DECR}) = -1;\ \chi(\texttt{SAME}) = \chi(\texttt{BLNK}) = 0;\ \chi(\texttt{INCR}) = 1 \\
\delta'(v_{FR_1}...v_{FR_n}) = stdev(\chi(v_{FR_1})...\chi(v_{FR_n})) \\
\delta(s_1...s_n) = \delta'(v_{FR}^{s_1}...v_{FR}^{s_n})
\end{gathered} \tag{55}$$

The third phase (revise) requires a way to evaluate the solution, but current performance measure (total time) is unknown until the end of execution (the credit assignment problem [32]). As we are working on this topic in the P2P scenario, the current implementation does not revise the proposed solution.

[13] In our experiments we use a `MAX_DIV` smaller than two. Thus, if `retrCases` involve `DECR` and `INCR`, then they will satisfy line 2 and will be replaced by a single `expertCase`. Consequently, there is no possible tie among less conservative actions.

It is important to remark that the notion of case-based reasoning [1] does not only denote a particular reasoning method, irrespective of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. The fourth phase of the CBR cycle is devoted to retain new experience into the case base. When a problem is successfully solved, the experience may be retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for failure is identified and remembered in order to avoid the same mistake in the future. In short, the retention phase consists of incorporating what is useful to store from the new problem-solving experience into the existing case base. As in humans, experience comes from doing or from cognition. The former refers to retain successful experience or remember failures. The latter considers to become acquainted with an external source.

In our case, the CBR starts with an empty case base. That is, there is no previous experience available. For this reason, the CBR focus on learning from an external source (i.e. in this case, the expert) that completes the experience of the CBR—at least at first cycles. Similarly to humans, our experience comes from cognition. The use of an expert is common in CBR and it is particularly necessary when the case base does not contain enough experiences. In our CBR, we estimate that it might be a lack of experience when there is not a minimum trusted similarity or when there is a maximum trusted divergence. Under one of these circumstances, the system uses the expert to learn and to store his experience in the case base as a new case. The addition of new cases into the case base increases experience of the CBR and this experience play a fundamental role in CBR learning. Note that the case base participates throughout the CBR cycle, and so it is vital for the system's problem-solving efficiency. It is clear when the CBR lacks of experience that it is necessary to retain as much experience as possible. An open issue that will be addressed in our future work consists of retaining experience from doing. Retention strategies are not as simple as storing all solved problems, since too much information may affect the utility [46] of a CBR system. The utility problem occurs when the cost of maintaining and searching in a large case base outweighs the benefit of storing this experience.

5 Empirical evaluation

In order to empirically evaluate our general model in the P2P case study, we have implemented a P2P adaptive OCMAS simulator [10]. It provides various facilities for executing tests and analysing results. As it simulates both agents and network components, it allows us to execute different sharing methods with identical initial conditions. We use it to obtain some results about alternative coordination models among adaptive and non-adaptive approaches. In brief, they show that adaptive coordination models outperform non-adaptive ones, especially the approach that uses learning to adapt norms.

5.1 Simulator

Our simulator is implemented in Repast Simphony, extending its original capabilities to deal with OCMAS and the P2P scenario. Its internal architecture clearly isolates

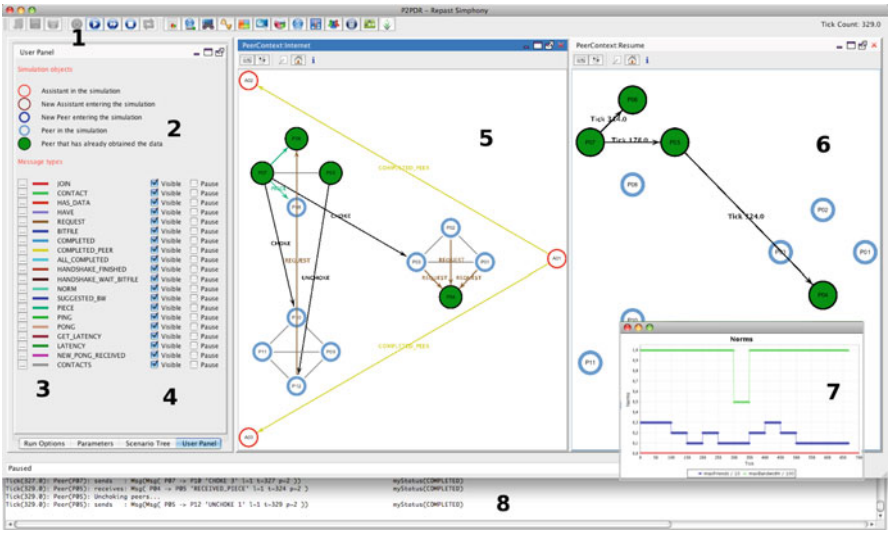

Fig. 8 P2P adaptive OCMAS simulator

different functionalities. It has three components: a Graphical User Interface (GUI), an Agent Simulator and a Network Simulator.

Figure 8 depicts its Graphical User Interface (GUI) to illustrate its general features. The *Control toolbar* (1) pertains to the original Repast GUI and makes it possible, among other things, to play the simulation, pause it or execute it step by step. In the left area, the *Legend panel* shows information about what each layout object represents (2), the colours of the different messages exchanged between agents (3), whether they are visible or not, and whether execution will pause upon sending messages of this kind (4). All these options can be modified by users. Thus, the legend allows easy identification of each agent and message to interpret what is happening in the simulation at every moment. The *Main layout* (5) shows the participants in the organisation and the communications between them. Peers and assistants are drawn according to its logical cluster topology, while messages are displayed as arrows between them with the corresponding colour defined in the legend panel. There is also a *Resume layout* (6) that shows how data have been distributed between different peers. It highlights completed peers and displays arrows connecting source and receiver agents. These arrows are labelled with the time step at which the datum was received. Moreover, our application can plot the evolution of different parameters through the simulation, such as norm updates (7). In addition, the simulator generates log files containing all the events occurring during executions (8). Afterwards, an (off-line) module facilitates the analysis of these logs by extracting relevant information that can be subsequently processed. For instance, it can compare the time spent sharing data in different configurations, or using different sharing methods. At the end of simulation, different metrics are shown as an execution summary. These metrics are also stored in the log file. Thus, once a user has used the GUI to understand an approach behaviour, he can evaluate and compare it just using logged information. Accordingly, our simulator

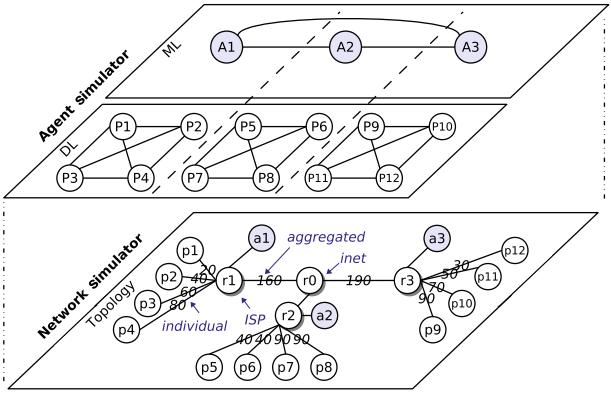

Fig. 9 Agent Simulator component over Network Simulator component

allows the execution of multiple simulations with different run options in batch mode, generating a log file for each simulation.

Regarding the *Agent Simulator* component, it provides an implementation of our suggested conceptual MAS architecture (2-LAMA)—see the top part of Fig. 9. At *domain-level*, it offers a peer code skeleton of state-based agents that follow a P2P protocol. The current implementation offers two sorts of derived peers: one that follows our protocol and one that follows the standard BitTorrent protocol (BT, see Sect. 5.2.1). Thus, we can simulate both alternatives and empirically compare their performance. In addition, at *meta-level*, the Agent Simulator component provides an assistant code skeleton where social structure and norm adaptation mechanisms have to be coded. Currently, there are three derived assistant implementations: one that only performs social structure adaptation (2L-SS, see Sect. 4.2), another one that also performs norm adaptation using the heuristic approach (2L-SS-N-Heu, see Sect. 4.3.1), and one that uses the learning approach instead (2L-SS-N-CBR, see Sect. 4.3.2). This facilitates the comparison of these adaptation approaches or even to add new ones by extending the current assistant skeleton. Next Sect. 5.2 contains further implementation details of each of these approaches.

On the other hand, the *Network Simulator* is used by agents when sending messages. It simulates a packet switching network to transport messages. Specifically, when an agent wants to send a message, it injects it into its network adaptor, which is a Network Simulator component—see Fig. 9. Then, this message is split into packets that travel along links and follow their path by switching at routers depending on network topology—see Sect. 3.3. The destination agent is informed when each packet reaches its network adaptor. Eventually, when all packets of a message arrive, the network adaptor also delivers the whole message to the destination agent. Hence, agents can pay attention to packets or just wait for entire messages. The latency of a message from one network adaptor to another depends on its size, the number of links, their bandwidth and the current traffic through them. Notice that in the current implementation, the network status changes dynamically depending on MAS activity, but we could even introduce extra non-MAS traffic to add more disruption. The lower part of Fig. 9 depicts the Network Simulator component, given the network topology used in

current tests. Clusters are formed by peer connected to the same ISP. In this example, peers `P1` and `P2` belong to the same cluster, which at the network level means that their corresponding network terminations `p1` and `p2` are connected to the same ISP (`r1`). We also have the agent `a1`, which is the assistant of these peers, connected to the same ISP (`r1`) through its corresponding network termination `a1`. Each cluster is connected to the others by means of links, so `r1` and `r2` have aggregated links connected to `r0`, which represents the interconnection through Internet, implementing the network abstraction explained in Sect. 3.3

5.2 Coordination models

The current implementation offers non-adaptive and adaptive coordination models. We implemented the BitTorrent protocol (BT, see Sect. 5.2.1) as a non-adaptive coordination model. This protocol is widely used in P2P sharing networks, so we use it as a baseline when evaluating our adaptive approaches. On the other hand, we provide three different adaptive coordination models. All of them include the Organisational Adaptation service defined in our general model—see Sect. 2.2.2. Specifically, they use an implementation of generic 2-LAMA architecture to provide it—see Sect. 2.3. Their differences reside in which organisational components they adapt and how they do it. We now describe specific details of their implementation.

5.2.1 Non-adaptive coordination model

Our protocol—see Sect. 3.2—is based on the BitTorrent protocol [4], which is widely used in P2P sharing networks. We regard a BitTorrent network from an agent-centred multi-agent system (ACMAS) perspective, where peers set their own net of relationships. We construe it as a non-adaptive coordination model, since peers always use the same mechanisms to coordinate. In contrast, in our approaches, meta-level is able to update domain-level organisation components—e.g. norms. Thus, we say that our approaches have adaptive coordination mechanisms.

We have implemented a BitTorrent version (BT [11]) based on the original protocol. The main difference is that in our implementation the data have only a single piece. The rest is equivalent, so BT has a single agent (tracker) that informs about connected peers, instead of providing further assistance like our distributed meta-level. Following the original protocol, new peers without data (interested) show their interest to peers having the datum (sources). Afterwards, sources start serving only at given intervals (`unchoke_interval`). In particular, at these intervals, a source peer communicates to four of its previously interested peers (selected peers) to say that it can serve them. Next, these peers can request the datum and all of them are served. The *selected peers* are those that were interested most recently. If two of them were interested at the same time, the one with the larger network bandwidth (`upload_bw` [14]) is selected.

[14] In a multi-piece scenario, this measure is estimated from previous piece interchanges. However, since in a single-piece implementation no estimation can be performed, its actual value is taken from the network topology. In contrast, our approach is actually estimating connectivity by sending partial data messages—see Sect. 3.2. Notice that this gives advantage to BT as peers do not have to exchange extra data messages.

In fact, if a peer's interest is older than a defined interval (`aging_period`), its age is ignored and only its peer's `upload_bw` is compared. In addition, in two out of three `unchoke_interval` selection processes, the fourth peer is *randomly* selected.

In our experiments, BT uses an `unchoke_interval` of 250 time units (ticks). It is approximately the time required to send four data messages in current topology—along an average peer individual link. Thus, it is the average time that a server peer can invest sending data to four selected peers—i.e. the number of simultaneous starting servings in BT. Also, we use an `aging_period` of 130 ticks to keep the ratio defined by the official protocol.

5.2.2 Adaptive coordination model

In contrast to BitTorrent, our approaches have an adaptive coordination model, since they include the Organisational Adaptation service proposed in our general model—see Sect. 2.4. In particular, we present three alternatives: one that only performs social structure adaptation (2L-SS), another that also performs norm adaptation using a heuristic approach (2L-SS-N-Heu) and one that uses the learning approach instead (2L-SS-N-CBR). The previous sections detail all these alternatives, especially Sect. 4. Thus, this section presents just a short description of them, stressing the value of some implementation parameters in our simulations.

First, the initial values of norm parameters ($\mathtt{max}_{\mathtt{BW}}$, $\mathtt{max}_{\mathtt{Friends}}$, $\mathtt{max}_{\mathtt{Has}}$) have to be fixed. In order to make a fair comparison among BT and 2-LAMA approaches, we have used the following initial norm parameters: $\mathtt{max}_{\mathtt{Has}} = \infty$, $\mathtt{max}_{\mathtt{BW}} = 100\%$, $\mathtt{max}_{\mathtt{Friends}} = 3$. These norms lead 2-LAMA approaches to a similar initial behaviour than BT because: $\mathtt{max}_{\mathtt{Has}} = \infty$ does not restrict communications among clusters, $\mathtt{max}_{\mathtt{BW}} = 100\%$ does not limit peer communication and $\mathtt{max}_{\mathtt{Friends}} = 3$ is equivalent to the three non-random selected peers. In our current implementation agents always fulfil these norms.[15] The 2L-SS alternative keeps these norms during all execution and just updates social structure as described in Sect. 4.2. In contrast, in norm adaptation approaches (2L-SS-N-Heu, 2L-SS-N-CBR), $\mathtt{max}_{\mathtt{BW}}$ and $\mathtt{max}_{\mathtt{Friends}}$ are updated by assistants at run-time at certain adaptation intervals ($\mathtt{adapt}_{\mathtt{interv}}$). Each assistant computes their desired values for each norm taking into account the information collected from its cluster and the information received from other assistants. Assistants use a voting scheme as a group decision mechanism to choose the actual norm updates before notifying their peers. Currently, in order to perform these adaptations, assistants aggregate their local and remote information by giving them the same importance ($\forall_i w_{R,i} = w_L$, see Sect. 4.1). With this information, 2L-SS-N-Heu approach uses the heuristic described in Sect. 4.3.1 to change norms, whereas 2L-SS-N-CBR uses the CBR approach described in Sect. 4.3.2. The latter requires some extra parameters, like the weights applied to each attribute when computing the case similarity (see Eq. 51): $w^{\Theta}_{srvCap} = 0.13$, $w^{\Theta}_{netSat} = 0.38$, $w^{\Theta}_{waiting} = 0.25$, $w^{\Theta}_{shareRatio} = 0.13$, $w^{\Theta}_{bwUsg} = 0.13$. Also, it uses 0.8 as the minimum similarity

[15] Otherwise, we could assume there is an infrastructure mechanism at ISPs that detects and filters out messages that exceed the bandwidth limit ($\mathtt{max}_{\mathtt{BW}}$) or the simultaneous data messages limit ($\mathtt{max}_{\mathtt{Friends}}$).

threshold (i.e. `MIN_SIM= 0.8`) and 1 as the maximum divergence threshold (i.e. `MAX_DIV= 1`) —see algorithms 2–3. Notice that in this approach, assistants start with an empty case base. Thus, they start calling the expert to generate an initial case. Afterwards, if a problem is similar to previous ones, they do not call the expert again, but reuse their knowledge.

The protocol they use to exchange the data is a modification of BT protocol to include our distributed assistance meta-level as explained in Sect. 3.2. In 2-LAMA, interested peers contact only a subset of sources, because their assistants promote a social structure ($SocStr_{DL}$) based on communication latencies. Then, sources can start serving at any moment, but there is a norm ($normFriends_{DL}$) that limits their maximum simultaneous servings. Moreover, interested peers can change their source if they find a faster one. In any case, communication speed is regulated by assistants through the $normBW_{DL}$.

5.3 Results

All simulations have been executed using the network topology depicted in Fig. 9. Notice that in BT alternative, a single tracker is linked to $r0$, whereas in our 2-LAMA approaches there is an assistant connected to each ISP ($r1..r3$). In any case, these elements (tracker/assistants) have an infinite bandwidth—as if they were almost located at the ISP. We have used data messages of 5,000 data units, and the rest of messages of a single data unit. Additionally, peers exchange messages of 150 data units to estimate the communication latencies between them—see "`lat_req`"/"`lat_rpl`" messages in Sect. 3.2.

We have tested all the alternatives described above by varying the peer that initially has the datum. For instance, when using the 2L-SS-N-CBR method, there is a first execution (*round*) with the data in a single initial position. In this first round, assistants ask peers to measure the communication latencies among them. Next, they give a social structure to peers based on these latencies and on who has the datum—see Sect. 4.2. Afterwards, they adapt the norms every $\texttt{adapt}_{\texttt{interv}}$ ticks—see Sect. 4.3. Once all peers have the datum, another round is performed with the data in another initial position. Notice that this example is based on 2L-SS-N-CBR, which uses learning. Hence, in subsequent rounds, assistants already know some previous cases since the case base is kept when sharing more data among the same agent community. This process is repeated until the data have been initially in all peers (*multiple-round*).

Due to the random nature of the BT—some served peers are selected haphazardly—, the results show the average of executing a multiple-round 50 times (i.e. $12 \times 50 = 600$ rounds, where the 12 corresponds to all possible initial data positions in a round, and the 50 corresponds to repeat the unique multiple-round). In contrast, a multiple-round does not need to be repeated when using 2-LAMA approach, because they do not present random issues. Hence, 2L-SS and 2L-SS-N-Heu have only been executed on a single multiple-round (i.e. 12 rounds). However, as 2L-SS-N-CBR's assistants learn at each round, the order of initial data positions influences the 2L-SS-N-CBR alternative. Thus, 2-LAMA-SS-N-CBR results show the average of executing 50 random

Table 2 Results for BitTorrent (*BT*) and 2-LAMA approaches (*2L-SS*: social structure adaptation only, *2L-SS-N-Heu*: adds norm heuristic adaptation, *2L-SS-N-CBR*: uses norm learning adaptation instead)

	Time	cNet	h	Data	cML
BT	941.23	205,344.1	3.4	11.0	–
2L-SS	849.71	345,060.2	3.2	40.1	3749.9
2L-SS-N-Heu	834.91	293,526.7	2.9	34.9	5133.3
2L-SS-N-CBR	741.53	292,357.7	3.0	32.8	4694.1

multiple-rounds (i.e. $12 \times 50 = 600$ rounds, where the 50 corresponds to different multiple-rounds with distinct order of 12 initial data positions).

Table 2 shows the average per round of the following metrics: *time* as the total time required to spread the datum among all peers; *cNet* which is the network cost consumed by all messages (each message cost is computed as its length times the number of links it traverses); *h* as the average number of links traversed by each message (hops); *data* as the total number of sent data messages; and *cML* that is the cost of all messages related with the meta-level—i.e. all messages sent to or by assistants. Notice that the *data* metric refers to all data messages that have been sent. Nevertheless, some of them may not be totally transmitted if: (i) a destination peer sends a cancel message to its source peer because it has found a better source or (ii) a source peer stops sending data to fulfil an updated $normFriends_{DL}$.

If we compare the performance of both the BT and 2-LAMA alternatives, we see that our proposals require less *time* to share the datum. This means that the time invested in communicating with our suggested meta-level is less than the time benefit of having this additional level. In contrast, the *network cost* (*cNet*) is larger in 2-LAMA. This means that, in our approaches the network is intensively used along the whole execution without achieving saturation—otherwise, the time would increase. Our proposal requires more communication because: (i) it has extra communications due to the meta-level, (ii) it sends more data messages, and (iii) it initially measures latencies to adapt $SocStr_{DL}$. Having a meta-level implies that coordination messages are exchanged among peers and assistants and also between assistants. However, the derived network overload (*cML*) is small since these control messages are very small compared with data messages. In contrast, having more data messages (*data*) consumes a significant amount of network resources. These extra data messages are created because 2-LAMA peers compare data sources by retrieving some data from them—they replace their current data source whenever they find a faster one. Thus, we expect to minimise this network consumption when dealing with more than one piece of data, since peers could compare sources depending on the pieces previously retrieved. Besides, latency measurements represent up to a 20% of the network cost increment. Notice, though, that these measures are used to improve system-wide data-paths—by providing certain neighbours to each peer. Regarding the number of links traversed by messages (*h*), our 2-LAMA approaches have more local communications—i.e. intra-clusters—than BT. This is convenient because local messages have lower latencies and costs, since they are usually performed within the same cluster.

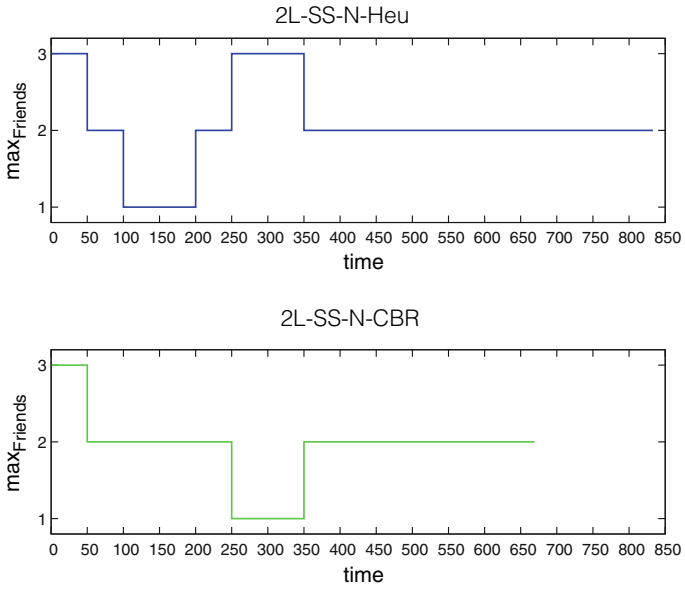

Fig. 10 Evolution of $Norm_{Friends}$'s parameter $\texttt{max}_{\texttt{Friends}}$ using the heuristic (2L-SS-N-Heu) or CBR (2L-SS-N-CBR) alternative when data was initially in P11

Finally, the results show that norm adaptation approaches (2L-SS-N-Heu, 2L-SS-N-CBR) performs better than the approach that just adapts the social structure (2L-SS), since they require less time. In fact, in our tests, no single combination of norm parameters applied during a whole execution outperforms an execution that starts by using the same parameter combination but adapts it later on. Thus, our P2P scenario has a sufficiently dynamic environment that justifies organisational adaptation. Furthermore, our learning approach (2L-SS-N-CBR) achieves better results than our heuristic approach (2L-SS-N-Heu). This means that our heuristic performs a good estimation of the mapping between system status, norms and outcomes, but it can be enhanced. In fact, our current CBR implementation is already improving this estimation. As an illustration, Fig. 10 shows the adaptation of $Norm_{Friends}$ in a single execution (when data was initially in P11) using both approaches.[16] Specifically, Fig. shows how the CBR approach (2L-SS-N-CBR) adapts $\texttt{max}_{\texttt{Friends}}$ in a different manner than the heuristic one (2L-SS-N-Heu) and obtains a shorter sharing time (669 ticks when using CBR versus 833 ticks when using the Heuristic).

6 Related work

In the literature, there exist several formalisations to model an organisation in Organisation Centred Multi-Agent Systems (OCMAS [22]). The brief ones simply define

[16] Before executing the CBR approach with the data initially in P11, the system was trained by sharing the data ten times (i.e. 10 rounds). In particular, the data was initially placed in each peer with a lower identifier (i.e. $P1 \ldots P10$ in ascendent order).

the *social structure* of participant agents, i.e. the relations among agents playing certain roles (e.g. [15]). Whereas the extended ones also define *social conventions* and organisational *goals*. For instance, EI's organisational model [21] defines the activities agents can engage on as a network of interaction emphprotocols, called Scenes. Similarly, Moise's organisational model [29] describes *protocols*—so called Contextual Specification—but also system tasks—so called Functional plan—that are related to organisational goals. Accordingly, these tasks are decomposed into sub-tasks and assigned to participants. Thus, we refer to these organisational models as *task-oriented models*.[17] Even more, social conventions' specification usually includes the definition of *norms* (e.g. EI and Moise extensions [5,23], or THOMAS [2]). Thus, our organisational model aims to be an overview of existing ones, having all mentioned components—i.e. a social structure (roles and relations), a social conventions (protocols and norms) and some goals. Moreover, in order to deal with scenarios that lack of an explicit relation among goals and tasks (i.e. *non-task-oriented*), we use a definition of goals that is not based on task descriptions but on desired system outcomes.

In addition to propose some organisational models, several OCMAS approaches—like the previously cited—provide an infrastructure to enable the execution of a specified organisation. In our Coordination Support model, we refer to this enabling infrastructure as the Organisational Layer. As an illustration, it is worth to mention Moise's extension ORA4MAS [26], in which the organisation is explicitly accessible by participants through some environmental objects (so called Artifacts). For instance, if an agent a gives the power to agent b to join a group that performs activity c, the former (a) transfers a key-artifact to the latter (b). Such a key-artifact let the latter (b) open a door-artifact that leads the agent enter a virtual space—so called Workspace—where activity c is performed. These artifacts are provided and controlled by MAS infrastructure. Thus, participant's activity depend on the interaction among agents and between agents and artifacts. In other words, an organisational specification is translated into run-time artifacts that enact the corresponding organisation in a Computational Environment [37]. Furthermore, some approaches also provide some assistance services, that we see as embryonic implementations of the Assistance Layer. For example, the Information Services in THOMAS provide information about all the organisational components to internal agents. Further, Moise's OrgManager informs participant agents about acquired obligations, such a new task to be performed. Moreover, in EI, Scene Managers inform participants about entering/leaving agents and Governors informs them when an action has been filtered out because it violated some protocol. Thus, we regard these features as an illustration of the *Information service* of the Assistance Layer. Even more, EI's information about filtered actions can be regarded as an example of our suggested *Justification service*.

Furthermore, several OCMAS approaches also define how to perform the adaptation of their organisational models. Such an adaptation can be seen as the *Adaptation service* of our Coordination Support model. On the one hand, the task-oriented models usually derive new tasks to fulfil original goals when there are environmental/population

[17] In *task-oriented models*, participant agents must accept the assignment of goals and try to achieve them. Thus, organising such systems mainly consists in assigning roles and tasks to participants depending on their capabilities (e.g. [18]).

changes. Also, they may replicate agents and/or update their social structure to improve systems' performance. For instance, in Moise there is a special role (Reorg) that has assigned the task of re-organising participant agents. Thus, it decomposes this task and assigns sub-tasks to other agents (ReorgExperts) in charge of analysing which changes are required. In particular, in [29], these agents use reinforcement learning to perform such a task. However, different techniques can be applied to these task-oriented models, such as diagnosis [27], generalised partial global planning [35] or a knowledge base of organisational structures [44].

On the other hand, there are also some works about organisational adaptation in non-task-oriented OCMAS. For example, in AEI [6], the norms of an electronic institution are adapted when certain system-wide measures differ from the expected ones—i.e. the goals. Although this approach uses CBR [1] to reason about the adaptation process, it follows a centralised scheme instead of our distributed approach. Thus, it does not deal neither with local information nor with an agreement process. In contrast, our approach goes a step further since we aim to have a distributed processing (multiple agents) and a distributed knowledge (multiple case bases)—see distributed-CBR taxonomy [39]. Above all, AEI was a basis for our proposal, and their exploration about open MAS issues inspires part of our future work—e.g. we use *social power* (see [16]) to spread norms, whereas this work uses a *sanction mechanism* that let it deal with norm violator agents. In fact, in SAEI [12] we formalised the adaptation scheme in an EI—i.e. the AEI's adaptation process—and suggested a mechanism to attach such feature to an existing MAS—so called SEI—which is the precedent of adding an assistance layer on top of MAS' domain activity. Moreover, our current work is an evolution of these previous approaches, but dealing with organisations in general—instead of just EI—and suggesting a two-layer distributed adaptation architecture.

Regarding OCMAS adaptation distributed architectures, the most of task-oriented cited approaches [27,29,35] distribute the adaptation task among a set or specialised agents. To the best of our knowledge, the task-oriented proposals closer to our two-level approach are MASPA [50] and Adaptive-MAS [25]. The former has a distributed mechanism composed by supervisor agents that have a partial view of the whole system. As our meta-level, its Multi-level Supervision Organisation has agents in charge of adapting the organisation of clusters of Workers—which is equivalent to our Org_{DL}. These agents provide Rules and Suggestions to agents in their previous layer—Suggestions are optional local conventions whereas 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. This way, their Supervision agents integrate global information into the Multi-Agent Reinforcement Learning (MARL) algorithm executed by its workers. In other words, MASPA aims to create adaptive MAS developing all its components, whereas our mid-term purpose is to deal with open MAS—where agents are developed by third parties, so there is no control over their development and corresponding behaviour. Similarly, the latter—the Adaptive-MAS [25]—differs from our approach since it assumes that the adaptation mechanism has also control over domain agents. Despite this relevant difference, our two-level architecture is similar to its proposal. In particular, it has an

Organisation-level—i.e. a meta-level—set of Monitor agents that observe and control the Micro-level's agents—i.e. Ag_{DL}. Each Monitor agent is in charge of a single domain-level agent and sends its information to a Host Monitor agent. Next, these Host Monitors act as a hub of information by building a global view of the system combining the information received from other Host Monitors—notice that our assistants are related to both monitor roles. Overall, cited distributed adaptation mechanisms use a hierarchical social structure to achieve an agreement about final adaptations. This agreement scheme can also be handled in 2-LAMA—instead of current voting scheme—by specifying a hierarchical $SocStr_{ML}$. Even more, there exist more sophisticated agreement techniques that could be used by meta-level, like the Argumentation protocols [3].

As we mentioned, since we also want to deal with non-task-oriented scenarios—where it is not possible to delegate sub-tasks—our approach use norms to influence agent behaviour. Thus, norms are an indirect tool to vary system's organisation while preserving agent's autonomy. However, the mapping between norms and system outcomes may be more complex than the mapping between tasks and goals in a task-oriented scenario. Due to this complexity, our assistants use learning in one of the norm adaptation alternatives. In fact, the cooperative MAS learning taxonomy in [38] highlights the complexity of such a task since agent interactions may bring unexpected joint behaviour—i.e. it claims that organisation/outcomes mapping is complex. On the one hand, this categorisation defines as *team learning* a centralised approach to discover a set of behaviours for a set of agents. On the other hand, it classifies as *concurrent learning* those approaches where there are multiple learners. They require that the search space can be split in disjoint parts that require disjoint actions—i.e. to decompose the problem and the solution. However, our case joins both domains, because we look for a distributed learning about an organisational level —instead of a local one.

In addition to previous organisation-centred approaches, there are several works [17, 34,40,42] that focus on the emergence and/or adaptation of norms from an agent-centred perspective (ACMAS). However, these approaches use methods that depend on participants' implementation or that assume cooperation among agents or that are driven by individual goals that may not be aligned with the social welfare.

Finally, regarding our P2P case study, some research work follow a MAS paradigm whereas others consider a network management approach. As for MAS, the work in [24] focuses on adaptation of two types of norms: *rules* (global and mandatory) and *conventions* (local and optional). *Rules* are enforced by restricting interaction, thanks to a reputation service offered by a *meta-level*. This *meta-level* also offers information about local *convention* violations, which favours the emergence of groups of agents using similar *conventions*. In their approach, agents can adapt their *conventions* but not the *rules*, *social structure* is derived from norm violations, and *meta-level*'s agents are just individual supervisors. On the contrary, our *assistants* can adapt both *social structure* and *norms* taking into account information about more than one *peer*. On the other hand, from a network management perspective, there are several works that enhance P2P systems based on a locality criteria—based on different network measures such as latencies. Some of them try to achieve it without ISPs involvement (e.g. Ono [14]) and others involve ISPs (e.g. P4P [48]). However, they usually adapt the

social structure and cannot directly vary network consumption to balance net capacity and traffic.

7 Conclusions

MultiAgent systems are composed by autonomous entities that interact in order to achieve their common and/or individual goals. The achievement of such goals normally requires the coordination of participants activities. Furthermore, since these systems are situated in dynamic environments, coordination must be adaptive in order to continue being effective under unexpected (unplanned at design time) conditions. Organisations have proven to be an effective mechanism to establish a coordination model to regulate participants behaviours. However, the adaptation of the organisation in order to continue being effective under varying conditions remains an open issue.

In this paper, we have focussed on how to endow an organisation with adaptation capabilities, and on providing means to empirically evaluate adaptation mechanisms. We regard organisational adaptation as one of a new set of services, called assistance services, which we suggest, should be incorporated in organisational frameworks [7]. The aim of these services is to assist coordination both at agent and organisational level in MAS organisations. We have proposed an abstract architecture (2-LAMA) where a distributed meta-level is in charge of providing assistance services to the domain level. Hence, we refer to the meta-level as assistance layer. Specifically, each meta-level agent (assistant) receives partial information about the system status and uses it to provide assistance to the domain level. The assistance layer may have pro-active capabilities taking the initiative and acting intelligently, which is the case in the organisational adaptation service. We have presented a formalisation of the organisational adaptation service in the proposed architecture, and how it is distributed between assistant agents. Our 2-LAMA approach can be applied to highly dynamic domains that can be designed using an organisational approach.

In order to empirically evaluate our approach and adaptation mechanisms, we have implemented a simulator in a peer-to-peer sharing network scenario. In particular, in this paper we have presented results on social structure and norm adaptation. With regard to the latter, we have compared a designed adaptation algorithm (heuristic) versus the use of Case-Based Reasoning to learn how to adapt norm values from previous experience: in other words, a comparison between defining the adaptation mechanism at design time or learning it at run time. Notice that while social adaptation is performed individually by each assistant within its cluster, in norm adaptation assistants have to reach an agreement on new norm values. Specifically, each assistant computes its new desired norm values and later must reach an agreement on each norm value. We believe that agreement technologies can play a key role in this process. Hence, our simulator can be used to test different approaches for reaching agreements among autonomous agents in the context of norm adaptation.

In future work, we plan to address open MAS issues such as how the system should react to agents joining or leaving the MAS at any point, or transgressing its

organisational restrictions. In fact, we already have preliminary results about *norm* violations that show how the system re-adapts to counter violation side effects.

References

1. Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. Artif Intell Commun 7:39–59
2. Argente E, Botti V, Carrascosa C, Giret A, Julian V, Rebollo M (2008) An abstract architecture for virtual organizations: the THOMAS project. Technical report, Grupo de Tecnología Informática-Inteligencia Artificial (GTI-IA), Universidad Politécnica de Valéncia
3. Artikis A, Kaponis D, Pitt J (2009) Dynamic Specifications of Norm-Governed Systems. In: Multi-agent systems: semantics and dynamics of organisational models. V. Dignum, IGI Global, pp 460–479
4. BitTorrentInc (2001) BitTorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html
5. Boissier O, Gâteau B (2007) Normative multi-agent organizations: modeling, support and control. In: Boella G, van der Torre L, Verhagen H (eds) Normative multi-agent systems. Dagstuhl Seminar Proceedings, vol 07122. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, pp 1–17
6. Bou E, López-Sánchez M, Rodríguez-Aguilar JA (2009) Autonomic Electronic Institutions' Self-Adaptation in Heterogeneous Agent Societies, vol 5368. Springer, New York, pp 18–35
7. Campos J, López-Sánchez M, Esteva M (2009) Assistance layer, a step forward in multi-agent systems coordination support. In: Eighth international conference on autonomous agents and multi-agent systems, pp 1301–1302
8. Campos J, López-Sánchez M, Esteva M (2009) Assistance layer in a p2p scenario. In: Engineering Societies in the Agents World X (ESAW'09). Lecture notes in artificial intelligence, vol 5881. Spinger, New York, pp 229–232
9. Campos J, López-Sánchez M, Esteva M (2009) Multi-Agent System adaptation in a peer-to-peer scenario. In: ACM SAC09-Agreement Technologies, pp 735–739
10. Campos J, López-Sánchez M, Esteva M, Morales J (2009) A simulator for a two layer MAS adaptation in P2P networks. In: WAT09-Workshop on Agreement Technologies
11. Campos J, López-Sánchez M, Esteva M, Novo A, Morales J (2009) 2-LAMA Architecture vs. BitTorrent protocol in a peer-to-peer scenario. In: Artificial Intelligence Research and Development-CCIA09, vol 202. IOS Press, pp 197–206
12. Campos J, López-Sánchez M, Rodríguez-Aguilar JA, Esteva M (2009) Formalising situatedness and adaptation in electronic institutions. In: Coordination, organizations, institutions, and norms in agent systems IV. Lecture notes in artificial intelligence (LNAI), vol 5428. Springer, New York, pp 126–139
13. Carley K (1995) Computational and mathematical organization theory: perspective and directions. Comput Math Organ Theory 1(1):39–56
14. Choffnes D, Bustamante F (2008) Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems. SIGCOMM Comput Commun Rev 38(4):363–374
15. Costa ACR, Demazeau Y (1996) Toward a formal model of multi-agent systems with dynamic organizations. In: Proceedings of the international conference on multi-agent systems. MIT Press, Kyoto
16. Savarimuthu BTR, Cranefield S (2009) A categorization of simulation works on norms. In: Normative multi-agent sysytems, Dagstuhl Seminar Proceedings, vol 09121. Leibniz, Germany, pp 39–58
17. Savarimuthu BTR, Cranefield S, Purvis M, Purvis M (2008) Role model based mechanism for norm emergence in artificial agent societies. In: LNCS—Proceedings of the coordination, organizations, institutions, and norms in agent systems III, vol 4870. Springer-Verlag, New York, pp 203–217
18. Deloach SA, Oyenan WH, Matson ET (2008) A capabilities-based model for adaptive organizations. Auton Agents Multi-Agent Syst 16(1):13–56
19. Dijkstra EW (1959) A note on two problems in connection with graphs. Numer Math 1:269–271
20. Esteva M (2003) Electronic Institutions: from specification to development. IIIA PhD, vol 19
21. Esteva M, Rodríguez-Aguilar JA, Sierra C, Garcia P, Arcos JL (2001) On the formal specifications of electronic institutions. In: Dignum F, Sierra C (eds) AgentLink. Lecture notes in computer science, vol 1991. Springer, New York, pp 126–147
22. Ferber J, Gutknecht O, Michel F (2004) From agents to organizations: an organizational view of multi-agent systems. In: Giorgini P, Müller JP, Odell J (eds) Agent-oriented software engineering IV. Springer, New York, pp 214–230

23. García-Camino A, Rodríguez-Aguilar JA, Sierra C, Vasconcelos W (2009) Constraint rule-based programming of norms for electronic institutions. Auton Agents Multi-Agent Syst 18(1):186–217
24. Grizard A, Vercouter L, Stratulat T, Muller G (2007) A peer-to-peer normative system to achieve social order. In: LNCS—Proceedings of the coordination, organizations, institutions, and norms in agent systems II, vol 4386. Springer, New York, pp 274
25. Guessoum Z, Ziane M, Faci N (2004) Monitoring and organizational-level adaptation of multi-agent systems. In: AAMAS '04: Proceedings of the third international joint conference on autonomous agents and multiagent systems, Washington, DC, USA. IEEE Computer Society, pp 514–521
26. Hübner JF, Boissier O, Kitio R, Ricci A (2009) Instrumenting multi-agent organisations with organisational artifacts and agents. Auton Agents Multi-Agent Syst, pp 1–32
27. Horling B, Benyo B, Lesser V (2001) Using self-diagnosis to adapt organizational structures. In: AGENTS '01: Proceedings of the fifth international conference on autonomous agents, New York, NY, USA. ACM, pp 529–536
28. Horling B, Lesser V (2004) A survey of multi-agent organizational paradigms. Knowl Eng Rev 19(4):281–316
29. Hübner JF, Sichman JS, Boissier O (2004) Using the $\mathcal{M}$oise+ for a cooperative framework of mas reorganisation. In: LNAI—Proceedings of the 17th Brazilian symposium on artificial intelligence (SBIA'04), vol 3171. Springer, New York, pp 506–515
30. Hübner JF, Sichman JS, Boissier O (2005) S-MOISE$^+$: a middleware for developing organised multi-agent systems. In: AAMAS workshops. LNCS, vol 3913. Springer, New York, pp 64–78
31. Jennings N, Sycara K, Wooldridge M (1998) A roadmap of agent research and development. Auton Agents Multi-Agent Syst 1(1):7–38
32. Jones J, Goel AK (2004) Revisiting the credit assignment problem. In: Challenges of Game AI: Proceedings of the AAAI, vol 4, pp 04–04
33. Kephart JO, Chess DM (2003) The vision of autonomic computing. IEEE Comput 36(1):41–50
34. Kota R, Gibbins N, Jennings N (2009) Decentralised structural adaptation in agent organisations. In: AAMAS workshop on organised adaptation in multi-agent systems, Estoril, Portugal. Springer, New York, pp 54–71
35. Lesser V, Decker K, Wagner T, Carver N, Garvey A, Horling B, Neiman D, Podorozhny R, Prasad MN, Raja A et al (2004) Evolution of the GPGP/TAEMS domain-independent coordination framework. Auton Agents Multi-Agent Syst 9(1):87–143
36. Lewis D (1969) Convention: a philosophical study. Harvard University Press, Cambridge
37. Omicini A, Ricci A, Viroli M (2008) Artifacts in the A&A meta-model for multi-agent systems. Auton Agents Multi-Agent Syst 17(3):432–456
38. Panait L, Luke S (2005) Cooperative multi-agent learning: the state of the art. Auton Agents Multi-Agent Syst 11(3):387–434
39. Plaza E, McGinty L (2006) Distributed case-based reasoning. Knowl Eng Rev 20(03):261–265
40. Pujol JM, Delgado J, Sanguesa R, Flache A (2005) The role of clustering on the emergence of efficient social conventions. In: IJCAI'05: Proceedings of the 19th international joint conference on artificial intelligence, pp 965–970
41. Riesbeck CK, Schank RC (1989) Inside case-based reasoning. Lawrence Erlbaum Associates, Hillsdale
42. Salazar-Ramirez N, Rodríguez-Aguilar JA, Arcos JL (2008) An infection-based mechanism for self-adaptation in multi-agent complex networks. In: Brueckner S, Robertson P, Bellur U (eds) 2nd IEEE international conference on self-adaptive and self-organizing systems, SASO pp 161–170
43. Di Marzo Serugendo G, Gleizes MP, Karageorgos A (2006) Self-organisation and emergence in MAS: an overview. Informatica 30:45–54
44. Sims M, Corkill D, Lesser V (2008) Automated organization design for multi-agent systems. Auton Agents Multi-Agent Syst 16(2):151–185
45. Smith BC (1982) Reflection and semantics in a procedural language. Technical Report MIT/LCS/TR-272
46. Smyth B, Cunningham P (1996) The utility problem analysed: a case-based reasoning perspective. In: European workshop on case-based reasoning, pp 392–399
47. Wooldridgey M, Ciancarini P (2001) Agent-oriented software engineering: the state of the art. In: Agent-oriented software engineering. Springer, New York, pp 55–82
48. Xie H, Yang YR, Krishnamurthy A, Liu Y, Silberschatz A (2008) P4P: provider portal for applications. ACM SIGCOMM Comput Commun Rev 38(4):351–362

49. Zhang C, Abdallah S, Lesser V (2008) MASPA: multi-agent automated supervisory policy adaptation. Technical Report 03
50. Zhang C, Abdallah S, Lesser V (2009) Integrating organizational control into multi-agent learning. In: Proceedings of The 8th international conference on Autonomous Agents and Multiagent Systems, vol 2. International Foundation for Autonomous Agents and Multiagent Systems, pp 757–764