

Proceedings of the
10th European Agent Systems Summer School
Student Session

Preface

This volume contains the papers presented at the Student Session of the *10th European Agent Systems Summer School (EASSS)* held on 7th of May 2008 at the New University of Lisbon, Portugal.

The Student Session, organised for students by students, is designed to encourage student interaction and feedback from the community. By providing the students with a conference-like setup, both in the presentation and in the review process, students have the opportunity to prepare their own submission, go through the selection process and present their work to each other and their interests to their fellow students as well as internationally leading experts in the agent field, both from the theoretical and the practical sector.

The goal of the Student Session is to provide the speakers with constructive feedback and means to be introduced to the community. Therefore, the competitive elements often found in conferences (best paper award, best presentation award) are intentionally omitted. Preparing a good paper is a difficult task, practicing it is the benefit of the Student Session.

All submissions were peer-reviewed and accepted paper submissions were assigned a 25 minute slot for presentation at the Summer School. Typically a presentation either detailed the intended approach to a problem or asked a specific question, directed at the audience.

The review process itself was extremely selective and many good papers could not be accepted for the final presentation. Each submission was reviewed by 4 programme committee members on the average, who decided to accept the 6 papers that are presented in these proceedings.

Overall, the EASSS'08 Student Session as well as the Summer School in general were a great success, which would not have been possible without the help of a number of people. We, the organisers of the Student Session, would like to thank the whole EASSS'08 committee, especially João Leite and Mehdi Dastani, for giving us valuable advice, as well as Hanno Hildmann, who introduced us into chairing the Student Session. Furthermore we like to thank the reviewers for giving feedback to the students. Finally our thanks go to Luis Antunes, who volunteered in chairing the second track of the Student Session. We had a great time, and we are looking forward to seeing all of you next year in Hungary.

May 2008

Tina Balke
Tobias Küster

Student Session Organization

Programme Chairs

Tina Balke
Tobias Küster

Local Organization

Tina Balke

Programme Committee

Luis Antunes
Andrea Addis
Suraj Ajit
Haris Aziz
Rutger Claes
Fabiano Dalpiaz
Mehdi Dastani
Marina De Vos
Jurgen Dix
Andrew Dowell
Edith Elkind
Torsten Eymann
Angela Fabregues Vinent
Berndt Farwer
Thomas French
Carlos García-Montoro
Nicola Gatti
Davide Grossi
Feng Gu
Paul Harrenstein
Anthony Hepple
Hanno Hildmann
Benjamin Hirsch
Sebastian Hudert
Wojciech Jamroga
Harjot Kaur
Rama Chandra Kota
Stefan König
Benoit Lacroix

João Leite
Guillaume Muller
Christoph Niemann
Julian Padget
Mario Paolucci
Eric Platon
Katia Potiron
Hongyang Qu
Xavier Rafael-Palou
Abdur Rakib
Guillaume Ravilly-Abadie
Jordi Sabater Mir
Peer-Olaf Siebers
Robert Siegfried
Michal Sindlar
Martin Slota
Dmytro Tykhonov
Laurent Vercouter
Dani Villatoro
Iain Wallace
Danny Weyns
Cees Witteveen
Chetan Yadati Narasimha
Pablo Lucas dos Anjos
Leon van der Torre

Table of Contents

Using Multiagent Systems for Mobility Management in Heterogeneous Network of Things	1
<i>Atiq Ahmed, Leila Merghem Boulahia, and Dominique Gaïti</i>	
Designing Open and Distributed Infrastructure for Multi-Agent Organisation	9
<i>Rosine Kitio</i>	
Adaptable Middleware for Heterogeneous Wireless Sensor Networks	17
<i>Edison Pignaton Freitas, Per Sderstam, Wagner Ourique de Morais, Carlos Eduardo Pereira, and Tony Larsson</i>	
A Cooperative Multiagent Architecture for Turkish Sign Tutors	25
<i>İlker Yıldırım</i>	
Resource Management for a Peer-to-Peer Service Oriented Computing System	31
<i>Mircea Moca</i>	
Verification of Resource Requirements of distributed rule-based Reasoners	39
<i>Abdur Rakib, Nguyen Hoang Nga, Natasha Alechina, and Brian Logan</i>	

Using Multiagent Systems for Mobility Management in Heterogeneous Network of Things

Atiq Ahmed, Leila Merghem Boulahia, Dominique Gaïti
{atiq.ahmed, Leila.merghem_boulahia, Dominique.gaiti} @utt.fr

ICD/ERA, FRE CNRS 2848,
Université de Technologie de Troyes,
12 rue Marie Curie, 10010 Troyes Cedex, France

Abstract. Today, there is a strong trend of people becoming more and more dependent on wireless networks because of that, for the past few years, in particular, mobility management and service continuity have become increasingly difficult tasks, and network applications often need to maintain efficient connectivity graphs for various purposes. A very interesting paradigm, ‘The knowledge Plane’ was proposed by Clark as a new construct to improve network management and applications. For improving handover decisions by taking into consideration Quality of Service (QoS) parameters (delay, bandwidth, packet loss, etc.), we propose to use a knowledge plane in heterogeneous network environments, which will be managed by the Multiagent system. The knowledge plane will be highly dependent upon knowledge derived from information contained in participating nodes of the network. It will be distributing the agents in the wireless nodes across the network for taking Handover decisions with the help of knowledge present in the network.

Keywords: Heterogeneous Networks, Internet of Things, Knowledge Plane, Mobility Management, Multiagent Systems

1 Introduction

In the past few years, Wireless Networks have emerged as the future of Internet changing an overall infrastructure of the network. As computing and communication devices become smaller and cheaper, the potential for “ubiquitous computing” becomes more real. People begin to imagine computers as things that are embedded into the environment, rather than placed on a desktop or carried around by them everywhere. The need is to have such an autonomy that can provide access all the time to every thing hence providing connectivity everywhere. Many new concepts have arisen from this idea, including terms such as “smart environments”, “ubiquitous computing” and “pervasive computing”. However, the underlying concept is simple: If computing and sensing units become small enough, they might be integrated in every place providing a rich flow of anytime-anywhere services and this concept also gives rise to Internet of Things (IoT) (Fig 1) [12]. Radio Frequency IDentification (RFID) is one of the candidates, so far, for the IoT, including other technologies like Wi-Family, GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunication System), ZigBee, Sensors, UPnP (Universal Plug and Play), DLNA (Digital Living Network Alliance), etc.

The current telecommunications market consists of many large networks built in a range of different technologies (for example GSM, UMTS, Wi-Family, etc.). Generally, when a customer uses a service for which network access is needed, only a single network

and one end-to-end path are used, even when many other suitable network connections are available to this user. If the access network has to be altered, for example, due to insufficient signal strength, in many cases the connection is first broken and then re-established via a new network. This wastes time and is inconvenient for the user. Hence, Mobility and continuity of service between different networks (for example Wifi to GPRS and GPRS to WiMax) is a very important issue to be dealt with. For a better service and more efficient use of the network, a switch from one network to another should be made seamlessly, while the connection is maintained at the same QoS level during the mobility.

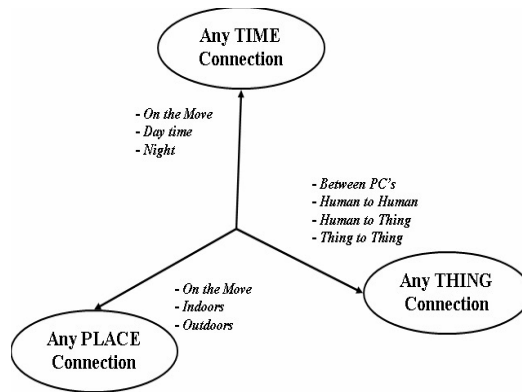


Fig 1: Internet of Things [12]

The wireless technologies currently deployed focus on providing network connectivity. Each service has its own requirements with respect to bandwidth, delay, jitter and packet loss, etc. jointly called the quality of service parameters. We need an architecture that makes it possible to set these parameters for every segment of the end-to-end network, in order to realize the service. The architecture should target mobile services that will use several of these wireless technologies like for example, video streaming starting at the home WLAN and continuing outside. In some cases, QoS parameters cannot be maintained across networks because while doing the handover between two different networks, normally, there is a possibility of packet loss and delay. As the focus is on the service and the user experience, service adaptation needs to be integrated in the network architecture and realised by the agents. We have organised this paper in the following manner: in the second section, we will present the description of our context of IoTcR (Internet of Things connected by Radio) and some of the technologies studied in this context. In section 3, some existing work related to handover techniques will be presented, which will be followed with our proposition of using a knowledge plane (KP) with Multiagent systems in section 4. And finally, we will conclude the article.

2 Internet of Things Connected by Radio

IoTcR (Internet of Things connected by Radio) can be treated as a particular case of Internet of things defined in the context of mobile and wireless networks. In this IoTcR network, the major challenge is to ensure end-to-end service continuity with a certain amount of quality of service during mobility. It must support the mobile users to be able

to pursue their applications through an interconnection of IoTcR cells. Now, we will describe our scenario, which is developed with our partners in the SUN (Situating Ubiquitous Networks) project, financed by ANR (Agence Nationale de Recherche), National French Research Agency. The goal of this project is to ensure end to end continuity of service with QoS during mobility in heterogeneous networks. Following are some of the technologies that have been studied in the context of IoTcR network.

2.1 RFID

In IoTcR network, RFID is one of the candidates. It is a method for remotely storing and retrieving data using devices called RFID *tags* or transponders. An RFID tag is a small object, such as an adhesive sticker, that can be attached to or incorporated into a product. RFID tags are composed of an antenna connected to an electronic chip. These chips transform the energy of radio-frequency queries from an RFID *reader* or transceiver to respond by sending back information they enclose. Finally, a computer hosting a specific RFID application or middleware pilots the reader and processes the data it sends [12]. RFID has very interesting characteristics like, it is possible to scan tags in motion; and the tags do not need to be in direct line of sight of the RFID reader. Having labelled or tagged objects being identifiable in an ubiquitous and flexible manner is already a good start. Building a network out of these objects, so that with a unique number one can easily retrieve information about them, would enable much more interesting use cases.

2.2 Wi-Family

Wireless technologies have evolved into a heterogeneous collection of network infrastructures providing a wide variety of options for user's access such as WiFi and cellular networks. Based on WiFi technology, Wireless Local Area Networks (WLANs) have demonstrated an exceptional success in recent years because of their high-speed data transmission and flexible deployment. WiMAX (Worldwide Interoperability for Microwave Access), is a telecommunications technology aimed at providing wireless data over long distances in a variety of ways, from point-to-point links to full mobile cellular type access. It is based on the IEEE 802.16 standard, which is also called Wireless MAN. The WiMAX forum describes WiMAX as "a standards-based technology enabling the delivery of last mile wireless broadband access as an alternative to cable and DSL"[10]. The bandwidth and reach of WiMAX make it suitable for the following potential applications:

- Connecting Wi-Fi hotspots with each other and to other parts of the Internet,
- Providing high-speed data and telecommunications services,
- Providing nomadic connectivity and so on.

Apart from Wi-family, ZigBee and UMTS are also very important types of networks present currently. ZigBee is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications. ZigBee is expected to provide low cost and low power connectivity for equipment that needs battery life as long as several months to several years but does not require data transfer rates as high as those enabled by Bluetooth. In addition, ZigBee can be implemented in mesh networks larger than is possible with Bluetooth. Universal Mobile Telecommunications System (UMTS) is one of the third-generation (3G) cell phone technologies, which is also being developed into a 4G technology. Currently, the most

common form of UMTS uses W-CDMA (Wideband Code Division Multiple Access) as the underlying air interface. It is standardized by the 3GPP (3rd Generation Partnership Project), and is the European answer to the IMT-2000 (International Mobile Telecommunications-2000) requirements for 3G cellular radio systems.

We can see that all the above described technologies are very different in nature and hence, they create a heterogeneous network of things. In this section, we have tried to explain the heterogeneity in our IoTcR Network while depicting some of the technologies. In the next section, we will present some of the related work in the mobility management.

3 Mobility In IoTcR Network

In order to support real-time or multimedia applications using end-to-end IP, it is necessary to support the requirements of mobile or broadband wireless access. Mobility management and continuity of service has been very important issues to handle for the past few years. Mobility management supports mobile terminals, allowing users to roam while simultaneously offering them incoming calls and supporting calls in progress. In the next section, we will describe some of the techniques of mobility that are present in literature.

Various Approaches for Mobility

There are many solutions in literature that have been presented for the Mobility/Handover Management. Mobile IP (MIP), the mobility enabling protocol for the Internet, introduces several new concepts in the case of handover on Layer 3. MIP enables terminals to move from one sub-network to another as packets are being sent, without interrupting this process with the help of Home agent and a Foreign agent with a care-of address [13] (Fig 2). The home agent intercepts any packets destined to the mobile node, and tunnels them to the mobile node's current location. Thus, it is necessary for a mobile node to register its location at the home agent. The time taken for this registration process combined with the time taken for a mobile node to configure a new network care-of address in the visiting network, amounts to the overall handoff latency. Thus, the handoff latency in Mobile IP is primarily due to two procedures, namely, the address resolution and the (home) network registration.

Another solution for mobility management is present, known as Session Initiation Protocol (SIP). Although SIP performs better than MIP in all aspects, including file download time, congestion window recovery time, and sequence number sent. Nevertheless, the SIP-based approach requires IP encapsulation at both ends as well as tunnelling and also handoff takes longer time to complete [13].

MIP and SIP could be used in order to support inter-domain mobility (mobility across different administrative domains), they both are unsuitable for handling intra-domain mobility. In the context of IoTcR, we are still studying mobility protocols in order to choose a protocol, which can provide end to end, fast, and flexible continuity of service.

In the next section, we propose the use of a Knowledge Plane for taking handover decisions with the help of agents.

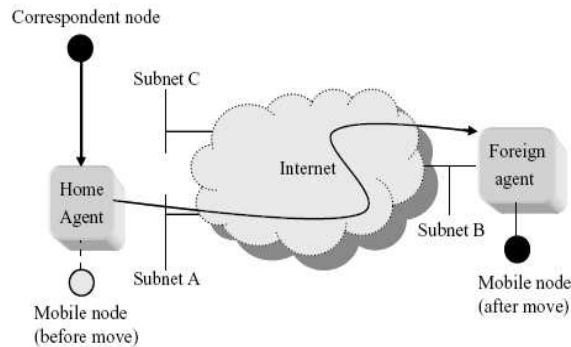


Fig 2: MIP Mobility [8]

4 A Knowledge Plane with Multiagent Systems

It has been the rule for many years to assume that communication systems could be monitored and controlled by a central entity. However, with the growth of the size and complexity of communication systems, it has become clear that this assumption is no longer valid. One of the paradigmatic cases of a communication system that has had to evolve to become autonomic and self-organized is the Internet whose initial centralized structure has had to evolve for fully distributed control and management. This trend is slowly expanding to all kinds of communications systems. It has given a way to new models and paradigms and has opened the door to migrating new research techniques from other fields for use in these systems. Among these concepts, there is a concept of knowledge plane as well, which is used for bringing the autonomy in the future networks. In the next section, we will describe the concept and need of knowledge plane.

4.2 The Knowledge Plane

A number of technologies are evolving that will formulate more adaptive and robust network architectures intended to operate in dynamic network environments. Thus, we propose to use a Knowledge plane (KP) (Fig 3) inspired by the work of Clark [4] that will be run by agents. A KP is a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network. The KP has been proposed as a distributed and decentralized construct within the internet to gather, aggregate and act upon information about network behaviour and operation [4]. We assume that knowledge is understood as a set of organized data. The main difference between information and knowledge is that knowledge is in context. While information is a composition of raw data, knowledge is a set of interrelated data.

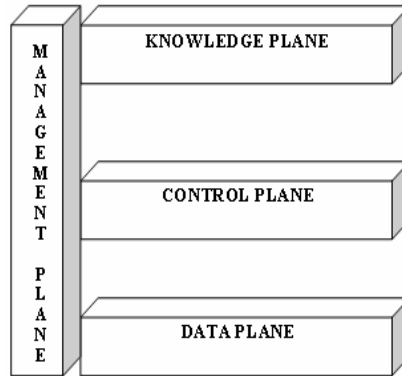


Fig 3: Knowledge Plane

The area of Multiagent Systems (MAS) is used for autonomous decision making under varying environmental conditions in the networks. An MAS is a system composed of multiple interacting intelligent agents, which are able to communicate together, possess their own resources, perceive their environment, have a behaviour and have partial representation of their environment [5]. Multiagent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve on the basis of the following characteristics.

4.3 Characteristics of MAS

We have chosen MAS because of the following characteristics of the agents [6]:

- *Decentralization*, it means that no agent has a global vision of the system and the decisions are taken in a totally decentralized way;
- *Reactivity*, an agent is a part of an environment and its decisions are based on what it perceives from his environment and on its current state. It takes a situated view of its environment;
- *Pro-activity*, it is the ability of setting goals and realizing them;
- *Sociability*, it is the ability to distribute the intelligence between the different agents and cooperating with other agents in the system. It means that some of the tasks agent can do by itself but not the entire task.
- *Adaptability*, the agent adapts its actions according to the incoming events and to its vision of the current system state.

After seeing the characteristics provided by MAS that are well suited for the development of an autonomous system, we propose the use of agents to build the knowledge plane.

4.4 Description of the Knowledge Plane

The Knowledge Plane is a building block for a network that is more reliable. In our view point, if possible, KP should be embedded within the network components (for example, access points, routers, etc.) so that it can gather useful information from usual control algorithms, for example, network connectivity, load, routing information, mobility and so on. The advantage of this is that information gathering activity can be performed

permanently, with very little impact on network performance. Then, KP should give a meaning to the collected data and should convert it into useful knowledge. This data can also be called as localised data, as it is collected with a situated view of the environment. Next function of the knowledge plane in our view is, providing/distributing the knowledge where it shall be useful, in a format that matches the needs and understanding capabilities of the recipients, which will be a very important task of the knowledge plane.

As the information will be distributed in the network, we need to put our agents correctly as well so that they can timely gather the knowledge. Agents can be placed in the correspondent nodes/ Access Points for doing handover and/or can be distributed towards the edge of the wireless network (Fig 4). Multiple wireless networks could be present in the environment. KP, built with the agents, will choose the best handover algorithm to reach the goal and it will also decide about the values of all the parameters of the algorithm chosen. The KP has to configure the control plane which itself configures the data plane. Knowledge plane will provide valuable knowledge about the Internet to mobility management and applications in a scalable and efficient way. The knowledge in the knowledge plane will include among others network topology, and performance information (fault, connectivity, latency, bandwidth, etc.).

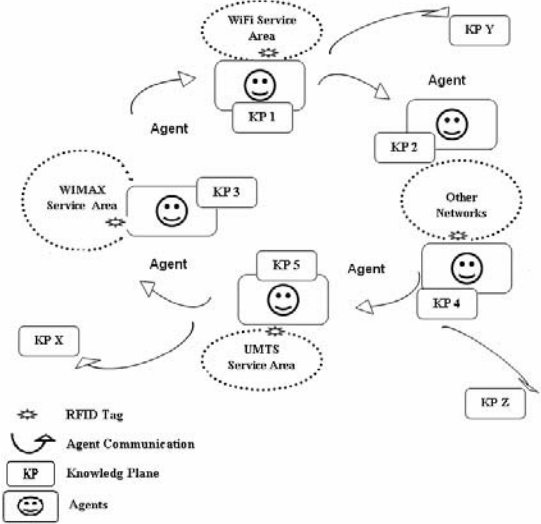


Fig 4: Agents and KP

The agents would be responsible for collecting and sharing knowledge. Apart from this, agents can send to each other the keep-alive messages, to know whether the agent in the neighbourhood is active or not. Properly implemented, the knowledge plane should continue to improve the network [3]. As we add more knowledge to the knowledge plane, it should become more valuable and useful overall.

Conclusion

Autonomic networking is a grand challenge, requiring advances in several fields of science and technology, particularly systems, software architecture and engineering, human-system interfaces, policy, modeling, optimization, and many branches of artificial intelligence such as planning, learning, knowledge representation and reasoning,

Multiagent systems, and negotiation. Handover management in a heterogeneous network environment is a big challenge because the interconnecting technologies are totally different. We have proposed to use a Knowledge plane for controlling the network decisions related to continuity of service and mobility which is just the state of art of our work done so far. The ultimate goal of the knowledge plane is to build a new generation of network that can drive its own deployment and configuration, diagnose its own problems, and make decisions to resolve them. The KP is a building block for a network that is more reliable. Properly implemented, it should continue to improve the network. As we add more knowledge to the knowledge plane, it should become more valuable and useful overall. For the future work, we are designing an architecture for the implementation of our agent approach that can support seamless mobility and could be able to take the handover decisions autonomously in the context of IoTcR.

References

1. A. Akella, G. Judd, S. Seshan, P. Steenkiste, "Self-management in chaotic wireless deployments", Proceedings of the 11th annual international conference on Mobile computing and networking, ACM publisher, pages:185-199, Cologne, Germany, Aug-Sep, 2005.
2. A. Greenberg, G. Hjalmtysson, A. Maltz, A. Myers, H. Zhang, "A clean slate 4D approach to network control and management", ACM SIGCOMM Computer Communication Review, v.35 n.5, pages:41-54, October 2005.
3. "Common Information Model By Distributed Management Task Force (DMTF)", Available: <http://www.dmtf.org/standards/cim/>
4. D. Clark, C. Partridge, J. Christopher Ramming, T. John Wroclawski, "A knowledge plane for the internet", in proceedings of ACM SIGCOMM, pages: 3 – 10, Germany, 2003.
5. D. Clark, "Content-directed aggregation and propagation in the knowledge plane", *Unpublished*, 2005.
6. D. Gaiti, G. Pujolle et al, "Autonomous Network Equipment", WAC' 05, 1-3 Athens. LNCS, Autonomic Communication, No: 3854, pages: 177-185, SpringerLink, September 2006.
7. A. Farbod, G. Liang, B. Liang, "Vertical Handoff Provisioning and Capacity Planning in the Deployment of Hybrid Networks", IEEE Biennal Symposium on communications, pages: 266-269, 2006.
8. G.D. Modica Calvagna , "A cost-based approach to vertical handover policies between WiFi and GPRS". Wireless Communications & Mobile Computing Volume 5, Issue 6, pages: 603 - 617 ISSN: 1530-8669, Wiley Interscience, September 2005.
9. H. Ballani, P. Francis, "CONMan: taking the complexity out of network management", Proceedings of the 2006 SIGCOMM workshop on Internet network management, pages: 41-46, Pisa, Italy September, 2006.
10. H. V. Madhyastha, T. Isdal, M. Piatek, et al, "IPPlane: An information plane for distributed services," in OSDI 2006, pages: 367-380, USENIX Association C.A, November 2006.
11. H. Chan, C. Chieu, "An approach to monitor application states for self-managing (autonomic) systems", Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pages: 312-313, CA, USA, October, 2003.
12. "Internet of things", International Telecommunication Reports, 2005.
13. J. Ferber, "Multi-Agent System: An Introduction to Distributed Artificial Intelligence" Addison Wesley Longman, ISBN: 0-201-36048-9, 1999.
14. "Management Information Base", IETF RFC 4220, November 2005.
15. "SUN: Situated Ubiquitous Networks", Available: <http://projet-sun.int-edu.eu/>
16. R. Rahim-Amoud, L. Merghem-Boulahia, D. Gaiti, "Autonomous Agents for Self-managed MPLS DiffServ-TE Domain", in proceedings of IFIP6, LNCS Autonomic Networking, No:4195, pages:119-131, Paris, France, 2006.
17. R. Kornelije, M. Mirko, B. Miroslav, "A Survey of the Properties of Agents", Journal of Information and Organizational Sciences (0351-1804), Vol: 30 pages: 155-170, Published by University of Aghreb, Faculty of Organization and informatics Croatia, November 2006.
18. S. Schmid, L. Eggert, M. Brunner, J. Quittek, "TurfNet: An Architecture for Dynamically Composable Networks" Proc. 1st IFIP Int. Workshop on Autonomic Communication, LNCS, No: 3457, pages: 94-114, Berlin, Oct. 2004.
19. S. Dobson, D. Gaiti, E. Gelenbe et al "A survey of autonomic ommunications", ACM Transactions on Autonomous and Adaptive Systems (TAAS), vol:1, pages: 223-259, December 2006.

Designing Open and Distributed Infrastructure for Multi-Agent Organisation

Rosine Kitio

PhD student at Multi-Agent System departement
Ecole National Superieur des Mines, Saint-Etienne
kitio@emse.fr

Abstract. In the field of multi-agent systems, the social and organisational aspects of agency become a major focus of interest. Various methodologies and infrastructures provide tools for modelling and managing organisation. However, the management of openness and large-scale organisations remains an open research issue. Particularly, considering the management of constraints and norms while keeping the agents' autonomous need to provide suitable abstractions both at conceptual and implementation step of an organisation infrastructure.

In this paper, we present an approach based on the A&A (Agents and Artifacts) meta-model. A&A promotes that agents are situated in an environment populated by artifacts that provide them some services. In the same trend, we are interested to provide an organisation infrastructure based on special kinds of artifacts called *organisational artifacts*. These artifacts manage enforcement of organisational constraints and we dedicated the organisation supervision and regulation to *organisational agents*.

1 Introduction

In recent years, organisation has become an important research field in Multi-Agent Systems [3, 1]. Many models [6, 4] have been, and are still, proposed. Most of them deal with the control of the autonomous agents so that, at the system level, coherent functioning is ensured. In [2] it claimed that the different proposals of organisation centered point of view are define with two important components may be considered: a declarative *Organisation Modelling Language* (OML) and an *Organisation Infrastructure* (OI). The former is used to define constraints, rules and the cooperation patterns of an organisation. The resulting description represent what we call an *Organisation Specification* (OS). The OI is the infrastructure where a set of agents are bind to accomplish the overall goal defined in the OS. An instantiation of an OS and its agents form an *Organisation Entity* (OE).

Our work is interested in providing tools for an OI with the following requirements: (1) Provide an open system which can enable heterogeneous agents¹ to enter and participate in an OE; (2) Enable the management of large-scale, distributed multi-agent organisations; (3) Provide a normative management in order to regulate agents compliance according to a defined organisation while keeping agents' autonomous. In this

¹ Agents heterogeneity is considered here in terms of different implementation languages and reasoning techniques

paper we present the first steps toward these objectives with our proposal ORA4MAS (Organisation Artifact and agents for open Multi-Agent Systems).

The remainder of this paper is structured as follow: section 2 presents related works and our motivations. Following, section 3 describes the basic concepts of our proposal ORA4MAS. Section 4 presents an overview of its current architecture with the mapping on the OML Moise⁺ [9]. Finally section 5 highlight our future works.

2 Related works and motivations

In the literature concerning organisation in multi-agent system, different OML are proposed to design agent cooperation patterns. Among them, AGR [6] is based on the notions of Agents, Groups and Roles. Each agent can plays several roles within different groups of an organisation. However there is no means to define the relationship between the groups and the roles. Islander [4], provides tools to define institutions structure and interactions. The designer may define the roles of the institution, their relationship (communication protocols) in different scenes and the norms assigning agents' rights and duties according to their roles. Moise⁺ [9] and Moise^{Inst} [8] propose the modelling of an organisation in different dimensions: the structural dimension is composed by the hierarchy of groups, each group with its roles, the cardinality of each role and the relations between these roles. The functional dimension is used to define the missions, the goals and the plans to be achieved in order to attain the overall goal of the organisation. The deontic or normative dimension is used to specify the norms defining obligations, permissions or interdictions related to roles and missions. The contextual dimension providing by Moise^{Inst} is used to define the different organisational state (contexts) related to norms.

The corresponding organisation infrastructures assuring the management of an OS defined with these OML are Madkit for AGR, Ameli [5] for Islander, S-Moise⁺ [9] for Moise⁺, Synai [7] for Moise^{Inst}. As illustrated in [2] they follow the general architecture depicted in Fig. 1.

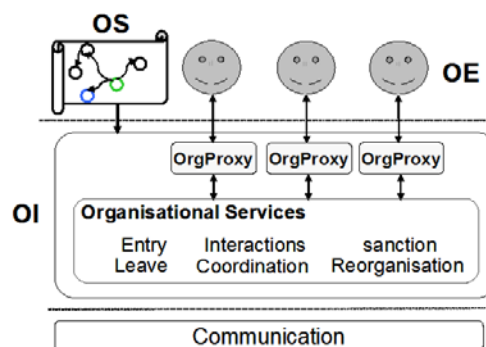


Fig. 1. General architecture of organisation infrastructure

The agents interact with the OI through a proxy (e.g. *Governor* for Ameli, *OrgBox* for S-Moise⁺, *OrgWrapperAg* for SYNAI) which provide to the agents, predefined primitives to perform any actions in their organisation. The problem with these proxies is that, they are implemented in order to take some decision independently to theirs agents. For instance, if an agent want to perform an action forbidden by a norm the execution of this action will be avoided by its proxy. Thus, the proxies take the decisions without taking in account agents' autonomous.

The different organisational services such as registering agents roles, maintaining organisation' state according to constraints defined in the OS, coordinate agents' cooperation, deciding to sanction or not an agent when it does not comply with some rules or norms are managing by special "*agents manager*". S-Moise⁺ and Madkit centralise all the organisational services in a same "agent manager": *OrgManager* for S-Moise⁺; *GroupRoleManager* for Madkit. These singles "agent manager" will be probably overloaded in case of managing large-scale organisation. Ameli and Synai try to distribute organisational services between different "agents manager": *SceneManager(s)*, *TransitionManager(s)* and *InstitutionManager(s)* for Ameli; *Struct-ManagedAg(s)*, *FunctManagerAg(s)*, *ContextManagerAg(s)*, *NormManagerAg* for Synai. Even this can be a good way for managing large scale organisation, these approaches as the centralise one meld the management of (1) the enforcement of organisational constraints and coordination functions; (2) The reasoning and decision making. Considering the fact that agents are autonomous pro-active entities, and not merely providers of services without any reasoning mechanisms, it seems that the both organisational aspects state above cannot be managed at the same abstraction level. Thus, we need to define suitable ways to manage these aspects and then providing an organisation infrastructure with the requirements highlight in Sec. 1.

Recent works done on MAS environments provide interesting mechanisms to manage some MAS dimension such as coordination and openness and to answer to some of the limitations stated above. More particularly, we will focus on the A&A (Agents and Artifacts) approach [12] in this paper. The A&A approach is built upon the notions of *agents* and *artifacts*. Artifacts are environment abstractions designed to provide some functions and services to support agents' activities. They can then be useful for an organisation designer to encapsulate mediation functionalities between the agents and their organisation.

3 ORA4MAS Approach

Designing a normative open infrastructure for large-scale multi-agent organisation need to address the following issues: (i) provide features that enable heterogeneous agents to be aware of the rules defined in the organization specification (OS); (ii) distribute the management of organisation constraints through different entities which maintains a consistent state of the OE. (iii) decentralise the supervision to different agents in order to ensure a coherent state while keeping agent autonomous.

In the following subsection, we present how we address these issues by introducing first the A&A model. Secondly we use its interesting properties to define the building blocks of our approach.

3.1 Agents and Artifacts Model

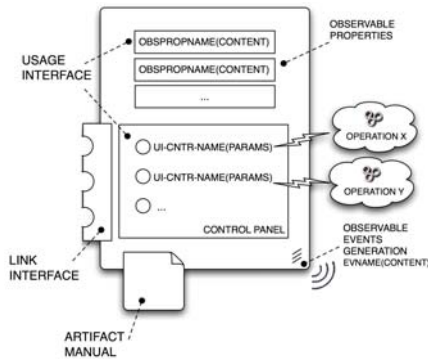


Fig. 2. Representation of an Artifact.

- an *usage interface* (UI), which possibly includes a set of operations that agents can trigger to get artifact services, and a set of *observable properties* (OP) that the agents can inspect (observe) without necessarily executing operations on it. The execution of an operation upon an artifact can result both in changing the artifact’s state, and in the generation of a stream of *observable events* that can be perceived by agents that are using or simply observing the artifact;
- a *link interface* (LI) is the set of operations provided by an artifact to another artifact to enable composed functionalities;
- a *manual*, that is composed of: (i) the *function description* which is the formal description of the purpose intended by the designer —the formal description of artifact UI and OP— (ii) the *operating instructions* providing the formal description of how to properly use the artifact so as to exploit its functionalities. Such a manual is meant to be essential for creating open systems with intelligent agents that dynamically discover and select which kind of artifacts could be useful for their work, and then can use them effectively.

Agents exploit artifacts functionality by calling provided operations and perceiving the generated events in order to get the result of their requested services. They can also be aware of artifact observable state by observing OP. This is done by mean of primitives provided by CARTAGO [11] (Common ARTifact Infrastructure for AGent Open environment) which is the platform implementing the A&A model.

3.2 ORA4MAS building blocks

Before presenting the building block of our approach, we assume that, an OS defined with an OML is provided as input of our infrastructure. We define an OML as follows:

The A&A model uses in our approach is based on the concept of working environment . populated by agents and artifacts. The *artifacts* are abstraction representing function-oriented entities. They are designed by MAS designers to encapsulate some kind of functionality, by representing (or wrapping existing) resources mediating agent activities. The agents are goal-oriented pro-active entities which can create and use artifacts services to perform their individual and social activities. Fig. 2 shows an abstract representation of an artifact as defined in the A&A meta-model, each artifact is mainly composed of:

Definition 1: an **OML**(organisation modelling language) is used to defined the specific *constraints, norms* and cooperation patterns representing an explicit *Organisation Specification* that agents may fulfill to achieve the overall goal of their organisation.

- The **constraints** are rules which define the static structure of an organisation. An agent cannot violate a constraint since the related action is executed after the verification of the constraint. We call the management of constraints the *regimentation*.
- A **norm** is a rule defining the expected behaviours of an agent in function of its role. An agent can be not compliant to a norm, however it exposes itself to possible sanction(s). The management of norms is then done in two step. First, *detection* and registering of the actions related to norms. Secondly, the *regulation* which concerns the process of decision mainly when the violations of norms occur.

From these definitions of constraints and norms, we distinguish two types of organisations management: enforcement and supervision. The *enforcement* concerns the regimentation of constraints and the detection of norms' actions. This type of management do not need any kind of reasoning mechanism. It concerns mechanisms of verification of constraints and computing/registering of norms' actions. The *supervision* concerns the monitoring of the organisation and the regulation of norms. Its need reasoning mechanisms in order to take decisions related to norms actions and the good functioning of an organisation.

To assure these managements, we defined a new type of artifact called *organisational artifact* for the enforcement. The supervision is dedicated to agents called *organisational agent*. We define the building blocks of ORA4MAS as follow:

Definition 2: **Organisational artifacts** (OrgArts) are those artifacts that agents may have to use in order to participate in organisation activities on ORA4MAS infrastructure. They encapsulate organisational constraints defined in the OS and provide organisation operations mediating agents' interactions with their organisation and assure coordination services with the other OrgArts to maintain consistent state of the OE.

Definition 3: **Organisational agents** (OrgAgent) are agents whose responsibilities concern the monitoring of organisational artifacts and reasoning about organisation dynamics in order to take appropriate decision both to sanction *member agents* which are not compliant with norms or to modify OS in order to improve achievement of organisation goals. Such activities take place almost by creating OrgArts, perceiving generated events of OrgArts operations and observing updates of OrgArts observable properties.

Definition 4: **Member agents** (MembAgent) are agents which participate to the achievement of organisation goals by interacting with OrgArts and by fulfilling constraints and norms defined in OS.

An ORA4MAS organisation entity (OE) is then composed by the sets of OrgArt_s, OrgAgent_s and MembAgent_s.

The particular set of OrgArt_s of an organisation are designed accordingly to the underlying OML. For instance, an organisational model based on the concepts of roles being played in groups, like AGR may have OrgArt_s to manage its groups. Each OML will

then have different types of OrgArts that support the management of the different considerate dimensions. The following Sec. 4 illustrates this approach by presenting with OML Moise⁺.

4 ORA4MAS with OML Moise⁺

In this section we present the shaping of our approach on Moise⁺. Since the supervision management is still a work in progress, we focus on the different OrgArts defined for this OML. These OrgArts are directly found with the dimensions considered by this OML, i.e. structural, functional and deontic dimensions.

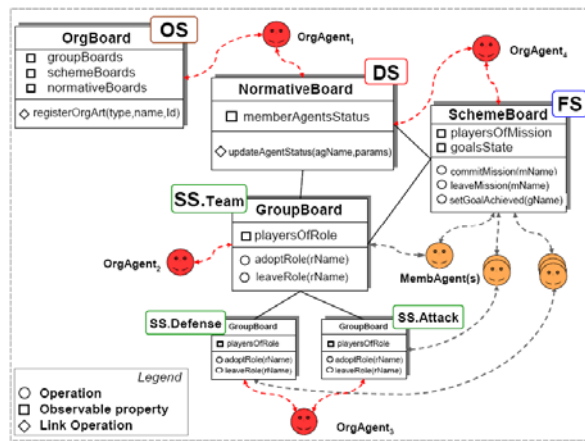


Fig. 3. ORA4MAS environment with Moise+ OrgArts.

- The **GroupBoard** is the organisational artifact which enforces and manages a group instance created from a group specification. Example: SS.Team on Fig. 4. It provides operations to agent to adopt role(s) in the group and maintains the state of the group that is accessible with its observable properties: *playableRoles*, *playersOfRole* which are respectively the set of roles which can be adopted in the group and the list of players of each role.
- The **SchemeBoard** is the organizational artifact that encapsulates and provides the operations for managing a social scheme defined in the functional dimension (FS). It maintains the state of missions, goals and plans according to constrains defined in the scheme specification. Its observable properties enable agents to be aware of the players of a missions, the possible goal(s) to achieve (cf. *goalsStatus* and *playersOfMission* Fig. 4).

- The **NormativeBoard** is the organisational artifact which manages the norms defined in the deontic dimension (DS) within an OE. It maintains information concerning the agents compliance or not according to norms. These information are accessible in the observable properties *agentsStatus*. They are used by organisational agent in their regulation task to take appropriate decisions for norms violation.
- The **OrgBoard** is an organisational artifact which encapsulates the whole organisation specification of an OE. Each OE may have only one OrgBoard. All the created instances of GroupBoard, SchemeBoard and NormativeBoard of an OE are registered in the OrgBoard.

The coordination between the different instances of the OrgArt of an OE are managed through link operations. Example, due to the composition link between groups and subgroups, a GroupBoard coordinates itself with the sibling descendant or ascendant GroupBoards. For instance a soccer game team is composed with a sub group attack and a sub group defence. In a same vein, GroupBoard and SchemeBoard are linked in order to provide to each other information for the management of the consistent and coherent state of an OE. The GroupBoard and SchemeBoard are respectively linked to a NormativeBoard in order to update information of agent compliance with the norms of the OE. The OrgArts and their links of an ORA4MAS organisation entity of a soccer team are illustrated on Fig. 4. The OrgAgents attached to each OrgArt are just for illustrate the supervision of the OrgArts without any specific repartition.

Because of lack of space we cannot detail the features of each type of OrgArt. The readers are invited to read paper [10] for more details.

5 Conclusion and Perspectives

In this paper, we have followed the A&A approach to propose on the one hand the organisational artifacts which encapsulate the functional and coordination aspects of an organisation, and on the other hand the organisational agents, which are responsible for the decision and reasoning which concern the management of the supervision and regulation in an organisations. We have described a reification of ORA4MAS on Moise⁺ organisational model resulting in a concrete architecture implemented on top of CARTAGO [11].

One benefits of this proposal consists in providing an open system for heterogeneous agents. More precisely, we expect to formally describe the manual of each type of organisational artifact that the agents will use with their own implementation language will to fulfil the requirements for they actions within the ORA4MAS infrastructure. In another side, due to possibility to dynamically create instances of organisational artifacts, link them together and observe their state, they appear as an interesting way to distribute the management of a large scale organisation and decentralise the supervision of an organisation entity.

However the work presented here is still in progress. We intend to instantiate this approach with different OML such as institutional one (e.g Islander [4] or Moise^{Inst} [8]). We then expect to improve features of the different types of OrgArts and provide a generic OI running with different OML. Another point of investigation is the definition

of the supervision process concerning organisational agents responsibilities. The evaluation of this work will consist to run our infrastructure with a relevant application in order to experiment the efficiency of our approach to manage large scale organisation with the agents of different implementation languages.

References

1. G. Boella, L. W. N. van der Torre, and H. Verhagen. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory*, 12(2-3):71–79, 2006.
2. O. Boissier, J. F. Hübner, and J. S. Sichman. Organization oriented programming from closed to open organizations. In G. O’Hare, M. O’Grady, O. Dikenelli, and A. Ricci, editors, *Engineering Societies in the Agents World VII*, volume 4457 of *LNCS*. Springer-Verlag, 2007.
3. O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vázquez-Salceda, editors. *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *LNAI*. Springer Verlag, 2006. AAMAS 2005 International Workshops on Agents, Norms, and Institutions for Regulated Multiagent Systems, ANIREM 2005 and on Organizations in Multi-Agent Systems, OOOOP’06.
4. M. Esteva, D. De La Cruz, and C. Sierra. Islander: an electronic institutions editor. In *AAMAS 02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1045–1052, New York, NY, USA, 2002. ACM Press.
5. M. Esteva, J. A. Rodríguez-Aguilar, B. Rosell, and J. L. AMELI: An agent-based middleware for electronic institutions. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 236–243, New York, 2004. ACM.
6. J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations : an organizational view of multiagent systems. In *LNCS*, editor, *AOSE 03: 4th Agent-Oriented Software Engineering*, volume 2935 of *LNCS*, 2003.
7. B. Gâteau. *Modélisation et Supervision d’Institution Multi-Agents*. PhD thesis, Ecole Nationale Supérieure des Mines, Saint Etienne, 2007.
8. B. Gâteau, O. Boissier, D. Khadraoui, and E. Dubois. Moiseinst: An organizational model for specifying rights and duties of autonomous agents. In *Third European Workshop on Multi-Agent Systems (EUMAS 2005)*, Brussels Belgium, December 7-8 2005.
9. J. F. Hübner, J. S. Sichman, and O. Boissier. *S-MOISE⁺*: A middleware for developing organised multi-agent systems. In O. Boissier, V. Dignum, E. Matson, and J. S. Sichman, editors, *Proceedings of the International Workshop on Organizations in Multi-Agent Systems, from Organizations to Organization Oriented Programming in MAS (OOOP’2005)*, volume 3913 of *LNCS*. Springer, 2006.
10. R. Kitio, O. Boissier, J. F. Hübner, and A. Ricci. Organisational artifacts and agents for open multi-agent organisations: “giving the power back to the agents”. In J. Sichman, P. Noriega, J. Padget, and S. Ossowski, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *LNCS*, pages 171–186. Springer, 2008. Revised Selected Papers.
11. A. Ricci, M. Viroli, and A. Omicini. *CARTAgO*: A framework for prototyping artifact-based environments in MAS. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*. Springer, 2006. 3rd International Workshop (E4MAS 2006).
12. A. Ricci, M. Viroli, and A. Omicini. “Give agents their artifacts”: The A&A approach for engineering working environments in MAS. In E. Durfee, M. Yokoo, M. Huhns, and O. Shehory, editors, *6th International Joint Conference “Autonomous Agents & Multi-Agent Systems” (AAMAS 2007)*. IFAAMAS, 2007.

Adaptable Middleware for Heterogeneous Wireless Sensor Networks

Edison Pignaton Freitas^{1,2}, Per Söderstam¹, Wagner Ourique de Morais¹, Carlos Eduardo Pereira^{2,3}, and Tony Larsson¹

¹ School of Information Science, Computer and Electrical Engineering, Halmstad University, Halmstad, Sweden

² Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil

³ Dep. Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Brazil
{edison.pignaton, per.soderstam, wagner.demorais, tony.larsson}@hh.se,
cpereira@ece.ufrgs.br

Abstract. The use of sensor networks in different kinds of sophisticated applications is emerging due to several advances in sensor technologies and embedded systems. However, the integration and coordination of heterogeneous sensors is still a challenge, especially when the target application environment is susceptible to changes that the system must track and adapt itself to in order to fulfil the users' requirements. These changing scenarios require services being provided in different places during the system runtime, and to fulfil this, a support for adaptability is needed. In this paper we present some initial ideas to use multi-agents in a middleware that aims to provide the necessary support to sophisticated sensor network applications.

Keywords: Agent-based Adaptable Middleware, Wireless Sensor Networks, Heterogeneous Sensor Networks.

1 Introduction

Sensor network applications are becoming more useful with the possibility to use different kinds of mobile sensors to provide more sophisticated functionalities [1] and be deployed also in complex scenarios, like where context-awareness is needed [2]. However, to support those emerging applications, an underlying infrastructure is necessary, and the current proposal is the use of middleware, such as TinyDB [3] and COUGAR [4]. The main drawbacks of these state-of-the-art middleware's that make them not very suitable for future envisaged applications, are the assumptions that the network is composed only by a homogeneous set of basic or meagre sensors, and thus a lack of the intelligence in the network to provide adaptability required to face changing operation conditions. Adaptability is a major concern that needs to be mainly supported for two reasons. The first is that long deployment time of wireless sensor networks may require flexibility in order to make changes according to the user requirements that may probably change within the usage life time of the network. The second is the fact that wireless sensor networks are being deployed in highly

dynamic environments, implying that applications have to be flexible enough in order to continue being useful in changing scenarios.

This paper presents a work in progress related to the development of an adaptive middleware to support sophisticated sensor network applications that must change their behaviour according to influences from the environment and the application demands. The idea is to use a multi-agent approach to help in the following issues: adaptation by code migration and decision planning.

The remaining of the text is organized as follows: Section 2 presents the overview of the proposed middleware. Section 3 describes how the use of agents will help in providing middleware adaptation. Section 4 presents some related works and finally, section 5 ends the paper with some concluding remarks and future works directions.

2 Overview of the Proposed Approach

The general idea is to develop a flexible middleware that can be used to support applications in heterogeneous sensor networks. By heterogeneity we mean that nodes in the network may have different sensing capabilities, computation power, and communication abilities and running on different hardware and operating system platforms. The goal is that this middleware fits both meagre and rich sensors. Meagre sensors are those with limited resources capabilities, such as piezoelectric resistive tilt sensors, with limited processing support and communication capability. Rich sensors comprehend powerful devices like radar, cameras or infrared sensors that are supported by moderate to high computing and communication resources. In order to achieve this goal, it must thus be lightweight, while being scalable in order to provide the demands of more sophisticated sensors. The proposed middleware might handle a node's resource usage, in order to assist in distributing tasks among different nodes that are capable to accomplish them. Another feature that the middleware may provide is the quality of the data required to reply a certain user's demand. Better results can be achieved by choosing the correct set of sensors to perform the measurements and collect data. These sensors can be static or moving, on the ground or flying over the target area in which the observed phenomenon is occurring. The mobility characteristic is also related to the heterogeneity that the middleware will address.

The input to the sensor network system, coordinated by the proposed middleware, will be seen as a "mission" that the whole network has to accomplish. In order to allow that, a high-level Mission Description Language (MDL) is being formulated based on the C/ATLAS test language [5]. This language will allow the user to specify - at a high level of abstraction - the data in which he/she is interested, including constraints regarding timing and location limits, as well as the measurement rate or accuracy desired. The proposed language will also allow hierarchical description of the mission goals, with establishment of priorities and other refined details, for instance, an application of comparison metrics to evaluate the how well the mission is being accomplished.

Given that the middleware must perform its actions also in very dynamic and changing scenarios, one has to take into account that a set of sensors chosen in the

beginning of a mission may not be the most adequate during for the whole mission. As an example, an area surveillance system receives the mission to survey an area that may not allow traffic of certain kinds of vehicles. Ground sensors are set to alarm in the presence of undesired vehicles, and unmanned aerial vehicles equipped with visible-light cameras are set to fly to the area where a ground sensor has issued an alarm to verify the occurrence. However, a sudden change in the weather, for instance becoming cloudy and foggy, turns the employment of a visible-light camera useless. The adaptation to this type of change in operational conditions will be supported by the middleware in order to choose a better alternative, among a set of options, for instance by choosing an infrared camera instead.

The middleware is divided in three parts or layers indicating that they are partly using each other in a specific order. Figure 1 presents the overview of the layers of the proposed middleware, and a description of each layer is provided in the following.

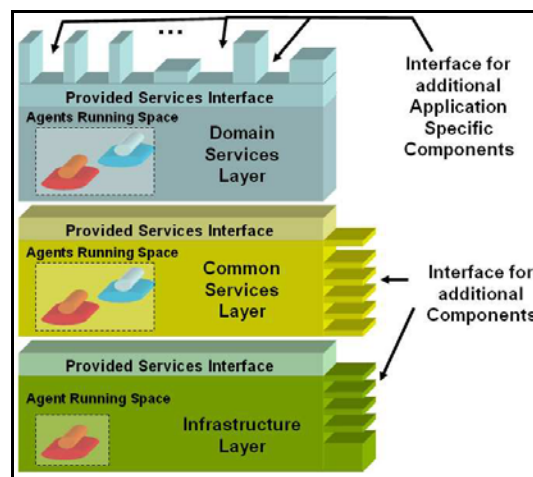


Fig. 1. Overview of the Middleware Layers.

The bottom is called Infrastructure Layer. This layer is responsible for the interaction with the underlying operating system and for the management of the sensor node resources, like available communication capacities, remaining energy, and sensing capabilities. This layer also coordinates the resource sharing based on application needs passed through the upper layers.

The intermediate layer is called Common Services Layer. This layer provides services that are common to different kinds of applications, such as QoS negotiation and control, and quality of data assurance.

The top layer is called Domain-Services Layer and has the goal to support problem-domain specific needs, such as data fusion support and specific data semantic support.

Multiple applications can run concurrently in the network. The middleware handles resource sharing and provides data sharing among applications that need the same type of data, allowing a better energy use in resource constrained nodes. In powerful

nodes, the middleware can provide more complex services aimed to handle rich data, like those related to image processing, and pattern matching. This also means that such nodes can take some of the burden from more resource constrained nodes.

3 Agent-Based Adaption

As stated before, the use of agents in the proposed middleware should help in the task of providing adaptive behaviour. To achieve this goal these agents are divided in two classes. The first is responsible for adaptation features in the services provided by the middleware, using code migration and updating. The second handles reasoning involved in the decision planning concerning the entire network. In the following these two ideas are explained, as well as an example of adaptation is provided.

3.1 Adaptation by Code Migration

As the network are composed by heterogeneous sensor nodes, those nodes that have constraints about their energy consumption, memory space or processing power, should run a minimum number of tasks as possible. However, as they are supposed to be used in dynamic environments, with changing conditions, requiring different kinds of handling, different task set allocations must be used in order to fulfil the actual needs in a certain time interval. We propose to use the idea of multiple mobile agents to provide services that can be used by the node if the agent is allocated in that node. We call them service-agents. This technique is not new; there are related works like Agilla [6] that use the same approach. However, our approach uses also multi-agents to decide which service-agent is needed in each node of the network at a certain time. We also consider other technologies to help in the middleware adaptation, like aspect-orientation and component-based design, although not discussed further in this paper. We refer to [7] for more details about these two technologies.

The service-agents can be allocated in the Common Services Layer or in the Domain Services layer, according to the type of the services they provide. Several service-agents can run in a single node, depending on the computation and memory limitations of the hardware platform. Meagre platform nodes usually need to allocate simpler service-agents, but more powerful nodes can host more sophisticated service-agents providing more complex services.

Service-agents can migrate from one node to another, clone and move to another node, or simply be unallocated, leaving space for another service-agent. The simple migration is used when its services are needed in other locations, and thus no more in the present one. The clone of a service-agent occurs when its services also are needed in another node, and it is unallocated when there is no node that needs its services. This flexibility of service-agents allows the desired adaptation of the network face changes in the environment that require different runtime services availability.

3.2 Adaptation Planning

We propose the use of a multi-agent approach to distribute intelligence over the network in order to provide a distributed way to decide several key issues about the network setup and configuration during its runtime. As our target applications are heterogeneous wireless sensor networks deployed in changing environments, and so, demanding system adaptations, we need a way to reason about the necessary changes and how to implement them.

The basic idea is that each node attached to the network receives a set of tasks (*Node-Missions*) to perform, and a type of agent, called planning-agent that is in charge of help the node in the accomplishment of its node-missions. A node-mission characterizes the node contribution to the whole system mission achievement, which is called *Global-Mission*. The node's internal tasks are called *Node-Tasks*, and they represent the finer-grained tasks that must be performed by a node to accomplish its node-missions. Node-tasks are related to the individual nodes concerns, for example power consumption handling and sensor capabilities, as well as the management of the services provided by the node.

The planning-agent is responsible for monitoring information about the current node's and assigned node-tasks state during the system runtime, including available resources, e.g. communication bandwidth and energy. As some condition changes in the environment, like when the weather changes or if one kind of measurement is no more possible. For instance, the planning agent has to reason about the node-mission assigned to its node's, and the service-agent(s) that perform the necessary services that the node has to provide. They also have to take in account the available resources to accomplish the node-mission. The planning-agents in all nodes perform this reasoning and, by a consensus, agree in a new distribution of the service-agents in order to accomplish the global-mission assigned to the network or in the employment of a different set of sensors due to a certain change in the network or environment conditions.

3.2 Example of Adaptation

As we stated before, the intention in using both planning-agents and service-agents is to help in the adaption of the system, face changes in the operational conditions due to environments or user requirements changes. In Figure 2, an example of a change in operation conditions followed by an adaptation is provided.

In the scenario presented in Figure 2, the network receives a mission, which is partitioned in four sub-missions, one for each sensor node. Service-agents are distributed around the sensor nodes according to the initial conditions to accomplish their sub-missions. In the left part of the Figure 2, t_1 represents the initial configuration established for sensor node 1 and 2 to accomplish their respective sub-missions. However, a change in operating conditions occurs, due to environment or user requirements change. The initial configuration does not meet anymore the system needs to accomplish the mission. So, planning-agents allocated in the nodes exchange information and agree in a new configuration, what occurs in time t_2 , in the middle of

the Figure 2. After the agreement, a service-agent from node 1 is migrated to node 2 and one service-agent from node 2 is unallocated, as shown in t_3 .

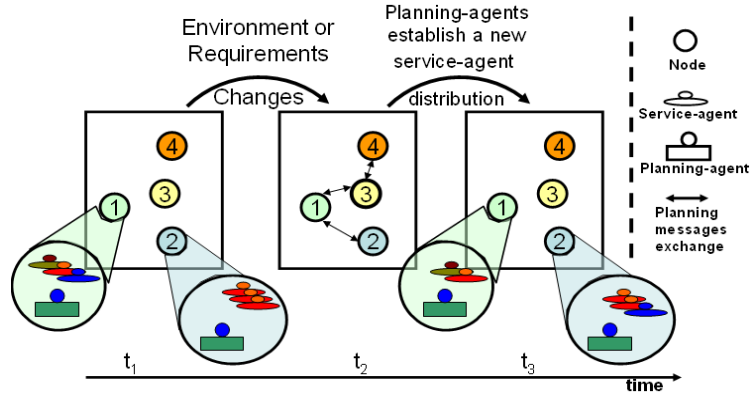


Fig. 2. Changing Scenario and Adaptation Example

4 Related Work

Code replication and migration are ideas that have inspired parts of the present proposal. Some state-of-the-art middleware for sensor networks uses this kind of ideas, as discussed in the following.

Impala [8] is a middleware used in applications dedicated to the study of wildlife. It has some strength in relation to adaptabilities, update and fault-tolerance mostly due to its event-based programming with highly modularized code. The drawbacks of this approach are that it has no support for data fusion and that the domain application is rather simplistic. Comparing with our proposal, we use multi-agents not only to move functionalities within the system, but also to provide intelligent behaviour and reasoning to support mission and environment adaptations.

Agilla [6] is the first mobile agent middleware of WSN implemented in TinyOS [9]. This approach uses agents that can move from one node to another, and it also allows multiple agents to run in the same node. These characteristics provide the desired features of energy saving, as the agents can run near to the data avoiding unnecessary communication, and it allows multiple applications to run concurrently in the network. The major drawbacks are related to the lack of an authentication policy to monitor agents' activities, and the difficult maintainability due to its programming model. We have some ideas in our work that go close to the ones presented in Agilla. However, the same comments made regarding Impala fits also in a comparison with Agilla. Another drawback is that Agilla runs only over TinyOS, and we propose not to be restricted to one operating system. Agilla use the mobile agents only to run application-specific tasks, while in our approach, a broader idea is proposed, using the agents to provide services that can be more specific to a certain domain application

(being hosted in the Domain Services Layer), or more general, supporting different kind of applications (agents hosted in the Common Services Layer).

Maté [10] consists of a virtual machine which runs atop of TinyOS hiding asynchrony and race conditions. Its strength is represented by the division of the program into small self-replicating capsules that are self-forwarding and self-propagating. The major drawbacks are that it has high energy consumption and that its programming model is not flexible enough to support a wide range of applications; an update of a capsule can not be done only in a single node, it is spread over the whole network. On the other hand, we propose to support the flexibility to run a broader range of applications, adapting, updating or modifying only specific nodes in the network, and like this also save energy, as only the nodes that need a certain adaptation are reached. This feature also focuses the support for heterogeneity.

5 Concluding Remarks and Future Work

This paper reports about an ongoing project that is in its initial stage. The study of literature in the area of wireless sensor networks indicated that there is a need for research in the area of adaptable middleware for heterogeneous sensor networks [1] [11]. Wireless Sensor Networks - composed both by small and simple resource constrained sensors as well as more sophisticated sensor nodes - need an approach that find the best trade-off to handle the different capabilities in order to provide meaningful information based on the gathered data.

An ongoing study is being performed in order to find the best suited partitioning of services to be hosted in each of the layers of the proposed middleware. In relation to that, the group is also discussing the best possible distribution of functionalities among components and services provided by agents. It is an important issue as we intend to address heterogeneous nodes with different capabilities that can accommodate different sets of services. Another important topic under discussion is the formal definition of the MDL (Mission Description Language). Some directions are under analysis in which a promise one is the use of scripts, such as those used in SensorWare [12], but with a higher abstraction level approach.

However, as this work is in its initial phase, we did not discuss about underlying details like how to achieve consensus among the planning-agents yet. It is certainly a major concern that we have to address during the development of the planning strategy and the methods for system coordination. Related works in this area, such as [13], are being analysed. Besides that, we are aware about the energy issues related to this concern. The message exchange in order to achieve the consensus among planning-agents must be minimal. Another important topic that we are analysing is how to implement the agents running space and the coordination among these spaces. Two ideas considered are the use of tuple space, as used by Agilla and a virtual machine like proposed in Maté [10]. The advantages and drawbacks of each are being studied as well as the search for other alternatives.

References

1. D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer*, vol. 37, no. 8, pp. 41–49, 2004.
2. K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom), pages 77–86. IEEE Computer Society, March 2004.
3. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.
4. P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards sensor database systems. In 2nd International Conference on Mobile Data Management (MDM), volume 1987 of Lecture Notes, 2001.
5. IEEE Std 716-1995, 1995. IEEE standard test language for all systems-Common/Abbreviated Test Language for All Systems (C/ATLAS), IEEE, Inc.
6. C.-L. Fok, G.-C. Roman, and C. Lu, "Rapid development and flexible deployment of adaptive wireless sensor network applications," in Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05), 2005.
7. A. Tesanovic, et al. "Aspects and Components in Real-Time System Development: Towards Reconfigurable and Reusable Software", *Journal of Embedded Computing*, IOS Press, v.1, n.1, 2005.
8. T. Liu and M. Martonosi, "Impala: A middleware system for managing autonomic, parallel sensor systems," in ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2003.
9. TinyOS website. [Online]. Available: <http://webs.cs.berkeley.edu/tos/>
10. P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," in International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, Oct. 2002. [Online]. Available: citeseer.ist.psu.edu/levis02mate.html
11. K. Romer, O. Kasten, and F. Mattern, "Middleware challenges for wireless sensor networks," *ACM SIGMOBILE Mobile Communication and Communications Review*, vol. 6, no. 2, 2002.
12. A. Boulis, C.-C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 187–200, New York, NY, USA, 2003. ACM Press.
13. Y. Elmaliach, N. Agmon and G. Kaminka, "Multi-Robot Area Patrol under Frequency Constraints", in *Proc. of 2007 IEEE International Conference on Robotics and Automation*, IEEE Computer Society, 2007, pp. 385-390.

Acknowledgments

We would like to acknowledge Kristoffer Lidström and Andreas Persson for the value contributions in discussions about this paper.

A Cooperative Multiagent Architecture for Turkish Sign Tutors ^{*}

İlker Yıldırım

Department of Computer Engineering
Boğaziçi University
Bebek, 34342, Istanbul, Turkey
ilker.yildirim@boun.edu.tr

1 Introduction

A sign is a combination of hand movements, facial expressions, and head movements. A sign with only hand movements is called manual signs, whereas signs which also include head movements or facial expressions are called non-manual signs. The aim of the *Sign Language Tutoring Tool* is to help users learn isolated signs by watching recorded videos and enable them try the same signs [1]. The system records a user's video, while she is performing a sign. After analyses of user's sign performance, the system gives the user feedback both verbal and animated. The system can recognize not only manual signs, but also non-manual, complex signs, that involve both hand movements and head movements and face expressions. The system uses a classifier for recognition, by which it can assess some similarity value with a level of certainty for user's sign performance. The Turkish Sign Language Tutoring Tool is specialized for Turkish Sign Language.

The current system is a stand-alone application. However, in reality the same application will need to be run in different locations with different data. This obviously calls for a distributed architecture. A possible architecture would be to have a collection of stand-alone applications in which there is no interaction between agents. In such a system although agents may improve their classification capabilities due to learning and experience, they will not be able pass their experiences to others and they will be not be able benefit from experiences of others.

Another possible architecture would be a centralized one, in which agents are geographically separated, but they share a common database and a common classifier. But in reality we know that agents may not be online all the time, hence they cannot access to the database and the classifier. In addition to that, videos may belong to a certain individual who do not want to share it.

Taking all these into account, we propose a cooperative multiagent system of Sign Language Tutoring Tools, which consist of many agents distributed geographically [2]. Each agent represents an agent that runs a sign language tutor.

^{*} The author is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant 107E021.

Agents are connected via Internet. Each agent is associated a local database and a classifier. Agents can improve its classification performance due to its own experience. An agent may decide to include a practice sign in its training data or a sign language teacher may add a new training data. Moreover, agents can help each other classify signs by exchanging classification requests. Since agents have their own local databases they can make a decision even they are offline.

2 Possible Challenges with the Architecture

In this section we state several example cases in order to better show different challenges. We will use Figure 1 as an example setup. In this figure we have four agents. Each agent can talk with each other. Agent 1 (the agent at station 1) is asked to make a decision on user's sign performing video, i.e. the sign of "school". That is Agent 1 will either decide that the user-performed sign is "school" or not. In order to come up with a better decision Agent 1 passes the user's video of sign school to other agents and requests them to state their ideas about the user's video. The following examples depict interesting situations that require systematic decision making.

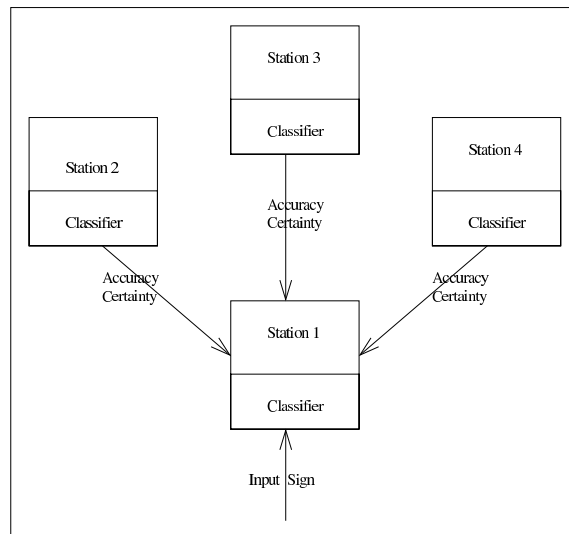


Fig. 1. A Setup for the Distributed Sign Language Recognition System

Example 1 Each agent can decide if a sign is performed *right* or *wrong*. However agents may not be confident on their decisions, hence they can request others to provide opinions (i.e. in form of votes). In Figure 1 Agent 1 receives votes for

”right” from Agents 2 and 4 which means they find it similar enough to actual sign of school, whereas it receives a ”wrong” vote from Agent 3 which means it does not find user’s performance similar enough. In addition to ideas of other agents Agent 1 also has its vote for ”right”.

Example 2 The previous case assumes all agents equally knowledgeable. Actually, there are two important criteria for decision: *Accuracy*: How similar is the performance of the user to the real sign, *Certainty*: How certain about that accuracy. In Figure 1, Agent 2 says the user’s video is the sign of school with a high accuracy but with a low certainty. Agent 3’s response is, similar to Agent 2’s, ”The performed sign is very similar to sign of school but I am not sure”. Apart from those Agent 4 says ”I am sure that the user’s performance is not the sign of school”. In addition to the ideas of other agents, Agent 1 has no idea on the user’s video, meaning its certainty level is very low or zero (But it has many ideas to run a weighted majority voting on it).

Example 3 Before sending user’s video, Agent 1 sends the word whose sign is performed to each agent. After each agent receives the word, they have vote to elect the one from whom to request a decision for the user’s performance. Hence, after all agents agree on the agent which will make the decision, Agent 1 sends the user’s sign performance to that agent.

Example 4 It is very likely that we have high number of agents. In such a case, it is possible that most agents have low certainty. Hence the aggregation of their votes may influence the system in a wrong way. But instead, each agent can model others based on expertise and direct queries accordingly. Let there be several other agents in Figure 1 and let Agent 1 be requesting votes of other agents on the sign of ”theater”. Since agent 1 thinks that an agent knows very well signs about art and literature, it directs query accordingly, and not request from every other agent in the system.

3 Parameters of Decision Making

There are several parameters that affect an agent while it evaluates an idea of accuracy with some level of certainty about a user video:

- How well my classifier is trained for this particular sign?
- Who entered my training data, a deaf person, a sign language teacher or a sign language student?
- When my training data was entered? For instance the resolution of my training videos may be low, because they are old.
- What is the amount of data used to train my classifier?

In Example 1 each agent has an idea about whether the user’s video is the sign of school, meaning right or the user’s video is not the sign of school, meaning wrong. Consider we apply a plurality voting protocol which means the candidate with highest number of votes wins [3]. Our candidates are *the user’s sign*

performance video is right or *the user's sign performance video is wrong*. Each agent votes by responding to Agent 1. In Example 1 *the user's sign performance video is right* receives 3 votes whereas *the user's sign performance video is wrong* receives only 1 vote. Hence Agent 1 concludes that the user performed the sign of school right.

In Example 2 each agent responds how similar the sign of school and the user's video are similar (its accuracy), and how certain it is about that accuracy (its certainty). In Example 2 although two of agents argue that the user's video has high accuracy, Agent 1 will decide that the user's video is wrong, because Agent 4 is certain that the user's video is not the sign of school. In order to be able come up with such a conclusion, Agent 1 considers accuracy values of other agents using their certainty values.

Before considering Examples 3 and 4, an agent is required to have a model of other agents in order to enhance its decision making process. One way to make agents construct models of other agents is to give feedback about its decisions up to now. For instance Agent 1 decided that the subject performs the corresponding sign of the word "school" correctly relying on the certainty of Agent 3. After sometime, a teacher manually checks decisions of Agent 1 and recognizes that while the subject could not perform the sign of the word "school" correctly, somehow, Agent 1 accepted it. The teacher inputs that feedback to the agent, and immediately Agent 1 changes its beliefs about Agent 3 in some sense negative direction. The reverse can also happen. Models can have several dimensions, i.e. expertise on an ontological subject, how much to trust to certainty value, how much to trust to accuracy.

In Example 3, agents elect one of them to make the decision. It is equal to saying we have number of agents many candidates and number of agents many voters. We can apply any voting mechanism to elect the decision maker. Let's consider *Single Transferable Vote (STV)* [3]. In STV the winner is determined after rounds. In each round each candidate receives points as the number of votes in which it ranks highest. In each round the candidate with the lowest score is eliminated. The last remaining candidate is the winner. The name comes from that the votes are transferred from most popular to the next most popular candidate in terms of voters of the eliminated candidate. After electing the agent to make decision, Agent 1 can get this decision and reply to the subject.

In Example 4 each agent models other agents in the system. An agent, when it is requested by the user to recognize a performance, requests a subset of other agents for help. The agent decides which subset of agents to ask using its models. Since agents improve via experience, these models need to be dynamic. So we need methods for updating models of others. Since there are a lot of agents, each agent may not be able to model each other. If an agent does not know whom to ask exactly, it can ask to another agent that she thinks that may know. The queried agent may either reply an answer or *refer* to another agent.

In Figure 2, our generic multiagent protocol, an agent can request classification of a sign performance. And an agent may respond with as *correct* or *wrong*, *certainty and accuracy tuple*, or a *referral*.

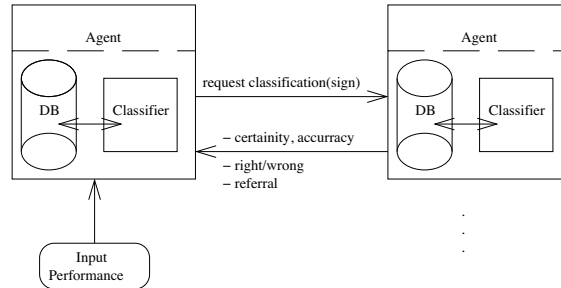


Fig. 2. Generic Multiagent Protocol

4 Discussion

In this study, we develop a cooperative multiagent system for sign language tutoring. We study different techniques for distributed decision making. After implementation of the system, we also have plans to experimentally evaluate the performance of such a system.

A training data will be distributed to each agent. Each agents training data is not disjoint from others, that means there may be intersections. After receiving its portion, each agent will train its classifier using its portion of training data. The training data may include errors or variations of the same sign on purpose, since we suppose to have such cases in reality. To test majority and weighted majority voting, we will run test data over the system and calculate the percentage of success, how many times agents made the correct decision. Whereas for model-based approaches we will measure how successful the models of agents are.

There are also some issues on which we are not clear about. The first is the compositional decision case. Agent is requested to recognize a sentence in sign language. We can assume that the sentence is divided into its parts. Agent decides on each part individually with support of other agents using one of the models described above. And then it makes a final composed decision. The question is *How to combine individual decisions?*

The second issue is dialects of the sign languages. There are dialects of Turkish Sign Language. Assume there are two different signs for the word father. And two different agents are educated for the different fathers. Then here is the problem: For the performance of the sign father one agent says right, whereas other says wrong and actually they are both saying the correct thing. How to recognize and deal with dialects?

Durfee discusses challenges in Distributed Problem Solving and Planning, using several problems and possible and proposed solutions to those problems [4]. He states that distributed problem solving requires marshalling of distributed capabilities in the right ways which thus requires to solve the problem of how to solve together. To come up with such a solution, agents need to communicate

for task sharing and result sharing. For task sharing, he discusses contract-net protocol in terms of several problems and possible solutions to those problems. He states that in order to achieve a confident and complete distributed problem solving agents should share their results.

In [5], a proposal of cooperative decision making and support system is given for general organization. There are different realizations of a cooperative decision support system: It may be a communication system between decision makers, who are people or it may be a system of decision making units in cooperation with other elements of the system. They argue that in organizations, partners should be able model others in enough depth to be able to request complex tasks from others. Further, partners should be able to go into necessary level of communication in order to use these complex detailed.

References

1. Aran, O., Ari, I., Benoit, A., Carrillo, A.H., Fanard, F.X., Campr, P., Akarun, L., Caplier, A., Rombaut, M., Sankur, B.: Signtutor: An interactive system for sign language tutoring, *IEEE Multimedia*, to appear (2008)
2. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. *Knowledge Engineering Review* **10**(2) (1995) 115–152
3. Conitzer, V., Sandholm, T.: Vote elicitation: Complexity and strategy-proofness. In: *In Proceedings of the National Conference on Artificial Intel ligence (AAAI)*, Edmonton, Canada (2002)
4. Durfee, E.: Distributed problem solving and planning. *Multi-Agent Systems and Applications* (2001) 118–149
5. Soubie, J.L., Zarat, P.: Distributed decision making: A proposal of support through cooperative systems. *Group Decision and Negotiation* **14**(2) (2005) 147–158

Resource Management for a Per-to-Peer Service Oriented Computing System

Mircea Moca

Babeş-Bolyai University
Business Information Systems Department
Str. Theodor Mihali No. 58-60, 400591, Cluj-Napoca, Romania
`mircea.moca@econ.ubbcluj.ro`

Abstract. In this paper we set the directions to follow in order to build a resource management module as a component of service oriented computing system. We intend to use peer-to-peer architecture in designing the system for its scalability, fault resistance and openness advantages.

1 Introduction

I am first year PhD student, finalizing PhD courses in summer 2008 and preparing a research proposal to be covered in my PhD thesis. By participating at EASSS08 I will use the received feedback concerning my research theme in order to make myself clear on the feasibility of the methods and tools I intend to approach and to get further expert guidance regarding my future work. An important issue for me is to assess a right research question and clarify the research directions I will follow during my PhD. This work was supported in part by the National Authority for Scientific Research under contracts PN II 91-049 and IDEI 573/2007.

Nowadays, computing services are done by grid systems, which harness a large quantity of resources in a centralized manner. Such systems are TeraGrid [9] in USA, Grid5000 [2] in France and National Grid Service [5] in UK. These networks form closed and strong controlled systems. Therefore, the access to these systems requires security certificates and their scope is mainly for running scientific experiments. Hence, the trust in these systems originates in their *closeness*. Another aspect we consider important to mention is that grid systems obey client-server paradigm. This implies failure points (server nodes) and sacrifices the scalability of the system. In this context, it is known the difficulty and inefficiency of adapting server parameters as the number of clients grow.

Besides, P2P systems are applications that aggregate resources (widely used for file-sharing) from Internet users [7]. The structured P2P systems proved to be scalable because of the lack of centralization, auto-adaptable - they succeed to solve the absence of structure using internal mechanisms, highly dynamic.

In order to better fulfill the above mentioned drawbacks, we aim to design a P2P system that delivers service oriented computing; we have in mind a market-like platform for managing resources of the system. Our research target is to:

- Choose an appropriate P2P architecture and
- Identify a specific behavior for peers.

After designing the system we intend to implement a software system using Sim-Grid platform; this would provide us with both simulation and real-life results. Therefore, the resulting software system would meet the following desiderata:

- Openness - users can easily join the system (without going through a human authentication), and also leave it at their wish.
- High scalability - the system copes with a great number of users; these can join the system from everywhere Internet is available.
- Dependability - the system is reliable, providing a quality service concerning correctness and trustworthiness of results.
- Market-place structure - peers can freely play roles both from demand and offer classes in order to fulfill own desires.
- Efficiency regarding the use of different types (size, quality) of resources, which we will further detail in section 4

From the previously discussed features emerge the research challenges, which we will present in next following sections. We will consider in our presentation the interdisciplinary character of our proposal, searching solutions in multi-agent systems and economics field. The interference with these domains is needed for designing peers behavior and is further discussed in sections 3 and 4.

2 System architecture

Our first concern regards the particular type of P2P architecture that we should employ for our system. We consider as suitable a structured P2P architecture, because protocols from this category supply highly satisfactory values [6, 8, 3] for the following parameters:

- resource location correct and guaranteed
- algorithm complexity - logarithmical

A still intricate issue of our architecture is whether to allow hybrid P2P architecture or not. We are conscious that equipping our system with super-nodes might improve the efficiency (faster service delivery). Apart from common peers, super-nodes can also play *global* (wider scope than neighboring zone) roles for certain functionalities. These two categories are detailed in next paragraph. A well-known and undesirable consequence of having centralized functionalities is the security threats, because bearing nodes become failure points. A server failure generally leads to disabling the entire sub-system dependent on it.

On the other hand, pure P2P architectures are able of dispensing with any participating node. If a node fails, its functionality is absorbed by a part of remaining participants. This is a strong advantage, insuring a permanent stay-up for the system, though supporting system reliability. However, in this situation the system loses execution speed when centralized information is needed. Such a situation is tackled by the management of information about aggregated resources, aspect that we will widely discuss in section 3. Since each approach hybrid and pure architectures has both strong and weak points we assume that an optimal solution would be a balance between them. Therefore, if a mix proves to be more suitable, we would compose it by deciding for every functionality whether it should be centralized or not.

Having in mind the particularities of different types of functionalities and also making a link to the hybrid-pure architecture problem we present two types of peers:

- **Common peer**, which can:
 - inject a task in the system request for a service
 - contribute with resources for:
 - * computing or
 - * storing purposes
- **Super-peer**, which is a common peer enhanced with resource management extra-capabilities. The existence of nodes of this type brings into discussion the hybrid architecture.

We further define the notion of accumulator i.e. a peer equipped with resource management capabilities. We will use this denomination in section 3, where the resource management problem is detailed.

2.1 Overlaying Internet consequences

The issue of employing structured P2P architecture for our system, also discussed in previous section brings scalability performance. Hence, with such a structure the system will scale well (logarithmic) [6, 8, 3] even at the Internet extent. This advantage supports our will to use resources of any computer connected to Internet.

Since Internet is such widely spread and continuously extending, our overlaying system must cope with its increasing dimensions. Since any Internet user is supposed to freely join and leave the system, the overall system is highly dynamic. Thus, a few research questions arise from this direction:

- How should be a specific task carried out by peers in order to deliver results when *currently-working* nodes fail (by any reason)?
- How to deal with *bad-behavior* peers? Some participants can intentionally deliver incorrect results as answers to their received tasks or place useless heavy-duty tasks so that other service-demanding nodes are debarred from service. These kinds of peers are considered to affect our system just like

viruses do with a computer. Thus, we search a feasible solution to minimize their effects. This issue is another missing brick in the system-trust wall that we are willing have in our system. Thus, we consider useful employing a reputation system as in [4] for influencing negotiating peer behavior - discussed in next section. By this method, peers are categorized based on their previous actions performed in the system, and all users act accordingly to this information.

3 Resource management

The problem of managing resources is central to our work. The way we will tackle it will influence all other system functionalities design. In this section we will discuss the assumed approaches which are determined by the P2P structure we would employ.

We firstly divide the resource management problem in two main actions to be performed by our system:

- Resource **aggregation** encompasses all actions to be done, all collaborations and negotiations between peers so as the shared *atomic* resources virtually become one globally available one.
- Resource (service) **delivery** - allocating (delivering) the aggregated resource to demanding users.

Aggregation

In the next paragraphs of this section we will present the action of aggregating the resources, with its two variants:

- Centralized and
- Distributed

According to MARA survey [1] there are two allocation procedures: centralized and distributed (decentralized). Each of them has important characteristics, which we will further discuss in the context of our system; in our discussion we will consider both aggregating and allocating actions.

The centralized paradigm would allow in our system accumulators as super-peers, leading to a hybrid structure. These accumulators would be responsible for centralizing information about other peers that share resources. In this situation, there must be a periodical running process of updating the respective information; that is, *tracked* peers announce the accumulator of modifying own shared resource quotas or even its premeditated system departure. A question here would be about the number of peers to be tracked by the super-peer; this value has to be artificially set as the system evolves. This aspect is implied by the client-server paradigm drawbacks mentioned in section 2.

Hence, we can assert that the centralized resource aggregation procedure is static, since there is a location where information about a group of shared resources is kept and is periodically updated. The update procedure is a both-ways communications between an accumulator and the resource owner: from the owner side for announcing sharing parameters or resource availability and from the accumulator when checking tracked resource presence. Obviously, maintaining this information brings an extra-charge to the accumulator.

Besides, the distributed approach would determine us to equip each peer with accumulating behavior. That is, moving the resource management capacity from super-peers (privileged nodes) to each peer. In this case, the resource management information maps on the neighboring scheme; this means that the number of peers to be tracked is natively set, thus the system remains scalable. This latter rendering has the advantage of keeping the purity of the structure, which avoids central points of failure. We also mention portability as a consequence of this approach, since every structured P2P architecture employs a neighboring scheme.

We will present now another distributed allocation procedure which is not based on the neighboring scheme, but on the DHT mechanism itself. This would be as follows: any peer can receive a service demand, and on its own, will follow the next steps to delivering the resource:

- First, evaluate the demand amplexness
- Fragment the demand and associate random numbers to each chunk (group of tasks)
- For each chunk number, find the corresponding participating node and try to hand over the chunk

When negotiation with a node fails, another number is generated and associated to the respective chunk and the process redone. We notice in this situation that resource aggregation is done *live* (right after receiving the service request); and apart from the centralized approach, information about the shared resource is more accurate because the moment of using the resource is closer from the moment it has been contacted. This particular approach totally avoids centralizing information about resources. We also consider it a possible alternative, since we not know the feasibility of any of the two presented.

Allocation using market-place like algorithms

As previously mentioned, the second main action to be performed in resource management is allocation. Having on one hand a queue with tasks from different demanding peers and peers available to *work* on the other, allocation means deciding what task goes to which peer. This is the point where we expect economics and agents realm to help us find a suitable solution. Aggregating a resource means to keep track of location and other details of a resource, but allocating (in our system) implies negotiation. An agreement between the resource

holder and the demander has to be done for a successful service transaction. Thus, we believe that peers should be equipped with a well structured negotiating behavior.

The following discussion is connected to the resource allocating problem and shows that peers can be financially motivated to participate with resources. We assume that motivating peers will make them improve the quality of the shared resource stay-up time. We envision our system as a market-place for peers that share resources (offer) and peers that need resources (demand). This means that sharing peers ask for a price for their resources and asking ones pay a certain price; though, the computing service is financially evaluated. The aspect of effectively allocating resources implies considering a price limit the demanding peer is willing to pay and another limit under which sharing peer will not cooperate. The challenge here is to find a suitable protocol for connecting peers from offer with those from demand. Achieving this, will lead to successful transactions and also keeps to minimum centralization.

Solving the problem of paying is a challenge regarding how the entity responsible for this functionality looks like. This question should have answers that go along with our previously presented desiderata. However, paying the service is a more seldom executed operation, so it is not worth charging each peer with such functionality. This observation conveys us to a possible solution of considering super-peers to be responsible for the payment procedure for peers who shared resources.

4 Resource granularity

As each Internet user can freely participate with resources, we expect many of them to contribute with relatively small quantities; that is contributing with a reduced number of FLOPS. This fact raises the question whether a minimum resource quantity limit should be assessed in order to consider it a *resource-unit*. If service-computing is freely spread to Internet, we think this will imply a higher resource fragmentation degree compared to one from classic grid systems. Though, having in mind the principle of efficiency, which is connected to the quantity a good or service is delivered, we consider worthwhile for our system to organize *low-contributing* peers into *alliances*. Thus, a *leader* peer will negotiate in the name of and will represent peers affiliated to a particular alliance. We parallel the two kinds of peers, considering their contributing capacity with peers and super peers; these types differ one from another based on speed connection and are treated differently by the system [10].

Figure 1 shows a fragment of our system, in which light-colored peers form alliance A; these nodes contribute with a small number of FLOPS and are represented in the system by node tagged with *L* (leader). Dark-gray nodes denote peers with enough available number of FLOPS to be considered a resource unit.

Hence, they negotiate for themselves only.

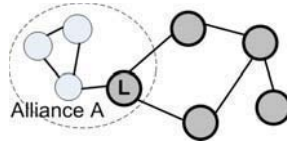


Fig. 1. A system fragment, with 4 peers forming alliance A

The reasons for a peer to join an alliance would be a greater chance to receive tasks and a better negotiated price for its work. It is generally known that poor individuals have fewer strong points into negotiation process than organized groups. Organizing low-contributing peers in alliances brings two main advantages on both sides:

- Makes resources more usable because system deals with a larger (grouped) resource. This principle obeys an efficiency criterion, since a demanding peer would *deal* only with L peer and benefit from the resources of entire alliance; *deal* means the process through which demanding peer goes from the first contact with another peer to obtaining final results from it.
- Better satisfies the alliance members; these get more tasks and money.

5 Conclusion and future work

As we presented, we have questions about the solutions for our aimed resource management system considering a P2P architecture. For achieving the system goals, we plan to start from basic version of system with its main functionalities and simulate its behavior. Based on the obtained results, we will continue by modifying peers behavior and add functionalities; with each version we try to move closer to an efficiently resource management system which financially motivates users.

References

1. Yves Chevalere, P. E. Dunne, U. Endriss, M. Lematre J. Lang, N. Maudet, J. Padget, S. Phelps, J. A. Rodriguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.
2. Grid5000. <https://www.grid5000.fr/>.
3. Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 314–329, New York, NY, USA, 2003. ACM.

4. Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152, New York, NY, USA, 2003. ACM.
5. NationalGridService. <http://www.grid-support.ac.uk/>.
6. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.
7. Clay Shirky. What is p2p... and what isn't. 2000.
8. Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.
9. TeraGrid. <http://www.teragrid.org/>.
10. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *IEEE International Conference on Data Engineering*, March 2003.

Verification of Resource Requirements of distributed rule-based Reasoners

Abdur Rakib, Nguyen Hoang Nga, Natasha Alechina, Brian Logan*

School of Computer Science and IT, University of Nottingham
Nottingham NG8 1BB, UK

(rza, hnn, nza, bsl)@cs.nott.ac.uk

Abstract. We present a framework for the automated verification of time and communication requirements in systems of distributed rule-based reasoning agents which allows us to determine how many rule-firing cycles are required to solve the problem, how many messages must be exchanged, and the trade-offs between the time and communication resources. We extend CTL* with belief and communication modalities to express bounds on the number of messages the agents can exchange. We propose a temporal epistemic logic, \mathcal{L}_{CRB} , that can be used to express both bounds on time and on communication. We provide an axiomatisation of the logic and prove that it is sound and complete.

1 Introduction

A key application of multi-agent systems research is distributed problem solving which allow groups of agents to collaborate to solve problems which no single agent could solve alone (e.g., because no single agent has all the information necessary to solve the problem), and/or to solve problems more effectively, e.g., in less time than a single agent. For a given problem and system of reasoning agents, many different solution strategies may be possible, each involving different commitments of computational resources and communication by each agent. For a given system of agents with specified inferential abilities and resource bounds it may not be clear whether a particular problem can be solved at all, or, if it can, what computational and communication resources must be devoted to its solution by each agent. For example, we may wish to know whether a goal can be achieved if a particular agent, perhaps possessing key information or inferential capabilities, is unable (or unwilling) to contribute more than a given portion of its available computational resources or bandwidth to the problem.

There has been considerable work in the agent literature on distributed problem solving e.g., [3–5] and on distributed reasoning e.g., [6, ?]. Much of this work analyses the time and communication complexity of distributed reasoning algorithms. However, while we have upper (and some lower) bounds on time requirements for reasoning in distributed systems, possible trade-offs between resources such as time and communication are less clear. In [1], we investigated resource requirements for time, memory and communication for systems of distributed resolution reasoners.

* Abdur Rakib and Nguyen Hoang Nga are PhD students, Natasha Alechina is an Associate Professor and Brian Logan is a Lecturer at the University of Nottingham.

In this paper, we focus on a more detailed investigation of time and communication trade-offs for *rule-based reasoners*.

2 Systems of communicating rule-based reasoners

The system consists of n_A agents, $\mathcal{A} = \{1, \dots, n_A\}$, where $n_A \geq 1$. We will assume that each agent has a number in \mathcal{A} , and use variables i and j over \mathcal{A} to refer to agents. Each agent has a *program*, consisting of propositional Horn clause rules, and a working memory, which contains facts (propositions).¹ If an agent i has a rule $A_1, \dots, A_n \rightarrow B$, the facts A_1, \dots, A_n are in the agent's working memory and B is not in the agent's working memory in state s , then the agent can fire the rule which adds B to the agent's working memory in the successor state s' .

In addition to firing rules, agents can exchange messages regarding their current beliefs. We assume that there is a bound on communication for each agent i which limits agent i to at most $n_C(i)$ messages. Each agent has a communication counter, c_i , which starts at 0 and is not allowed to exceed the value $n_C(i)$. The exchange of information between agents is modelled as an abstract *Copy* operation: if a fact A is in agent i 's working memory in state s , A is not in the working memory of agent j , and agent j has not exceeded its communication bound ($c_j < n_C(j)$) then in the successor state s' , A can be added to agent j 's working memory, and c_j incremented. Intuitively, this corresponds to the following operations rolled into one: j asking i for A , and i sending A to j . This is guaranteed to succeed and takes one tick of system time. The only agent which pays the communication cost is j . These assumptions are made for simplicity; it is straightforward to modify our definition of communication so that the 'cost' of communication is paid by both agents, communication takes more than one tick of time, and communication is non-deterministic. An agent can also perform an Idle operation (do nothing).

A problem is considered to be solved if one of the agents has derived the goal. The time taken to solve the problem is taken to be the total number of steps by the whole system (agents firing their rules or copying facts in parallel, at most one operation executed by each agent at every step). The communication cost for each agent is the value of communication counter for that agent.

As an example, consider a system of two agents, 1 and 2. The agents share the same set of rules:

RuleB1 : $A_1, A_2 \rightarrow B_1$ **RuleC1** : $B_1, B_2 \rightarrow C_1$
RuleB2 : $A_3, A_4 \rightarrow B_2$ **RuleC2** : $B_3, B_4 \rightarrow C_2$
RuleB3 : $A_5, A_6 \rightarrow B_3$ **RuleD1** : $C_1, C_2 \rightarrow D_1$
RuleB4 : $A_7, A_8 \rightarrow B_4$

The goal is to derive D_1 and agent 1 has A_1, A_2, A_3 and A_4 in its working memory, and agent 2 has A_5, A_6, A_7, A_8 in its working memory. In this example, the agents require one communication and five time steps to derive the goal. (In fact, this is an optimal use of resources for this problem, as verified using Mocha, see section 5). In

¹ The restriction to propositional rules is not a very drastic assumption: if the rules do not contain functional symbols and we can assume a fixed finite set of constant symbols, then any set of first-order Horn clauses and facts can be encoded as propositional formulas.

this work, we investigated the variations on this synthetic ‘binary tree’ problem. We vary the number of rules and the distribution of ‘leaf’ facts between the agents. For example, a larger system can be generated using 16 ‘leaf’ facts A_1, \dots, A_{16} , adding extra rules to derive B_5 from A_9 and A_{10} , etc., and the new goal E_1 derivable from D_1 and D_2 . We have chosen this form of example because it is typical of distributed reasoning problems and can be easily parameterised by the number of leaf facts and the distribution of facts to the agents.

3 The logic \mathcal{L}_{CRB}

We begin by defining an internal language for each agent. This language includes all possible formulas that the agent can store in its working memory. Let P be a finite set of common alphabet of facts. Let Π be a finite set of rules of the form $p_1, \dots, p_n \rightarrow p$, where $n \geq 0$, $p_i, p \in P$ for all $i \in \{1, \dots, n\}$ and $p_i \neq p_j$ for all $i \neq j$. For convenience, we use the notation $pre(\rho)$ where $\rho \in \Pi$ for the set of premises of ρ and $con(\rho)$ for the conclusion of ρ . For example, if $\rho = p_1, \dots, p_n \rightarrow p$, then $pre(\rho) = \{p_1, \dots, p_n\}$ and $con(\rho) = p$. The internal language IL , then, includes all the facts $p \in P$ and rules $\rho \in \Pi$. We denote the set of all formulas of IL by $\Omega = P \cup \Pi$, where Ω is finite.

The syntax of \mathcal{L}_{CRB} includes the temporal operators of CTL^* and is defined inductively as follows:

- \top (tautology) and *start* (a propositional variable which is only true at the initial moment of time) are well-formed formulas (wff) of \mathcal{L}_{CRB} ;
- $cp_i^{\bar{n}}$ (which states that the value of agent i ’s communication counter is n) is a wff of \mathcal{L}_{CRB} for all $n \in \{0, \dots, n_C(i)\}$ and $i \in \mathcal{A}$;
- $B_i p$ (agent i believes p) and $B_i \rho$ (agent i believes ρ) are wffs of \mathcal{L}_{CRB} for any $p \in P$, $\rho \in \Pi$ and $i \in \mathcal{A}$;
- If φ and ψ are wffs of \mathcal{L}_{CRB} , then so are $\neg\varphi$ and $\varphi \wedge \psi$;
- If φ and ψ are wffs of \mathcal{L}_{CRB} , then so are $X\varphi$ (in the next state φ), $\varphi U\psi$ (φ holds until ψ), $A\varphi$ (on all paths φ).

Other classical abbreviations for \perp , \vee , \rightarrow and \leftrightarrow , and temporal operations: $F\varphi \equiv \top U\varphi$ (at some point in the future φ) and $G\varphi \equiv \neg F\neg\varphi$ (at all points in the future φ), and $E\varphi \equiv \neg A\neg\varphi$ (on some path φ) are defined as usual. For convenience, we also introduce the following abbreviations: $CP_i = \{cp_i^{\bar{n}} \mid n = \{0, \dots, n_C(i)\}\}$ and $CP = \bigcup_{i \in \mathcal{A}} CP_i$.

The semantics of \mathcal{L}_{CRB} is defined by \mathcal{L}_{CRB} transition systems which are based on ω -tree structures. Let (T, R) be a pair where T is a set and R is a binary relation on T . (T, R) is a ω -tree frame iff the following conditions are satisfied.

1. T is a non-empty set.
2. R is total, i.e. for all $t \in T$, there exists $s \in T$ such that tRs .
3. Let $<$ be the strict transitive closure of R , namely $\{(s, t) \in T \times T \mid \exists n \geq 0, t_0 = s, \dots, t_n = t \in T \text{ such that } t_i R t_{i+1} \forall i = 0, \dots, n-1\}$.
4. For all $t \in T$, the past $\{s \in T \mid s < t\}$ is linearly ordered by $<$.

5. There is a smallest element called the root, which is denoted by t_0 .
6. Each maximal linearly $<$ - ordered subset of T is order-isomorphic to the natural numbers.

A branch of (T, R) is an ω -sequence (t_0, t_1, \dots) such that t_0 is the root and $t_i R t_{i+1}$ for all $i \geq 0$. We denote $B(T, R)$ to be the set of all branches of (T, R) . For a branch $\sigma \in B(T, R)$, σ_i denotes the element t_i of σ and $\sigma_{\leq i}$ is the prefix (t_0, t_1, \dots, t_i) of σ .

A \mathcal{L}_{CRB} transition system M is defined as a triple (T, R, V) where:

- (T, R) is a ω -tree frame,
- $V : T \times \mathcal{A} \rightarrow \wp(\Omega \cup CP)$ such that for all $s \in T$ and $i \in \mathcal{A}$: $V(s, i) = Q \cup \{cp_i^{-n}\}$ for some $Q \in \wp(\Omega)$ and $cp_i^{-n} \in CP_i$. We denote $V^*(s, i) = V(s, i) \setminus CP_i$.

The truth of a \mathcal{L}_{CRB} formula at a point n of a path $\sigma \in B(T, R)$ is defined inductively as follows:

- $M, \sigma, n \models \top$,
- $M, \sigma, n \models \text{start}$ iff $n = 0$,
- $M, \sigma, n \models B_i \alpha$ iff $\alpha \in V(s, i)$,
- $M, \sigma, n \models cp_i^{-m}$ iff $cp_i^{-m} \in V(s, i)$,
- $M, \sigma, n \models \neg \varphi$ iff $M, \sigma, n \not\models \varphi$,
- $M, \sigma, n \models \varphi \wedge \psi$ iff $M, \sigma, n \models \varphi$ and $M, \sigma, n \models \psi$,
- $M, \sigma, n \models X\varphi$ iff $M, \sigma, n+1 \models \varphi$,
- $M, \sigma, n \models \varphi U \psi$ iff $\exists m \geq n$ such that $\forall k \in [n, m)$ $M, \sigma, k \models \varphi$ and $M, \sigma, m \models \psi$,
- $M, \sigma, n \models A\varphi$ iff $\forall \sigma' \in B(T, R)$ such that $\sigma'_{\leq n} = \sigma_{\leq n}$, $M, \sigma', n \models \varphi$.

The models of \mathcal{L}_{CRB} satisfy a set of constraints on the accessibility relation. Intuitively, each R is composed of an n_A -tuple of agents' actions performed in parallel. We will next define precisely the set of actions that each agent can perform. They are $Rule_{i,\rho}$, $Copy_{i,\alpha}$ and $Idle_i$ where $i \in \mathcal{A}$, $\rho \in \Pi$ and $\alpha \in \Omega$. $Rule_{i,\rho}$ is the action of an agent i firing ρ ; $Copy_{i,\alpha}$ the action of copying α from another agent and $Idle_i$ is when agent i does nothing and moves to the next state.

We set constraints on the set of models such that the two following conditions are satisfied: (i) any transition between two states of the model corresponds to the effect of actions done by all agents in \mathcal{A} and (ii) for any action of an agent in \mathcal{A} that is applicable at a state s of the model, then there exists another state s' and a transition from s to s' which corresponds to the effect of the action. To formalise those two conditions, we have the following definitions.²

Definition 1. Let (T, R, V) be a tree model. The set of effective transitions R_a for an action a is defined as a subset of R and satisfies the following conditions, for all $(s, t) \in R$

1. $(s, t) \in R_{Rule_{i,\rho}}$ iff $\rho \in V(s, i)$, $V(s, i) \supseteq \text{pre}(\rho)$, $\text{con}(\rho) \notin V(s, i)$ and $V(t, i) = V(s, i) \cup \{\text{con}(\rho)\}$. This condition says that s and t are connected by agent i 's rule-fired transition if the following is true: ρ is a rule of i , $V(s, i)$ contains all premises of ρ but not its conclusion and the conclusion of ρ is added to the next state t of i .

² Page limitation does not permit us to provide the axiomatisation systems and the proof of sound and completeness of \mathcal{L}_{CRB} , so we skip it completely.

2. $(s, t) \in R_{Copy_{i,\alpha}}$ iff $\alpha \in V(s, j)$ where some $j \in \mathcal{A}$ and $j \neq i$, $cp_i^{\bar{n}} \in V(s, i)$ such that $n < n_C$, $\alpha \notin V(s, i)$ and $V(t, i) = V(s, i) \setminus \{cp_i^{\bar{n}}\} \cup \{cp_i^{\bar{n}+1}\} \cup \{\alpha\}$. In this condition, s and t are connected by a *Copy* transition of agent i iff i has copied so far at most $n_C(i) - 1$ messages from other agents, at s , i does not have α in its working memory while another agent j does and at the next state t , α is added into the working memory of i and its message counter is increased by one.
3. $(s, t) \in R_{Idle_i}$ iff $V(t, i) = V(s, i)$. The *Idle* transition does not change the state.

Below, we specify when an action is applicable. Note that we only enable deriving a formula if this formula is not already in the agent's working memory.

Definition 2. Let (T, R, V) be a tree model. The set $Act_{s,i}$ of applicable actions that an agent i can perform at a state $s \in T$ is defined as follows:

1. $Rule_{i,\rho} \in Act_{s,i}$ iff $\rho \in V(s, i)$, $pre(\rho) \subseteq V(s, i)$ and $con(\rho) \notin V(s, i)$.
2. $Copy_{i,\alpha} \in Act_{s,i}$ iff $n < n_C(i)$ where n is from $cp_i^{\bar{n}} \in V(s, i)$, $\alpha \notin V(s, i)$, $\alpha \in V(s, j)$ for some $j \in \mathcal{A}$.
3. It is always the case that $Idle_i \in Act_{s,i}$.

Finally, the definition of the set of models corresponding to a system of rule-based reasoners is given below:

Definition 3. $M(n_C)$ is the set of models (T, R, V) which satisfies the following conditions:

1. $cp_i^{\bar{0}} \in V(t_0, i)$ where t_0 is the root of (T, R) for all $i \in \mathcal{A}$.
2. $R = \bigcup_{\alpha} R_\alpha$.
3. For all $s \in T$, $a_i \in Act_{s,i}$, there exists $t \in T$ such that $(s, t) \in R_{a_i}$ for all $i \in \mathcal{A}$.

4 Model checker encoding

It is straightforward to encode a \mathcal{L}_{CRB} model for a standard model checker, and to verify resource bounds using existing model checking techniques. For the examples reported here, we have used the Mocha model checker [2], due to the ease with which we can specify concurrently executing agents in *reactive modules*, the description language used by Mocha. The presence or absence of each fact in the working memory of an agent is represented by a boolean state variable $a_i A_j$ which represents the fact that agent i believes fact A_j . The initial values of these variables determines the initial distribution of facts between agents.³ In the experiments reported below which used the binary tree with 8 leaves, all derived (non-leaf) variables were initialised to *false*, and only the allocation of leaves to each agent was varied.

The actions of firing a rule, copying a fact from another agent and idling were encoded as a Mocha *atom* which describes the initial condition and transition relation

³ We can also leave the initial allocation of facts undetermined, and allow the model checker to find an allocation which satisfies some property, e.g., that there is a proof which takes less than 7 steps. However for the experiments reported here, we specified the initial assignment of facts to agents.

for a group of related state variables. In our encoding, each agent is represented by a module. A particular distributed reasoning system is then simply a parallel composition of the appropriate agent modules.

The specification language of Mocha is *ATL*, which includes *CTL*. We can express properties such as ‘agent i may derive belief α in n steps’ as $EX^n tr(B_i\alpha)$, where EX^n is EX repeated n times, and $tr(B_i\alpha)$ is a state variable encoding of the fact that α is present in the agent’s working memory (e.g. $tr(B_i\alpha) = a_iA_j$ if $\alpha = A_j$). To obtain the actual derivation, we can verify an invariant which states that $tr(B_i\alpha)$ is never true, and use the counterexample trace to show how the system reaches the state where α is proved. To bound the number of messages used, we can include a bound on the value of the message counter of one or more agents in the property to be verified. For example, $EX^n (tr(B_i\alpha) \wedge tr(cp_i^=0 \vee cp_i^=1))$, where $tr(cp_i^=0 \vee cp_i^=1)$ is translated to the statement $a_i_counter < 2$, bounds the number of messages used by agent i to be at most 1. The encoding of the models and translation of the properties from \mathcal{L}_{CRB} into the Mocha specification language does not involve a significant overhead in comparison to other model-checking problems.

5 Experimental results

We have experimented with the above mentioned ‘binary tree’ example with varying size considering single and multi-agent cases and then investigated different distributions of leaf facts between the agents.

Case	Agent 1	Agent 2	# steps	# messages
1.	$A_1 - A_8$		7	-, -
2.	$A_1 - A_7$	A_8	6	0,3
3.	$A_1 - A_7$	A_8	6	1,2
4.	$A_1 - A_7$	A_8	7	1,1
5.	$A_1 - A_7$	A_8	8	1,0
6.	$A_1 - A_6$	A_7, A_8	6	0,2
7.	$A_1 - A_6$	A_7, A_8	6	1,1
8.	$A_1 - A_6$	A_7, A_8	7	1,0
9.	$A_1 - A_4$	$A_5 - A_8$	5	1,0
10.	A_1, A_3, A_5, A_7	A_2, A_4, A_6, A_8	7	2,3
11.	A_1, A_3, A_5, A_7	A_2, A_4, A_6, A_8	11	0,4

Fig. 1. Resource requirements for optimal derivation in 8 leaves cases

Figure 1 shows the number of derivation steps and the number of messages for each agent for varying distributions of 8 leaves. Similar trade-offs are apparent for a problem with 16 leaves⁴. However in this case there are a larger number of possible distributions of leaves, and, in general, more trade-offs for each distribution.

⁴ Due to space limitations the experimental result table has been omitted.

Although these examples are very simple, they point to the possibility of complex trade-offs between time and communication bounds in systems of distributed reasoning agents. For more complex examples, we would anticipate that such trade-offs would be harder to predict *a priori*, and our framework would be of correspondingly greater utility.

6 Conclusions

In this paper, we proposed an approach to modelling and verifying resource requirements of distributed rule-based reasoners. We showed how to reason about time and communication bounds in such systems.

References

1. N. Alechina, B. Logan, N. H. Nga, and A. Rakib. Verifying time agents. In *Proc. of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, Estoril, Portugal.
2. R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Computer Aided Verification*, pages 521–525, 1998.
3. B. Faltings and M. Yokoo. Introduction: Special issue on distributed constraint satisfaction. *Artificial Intelligence*, 161(1-2):1–5, 2005.
4. H. Jung and M. Tambe. On communication in solving distributed constraint satisfaction problems. In *Multi-Agent Systems and Applications IV, Proc. 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005*, volume 3690 of *LNCS*, pages 418–429. Springer, 2005.
5. G. M. Provan. A model-based diagnosis framework for distributed embedded systems. In *Proc. of the Eighth International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pages 341–352. Morgan Kaufmann, 2002.
6. M. Wooldridge and P.E. Dunne. On the computational complexity of coalitional resource games. *Artif. Intell.* 170 (2006) 835871
7. E. Amir and S.A. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence* 162 (2005) 4988

Author Index

Ahmed, Atiq	1
Alechina, Natasha	39
Boulahia, Leila Merghem	1
Freitas, Edison Pignaton	17
Gaïti, Dominique	1
Hoang Nga, Nguyen	39
Kitio, Rosine	9
Larsson, Tony	17
Logan, Brian	39
Moca, Mircea	31
de Morais, Wagner Ourique	17
Pereira, Carlos Eduardo	17
Rakib, Abdur	39
Söderstam, Per	17
Yıldırım, İlker	25