View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Electronic Communications of the EASST (European Association of Software Science and...

Electronic Communications of the EASST
Volume 56 (2013)

Proceedings of the Combined workshop on
Self-organizing, Adaptive, and Context-Sensitive
Distributed Systems
and
Self-organized Communication in Disaster Scenarios
(SACS/SoCoDiS 2013)

Decentralized Coordination in Self-Organizing Systems based on Peer-to-Peer Coordination Spaces

Thomas Preisler, Ante Vilenica and Wolfgang Renz

12 pages

Guest Editors: Michael Zapf, Florian Evers
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Decentralized Coordination in Self-Organizing Systems based on Peer-to-Peer Coordination Spaces

Thomas Preisler[1], **Ante Vilenica**[2] **and Wolfgang Renz**[1]

[1] thomas.preisler;wolfgang.renz@haw-hamburg.de
Multimedia Systems Laboratory, Faculty of Engineering and Computer Science
Hamburg University of Applied Sciences, Berliner Tor 7, 20099 Hamburg, Germany

[2] vilenica@informatik.uni-hamburg.de
Distributed Systems and Information Systems, Computer Science Department
University of Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

Abstract: Coordination is an important aspect to realize Self-organizing Systems usually implemented as part of the functional properties of the system. This paper promotes a separation of concerns via a declarative approach to realize *decentralized coordination* in Self-organizing Multi-Agent Systems (MAS). In previous work the concept of *Coordination Spaces* was developed which provides explicit support for the task of coordination in MAS. Coordination Spaces are part of the agent environment and handle a declarative description of the coordination process. Thereby, this approach allows developers rather focusing on what to coordinate than on how to coordinate. Also by releasing the developer from programming coordination manually, the approach offers benefits like reusability and interoperability of coordination processes. This paper extends the approach by a *distribution concept* for coordination spaces. By using different techniques like remote service calls, group communication and publish/subscribe models the distribution of information among multiple platforms in a peer-to-peer like approach is achieved.

Keywords: Decentralized Coordination, Coordination Architecture, Self-Organization, Distributed Systems, Peer-to-Peer

1 Introduction

Today's distributed software systems are characterized by an increasing size and complexity. This demands for novel engineering approaches. The utilization of self-organizing processes for Multi-Agent Systems (MAS) has been proposed [SGK06] to enable adaptiveness of inherently decentralized system architectures. Self-organization refers to physical, biological and social phenomena, where global structures arise from local interactions of autonomous individuals (e.g. agents) [Pro08]. MAS are a well suited approach to realize complex distributed systems. The naturally decentralized architecture provides appropriate concepts to realize systems which inherently offer non-functional requirements like scalability, robustness and fault tolerance. To ensure these characteristics on a global level the agents need to be coordinate. The work presented in this paper is based on the idea that all aspects related to coordination should be handled separately within the MAS environment. This environment encapsulates all concerns related to

coordination which allows developers to easily apply different coordination strategies without the need to change the agents' functionality. Therefore, this paper advocates a clear separation between application functionality and coordination. Environments as a conceptual part of MAS have been widely recognized. The definitions of agents point out the occurrence of an environment in which the agent is situated [WJ95]. More recently it has been argued that environments have to be understood as first-class entities within MAS and therefore as an explicit concept to handle concerns apart from the core agent functionalities [WSR+05, KG06]. Following this concept, environment-mediated coordination [SGK06, WSR+05] as well as coordination by direct agent interaction has been analysed in order to propose a layered architecture with an agent coordination layer [SR08] in order to separate coordination from application programming.
The work presented here extends the concepts originally proposed in [SR09a] to handle coordination in MAS explicitly with environments treated as first-class entities. In [VSL+10] a model was presented that enables a declarative description of coordination entities and mechanisms in MAS. The environment can process this description automatically and therefore relieves developers from implementing coordination functionality manually. Also, this allows the coordination process to be modularized and coordination mechanism to be exchanged as well, offering benefits like reusability and interoperability of coordination processes. These environments are called *Coordination Spaces*. The work presented in this paper extends the Coordination Spaces by adding a distribution concept. As the Coordination Spaces are based on non-distributable environments in the Jadex framework [BP12b] they needed to be equipped with some sort of distribution mechanism to realize distributed applications. Therefore, this paper presents a generic reference architecture which allows the usage of different techniques to achieve the required distribution. This architecture allows the interaction of multiple local Coordination Spaces to form a Peer-to-Peer network of Coordination Spaces to achieve decentralized coordination in Self-organizing Systems. Therefore, three different example implementations - based on remote service calls, Distributed Hash Tables (DHT) and a publish/subscribe model - are described to illustrate the flexibility benefits of the approach as the distribution concept can be adapted to the needs of the application context or the network topology.
This paper is structured as follows. The next Section discusses related work, Section 3 describes the concepts and the reference architecture behind distributed Coordination Spaces. In Section 4 an example realization for the Jadex framework and a case study is described. Finally, Section 5 concludes the paper and gives an overview about future work.

2 Related Work

Decentralized coordination based on Coordination Spaces is a wide spread problem. There are several applications and approaches on how to coordinate distributed systems. For instance, [NVCO10] present a self-organizing infrastructure that offers coordination capabilities inspired by chemical reactions utilizing the *TuCSoN* Coordination Space concept. The TuCSoN Coordination Space relies on a multiplicity of independent communication abstractions, called tuple centers. These can be spread over Internet nodes and used by agents to interact with each other. Agents interact by exchanging tuples via tuple centers by means of a small set of communication primitives. TuCSoN exploits tuple centers as its coordination media, where a tuple center

enhances a tuple space with a behavior specification. Therefore, the tuple centers are a communication abstraction whose behavior can be defined to embed an overall law of coordination. This is similar to the approach presented in this paper which utilizes coordination media as communication abstractions. But it lacks a clean separation between application and coordination logic which is propagated by the presented approach.
In [JXJY06] another tuple space oriented approach for decentralized coordination is presented. *DTuples* is a peer-to-peer tuple space middleware built on top of a DHT. The approach aims at combining the advantages of peer-to-peer systems with the tuple space model firstly introduced in *Linda* [CG89]. Since a standard tuple space model like Linda is based on a central space for communication and coordination, it has to deal with problems like single point of failure and scalability issues. Peer-to-peer systems overcome these issues as they are based on a decentralized approach, but most of them lack coordination primitives like given in tuple spaces. Therefore, [JXJY06] builds a tuple space based on a DHT. Since in a dynamic peer-to-peer system network nodes can fail and new nodes can appear, for the realization of a fault-tolerant systems it is necessary that the the data stored in the distributed tuple space is replicated among multiple nodes and the data's integrity is insured by transactions. The approach presented in [LP05] is also based on a DHT. The presented *Coment* is a scalable peer-to-peer content-based Coordination Space for decentralized distributed environments. It provides a global virtual shared space that can be accessed by all system peers. By adding a transparent layer as an overlay network on the DHT, the system peers can access the virtual share space without requiring the host identifiers or location information of tuples. The usage of a DHT enables Coment to guarantee information lookup and performance. Both are essential requirements for a coordination middleware in decentralized environments. Transient Coordination Spaces are also supported to enable context locality to be explicitly exploited for improving system performance. Both presented approaches focus on transferring the tuple space model to peer-to-peer systems to overcome scalability and single point of failure issues, while ensuring performance and fault-tolerance. The approach presented in this paper introduces a more flexible architecture which facilitates the usage of different technical distribution solutions and utilizes DHT only as one possible solution. So, an optimal solution for the given application and network topology can be chosen.
A different approach is presented in [RVO08]. According to the presented Agents and Artifacts (A&A) metamodel a MAS consists of agents and so called artifacts. These represent elements of the environment which can be used by agents. For the purpose of coordination via artifacts, specific coordination artifacts can be devised. Such artifacts can include blackboards similar to the mentioned tuple spaces, maps or task schedulers. Similar to the approach presented in this paper, the artifact approach transfers the responsibility for the coordination to the environment. But the approach expects the agents to explicitly use the artifacts for coordination. This implicates that coordination is not brought transparently to the agents. Therefore, coordination has to be part of the application's functional system design contrasting to the approach presented in this work that advocates a separation of concern as introduced by [GC92]. The work on the A&A metamodel was extended in [RPV11] where the notion of environment programming in MAS was introduced and a concrete computation and programming model based on the artifact abstraction was described with the CArtAgO framework. By adopting the A&A metamodel dynamic sets of computational artifacts populate the environment as first-class entities. From the agents viewpoints the artifacts represent resources and tools that can dynamically instantiated,

shared and used to support their individual and collective activities. For a MAS programmer the artifacts are first-class abstractions to program functional environments that agents will exploit at runtime, including functionalities that concern agent coordination. From a software-engineering perspective this improves modularity, extensibility and reusability.

3 Distributed Coordination Spaces

This Section describes the concepts and foundations of decentralized coordination in distributed Coordination Spaces. First, the overall design concept for decentralized coordination is introduced, which is based on the clean separation of application and coordination logic. Then the Coordination Space approach is described which provides a separate space within the environment of MAS that is dedicated for coordination. The last part presents the Peer-to-Peer based reference architecture with enhances the Coordination Spaces with distribution mechanisms to realize flexible, distributed Coordination Spaces.

3.1 Coordination Enactment Architecture

The work on decentralized coordination in Self-organizing Systems is based on a tailored programming model for the purpose of software-technical utilization of coordination mechanisms as reusable design elements [SR09b]. This model provides both, a systematic modeling and configuration language called *MASDynamics* and the *DeCoMAS* reference architecture to enable the enactment of pre-described coordination models [SR09a]. The architecture is based on the clean separation of activities that are conceptually relevant to the coordination of system entities and the systems' functionality. This allows coordination models to be treated as first class design elements that define application-independent coordination interdependencies. Figure 1 illustrates the conceptual layered structure of the coordination enactment architecture. The functionality of the application is realized as a MAS within the application layer. The coordination logic is realized by an underlying coordination layer. This layer provides a set of coordination media which provide coordination mechanisms. They build the infrastructure that allows the agents to exchange application independent coordination information and control the information dissemination. They are accessed by generic publish/subscribe interfaces which decouple the participating agents. The agents communicate with the coordination media via their coordination enactments (cf. figure 1(B)). These endpoints observe and influence the agent activities (1) and via the coordination media, they exchange the information that is relevant for the coordination (2). The configuration of the whole coordination process is described in a declarative, external coordination model (3) written in the *MASDynamics* language. The configuration file contains all information relevant to coordination, e.g. when to publish coordination information, which information is to publish, which coordination mechanism is to be used and how is it configured and also to which agents should specific information be published. Coordination is declaratively prescribed to support the reuse of coordination pattern in different application designs. Following this architecture, the coordination logic is transparently separated from the application logic, meaning that the agent models are not modified by the coordination logic. So the coordination logic can be supplemented to an existing application design.

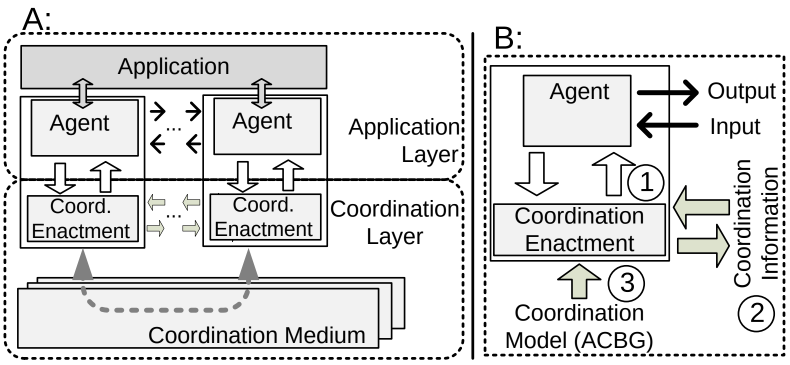


Figure 1: Coordination enactment architecture [VSL+10].

3.2 Coordination Spaces

The coordination enactment architecture was extended by [VSL+10] with the introduction of *Coordination Spaces*. The Coordination Space concept is based on the architecture and principles introduced in [SR09a, SR09b] and extends them by using environments as first class entities within MAS. It provides separate spaces within the environment of a MAS, that are used for coordination. Figure 2 illustrates the approach. Coordination Spaces are placed within the environment of a MAS, dedicated to all aspects related to coordination. All participating agents are observed by the space via *listeners*. Therefore, whenever the space perceives an event relevant to the coordination via its *perceive interface*, this event is published to the *coordination medium* which processes it and publishes it back to the Coordination Space by triggering an according event. The Coordination Space uses its *percept generator* to create a perceive event in the relevant agents, which are equipped with a *percept processor* to perceive coordination events. The triangle of *Agent Listeners*, *Percept Generator* in the Coordination Space and *Percept Processor* in the agent as shown in Figure 2 maps the Coordination enactments (cf. Figure 1) to the Coordination Space model. As in the previously described work, all the aspects of a Coordination Space are declaratively modeled and described in a separate configuration file [VSL+10]. Figure 2 also shows the three important interfaces the architecture relies on and which ensure the exchange of different coordination media and allow the usage of heterogeneous agent architectures. By providing the *ICoordinationMedium* interface it is ensured that all coordination media implement certain basic methods and have a similar structure. In order to support the usage of heterogenous agent architectures it is only necessary to implement one *agent listener/observer* as well as one *percept generator/processor* for the according architecture, as these interfaces encapsulate all actions and percepts related to coordination for a specific agent architecture.

3.3 Peer-to-Peer Coordination Spaces

The previously described Coordination Spaces allow the declarative description of coordination in MAS. By providing a separate space within in the environment of a MAS, which is dedicated to all aspects related to coordination, it is possible to separate the coordination logic transparently from the application logic. So, the coordination logic can be supplemented to an existing

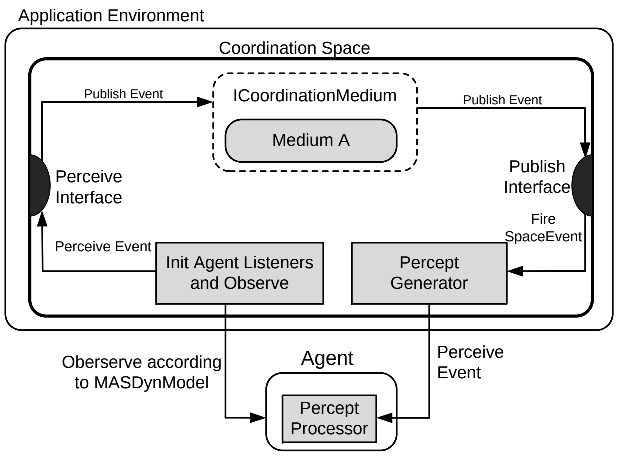


Figure 2: Conceptual model of Coordination Spaces [VSL+10].

application design and there is no need to modify the agent models to achieve coordination. The Coordination Spaces are conceptually designed to enable decentralized coordination in MAS on a local node. In order to support decentralized coordination in applications distributed over multiple nodes the Peer-to-Peer Coordination Spaces have been envisioned.

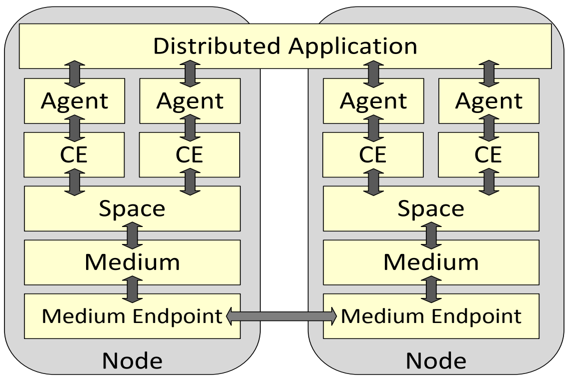


Figure 3: Peer-to-Peer Coordination Spaces architecture

Figure 3 shows how the Peer-to-Peer Coordination Spaces support the distribution of coordination information conceptually. To enable coordination of distributed applications running on multiple nodes *Medium Endpoints* are introduced. These endpoints can be equipped with different mechanisms to distribute coordination information among multiple coordination media and therefore connect them to a distributed coordination medium. In case of a distributed application with multiple nodes, on each node a number of agents are executed. All these agents are coordinated in a decentralized way by the same Coordination Space under the usage of their coordination enactments. Following the concept of Coordination Spaces the coordination media are used to process and publish coordination events perceived by the Coordination Space. Based on the newly added medium endpoints it is now possible to distribute these coordination

information not only to the local Coordination Space but also to Coordination Spaces on remote nodes. Whenever a local coordination medium processes a coordination event, it can use its endpoint to distribute the coordination information to remote nodes. On the remote node the medium endpoint is used to forward the received coordination information to its medium. A coordination event then is published to the according local Coordination Space the same way as locally perceived coordination information.
On a technical level the distribution of coordination information can be realized with different approaches. Currently coordination media and endpoints have been implemented which facilitate remote service calls, a group communication model based on DHTs and a message-queue inspired publish/subscribe model. These three approaches will be described in the next Section.

4 Realization

The Peer-to-Peer Coordination Space architecture has been implemented for the Jadex framework [BP12b]. Jadex is conceptually based on active components that combine properties of agents with software components. Jadex therefore allows the development of traditional MAS with BDI-agents[1], reasoning and environments, but also allows the usage of more lightweight active components which can show a reactive or proactive behavior. Additionally to message based communication among the agents, following the FIPA standard[2], Jadex also supports service-based architectures where the components can offer and require certain services. These services can also be offered as web services to allow an integration with other applications. Similarly, the active components are able to consume web services from third party applications [BP12a]. Jadex applications are described in a declarative way [PB10] akain to the MASDynamics languages described in [SR09b].
Figure 4 depicts the realization of the generic Peer-To-Peer Cordination Space architecture in the Jadex framework. Due to the lack of Jadex to inherently support fully distributed applications over network nodes the realization introduces the concept of *Coordination Contexts*. It is used as a glue between different Jadex application instances on multiple platforms that should form together a distributed application. The following three paragraphs describe different realizations that enable Peer-To-Peer Coordination Spaces in Jadex.

Remote Service Calls

A first approach in realizing the distribution of coordination information among multiple local Coordination Spaces, was to utilize the service infrastructure of the Jadex framework. Therefore, each coordination medium offers a coordination service interface as its endpoint. This interface is called from a remote endpoint in case some coordination information should be published to the local Coordination Space. Figure 5a illustrates the process. If a local Coordination Space perceives a coordination event, the event is forwarded to the local coordination medium to be processed (1). The coordination medium uses its endpoint, in this case realized as a coordination service, to publish the event to Coordination Spaces on remote platforms. Therefore, the

[1] Belief, Desire, Intension
[2] http://www.fipa.org/

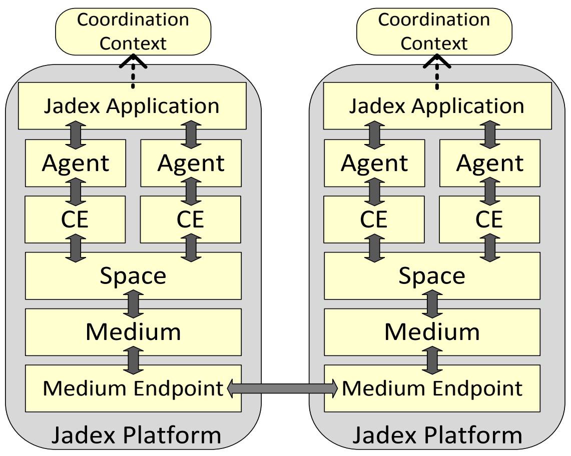


Figure 4: Realization of Peer-to-Peer Coordination Spaces in Jadex

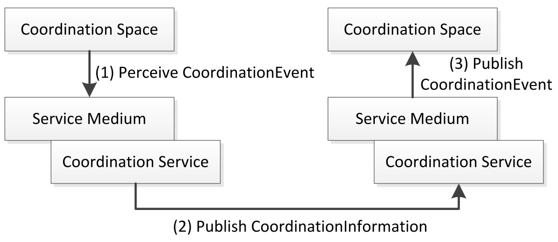
coordination service endpoint looks up all coordination services which are assigned to the same coordination context as itself and publishes the coordination information by calling the remote publish method (2). Finally, the remote coordination medium publishes the information as a coordination event by invoking the appropriate method on its local Coordination Space. Conceptually this resembles a broadcast push communication protocol.

Publish/Subscribe Model

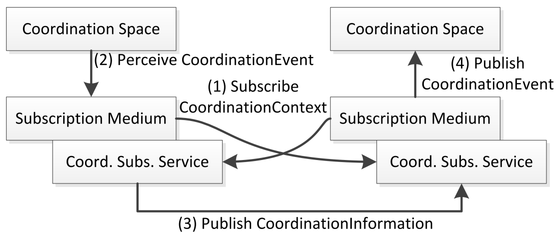
The next approach (cf. Figure 5b) realizes a publish/subscribe model for the distribution of coordination information, conceptually resembling a multicast push communication protocol. Each coordination medium, in this case called subscription medium, offers a coordination subscription service for its coordination context as its endpoint. Other coordination media endpoints can subscribe to this service, if they want to be informed in case any coordination events for the coordination context occur at the remote Coordination Space. In case an endpoint wants to be informed about all remote coordination events for a given coordination context, it has to lookup all subscription services and subscribe to them for the given coordination context (1). Again, if a local Coordination Space perceives a coordination event, it is forwarded to the local coordination medium to be processed (2). The medium publishes the coordination information to its endpoint and the service forwards it to all its remote subscribers (3). Finally, the remote subscription media will publish the coordination information as a coordination event on their local space (4).

Distributed Hash Table

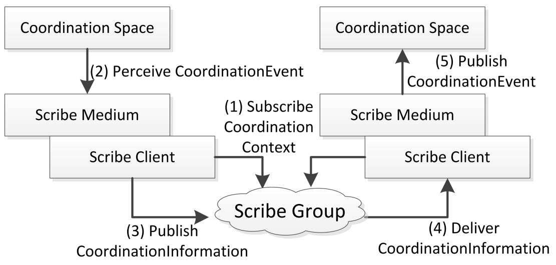
The last approach to realize the distribution of coordination information among multiple local Coordination Spaces was a proof of concept stating that the distribution could also be realized without utilizing the Jadex infrastructure. Therefore, Scribe [RKCD01], a large-scale event notification infrastructure for publish-subscribe applications, was utilized. Scribe is built on top of Pastry, [RD01] a generic peer-to-peer object location and routing substrate overlayed on the



(a) Remote Service Calls


Coordination Space
(2) Perceive CoordinationEvent
Subscription Medium
Coord. Subs. Service
(1) Subscribe CoordinationContext
Coordination Space
(4) Publish CoordinationEvent
Subscription Medium
Coord. Subs. Service
(3) Publish CoordinationInformation

(b) Publish/Subscribe Model


Coordination Space
(2) Perceive CoordinationEvent
Scribe Medium
Scribe Client
(1) Subscribe Coordination Context
Scribe Group
(3) Publish CoordinationInformation
Coordination Space
(5) Publish CoordinationEvent
Scribe Medium
Scribe Client
(4) Deliver CoordinationInformation

(c) Distributed Hash Table

Figure 5: Peer-to-Peer Coordination Space Realizations

Internet for the implementation of a DHT. Scribe uses Pastry to create groups for specified topics and to build an efficient multicast tree for the dissemination of events to the topic's subscribers. The usage of Scribe for the purpose of distributing coordination information is illustrated in Figure 5c. Each coordination medium has its own Scribe client as an endpoint which subscribes to a Scribe group for its coordination context (1). Like before, if a local Coordination Space perceives a coordination event, it is forwarded to its coordination medium (2). The Scribe medium uses its Scribe client endpoint to publish the coordination information to the Scribe group with the according coordination context (3). The coordination information is delivered by the Scribe group as a multicast message to all subscribers for this group (4). Finally, all Scribe client endpoints which have received the coordination information publish it as a coordination event on their local space by invoking the appropriate method.

4.1 Decentralized Coordination in Natural Disaster Scenarios

In order to demonstrate the feasibility of the Peer-To-Peer Coordination Spaces it is envisioned to conduct a case study called *rescue of humans in disaster situations*. More specific, it targets the coordination among different organizations in an emergency situation which is caused by a natural disaster like an earthquake or a flood. The case study will focus on the detection of humans in destroyed houses, their rescue and transport to hospitals. This objective requires coordination between different types of rescue forces, e .g., police units, fire services and ambulances. Thereby, efficient coordination among these forces relies on an appropriate communication infrastructure which cannot be assumed to work properly in the case of a disaster. Consequently, decentralized approaches of coordination seem to be better suited for these scenarios than centralized ones. From this perspective on, the case study will study the benefits of the concept of Peer-To-Peer Coordination Spaces and compare it with approaches that base on centralized coordination. The main focus is on the ability to coordinate different types of rescue forces in a decentralized fashion. Thereby, each entity is performing its task independently of others and has also its own perception of the environment. If the entity requires coordination with others it uses the Peer-To-Peer Coordination Space to publish events and to receive events from others. Despite this basic usage of the Peer-To-Peer Coordination Space, i.e. transmission of events, it can also be used as a component with computational abilities. In the case study for instance it will be used to aggregate or delay events, if necessary. Furthermore, it will also be used to compute distances between different forces and utilize this information for the transmission of events. Another benefit of the approach consists in its ability to offer multiple coordination media at the same time within one Coordination Space, i.e. different tasks can be realized with different ways of coordination. With respect to the case study this can be used to coordinate police units in a swarm based fashion whereas ambulance cars use digital pheromones. Moreover, the distributed character of Coordination Spaces allows for implementing a visual control console which can be used to display the available information about the situation in the disaster area. In conclusion this case study will evaluate the effectiveness of Peer-To-Peer Coordination Spaces.

5 Conclusion and Future Work

This work promoted a declarative approach to realize decentralized coordination in Self-organizing MAS based on Peer-to-Peer Coordination Spaces. Therefore it provides an architecture together with three alternative realizations to support the approach of explicit declarative coordination programming in MAS as described in previous work. This approach utilizes the environment of a MAS as a first-class entity dedicated to handle coordination. The approach is based on a declarative description of coordination entities and mechanisms which can be processed by the environment automatically and therefore relieves developers from implementing coordination functionality manually. It also allows the modularization of coordination which enables reusability and interoperability of coordination processes. This approach propagates a clear separation between application functionality and coordination, allowing developers to implement coordination without the need to change the agent's functionality. As this approach lacked a distribution concept, it was extended by a generic reference architecture which allows the usage of different techniques to achieve the required distribution for supporting complex, distributed systems. The resulting Peer-to-Peer Coordination Space architecture offers the flexibility to facilitate different implementations for the distribution of coordination information, depending of the needs of the application context and network topology. As an example three different distribution concepts (remote service calls, publish/subscribe model, DHT) were implemented to show this flexibility. The work presented in this paper still lacks a comparison of the three different concepts that have been implemented to support the distribution of coordination events, described in Section 4, and their advantages and disadvantages in different applications scenarios and network topologies. Therefore, future work will include such a comparison with the benchmarking and evaluation support framework for self-adaptive distributed systems described in [VL12]. Further future work will apply this approach to activate or deactivate certain coordination mechanisms during runtime. Either controlled by some sort of system administrator or by the distributed application itself. Therefore, a convergence medium will be realized which will support the autonomous decision making among the coordination enactments whether or not certain mechanisms should be activated or deactivated including the adaption of certain configuration parameters.

Acknowledgements: We would like to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting this work in a project on ”Self-organisation based on decentralized Coordination in Distributed Systems” (SodekoVS). Furthermore, we would like to thank Winfried Lamersdorf, Lars Braubach and Alexander Pokahr from the University of Hamburg for inspiring discussion.

Bibliography

[BP12a] L. Braubach, A. Pokahr. Conceptual Integration of Agents with WSDL and RESTful Web Services. In *10th Int. Work. on Prog. MAS (ProMAS'12)*. 2012.

[BP12b] L. Braubach, A. Pokahr. Jadex Active Components Framework - BDI Agents for Disaster Rescue Coordination. In Essaaidi et al. (eds.), *Software Agents, Agent Systems and Their Applications*. NATO Science for Peace and Security Series - D: Information and Communication Security 32, pp. 57–84. IOS Press, 2012.

[CG89] N. Carriero, D. Gelernter. Linda in context. *Commun. ACM* 32(4):444–458, 1989.

[GC92] D. Gelernter, N. Carriero. Coordination languages and their significance. *Commun. ACM* 35:97–107, February 1992.

[JXJY06] Y. Jiang, G. Xue, Z. Jia, J. You. DTuples: A Distributed Hash Table based Tuple Space Service for Distributed Coordination. In *Proc. of the 5th Int. Conf. on Grid and Coop. Comp.* GCC '06, pp. 101–106. IEEE Comp. Soc., Wash., DC, 2006.

[KG06] D. Keil, D. Goldin. Indirect Interaction in Environments for Multi-agent Systems. In Weyns et al. (eds.), *Environments for Multi-Agent Systems II*. Lecture Notes in Computer Science 3830, pp. 68–87. Springer Berlin / Heidelberg, 2006.

[LP05] Z. Li, M. Parashar. Comet : A Scalable Coordination Space for Decentralized Distributed Environments. In *Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems*. Pp. 104–112. 2005.

[NVCO10] E. Nardini, M. Viroli, M. Casadei, A. Omicini. A Self-Organising Infrastructure for Chemical-Semantic Coordination: Experiments in TuCSoN. In Omicini and Viroli (eds.), *WOA*. CEUR Workshop Proceedings 621. CEUR-WS.org, 2010.

[PB10] A. Pokahr, L. Braubach. The notions of application, spaces and agents - new concepts for constructing agent applications. In *Multikonferenz Wirtschaftsinformatik (MKWI): Multi-agent Systems: Decentral approaches for designing, organizing, and operating information systems*. 2010.

[Pro08] M. Prokopenko. Design vs. Self-organization. In Prokopenko (ed.), *Advances in Applied Self-organizing Syst.* Adv. Inf. and Knowl. Proc., pp. 3–17. Springer, 2008.

[RD01] A. Rowstron, P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM Int. Conf. on Distr. Systems Platforms (Middleware)*. Volume Nov. 2001, pp. 329–350. Heidelberg, Germ., 2001.

[RKCD01] A. Rowstron, A.-M. Kermarrec, M. Castro, P. Druschel. SCRIBE : The design of a large-scale event notification infrastructure. In *3rd International Workshop on Networked Group Communication (NGC2001)*. Volume November 2001. 2001.

[RPV11] A. Ricci, M. Piunti, M. Viroli. Environment Programming in Multi-Agent Systems – An Artifact-Based Perspective. *Autonomous Agents and Multi-Agent Systems* 23(2):158–192, Sept. 2011. Special Issue: Multi-Agent Programming.

[RVO08] A. Ricci, M. Viroli, A. Omicini. The A&A Programming Model and Technology for Developing Agent Environments in MAS. In Dastani et al. (eds.), *Progr. Multi-Agent Systems*. LNCS 4908, pp. 89–106. Springer Berlin / Heidelberg, 2008.

[SGK06] G. D. M. Serugendo, M. P. Gleizes, A. Karageorgos. Self-Organisation and Emergence in MAS: An Overview. *Informatica (Slovenia)* 30(1):45–54, 2006.

[SR08] J. Sudeikat, W. Renz. On the Encapsulation and Reuse of Decentralized Coordination Mechanisms: A Layered Architecture and Design Implications. *Communications of SIWN* 4:140–146, 7 2008.

[SR09a] J. Sudeikat, W. Renz. DeCoMAS: An Architecture for Supplementing MAS with Systemic Models of Decentralized Agent Coordination. In *IEEE/WIC/ACM International Conference on Web Intelligence*. Pp. 104 – 107. IEEE, 2009.

[SR09b] J. Sudeikat, W. Renz. MASDynamics: Toward systemic modeling of decentralized agent coordination. In *Komm. in Vert. Systemen (KiVS)*. Pp. 79–90. Springer, 2009.

[VL12] A. Vilenica, W. Lamersdorf. Benchmarking and Evaluation Support for Self-Adaptive Distributed Systems. In *The Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2012)*. Pp. 20–27. 7 2012.

[VSL$^+$10] A. Vilenica, J. Sudeikat, W. Lamersdorf, W. Renz, L. Braubach, A. Pokahr. Coordination in Multi-Agent Systems: A Declarative Approach using Coordination Spaces. In Trappl (ed.), *Proc. of the 20th European Meeting on Cybernetics and Systems Research (EMCSR 2010) - Int. Workshop From Agent Theory to Agent Implementation (AT2AI-7)*. Pp. 441–446. Austrian Society for Cybernetic Studies, 4 2010.

[WJ95] M. Wooldridge, N. R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review* 10(02):115–152, 1995.

[WSR$^+$05] D. Weyns, M. Schumacher, A. Ricci, M. Viroli, T. Holvoet. Environments in multi-agent systems. *The Knowledge Engineering Review* 20(2):127–141, June 2005.