Advances in Smart Systems Research ISSN 2050-8662
Vol. 7. No 1 : pp.33-42 : amsta19-043

Two layer approach: A modified Multi Agent System (MAS) as a network-independent adaptive framework for a distributed logistic network.

André Schäfer[1] , Volkmar Schau[1] , Marianne Mauch[1] , and Wolfram Amme[1]

[1] Friedrich Schiller University, Jena 07743, Germany
{andre.schaefer,volkmar.schau,marianne.mauch,wolfram.amme}@uni-jena.de

Abstract. The current living standard of industrial nations causes increasing CO^2 emissions, particulate matter and noise pollution. The interdisciplinary collaborative project "SMART DISTRIBUTION LOGISTIK" provides solutions for a successful and economical implementation of electric vehicles in short-distance freight transportation. A network will be established between logistics partners in which different variations of the vehicle fleet can be tested under model conditions. This network should be able to grow and shrink adaptively to logistics partners without influencing their actual interaction. In order to create this flexibility, a Network Structure Independent Model (NSIM) is set up in which the network architecture and the business relationships are separated from each other. Using this NSIM, an Application to Architecture Mapping will take place, in which an application with adaptable parameters will adapt to a discontinuous, widely distributed network.

Keywords: logistic, agent, mas, architecture, model, nsim, a2am

1 Introduction

As one of the largest logistics sectors in Germany, Media Logistics is about to completely restructure its logistics because of declining subscriber numbers and rising costs, resulting in a rethinking of logistics concepts.
This opens up the possibility of economically integrating electrically powered vehicles into the vehicle fleet. The research project "Smart Distribution Logistik"[1] (SDL) examines the possible applications and the economical, ecological and social impacts of electric vehicles in media logistics.
As a result, application models suitable for everyday use are to be developed, which can also be adapted for other sectors such as pharmaceutical, courier, express, parcel (CEP) and food logistics.
In order to achieve this goal, different variations of fleet compositions as well as other key parameters, such as human preferences or unpredictable traffic situations, will be modeled. Packages and letters, cars with their special characteristics, drivers, customers and other participants are to be represented as units. These should interact with

[1] http://sdl.uni-jena.de

Presented at KES-AMSTA-19 17-19 June 2019

each other and thus emulate the logistics chains. A packages should find the best way through different logistics networks. These logistics networks are to be tested with the help of different parameters in different variations in order to find an optimal constellation for the most efficient vehicle fleet. Unexpected exceptions, such as a traffic accident and the resulting traffic jam, will also be included in the model. How can the logistics network react? The individual entities should be able to decide for themselves how they react to such environmental conditions. In another scenario, it should also be possible for packages in a logistics network to communicate with the car and driver identities in order to find their own way through the network.
Regional media logistics companies are integrated as project partners of the SDL research project. Their daily processed, transported and delivered packages, delivery staff, vehicles and customers can reach enormous numbers. There are many frequently changing parameters, such as customers or suppliers. New customers will be added and removed dynamically. If one imagines a model of such a large, complex and distributed network, it becomes obvious that a central solution approach would quickly reach its limits. The media logisticians involved in our project belong to large media groups that consist of several media logisticians. Each of these companies has a specific server structure that must be supported. Furthermore, logisticians can be eliminated at any time or new group or cooperation members can join the conglomerate. These factors make this server structure very unsteady. An adaptable framework is needed, that adapts to the respective conditions of the currently available servers. Furthermore, the problem exists that there is currently no central server instance to which new servers log on. This makes it difficult for a calculation system to find all available servers. The solution for such a system is a cascading network. Each server knows a few more servers. Through this approach, our adaptable system would find the currently available servers at each scaling cycle and thus adapt to the server structure.
Therefore, problems mentioned above can be divided into two parts:

- The **business level** (application level) deals with trade relations between logistics providers and dynamically changing parameters. Examples of this would be that a media logistics *A* suddenly leaves the collaboration or a customer is added to media logistics *B*. Furthermore negotiations are to take place between the various logistics providers. Fleet compilations can also be modeled and mapped with different parameters.
- The **structure level** (architectural level), on the other hand, deals with the different types of servers of media logisticians and their discontinuity. Which operating system does a server have? Is the system switched on or is it possibly busy? Furthermore it determines how to perform a calculation on such a distributed and unsteady system.

In order to solve such a task, the technology used must consider both levels and offer solutions. The aim is to design a ***Network Structure Independent Model (NSIM)***. Both, the business level and the structure level, will be considered. Using this NSIM, an ***Application to Architecture Mapping*** will take place, in which an application with adaptable parameters will adapt to a discontinuous, widely distributed network.

The structure levels show how information about other server instances are stored and how this data can be retrieved. The application level describes the protocols on which trading relationships are based and how these software components can be generalized.
In order to better understand the requirements, the example of an ant colony is used. If the ant colony lacks food, many ants go searching for food, but if the ant colony is attacked, all ants will be available for defense. This characteristic of adapting to the conditions of the ant colony can be compared to the constantly changing and dynamic conditions of our business level. The structural level is also found in an ant colony. If a path to a food source is blocked, the ants look for new food sources and thereby also the paths to these.

2 Related Work

In order to move from an approach in the direction of a solution, typical technologies are now presented that would normally be used for similar problems. Service Oriented Architectures (SOA) come to mind as a first technology. With this technology, services are defined that can be addressed via a defined protocol and return a defined result [12]. Wang and Liao wrote to SOA:

> "The difference between SOA and traditional application architecture is SOA emphasizes interface, protocol, communication, coordination, working process, search, cooperation, publication. All these are through XML, SOAP, WSDL, UDDI and HTTP, by using of the common standard. That allows the development in different platform, and exchange data."[16]

This assessment roughly fits the requirements. A more detailed assessment of SOA is described later.
The client server approach can also be considered as another technology. One or more clients can be found on different systems and perform tasks, while they can make requests to a central server instance [14]. Lewandowski has well summarized the way Client Server works:

> "[Clients] make requests to servers for services or information and then use the response to carry out their own purpose. The server plays the role of the producer, filling data or service requests made by clients."[9]

This way of working seems to be suitable for the business level.
Microservices is another technological approach. Individual tasks are subdivided into completed processes that can communicate with each other via interfaces. This makes a modular structure of the entire application possible [11]. Kratzke wrote about it:

> "This architecture pattern is used to build big, complex and horizontally scalable applications composed of small, independent and highly decou-

> pled processes communicating with each other using language-agnostic application programming interfaces (API)."[8]

This description of Microservices exactly fits the required characteristics of the structural level.

3 Mobile Agents and Multi Agent Frameworks

The demand for a technology that combines the two concepts client server and microservices with each other without generating a considerable development effort arises. Features, such as many small units and the ability to travel between systems, are basic features of agent technologies. The usage analysis of broadened technologies with multi agent systems - as mentioned in Davidsson et al. [5] - is rarely done on special use cases and has therefore been presented here at least theoretically. Agents are software components that do or act on behalf of another person. A definition of agents in the field of computer science is the following:

> "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."[6]

With the help of agents, individual software components can be designed which are reusable and exchangeable. Furthermore, agents can be programmed very easily. The interesting aspect of agents, however, is that they can work together in a network instead of separately. This approach is also known from the ants in an ant colony or from the coexistence of humans. There are several agents/ants/humans who can do the same task. Through the cooperation of many small components a large system is created, which has an emerging intelligence. This emerging intelligence results from the fact that the great task is divided into many subtasks.
Properties[10] of agents are thus:

- Agents are *cooperative and social* because they help each other to solve problems
- Agents can act *dynamically and reactively*, e.g. change their behavior at runtime
- Agents are *robust* as they can react to negative influences of their environment
- Agents act *autonomously*, as they are not always dependent on other systems or resources
- Agents act *pro-actively*, as they decide on further steps themselves

> An extension of the properties of software agents is mobility.
> "Mobile software agents are computer programs that act as representatives in the global network of computer systems. The agent knows its owner,

> knows his or her preferences, and learns by communicating with its owner. The user can delegate tasks to the agent, which is able to search the network efficiently by moving to the service or information provider. Mobile agents support nomadic users because the agent can work asynchronously while the user is offline. Finally, the agent reports results of its work to the user through different communication channels such as electronic mails, Web sites, pagers, or mobile phones."[4]

With the help of mobility, agents can travel to different systems and work there. This "traveling" of agents is realized by an agent framework. One of the most common frameworks is the Java Agent Developing Framework (Jade). This agent framework supports the development of agents with all the features mentioned above. Migration is also supported. The migration with Jade, however, is limited to the fact that each Jade instance, to which migration is required, must first be transferred into a Jade cluster [2]. In our scenario, where several different logisticians are integrated in a logistics network, who would otherwise not cooperate any further, it would be difficult to set up a cluster via their different computer systems. A possible solution to this problem would be to connect an existing component to Jade, which takes over the migration without the need for a cluster. One such existing component would be the Kalong component [3]. The integration of this into the Jade framework involves considerable development effort [7].
A better solution is to use an agent framework that already has all the required components. One such framework is Ellipsis[2], an agent framework developed at Friedrich Schiller University Jena. This has the Kalong component integrated, with which a migration across cluster boundaries is possible. The communication between agents is based on the standard of the Foundation for Intelligent Physical Agents (FIPA) called Agent Communication Language (ACL), which is already integrated in Ellipsis. Furthermore the security aspect of the agents and the migration is covered by the Java Virtual Machine (JVM) and the Wildfly application server.

Of course, there are already some existing approaches and scenarios in which multi-agent systems have been used to map logistics networks and ultimately optimize them. Timm et al. [15] discussed, for example, the use of multi-agent systems, communication and security aspects in logistics. Anand et al. [1], on the other hand, described a framework for modeling ontologies to map all possible stakeholders and their interactions. Most work in the field of multi-agent systems in the logistics environment are of a theoretical nature, as Davidsson et al. [5] noted. A good overview of existing work can be taken from their paper. As a separation of these and similar existing works, this paper places particular emphasis on an approach for the technical implementation and on the technology decision. The model requirements of the two layers are to be mapped to the dynamic, changing architecture, which has not yet been published in this form. The theoretical approach of task division into the two layers together with the approaches to implementation constitute the main differences.

[2] http://www.ellipsis.uni-jena.de/

4 Solution for the NSIM

The solution for the business level, resulting from the negotiations between the logistics agents, could be as follows, using agents and Ellipsis: An agent of server *A* already knows other logistics agents and therefore knows that there is a server *B*. The agent now migrates to server *B* and asks for agents which would enter into a business relationship with our server *A* agent. Server *A*'s agent then interacts with these agents and trades. The agent of server *A* can now migrate to further servers and also make the same request for trade relations there. It can then always hand orders over to the agents of partner logistics companies, where it has the best conditions for itself. This creates a dynamic negotiation network. In order to recreate a delivery chain, the package always travels to the server on which the auto entity is located and communicates with it in order to recreate the modeling as best as possible. By carrying out these calculations with different parameters, such as using an electric car with different characteristics, the best fleet combinations can be determined with several runs.
If the initial logistics agent wants to expand its network, it needs information on where there could be more logistics agents and how it can migrate to these servers. The structural level of the NSIM approach is solved by the fact that there are programmatic structures in which the information about other server instances is stored [13]. This is comparable to an information card as shown in Figure 1. There is information about the address, the connection quality, where it can move from this server to other servers and software information of the server instance on this information card.

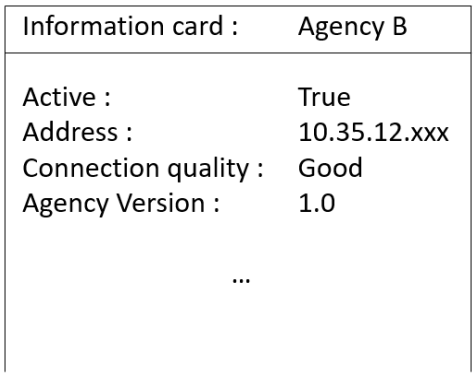

Fig. 1 : Information card

Each agent within such a system has stored information cards and can exchange them with other agents via an agent service. The workflow for distributing information

cards would therefore be as follows: An agent on server *A* only knows server *B*. Therefore, it migrates to server *B* and asks all available agents there, whether they have additional cards and whether they would share these cards with it. Now the agent of server *A* receives an information card about server *C*. It takes it back to its home server and shares the information about server *C* with all agents there. Thus cascading information about the current server landscape could be obtained. An update happens whenever an agent migrates to another server. The information about the server landscape is updated adaptively at runtime. If the migration to a known server fails several times, a "deactivated" note is made on its card. By implementing this level, Ellipsis with its agents becomes an *adaptive framework*.
The *Application to Architecture Mapping* was implemented by this principle of the two layers, which is illustrated by Figure 2. Here it is also clarified that the business level is detached from the network structure, since the available logistics agents and servers are kept up-to-date at all time by the structure level. Figure 2 also shows the structure of the NSIM. The agents with information cards and the agents trading with each other represent the two levels and are thus the components of the NSIM.

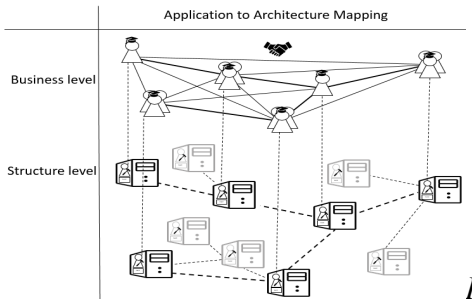

Fig. 2 : Business and Structure layer and their unification

The actual task, the reproduction of the logisticians and their trade relations, is always determined by the logisticians currently available at the architecture level. If further logisticians are added, trade relations will adapt, possibly expand or even shrink. Thus, the application is always mapped exactly to the architecture. An application model can therefore always be set up independently of the logisticians currently available in the network, because both the finding of further partners and the trade relationships between these partners are designed dynamically and flexibly. It is also not necessary that every server of the network is always running. Individual agents could stay on external servers and act until their own home server is available again. This is one of the decisive advantages of multi-agent systems compared to SOA applications from section 2. Through this advantage, our research project hopes for better entry chances, especially with smaller logisticians, who may not be able or willing to take care of the permanent availability of their own servers.

5 Evaluation

The solution approach will be evaluated using mobile agents, SOA, Microservices and client server. The evaluation with mobile agents was already done in section 4. In the following the other technologies will be evaluated. The business level and the structure level from section 1 are selected as tasks to be solved. It will be shown that the technologies presented in section 2 would not solve the problems or only with considerable additional development effort.
With an SOA architecture, very large networks with a large number of logisticians and many more negotiations between the individual entities create an enormous communication overhead over the Internet by addressing the respective services and their responses, so this approach is not suitable. In addition, services are not suitable for making autonomous decisions and reacting to environmental influences. Another difficulty is that all information must be transmitted to the negotiation platform before a negotiation can take place. Furthermore, even small logistics companies would have to operate a server with the corresponding services at all times. It would be better to have a system that can be scaled down or switched off when it is not in use.
The business level can also, as in the ant state, be implemented through many small units. The most obvious technology to implement such small units are microservices. Microservices are, according to the description, many small units that perform different tasks under different conditions. Exactly this ¨ant state¨-property is required for the business level. All entities, such as packages and letters, cars, drivers, customers and other participants could be visualized by communicating microservices. However, microservices cannot handle the structural level. Microservices are rigid and cannot easily be moved to communicate on other systems with other microservices from other logistics providers. This would only be possible by creating a cluster between the different logisticians, who otherwise do not have much to do with each other. Furthermore, in such a system, the microservices would have to know each other directly, which is not desired, since a loose heterogeneous union is required.
The client server approach of running programs on different systems that communicate with a major component fits in with the structural level requirement. Each system should run clients that communicate with a server, e.g. the home server. This makes it possible to find the servers that are currently available.
With Microservices you could handle the business level, with Client Server the structure level. A combination of these technologies would solve the overall problem at both levels. Frameworks that can solve the problems of both levels can be designed and programmed with enormous development effort. Some programming tasks for such a project are named below. A transfer of microservices by using client server principles would have to be implemented. In addition, further standards such as uniform communication protocols or the identification of other microservices using client servers would have to be designed and developed. A summary of the evaluation is shown in Table 1.

	Structure level	Business level
Microservices		x
Client-Server	x	
SOA	(x)	(x)
Mobile Agents	x	x

Table 1 : Requirements Evaluation

6 Conclusion

This paper presents an approach in which tasks can be accomplished with an NSIM implemented by agents. A simulation of the entire system is the most obvious application. All possible combinations and the procedure within the fleet can be simulated to find the best solution. The SDL research project is working towards this particular application. Agents provide the highest flexibility in terms of parameters and execution. Furthermore, the best way to distribute the calculation load among the participants is to migrate agents to less busy servers and work there. In addition, scaling such a network is not or only slightly limited, which has to be proven in further work.
Another aspect that arises from the NSIM implementation using agents is the negotiation between logistics agents and the smart-contracts resulting from this. There are wide-ranging questions to clarify. What does such a contract look like, which legal bases already exist? For future work, the aspect of data protection of customer data is interesting. All these things have to be clarified after the implementation of the NSIM in further work.

Acknowledgment

The project on which this report is based was funded by the Federal Ministry of Economics and Energy under the promotional reference number 01ME17001C.
The responsibility for the content of this publication lies with the author.

References

1. Anand, N., Duin, J.R.v., Tavasszy, L.: Framework for Modelling Multi-stakeholder City Logistics Domain Using the Agent based Modelling Approach. Transportation Research Procedia 16, 4–15 (2016), https://linkinghub.elsevier.com/retrieve/pii/S2352146516306160
2. Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: Jade Programmers Guide p. 49

3. Braun, P.: The migration process of mobile agents: implementation, classification, and optimization (May 2003), https://www.db-thueringen.de/receive/dbt_mods_00001059
4. Braun, P., Rossak, W.: Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit. Morgan Kaufmann Publishers Inc. (2004)
5. Davidsson, P., Henesey, L., Ramstedt, L., Törnquist, J., Wernstedt, F.: An analysis of agent-based approaches to transport logistics. Transportation Research Part C: Emerging Technologies 13(4), 255–271 (Aug 2005)
6. Franklin, S., Graesser, A.: Is It an agent, or just a program?: A taxonomy for autonomous agents. In: Carbonell, J.G., Siekmann, J., Goos, G., Hartmanis, J., van Leeuwen, J., Müller, J.P., Wooldridge, M.J., Jennings, N.R. (eds.) Intelligent Agents III Agent Theories, Architectures, and Languages, vol. 1193, pp. 21–35. Springer Berlin Heidelberg, Berlin, Heidelberg (1997), http://link.springer.com/10.1007/BFb0013570
7. Goos, G., Hartmanis, J., van Leeuwen, J., Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Rangan, C.P., Steffen, B.: Mobility Aware Technologies and Applications p. 354
8. Kratzke, N.: About Microservices, Containers and their Underestimated Impact on Network Performance. arXiv:1710.04049 [cs] (Sep 2017), http://arxiv.org/abs/1710.04049, arXiv: 1710.04049
9. Lewandowski, S.M.: Frameworks for component-based client/server computing. ACM Computing Surveys 30(1), 3–27 (Mar 1998)
10. Moonen, H.J.M.: Multi-Agent Systems for Transportation Planning and Coordinationa (Jun 2009), https://repub.eur.nl/pub/16208/
11. Namiot, D., Sneps-Sneppe, M.: On Micro-services Architecture 2, 24–27 (Sep 2014)
12. Papazoglou, M.: Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the 7th International Conference on Properties and Applications of Dielectric Materials (Cat. No.03CH37417). pp. 3–12. IEEE Comput. Soc, Rome, Italy (2003), http://ieeexplore.ieee.org/document/1254461/
13. Schau, V., Erfurth, C., Eichler, G., Späthe, S., Rossak, W.: Geolocated communication support in rescue management. In: M.A. Santos, L.S. (ed.) 8th International Conference on Information Systems for Crisis Response and Management: From Early-Warning Systems to Preparedness and Training, ISCRAM 2011. Information Systems for Crisis Response and Management, ISCRAM, Lisbon (2011)
14. Sinha, A.: Client-server computing. Communications of the ACM 35(7), 77–98 (Jul 1992), http://portal.acm.org/citation.cfm?doid=129902.129908
15. Timm, I.J., Woelk, P.O., Knirsch, P., Tönshoff, H.K., Herzog, O.: Flexible Mass Customisation: Managing Its Information Logistics Using Adaptive Cooperative Multi-agent Systems. In: Pawar, K.S., Rogers, H., Potter, A., Naim, M. (eds.) Developments in Logistics and Supply Chain Management, pp. 203–211. Palgrave Macmillan UK, London (2016)
16. Wang, Y.H., Liao, J.C.: Why or Why Not Service Oriented Architecture. In: 2009 IITA International Conference on Services Science, Management and Engineering. pp. 65–68. IEEE, Zhangjiajie, China (Jul 2009)