Decentralisation and Interaction Efficiency in Cooperating Autonomous Logistics Processes

Arne Schuldt

Centre for Computing and Communication Technologies (TZI)
University of Bremen, Am Fallturm 1, D-28359 Bremen

Abstract The efficiency of conventional centralised control in logistics is limited due to the complexity, the dynamics, and the distribution of logistics processes. The paradigm of autonomous logistics aims at overcoming these limitations by delegating decision-making to local logistics entities such as packages or containers. Represented by software agents, these entities must cooperate with each other to succeed in their logistics objectives. This paper introduces two interaction protocols for team formation of logistics entities. Which of them is adequate depends on the concrete application at hand. This decision is closely related with the limitations of autonomous logistics. One protocol aims at decreasing the communication effort, i.e., increasing the interaction efficiency. The other one aims at increasing the degree of decentralisation. This paper contributes a thorough investigation that supports system developers in choosing the right protocol for their demands.

1 Introduction

The complexity of logistics processes has increased significantly in the last decades. Traditionally linear supply chains have evolved into complex supply networks. Participants in these networks are highly distributed, often even over multiple continents. Furthermore, they are highly interconnected and thus highly dependent on each other. Every customer has many suppliers and vice versa. Efficiency of conventional centralistic control of such supply networks is limited. Furthermore, it is often not applicable. The limitations of centralistic control are as follows:

1. Complexity
2. Dynamics
3. Distribution

Logistics processes usually comprise a high number of participating entities and parameters to be considered. However, already problems that seem to be rather simple at first glance, like the Transport Problem and the Travelling Salesman Problem, exhibit a high computational complexity. Optimal plans might thus already be outdated as soon as their generation is finished. This problem is even aggravated by the dynamics of logistics processes because changes in the environment require frequent re-generation of plans. Finally, the physical distribution of supply networks prevents relevant information from being available for centralised planning.

The paradigm of autonomous logistics [4] addresses these issues by delegating decision-making to the local logistics entities. For instance, packages and shipping containers are expected to plan and schedule their way through the logistics network on their own (Section 2). In this approach, computational complexity can be reduced significantly because the number of parameters to be considered by each single container is limited. Robustness against dynamics is increased because replanning involves only the affected entities instead of the whole system. Furthermore, it is no longer necessary to transmit all information to a centralistic entity. Decentralised control, however, requires delegating both the ability and the autonomy to make decisions to the participating entities. To this end, intelligent software agents are employed to represent logistics entities and to act on their behalf.

However, there are also limitations in autonomous logistics. Autonomous entities can rarely succeed in their objectives on their own [10]. Instead, it is necessary to cooperate [12] which requires interaction with other entities. Obviously, the communication effort depends on the number of logistics entities involved. Therefore, it is important to find an appropriate granularity at which autonomous control is applied. Besides, communication complexity depends on the interaction mechanisms applied (Section 3). In general, there is a tradeoff between decentralisation and communication effort. The particular contribution of this paper are two interaction protocols (Section 4) for cooperating autonomous logistics processes: One with minimal communication effort (i.e., high interaction efficiency), the other one with a maximal degree of decentralisation. Which of them is appropriate depends on the concrete logistics task to be solved. A thorough examination (Section 5) helps choose the adequate protocol for implementing autonomous logistics processes. Both protocols have been implemented (Section 6) in an agent framework.

2 Cooperating Autonomous Logistics Processes

An application scenario is onward carriage in container logistics. Shipping containers arriving at a container terminal are expected to organise their transport into appropriate warehouses. This example involves three primary logistics functions: transport, handling, and storage. All of them require cooperation between containers for efficient process execution.

1. A shipping container selects an appropriate warehouse. This decision depends on properties, capacity, and costs. Cooperation is necessary because it is desirable to receive similar goods at the same location. With regard to the subsequent distribution, this helps decreasing the number of truckloads by preventing empty vehicle running.
2. Based on the warehouse chosen, the container selects a matching transport relation. Mass transport by barge or train is cheaper than transport by truck. Cooperation requires finding other containers with the same location and the same destination in order to share a train or barge.
3. The container requests a time window for receiving at the warehouse. Cooperation is necessary to ensure that containers are received in accordance with their priority. Containers waiting at the same warehouse should therefore coordinate their demands.

These examples illustrate the necessity for cooperation in decentralised logistics control. Each of the above functions requires similar containers to form teams in order to succeed in their goals. Similarity is either defined by the goods loaded, by the current location, or by the scheduled destination. The task for the agents representing the logistics entities is therefore to find potential partners for cooperation. The teams formed may differ for different tasks addressed.

3 Problem Definition and Related Work

A multiagent system (MAS) is populated by a set of agents $A=\{\alpha_1,\ldots,\alpha_n\}$. Each agent represents an autonomous logistics entity, e.g., a shipping container. The task described in the preceding section covers two aspects: Representing agent properties for team formation and the process of team formation itself. The finite set of descriptors $D=\{\delta_1,\ldots,\delta_m\}$ describes relevant properties of the logistics entities. The choice of the concrete description depends on the logistics problem addressed. The description should follow some formal language, e.g., temporal or description logics [10], so that agents can reason about it. The mapping

$$description : A \rightarrow D$$

maps agents to their descriptors. In the addressed scenario, agents with similar descriptors should form teams for successful cooperation. Note that this differs from applications where the capabilities of agents within one team supplement each other. The predicate

$$match(\delta_i, \delta_j)$$

indicates matching descriptors $\delta_i, \delta_j \in D$. A concrete specification of the predicate depends on the specific application in logistics as well.

The particular focus of this paper is therefore on agent interaction mechanisms for team formation. Previous work focused on formalising multiagent organisations [2] and internal states of agents during team formation [12]. However, con-

siderably less effort has been spent on actual *agent interaction* protocols for team formation. Distributed clustering approaches [3], as applied in wireless sensor networks, are not applicable. They focus on clustering by quantitative (spatial) data. By contrast, in the task addressed here the partitioning already exists implicitly in the agent descriptions. The task is thus rather finding potential team members *without prior knowledge* about the the other agents. Peer-to-peer approaches [6] provide each agent with an arbitrarily chosen set of other agents. Agents inform their peers about each other. Based on this foundation, they exchange their direct partners by others that are more similar. However, this setting is purely artificial for the addressed application in autonomous logistics. In particular, there is no meaningful choice for initial peers because the autonomous logistics entities are initially completely unaware of each other. A common approach to implement team formation is to apply the contract net [11] interaction protocol. If an agent intends to form a team, it could use the contract net to announce the team description to all interested agents. It is, however, not applicable to the particular problem addressed here. Teams are usually not static but *dynamic* in autonomous logistics. That is, there is no distinguished point in time at which all members jointly establish the team. Instead, agents may join the team after it has been established. In the application scenario (Section 2) consider, for instance, containers that arrive after others.

An existing interaction protocol [10] for this purpose involves a catalogue service for existing teams. This catalogue is queried by agents looking for potential partners. Subsequently, the agents send their description to the management agents of all existing teams. If one of them matches the description, the agent may join the team. Otherwise, it can itself register as a management agent for a new team with its description. This protocol leaves much autonomy regarding team formation to the agents and teams respectively. As a consequence, it has a comparatively high communication complexity. To prevent even higher communication efforts, the catalogue service still remains a centralistic entity. To summarise, the protocol is a tradeoff between decentralisation and interaction efficiency.

4 Team Formation Protocols for Autonomous Logistics

For reliable logistics processes, it is important to judge whether an interaction protocol is appropriate even before it is applied. To this end, boundary cases are of particular interest. Such cases include applications with a high demand for decentralisation or low communication effort respectively. The team formation protocol discussed in the preceding section balances these requirements. It has, however, shortcomings for boundary cases because it incorporates a centralistic entity and it exhibits a comparatively high communication complexity. Therefore, this paper introduces two derived protocols. The first one aims at minimising the communication effort (Section 4.1). The second one aims at maximising the degree of decentralisation (Section 4.2).

Agents that participate in the different team formation interaction protocols can take one or more of the following roles:

1. Participant
2. Manager
3. Broker

A participant agent aims at finding potential partners for cooperation in a specific logistics objective. A manager agent manages a team of agents that share a specific logistics objective. A broker agent administers a set of currently existing teams and their descriptors.

4.1 Minimising the Communication Effort by Broker

As a first step, this section aims at minimising the communication effort for team formation. This is important whenever communication should to avoided because it is expensive. As discussed in Section 3, there is a tradeoff between the interaction efficiency and the degree of decentralisation. That is, in order to minimise communication effort, one has to accept a decrease in the degree of decentralisation. The original protocol [10] already incorporates a centralistic entity, the catalogue service. To recapitulate, the catalogue only administers the list of the teams established. The most significant part of the communication effort arises from the fact that all teams must be contacted by agents looking for cooperation partners. It is therefore promising to delegate more responsibility to the catalogue service. In particular, it should be able to decide directly which team descriptions match the description of the agent. This turns the former catalogue into a broker agent.

The protocol flow is as follows (Figure 1). The protocol is initiated by a participant $\alpha_i \in A$ that is interested in team formation. This agent acts optimistically in that it initially assumes that no existing team matches its properties. In that case, it can itself register as the manager of a newly established team with its properties. To this end, the agent transmits its $description(\alpha_i) \in D$ to the respective broker agent in order to register itself as a team manager. The broker then compares this descriptor with those of all teams that are stored in its database. If there is no match, the agent itself is indeed registered as a new team manager. If the descriptions of α_i and an existing team manager $\alpha_j \in A$ resemble each other, i.e.,

$$match(description(\alpha_i), description(\alpha_j))$$

the registration of α_i fails. It may instead join the existing team of α_j.

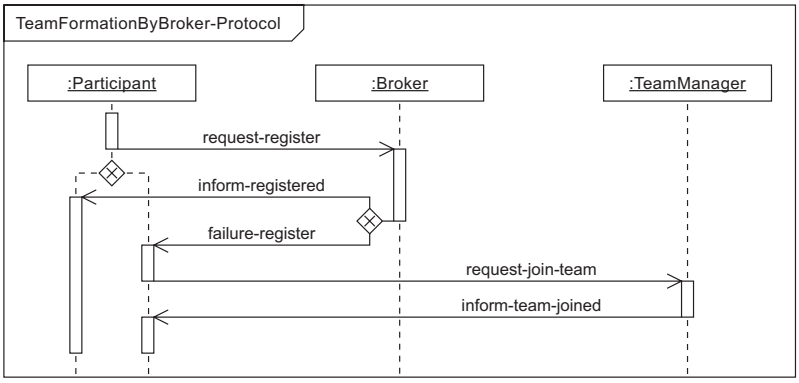

Fig. 1. Agent interaction protocol for team formation by broker. The notation is in accordance with the Agent Unified Modeling Language, in short AUML [5]. Note that exceptional messages are omitted for the sake of readability

4.2 Maximising the Degree of Decentralisation by Multicasting

Applying a broker agent significantly reduces the number of messages to be sent in the multiagent system. However, it also decreases the degree of decentralisation because all agents must contact this centralistic entity. The broker is thus a potential bottleneck of the system. In order to increase robustness, it is thus desirable to abolish centralistic entities. Of course, this also includes the catalogue service of the previous protocol [10]. To recapitulate, the catalogue is employed in order to administer the list of existing teams. That is, agents looking for cooperation partners can directly contact all team managers. Without this information, they would have to send a broadcast message to all agents because the intended recipients of the message are not known in advance. However, even if communication is affordable one should aim at reducing the number of messages sent. In general, a broadcast message is therefore not acceptable. As an alternative, multicasting can be applied. Computer network reference models like OSI or TCP/IP implement multicasting on the network layer. Hence, the application layer (which corresponds to agents) is disburdened from this task.

The protocol flow incorporating multicast messages is as follows (Figure 2). Like in the broker-based protocol (Section 4.1), the participant α_i acts optimistically, i.e., it assumes that it may establish a new team. Therefore, it contacts the Message Transfer Service (MTS) of its multiagent platform in order to receive future multicast messages on team formation. It may thus happen that multiple agents with the same description form new teams in parallel. However, this is not desirable for the application intended. The teams should clearly distinguish from

each other at least during team formation. Whenever smaller teams are preferable, the partners may split up into multiple teams, e.g., during plan formation [12].

In order to resolve potential conflicts between similar teams, the participant α_i sends its own properties to the respective multicast address. Therewith, it reaches all managers of existing teams to request a team match. The decision whether the description of α_i matches the one of a particular team is made by the managing agent itself. That is, all autonomy regarding team formation is left to the teams. If two descriptions match, α_i is informed that it must deregister and that it may join the older team of α_j. Otherwise, α_i has successfully established its own team.

Note that the optimistic behaviour of the participant distinguishes this protocol from the original catalogue-based approach [10]. The original protocol comprises an additional iteration of matchmaking before a participant registers itself as a team manager. Due to the concurrent execution of multiagent systems, this does not suffice to prevent redundant teams. Abolishing this step reduces the number of messages to be exchanged. In turn, it increases the number registrations and deregistrations for multicast addresses.

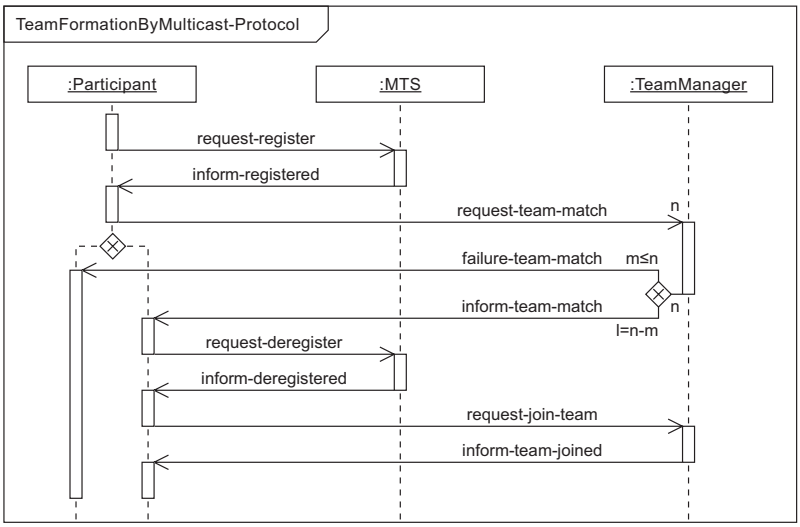

Fig. 2. AUML interaction protocol for team formation by multicast message service. Exceptional messages are omitted for the sake of readability

5 Protocol Analysis and Discussion

The protocols introduced in the preceding section address different boundary cases in autonomous logistics. In order to enable developers to judge which protocol is appropriate for a concrete logistics application, it is necessary to examine them

thoroughly. Several attributes for the categorisation of agent interaction protocols can be found in the literature (e.g., [7, 8]). The following attributes are considered here in order to compare the protocols w.r.t. their particular advantages and drawbacks (Figure 3):

1. Decentralisation
2. Autonomy of the participant
3. Autonomy of the team
4. Communication effort
5. Common language
6. Privacy

The degree of decentralisation indicates whether centralistic entities are required for protocol execution. The degree of autonomy measures how much decision-making is left to the participant and to the team respectively. The asymptotic communication complexity is an indicator for the communication effort, i.e., the interaction efficiency of the protocol. Furthermore, the extent to which a common language is required as well as the privacy are examined.

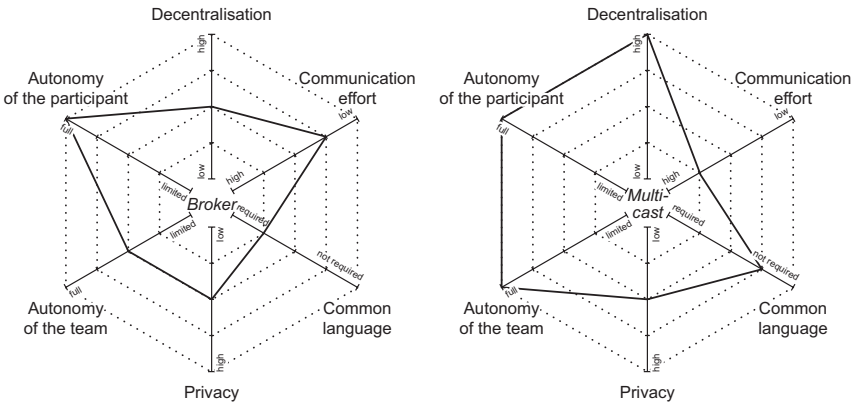

Fig. 3. Comparison of the team formation interaction protocols w.r.t. decentralisation, communication effort, common language, privacy, and autonomy

The team formation protocol based on multicasting has a high degree of decentralisation because no centralistic entity exists. All coordination is performed by the participating logistics entities themselves. The decentralisation of the broker-based counterpart is limited because the broker is a potential bottleneck. All entities looking for cooperation partners have to contact the broker agent. This is only acceptable in systems where the degree of decentralisation is less important than other attributes. In both protocols, the autonomy granted to the participants is high. Participants can deliberately decide whether they want to find partners and which team they want to join. Likewise, the multicasting-based approach does not restrict the autonomy of teams. By contrast, teams only have limited autonomy if a broker is applied. Decision-making whether a candidate matches the team descrip-

tion is delegated to the broker. However, the subsequent question whether a participant is actually accepted as a team member is again left to the team manager. If a broker is applied it must be able to understand all agent descriptions in order to decide whether they match. Hence, there is a demand for a common language all agents agree upon. Without a broker, only agents with similar goals must share a common language. Agents may then skip messages they do not understand. This also permits introducing descriptions for special purpose applications. Although for different reasons, both approaches are constricted regarding privacy. If a broker is applied, it is necessary to disclose decision processes about team matching to this centralistic entity. If multicasting is applied, agent descriptions are sent to all agents that claim to be team managers. However, virtually every agent can subscribe to this multicast channel. If there is a demand for security it is necessary to certify eligible agents.

The communication complexity of the broker-based protocol is as follows. Each agent exchanges one message with the broker. Agents that are not themselves team managers exchange two additional messages with their team manager. The complexity for every single agent is thus constant, either $O(2)$ or $O(4)$ respectively. The complexity for the whole system is thus linear, $O(4n-2m)=O(n)$ with n being the number of all agents and m being the number of the team managers. In the multicasting-based approach each agent communicates with all m team managers. The communication complexity ranges between $O(2m+2)$ and $O(2m+6)$ and is thus linear for single agents. The complexity of the whole system is thus quadratic for all n participating agents, $O(nm)=O(n^2)$. That is, the increased degree of decentralisation also increases the communication complexity.

Despite of their differences, the outcome of the introduced team formation interaction protocols equals. They are interchangeable because they all result in unique teams that can be flexibly extended. Moreover, the result is even equal to that of the catalogue-based protocol [10]. The catalogue-based approach can act as a fallback solution if multicasting is not available because it resembles its multicasting-based counterpart in most attributes.

6 Implementation and Application

The protocols introduced in this paper have been implemented in JADE [1], the Java agent development framework. The roles of agents (participant, manager, and broker) in the interaction protocols have been implemented as agent behaviours in this framework. Since JADE version 3.5, agents can subscribe to multicast topics. It is thus possible to benefit from multicast messages as demanded by the second protocol. The protocol implementation is generic in that it only incorporates agent interaction. Agent developers can add concrete agent descriptions demanded for the application intended. The protocols are currently applied in PlaSMA [9] in order to evaluate strategies for autonomous logistics. PlaSMA is a middleware that enhances JADE for parallel and distributed event-driven simulations.

7 Conclusion

This paper contributes two interaction protocols for software agents representing autonomous logistics entities. These protocols allow forming dynamic teams of agents sharing the same goals without any prior knowledge. A thorough examination helps agent developers choose the right protocol based on the logistics application intended. One protocol maximises the interaction efficiency, thereby also limiting the degree of decentralisation. The other one maximises the degree of decentralisation, thereby requiring higher communication efforts. Both protocols are generic regarding the descriptions for logistics entities.

Acknowledgement. This research is funded by the German Research Foundation (DFG) within the Collaborative Research Centre 637 “Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations” (SFB 637) at the University of Bremen, Germany.

References

1. F. Bellifemine, G. Caire, and D. Greenwood. Developing Multi-Agent Systems with JADE. John Wiley & Sons, Chichester, UK, 2007.
2. K. Fischer, M. Schillo, and J. H. Siekmann. Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems. In HoloMAS 2003, pages 71–80, Prague, Czech Republic, 2003. Springer-Verlag.
3. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In HICSS 2000, volume 8, pages 8020–8029, 2000.
4. M. Hülsmann and K. Windt, editors. Understanding Autonomous Cooperation and Control in Logistics. Springer-Verlag, Heidelberg, Germany, 2007.
5. J. Odell, H. Van Dyke Parunak, and B. Bauer. Representing Agent Interaction Protocols in UML. In AOSE 2000, pages 121–140, Limerick, Ireland, 2000. Springer-Verlag.
6. E. Ogston and S. Vassiliadis. Matchmaking Among Minimal Agents Without a Facilitator. In Agents 2001, pages 608–615, Montreal, Canada, 2001. ACM Press.
7. J. S. Rosenschein and G. Zlotkin. Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers. MIT Press, Cambridge, MA, USA, 1994.
8. T. W. Sandholm. Distributed Rational Decision Making. In G. Weiss, editor, Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence, pages 201–258. MIT Press, Cambridge, MA, USA, 1999.
9. A. Schuldt, J. D. Gehrke, and S.Werner. Designing a Simulation Middleware for FIPA Multiagent Systems. In WI-IAT 2008, pages 109–113, Sydney, Australia, 2008. IEEE Computer Society Press.
10. A. Schuldt and S. Werner. Towards Autonomous Logistics: Conceptual, Spatial, and Temporal Criteria for Container Cooperation. In LDIC 2007, pages 311–319, Bremen, Germany, 2007. Springer-Verlag.
11. R. G. Smith. The Contract Net: A Formalism for the Control of Distributed Problem Solving. In IJCAI 1977, page 472, Cambridge, MA, USA, 1977. William Kaufmann.
12. M. Wooldridge and N. R. Jennings. The Cooperative Problem Solving Process. Journal of Logic & Computation, 9(4):563–592, 1999.