Autonomous Units for Communication-based Dynamic Scheduling

Karsten Hölscher, Peter Knirsch, Melanie Luderer

Department of Computer Science, University of Bremen, Bremen, Germany

Abstract Logistics has to deal with dynamics and uncertainties. In order to cope with such problems we introduce a communication-based approach built on distributed autonomous systems. In this work graph transformation with autonomous units is used as a rule-based instantiation of multi-agent systems to model logistic systems. The approach will be presented using a scenario taken from transport logistics. Here loads which have to be transported are queued in order of their arrival but scheduled for further transportation according to their own constraints. In the paper we propose a negotiation between loads and the respective truck based on payment of transportation rates.

Introduction

Logistics is a field characterized by mainly two types of dynamics. First, the organized supply with goods and information demands for controlled material and information flows. In addition, these flows are from the operational point of view dynamic changes. To manage this kind of dynamics is the paramount challenge of logistics. Second, the whole setting is equipped with uncertainties and risks. Not all circumstances can be planed in advance; not all information needed is accessible, correct, or consistent at the point of scheduling. Occurring changes in the prerequisites can force to restart the whole scheduling process repeatedly. Especially this type of dynamics is hard to cope with using the classical approaches. However, these difficulties exist in real world scenarios and being able to react to them can result in a great competitive advantage.

The Bremen Collaborative Research Centre CRC 637 “Autonomous Cooperating Logistic Processes – A Paradigm Shift and its Limitations” tries to overcome these kinds of problems by avoiding a strictly centralistic view and by passing control capabilities to the logistic objects in order to make them smarter.

In this work, a communication-based approach will be introduced that incorporates autonomous units as a formal framework for modeling autonomous behaviors. A scenario taken from the transport logistics will demonstrate the usefulness of the chosen approach: Unit loads arriving at consolidation points for further transportation are queued (see, e.g., (Cooper 1981) for an introduction to queueing theory) according to their arrival time in a *first come, first served* manner but are scheduled according to their own constraints. Often the order of arrivals is rather arbitrary and does not reflect the real priorities of the transports to be accomplished. In addition, transport orders can arrive at any time, so early scheduling cannot react accordingly. Additionally, due to limited resources, the simultaneous transport of unit loads can be impossible and can force loads to pause. Imagine a queue of 100 unit loads where only one load at position 98 is having a hard time constraint to be met. Wouldn't it be fair to give this unit load the chance to be served first? In this case the load would have to pay a higher transportation rate while the remaining loads' disadvantages can afterwards be compensated monetarily.

In this work, a negotiation- and market-based approach is chosen to reorganize the processing of the queue in a decentralized way. For the sake of simplicity, this approach does not take into account the transhipment times.

Autonomous Units

For the structuring of the logistic system, we consider autonomous units as a basic entity having all means for modeling autonomous behavior of distributed entities. Autonomous units are a generalization of the concept of transformation units as studied in (Kreowski and Kuske 1999) and (Kreowski et al. 1997). They are a rule-based instantiation of the idea of multi-agent systems (see, e.g. (Weiss 1999)) as first introduced in (Knirsch and Kreowski 2000). A first discussion on the relation of these concepts can be found in (Timm et al. 2007).

An *autonomous unit* consists of a *goal g* (formulated in a proper logic or language), a set of identifiers *U* naming used autonomous units (that are imported in order to use their capabilities), a set *P* of rules (also called productions) specifying the operational capabilities, and some control condition *c*, which restricts the possible orders of actions.

The goal *g* describes the unit's intent. An autonomous unit acts in a specific environment, which it may change by applying a suitable rule from its productions *P*. It may also use help from other units in *U.* A unit is considered autonomous in the sense that the next action is rather based on a non-deterministic selection of a rule or of an imported unit than on control from outside the unit. A simple unit would always randomly decide on the next action. The control condition *c* provides the means for realizing a more sophisticated form of autonomy. A control condition may be very restrictive and thus eliminating the non-determinism completely or it may leave some room for non-deterministic choice.

An autonomous unit is typically chosen to represent an autonomous entity of the overall system to model. For this reason, a community of autonomous units is defined as a system comprising an overall goal *Goal*, an initial environment *Init* (both formulated in a proper logic or language), and the set *Aut* of the autonomous units that belong to the community.

Since the underlying rule-based concept of autonomous units is graph transformation (see e.g. (Rozenberg 1997)), the productions are graph transformation rules and the environment is specified as a graph. The changes of the environment happen in a well-defined way as application of graph transformation rules on the environment graph, yielding a rigorous formal operational semantics not only for a single autonomous unit but also for the community as a whole. The sequential semantics as discussed in (Hölscher et al. 2006b) is used if exactly one unit is performing an action at any given point of time. The parallel semantics, where a number of actions take place in parallel at the same time is investigated in (Kreowski and Kuske 2006). The concurrent semantics (see (Hölscher et al. 2007) for a short introduction) is used if there are no chronological relations between the acting units except for causal dependencies. In the context of this work, we consider the sequential semantics, i.e. the respective units act one at a time in sequential order.

Communication-based Dynamic Scheduling

Transport Networks

The sample scenario in this work is based on a simplified transport network. We assume a number of consolidation points in German cities and unit loads (ULDs) to be transported by trucks along road connections from one consolidation point to another one. The main relations of each truck are fixed, i.e. the routing has been arranged in advance as a regular service with timetables for each truck. Based on the knowledge of these timetables, the routes of the ULDs have also been planned in advance. What remains to be scheduled in our scenario is which of the waiting ULDs are actually transported by the respective truck. We propose a negotiation between the truck and the ULDs based on the payment of transportation rates.

In this scenario, trucks and ULDs are realized as autonomous units, called truck unit and load unit, respectively. Thus, the underlying environment comprises the consolidation points and their connections. This is modeled as a graph in a straightforward way. Now each truck unit and each load unit becomes part of the environment as a special truck resp. load node. Both kinds of autonomous units utilize tour nodes for a representation of their planned tours (as introduced in (Hölscher et al. 2006a). A truck tour is divided into tour sections, which are represented by tour nodes that are connected to the source and target consolidation points of the respective section. These tour nodes are labeled with the estimated time of departure, estimated time of arrival and the capacity for that tour section. A planned load unit tour is represented by tour nodes in a similar way, except for the labels. Load unit tour nodes are labeled with the weight of the ULD and a desired time of arrival (which is 'n.def' in the case that it does not matter when the ULD arrives). Fig. 1 shows an excerpt of a sample environment, with a truck that has started in Hamburg at noon and drives via Bremen to Dortmund, estimated to arrive at 5pm. There is no capacity left on the first tour section, since the truck transports the load unit lu_1 and others (which are not depicted for clarity). The capacity on the second tour section is eight, since it is not yet negotiated which load units are transported on that section. The load unit lu_1 with a weight 2 has planned a tour from Hamburg to Dortmund via Bremen, with no fixed arrival time. It is currently being transported from Hamburg to Bremen for a transportation rate of 4 (represented by an accordingly labeled edge between the corresponding tour nodes). Another load unit lu_4 with a weight of five has planned a tour from Bremen to Dortmund with an arrival time not later than 6pm.

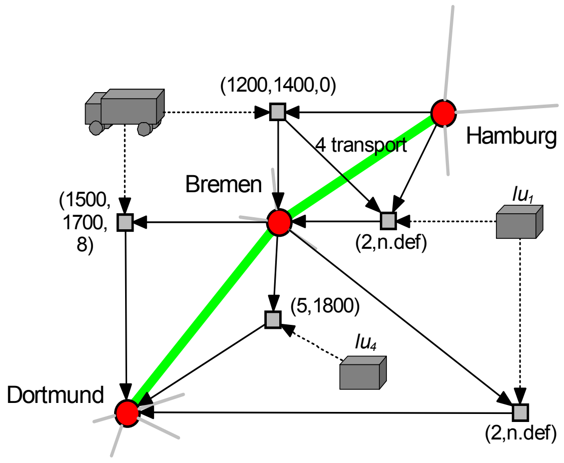

Fig. 1 Excerpt of a sample transport network

Sample Negotiation

Now consider the following concrete situation. A truck with a load capacity of eight has recently started its journey from Hamburg to Dortmund via Bremen, transporting different ULDs. One of these is lu_1, which has a weight of two and is scheduled for further transport to Dortmund, while all the others are unloaded in Bremen. Additionally three load units lu_2, lu_3, and lu_4 with the respective weights 4, 2, and 5, are queued for pickup in Bremen for transportation to Dortmund (here the position in the waiting queue is represented by the indices, in the environment the queue is represented by corresponding queue edges between the load unit nodes). ULDs are queued in the order of their arrival at the consolidation point, resp. the arrival of the corresponding transport order, preferring those ULDs that are already loaded on a truck.

Now each load unit may make an offer for transportation to the desired truck unit. The standard transportation rate in our simplified scenario is calculated to be the weight of the load unit multiplied by the transport time for the tour section in hours given by the timetable (so lu_2 would offer a rate of 4*2=8). The offer is inserted into the environment by a graph transformation rule, which inserts an edge from the respective tour node of the load unit to the corresponding tour node of the truck unit, labeled with the actual offer and a question mark. These offers have to be made until the truck arrives at the consolidation point where the ULDs are queued. The graph in the left-hand side of Fig. 2 shows an excerpt of the sample environment after all load units that desire transportation from Bremen to Dortmund have made their offers.

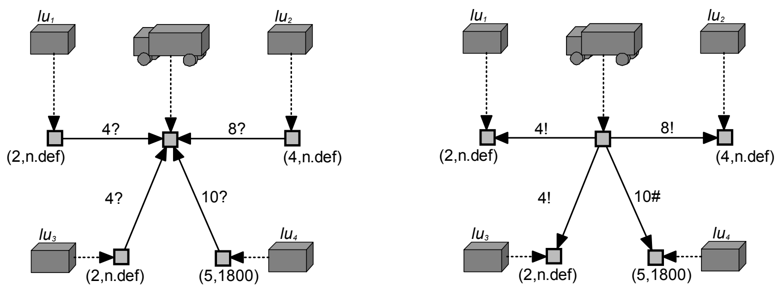

Fig. 2 Offers of load units and the truck unit

When the truck arrives at the consolidation point in Bremen, the corresponding truck unit scans the offers of the load units respecting the queue order. In the concrete example, it will accept the ULDs lu_1, lu_2, and lu_3, which will pay the overall rate of 16 and moreover result in a full truckload. Technically this is achieved by a truck rule, which considers the offer of every load unit in the order in which they are queued. This rule is applicable, if the weight of a load unit is below or equal to the remaining capacity of the tour section under consideration (skipping every load unit with a weight that exceeds the currently remaining capacity for the tour section). When applied, the rule replaces the edge labeled with the offer by a reversely directed edge labeled with the same offer and an exclamation mark. If the rule is not applicable anymore then there is not sufficient capacity left for additional load units. The rejection of offers for those load units is handled by a truck rule, which replaces all remaining offer edges with reversely directed edges labeled with the same offer and '#'. This is depicted in the graph of the right-hand side of Fig. 2.

In the next step each load unit answers to the accepted or rejected offer. If a unit accepts the truck's decision it labels the corresponding edge with an additional 'OK'. If it does not accept the decision of the truck, it labels the corresponding edge with an additional 'Not OK'. In our example, *lu*$_4$ would not accept the decision, as it has to be transported by the considered truck in order to arrive in Dortmund on time. The left-hand side of Fig. 3 shows the situation in the environment after each load unit has commented on the decision of the truck unit.

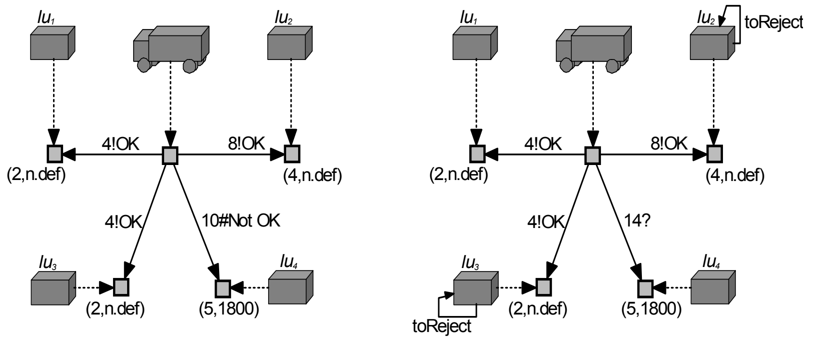

Fig. 3 Comments of the load units and new offer of the truck unit

The truck unit may now fix the transportation schedule if all edges are labeled with 'OK'. If an edge is labeled as 'Not OK', then the truck unit may calculate a new rate for the transportation of the corresponding load unit based on the potential rejection of other load units. The necessary amount to pay is calculated by the truck as the missing amount compared to the full truckload plus one for every previously accepted load unit that now has to be rejected. This additional amount is then paid to the now rejected load units, so that they gain a small advantage for further negotiations. In the concrete example, the transportation of *lu*$_4$ with a weight of five would only be possible by likewise rejecting *lu*$_2$ and *lu*$_3$, resulting in an overall payment loss of two and the usually desired full truckload. In the given scenario, there is no chance to transport *lu*$_4$ and to get a full truckload. The necessary amount to pay for *lu*$_4$ is now calculated by the truck in the following way. In order to transport *lu*$_4$, the load unit *lu*$_2$ definitely has to be rejected due to its weight. Because the load unit *lu*$_1$ is considered first in the queue (since it is already loaded onto the truck), the load unit *lu*$_3$ will also be rejected. This yields enough capacity in the truck to transport *lu*$_4$. The payment rate of *lu*$_1$ is 4, a full truckload would amount to an overall transportation rate of 16 (the capacity of 8 multiplied by 2 hours). Therefore, the load unit *lu*$_4$ will have to pay a rate of 12 for compensating the difference compared to the full truckload, and additionally one for every previously accepted and now rejected load unit, in this case 2. Therefore, the overall sum in this example would be 14. The much higher cost of 14 compared to 10 can be justified by the fact, that this load unit has a higher urgency and is waiting for a much shorter time (as can be seen by its position in the waiting queue). Technically the rule of the truck unit handling this situation would replace the label of the corresponding edge with the calculated amount and a question mark (the direction of the edge makes it distinguishable from a load unit's offer). It would also mark the potential load units to reject with a loop labeled 'toReject'. The right-hand side of Fig. 3 depicts this situation.

If the load unit accepts the suggested payment, it can apply a rule which adds 'OK' to it. This in turn makes a rule of the truck unit applicable, which replaces the corresponding question mark by an exclamation mark. It also marks the previously flagged load units as rejected by replacing '!OK' with '#' in their corresponding edge labels and removing the loop edges labeled 'toReject'. The situation of the concrete scenario is depicted in the left-hand side of Fig. 4.

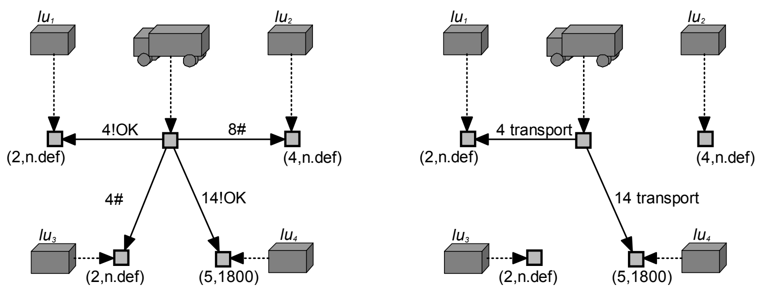

Fig. 4 Accepting a recalculated offer and finishing the negotiation

Now every rejected load unit again comments on the new decision by adding 'OK' or 'Not OK' to the corresponding edge labels. In the case of the concrete scenario, lu_2 and lu_3 would accept the truck's new decision, because they have no fixed arrival time and thus can wait for a later truck. In this case, the negotiation is considered finished which is handled by a rule of the truck unit. The application of this rule labels the edges connecting the truck tour node and the tour nodes of the transported load units with the negotiated transport rate and additionally 'transport'. The edges connecting the truck unit's tour node and the tour nodes of the rejected load units' tour nodes are removed. The right-hand side of Fig. 4 shows this situation.

However, if a previously accepted and now rejected load unit would disagree with the new situation, it would again add 'Not OK' to the respective edge label. Then the truck would in turn recalculate the now necessary amount to overrule the current proposition. This recalculation is done as before plus one for every negotiation round in order to avoid repeated transportation rates. An endless negotiation is not possible due to the fact, that the transportation rates are increasing with every negotiation round and every load unit has only a fixed amount at its disposal (and is of course not allowed to accept a transport rate which exceeds its own budget). An alternative could be the restriction of a maximum number of offer and accept/reject steps.

Conclusion

In this paper, we presented an approach to the intelligent scheduling of transports using communicating autonomous units. All logistic entities are represented by autonomous units having their own goals, their own capabilities, and their own control devices. It turned out that compared to traditional scheduling a sophisticated support of decentralized decision-making can significantly contribute to the performance of the overall system.

Autonomous units are a formal framework still under development. One aim is to use the formal framework to prove properties of the system and its components in general. An interesting aspect to prove in the context of the mentioned situations is the fact that every load unit will be transported to its final destination in time (provided that the overall constraints like capacities and timetables permit this).

Future work will regard extensions of the rather simple negotiation presented in this paper. Aspects like customer retention, transhipment costs, and competition between different logistic companies have to be considered for the negotiation to become more realistic. The possible effects on stability aspects, as investigated in e.g. (Scholz-Reiter et al. 2005), have to be considered. Alternative queueing techniques, as proposed in e.g. (Scholz-Reiter et al. 2006) should also be investigated.

A simulation tool that realizes executable autonomous units is currently being implemented. Once this tool is available, different negotiation concepts will be simulated and compared.

Although we presented a merely theoretical modeling concept, the prerequisites for a practical realization are already met (see, e.g., (Jedermann et al. 2006)).

Acknowledgments This research was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 "Autonomous Logistic Processes".

References

Cooper RB (1981) Introduction to Queueing Theory, 2nd Edition, Elsevier North Holland

Jedermann R, Behrens C, Westphal D, Lang W (2006) Applying autonomous sensor systems in logistics; Combining Sensor Networks, RFIDs and Software Agents. Sensors and Actuators A (Physical), 132:370 – 375

Hölscher K, Knirsch P, Kreowski H-J (2006a) Modelling Transport Networks by Means of Autonomous Units. In: Haasis H-D, Kopfer H, Schönberger J (eds) Operations Research Proceedings 2005, Springer, Berlin Heidelberg New York, pp 399 – 404

Hölscher K, Kreowski H-J, Kuske S (2006b) Autonomous Units and their Semantics – the Sequential Case. In: Corradini A, Ehrig H, Montanari U, Ribeiro L, Rozenberg G (eds) Proc. 3rd International Conference on Graph Transformations (ICGT 2006), Lecture Notes in Computer Science vol 4178, Springer, Berlin Heidelberg New York, pp 245 – 259

Kreowski H-J, Kuske S (2007) Autonomous Units and Their Semantics - The Parallel Case. In: Fiadeiro JL, Schobbens PY (eds) Recent Trends in Algebraic Development Techniques, 18th International Workshop, WADT 2006, La Roche en Ardenne, Belgium, June 1-3, 2006, Revised Selected Papers. Lecture Notes in Computer Science vol 4409, Springer, Berlin Heidelberg New York, in print

Kreowski H-J, Kuske S (1999) Graph Transformation Units with Interleaving Semantics. Formal Aspects of Computing 11(6):690 - 723

Kreowski H-J, Kuske S, Schürr A (1997) Nested Graph Transformation Units. International Journal on Software Engineering and Knowledge Engineering, 7(4):479 – 502

Knirsch P, Kreowski H-J (2000) A Note on Modeling Agent Systems by Graph Transformation. In: Nagl M, Schürr A, Münch M (eds) International Workshop of Graph Transformation with Industrial Relevance (AGTIVE 1999), Lecture Notes in Computer Science Vol. 1779, Springer, Berlin Heidelberg New York, pp 79 – 86

Rozenberg G (ed) (1997) Handbook of Graph Grammars and Computing by Graph Transformation, Vol 1: Foundations. World Scientific, Singapore

Scholz-Reiter B, Freitag M, de Beer C, Jagalski T (2006) Modelling and Simulation of a Pheromon based Shop Floor Control. In: Cunha P, Maropoulos P (eds) Proceedings of the 3rd International CIRP Sponsored Conference on Digital Enterprise Technology - DET2006. University of Setubal, Setubal

Scholz-Reiter B, Wirth F, Freitag M, Dashkovskiy S, Jagalski T, de Beer C, Rüffer, B (2005) Some remarks on the stability of manufacturing logistic networks. Stability margins. In: Proceedings of the International Scienific Annual Conference on Operations Research. Springer, Berlin Heidelberg New York, pp 91 – 96

Timm IJ, Knirsch P, Kreowski H-J, Timm-Giel A (2007) Autonomy in Software Systems. In: Hülsmann M; Windt K. (eds) Understanding Autonomous Cooperation & Control in Logistics – The Impact on Management, Information and Communication and Material Flow, Springer, Berlin Heidelberg New York, pp 255 – 274

Weiss G (ed) (1999) Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge, Massachusetts