ECOTRUCK: An Agent System for Paper Recycling

Nikolaos Bezirgiannis[1] and Ilias Sakellariou[2]

[1] Dept. of Inf. and Comp. Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
[2] Dept. of Applied Informatics, University of Macedonia,
156 Egnatia Str. 54124 Thessaloniki, Greece.
n.bezirgiannis@students.uu.nl iliass@uom.gr
http://eos.uom.gr/~iliass

Abstract. Recycling has been gaining ground, thanks to the recent progress made in the related technology. However, a limiting factor to its wide adoption, is the lack of modern tools for managing the collection of recyclable resources. In this paper, we present EcoTruck, a management system for the collection of recyclable paper products. EcoTruck is modelled as a multi-agent system and its implementation employs Erlang, a distribution-oriented declarative language. The system aims to automate communication and cooperation of parties involved in the collection process, as well as optimise vehicle routing. The latter have the effect of minimising vehicle travel distances and subsequently lowering transportation costs. By speeding up the overall recycling process, the system could increase the service throughput, eventually introducing recycling methods to a larger audience.

Keywords: Agent Systems, Contract-Net, Functional Logic Programming, Erlang

1 Introduction

The term *Recycling* refers to the reinsertion of used materials to the production cycle, the initial phase of which is collection of the recycled materials from the consumers. In a sense, this is a transportation problem i.e. collecting from various end-point users items in a highly dynamic manner. Due to this dynamic nature, we consider it to be an excellent area of application for multi-agent technology [1]. Although the latter has been around for many years, still presents a number of excellent opportunities for its application in new domains. In the light of new technological advances in the areas of telecommunications and portable computing devices, such applications can really “go” mainstream.

Paper recycling has gain significant attention due to a) the large quantities of paper used in everyday tasks, (offices, packaging, etc.) and b) possible reductions in energy consumption and landfill waste. Currently, municipal authorities employ a rather outdated procedure for the collection of recyclable large packaging cartons and other paper waste, that is based on telephone communication and manual truck assignment for pick up. The drawbacks of such a procedure are rather obvious: delays in the collection process, unoptimised use of resources (trucks, etc.), which can potentially lead to a low acceptance of recycling practise and thus to failure of the whole process.

EcoTruck aims to address the above issues, replacing the current “manual” collection process by a multi-agent system. This offers a number of advantages, such as distributed coordination of collection services, speed, robustness, absence of a central point of control, etc. The system is implemented entirely in Erlang, a concurrency-oriented declarative language. While Erlang is not widely adopted by the agent community, it is quite heavily used in industry to develop mission critical server products and soft real-time systems. We found that the language provides the necessary distributed infrastructure and carries enough expressiveness to allow the development of our multi-agent system.

Thus, the aim of this paper is twofold: firstly, the paper supports that the application of multi-agent technology can improve the recyclable paper collection process; secondly, we argue that the development of such multi-agent systems can be easily done in Erlang and prove the latter by presenting the implementation of our system in the language.

The rest of the paper is organised as follows. We introduce in more detail the problem we address in section 2. Section 3 presents the system’s architecture and cooperation protocol, i.e. the Contract-net protocol. Section 4 describes the implementation of the system in Erlang while section 5 presents related work on the problem. Finally, section 6 concludes the paper and presents potential extensions of the current system.

2 Collecting Paper for Recycling

While recycling technology has made significant progress, the collection of the recyclable materials still relies on old-fashion practises. Currently, municipal offices responsible for the paper collection and management operations act as mediators between companies (i.e. large companies, shopping centres, supermarkets) and dedicated truck vehicles for transporting paper from the former to the materials recovery facilities (MRFs). Typically, offices rely on telephone communication: a) the company contacts the office and places a request, b) the municipal office checks the current truck schedules and finds an appropriate truck for the task c) the truck is immediately informed of the new task and adds it to its current schedule. Of course the procedure described is an ideal “manual” operation; usually, offices collect requests and construct the next-day trucks’ schedule.

There are quite a few problems in this procedure, mainly due to the centralised approach followed, that imposes a serious bottleneck to the system. One can easily see that:

– Efficient resource allocation (trucks to tasks) cannot be easily achieved, since the municipal office is unaware of the current status or location of the trucks.
– Communication is slow, resulting to a low number of handled requests.

– Trucks do not coordinate their actions and thus resources are underexploited in most of the cases.
– Finally, customers have limited information on the progress of their request and this has an impact on the adoption of recycling by a wider community of professionals.

The approach proposed in this paper attempts to resolve most of the above problems. Imposing a distributed cooperation model between interested parties alleviates the need for a central coordination office, allows trucks to form their daily schedule dynamically based on their current state (e.g. load, distance from client, etc.), and increases the number of requests serviced. Finally, since the system's status is available, clients have access to information regarding the progress of their request.

3 EcoTruck Agents

A natural modelling of the problem is to map each interested party in the process to an agent. Thus, EcoTruck is conceived as a multi-agent system that consists of two types of agents: *Customer Agents*, each representing a company in the system and *Truck Agents*, each representing a truck, responsible to manage the latter's everyday schedule.

The *Customer agent* has the goal to satisfy "its" user's recycling request by allocating the latter to an available truck. Thus, a customer agent after receiving input, i.e. the paper amount the company wishes to recycle, initiates a cooperation protocol to find the best possible truck to handle the request. The best truck in this case is the one that can service the request in the *shortest attainable amount of time*. Furthermore, the agent is responsible for monitoring the entire progress and provide a user friendly display, as well as take action in the light of possible failures.

A *Truck agent* is modelled as a software agent, mounted on truck vehicles that operates as an assistant to the driver. The agent has the goal of collecting as much recyclable material as possible, thus tries to maximise the number of Customer agent requests it can service. Each incoming request is evaluated based on the current available capacity of the truck, its geographical position and the tasks (paper collections) it has already committed to, i.e. its plan. Once a new collection task is assigned to the agent, it is added to the truck's plan. Each Truck agent maintains its own plan, that is a queue of jobs it intends to execute. Additionally, the truck agent proactively adds a "paper unloading" job to the plan upon detecting current low capacity. Finally, by processing real-time routing information, the agent "guides" the driver inside the city, much like a GPS navigation system, reducing travel distances and response time.

The operation of the system is *cooperative*, in the sense that all involved parties work together in order to fulfil the overall goal of increasing the amount of recyclable material collected during a working day. Consequently, the interaction protocol that was selected for EcoTruck system was the Contract-Net protocol, as discussed in the subsection that follows.

3.1 Agent Cooperation

The Contract-Net protocol (CNP) [2, 3], is probably the most well-known and mostly implemented task sharing protocol in distributed problem solving. It offers an elegant yet simple way of task sharing within a group of agents and we considered it to be suitable for the EcoTruck system. In the latter, Customer agents play the role of *managers* while Truck agents are *contractors* according to the protocol's terminology. Thus, the overall process is the following:

1. A Customer agent initiates a CNP protocol by announcing the task to truck agents. This "call for proposals" (CFP) contains information regarding the geographical location of company and the paper quantity for collection. The latter plays the role of the *eligibility criterion* in the protocol.
2. Trucks evaluate the CFP and decide on their eligibility for the task. If they they can carry the paper load, they answer with a *bid* that contains the *estimated time of servicing* (ETS) the request; the latter is the sole criterion on which Customer agents decide on who to assign the task. Naturally, Trucks can also refuse a request, if they cannot handle it, are currently full or for any other reason, by issuing an appropriate message.
3. The Customer agent receives bids, processes them, decides on the best truck to assign the task, and broadcasts the corresponding messages (accept-proposal/refuse-proposal).
4. The *winner* truck adds the task to its current plan. Upon arriving at the designated location, it will collect the paper, mark the job as finished and inform the corresponding Customer agent.

Obviously [2], there is always the risk that the Customer agent receives no bids. This can occur when the paper quantity in the CFP exceeds either a) the current capacity of any truck in the system, or b) the maximum capacity of even the largest truck. It was decided that both these cases, although different in nature, are going to be treated uniformly: the Customer Agent decides to decompose the original request to two smaller ones of equal size and initiate the protocol on each one once more. While the decision is obvious in the case that the CFP quantity exceeds the maximum capacity of all trucks, it requires some explanation in case (a). Since for trucks to regain their capacity they need to unload their cargo in Material Recovery Facility (MRF), a rather time consuming process, it was considered better in terms of service time reduction to allow the decomposition of the task, instead of the Customer agent waiting for some truck with the appropriate capacity to appear in the community.

Another issue, that is also present in the original CNP specification, regards whether to allow contractors to participate in multiple call for proposals [4], [5]. Since, in the EcoTruck system such cases would naturally occur (multiple companies exist that might have simultaneous requests), it was decided that when a "winner" truck is awarded a contract (accept-proposal) then it will check it against its current schedule (note that the latter can be different than the one the agent had during the bidding phase). If there are no differences in the ETS then it simply adds it to the schedule; if there are then it sends back a *failure* message to the customer agent and the latter re-initiates a CNP protocol once more. Although, this approach is not very sophisticated like the one described in [4], and certainly leaves plenty of room for improvements, it was considered to be adequate for the specific case.

4 Implementing EcoTruck

4.1 Platform of Choice

Erlang [6], is a concurrent declarative language aiming at the development of large distributed systems with soft real-time constraints [7]. The language offers dynamic typing, a single assignment variable binding scheme and automatic garbage collection so as to facilitate program development. Support for concurrency is achieved through process based programming and asynchronous message passing communication. Erlang follows the now popular "execution through virtual machine" model and a distributed system is composed of a number of different Erlang nodes running in one or more computers. Each node being a unique instance of a virtual machine.

A useful extension to the standard Erlang language is the Open Telecom Platform (OTP), that is a set of libraries and tools to facilitate the design and development of large distributed systems. OTP includes a set of the most common design patterns (*OTP behaviours*) that occur in a distributed setting.

EcoTruck employs *Server behaviour* processes, for asynchronous client-server communication, *FSM behaviour* processes, for interactions that require state transitions, and *Supervisor behaviour* processes, that monitor other OTP processes and restart them if they crash. These processes are nicely packaged into three distinct applications, using another OTP behaviour, called *Application behaviour*. Thus, our agents in Erlang consist of a number of processes, each being an instance of a specific OTP behaviour. This approach greatly facilitated the design and implementation of the EcoTruck system, since it provides all the tools necessary to built the agents.

4.2 Directory Facilitator

Since our implementation did not rely on any agent specific platforms, in order to provide directory services for the agent community, a *Directory Facilitator* (DF) was introduced in the system. This entity stores the roles and addresses of every active agent.

EcoTruck agents subscribe to the system by passing relevant information to the DF; the DF saves this information in an Erlang Term Storage (ETS) table, i.e. a fast in-memory data-store, part of the standard Erlang distribution. The DF is implemented as a server OTP behaviour, that monitors all subscribed agent processes and if any of them becomes unresponsive, due to network failure or abnormal termination, automatically unsubscribes them from the system. The server process itself is monitored by a supervisor behaviour.

4.3 Agent Structure

A set of different Erlang processes constitute the customer agent. The *customer Server*, an OTP Server instance monitored by a *customer Supervisor*, is the master process of the agent, that stores agent's preferences and controls the user interface. When the user places a request, the server spawns a new *Manager process* and passes the relevant information.

The *customer Manager* process will acquire the list of truck agents from the DF and initiate a CNP interaction as described above. The latter is implemented as an OTP Finite State Machine (FSM) behaviour, for the reason that CNP requires successive steps of interaction. These communication steps are essentially the transition states of the Finite State Machine. Finally, the Manager will link itself with the "winner" truck and specifically with its Contractor process. This link can be perceived as a "bidirectional monitor"; if one of the linked processes exits abnormally, the other process becomes aware of it.This ensures that if a winner Contractor crashes, the Manager will notice it and start a new CNP interaction. When the requested job is completed, the Manager reaches its final state and gracefully dies.

A Manager process also gracefully exits in the case of a request decomposition: it spawns two new Manager processes, passing to each one a request divided in two, and terminates its execution. Compared to the customer Server which lives as long as the application is running, the Manager's lifetime span covers only the period of time from request announcement to collection. The described behaviours together with a GUI are packaged into an OTP application.

The truck agent is also composed of a number of supervised processes; the *truck Server*, that is responsible for maintaining the plan (list of jobs) of the agent. For every incoming CFP request, the Server will spawn a new *Contractor process* (FSM behaviour) and pass to it the request. An eligible Contractor places a bid and in the case that it is the "winning" bidder, it will instruct the truck Server to schedule the job for execution by appending it to the plan. Upon completing the job, the Contractor process will notify the customer Manager process and terminate.

A conflict that commonly arises in a multi-CFP setting is when two or more Contractors try to place a recycling job in the exact same place in the plan queue. EcoTruck resolves this, by setting the truck Server to accept only one of the Contractors and its job, while rejecting the rest. The rejected Contractors will signal a failure to the corresponding Manager, who will in turn restart a CNP interaction.

Finally, the *truck driver process* is an extra Server, which simulates the motion of the truck vehicle inside the city roads. Of course, this is not necessary in a production environment. These three behaviours constitute a truck application.

It should be noted that since Erlang follows the virtual machine model, the applications developed can execute both in desktop and mobile environments. This is particularly interesting in the case of the truck application, since it allows easily executing it on a mobile device mounted on truck vehicles. Figure 1 presents the agent processes and message exchange of EcoTruck agents.

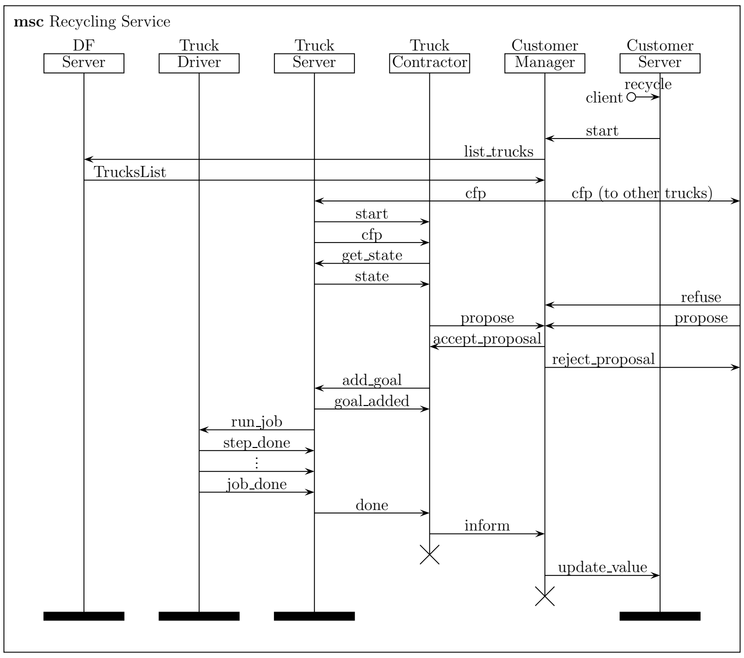

Fig. 1. Agent processes and message exchange in EcoTruck

4.4 Google Maps Integration

The Truck application uses Google Maps Directions Service to provide the truck driver with routing information as well as live traffic data where applicable. Additionally, both Customer and Truck applications have a GMAPS based web monitoring tool. The service responses are parsed with the *xmerl* Erlang parsing library. Each application relies on Google Maps Javascript API to display to the user a live view of the system's state. The Google Maps JSON encoded content along with any HTML files are displayed to the user by Misultin, a lightweight Erlang web server. The reason for not having, instead, a native GUI, is that a web interface can be more portable, thus the EcoTruck software will run on any hardware an Erlang VM exists for. Figure 2 shows the GUI of EcoTruck.

5 Related Work

The MAS approach has been applied to recycling/environmental processes in various contexts. For instance, the EU funded project E-MULT [8] aimed at developing a dynamic network of SMEs based on multi-agent solutions for the recycling of end-of-life vehicles, a process that is reported to be particularly complex. In [9] a simulation of an animal waste management system between producers and potential consumers (farms) is presented that allows to test different management scenarios.

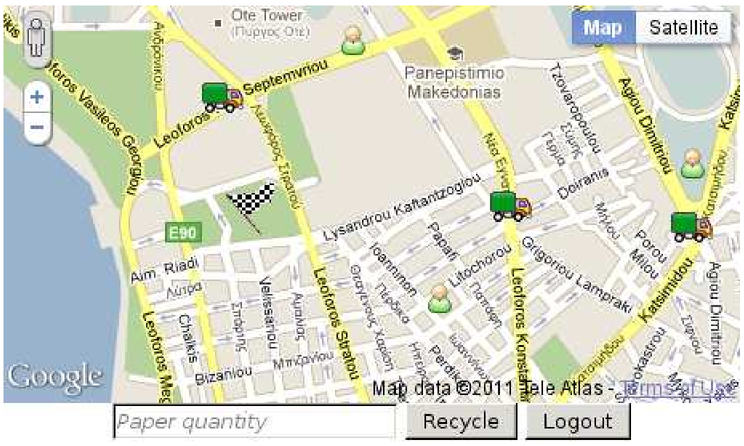

Fig. 2. Web Interface of EcoTruck Application

In [10] authors employ multi-agent simulation coupled with a GIS to assist decision making in solid waste collection in urban areas. Although the approach concerned truck routing, it was in a completely different context than the one described in the present paper: the former aimed at validating alternative scenarios by multi-agent modelling in the waste collection process, while EcoTruck aims at dynamically creating truck paths based on real time user requests.

The problem of efficiently collecting recyclable material from customers is in essence a dynamic transportation problem, although simpler in the sense that the destination location is fixed. A number of approaches in the literature have attacked the same problem, such as [5], where an extended contract net protocol (ECNP) is used to solve a transportation logistics problem. The ECNP protocol introduces new steps in the original contract net (temporal grant/reject and definitive grant/reject) and allows to dynamically decompose large demands (contracts). In [11] authors propose a new protocol the "provisional agreement protocol", where agents under the protocol are committed to bids sent only when they are (provisionally) granted the award. Thus, agents are allowed to participate simultaneously in multiple task announcements. The PAP protocol has been applied together with open distributed planning to the simulation of a real logistics data of a European company. Since the problem EcoTruck is dealing with is less complex than the dynamic transportation problem, EcoTruck follows a simpler approach: it allows agents to place multiple bids and allows multiple task announcements for an unserviced request if the bidder has already committed to another "contract" by the time it receives the award.

Finally, the Erlang language has been used to implement software agents with quite promising results. In [12] authors proposed the agent platform eXAT and support that it offers a number of benefits compared to the well-known JADE [13] platform, such as robustness and fault tolerance, more natural integration of the necessary components to implement such systems (FSM support, production rules components, etc). Although eXAT was not used in the implementation of the EcoTruck agents, we tend to agree with the arguments stated in [14] for the suitability of the Erlang language compared to JAVA based platforms.

6 Conclusions and Future Work

EcoTruck is a multi-agent system for the management of recyclable paper collection process. We believe that the specific approach can greatly facilitate and optimise the process, thus allow its wider adoption by the parties involved.

As discussed, the system's implementation is based on concurrency and distribution mechanisms of the Erlang language. We strongly believe that robustness and fault-tolerance are important qualities that a multi-agent system should meet. Although, the Erlang language is not a MAS platform it appears to have the necessary features to facilitate simple and elegant implementations of multi-agent applications.

There are quite a few features that could be incorporated in the current system among which the most interesting ones include:

– *Dynamic re-planning and scheduling.* In the present system, each truck agent maintains its own plan that has a static ordering of jobs, to which new jobs are inserted in a FIFO manner. A more intelligent planning process would include more dynamic features, such as prioritisation of jobs based on a number of criteria such as proximity to the current position and estimated time of arrival, and could help minimise the total travel path of the truck.
– *Smarter Truck Positioning.* Based on past data, truck agents can identify geographic areas where most requests appear in, place themselves closer to those areas and consequently increase system performance and success rate.
– *Better Decomposition of CFP's.* A more informed manner of splitting large requests could involve taking advantage of information attached in "refuse" messages of the CNP, and decomposing them in smaller ones of appropriate size, based on the current truck availability. Thus, the overall system's performance would increase, since fewer agent interactions would occur.

There are also a few improvements that could be done on the implementation level. For instance, the Erlang Term Storage (ETS), can be replaced by a Mnesia database, allowing exploitation of the fault-tolerance and distribution advantages of the latter. This change would allow to have multiple replicated instances of the DF database, and thus achieve robustness through redundancy.

Deployment of the application in a real-world environment, would lead to fine-tuning the system and examine possible patterns and procedures emerging in real-life situations. Since the system is based on a extensively tested, industrial strength platform (Erlang), we believe that the transition to a full-fledged real-world application can be accomplished with relative ease.

Finally, we should note that an earlier version of EcoTruck with a different structure and implementation by Nikolaos Bezirgiannis and Katerina Sidiropoulou, won the 1st prize on the Panhellenic Student Contest of Innovative Software (Xirafia.gr).

References

1. Jennings, N., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Journal of Autonomous Agents and Multi-Agent Systems **1** (1998) 275–306
2. Smith, R.: The contract net protocol: High-level communication and control in a distributed problem solver. Computers, IEEE Transactions on **C-29** (1980) 1104 –1113
3. Smith, R.G., Davis, R.: Frameworks for cooperation in distributed problem solving. IEEE Transactions on Systems, Man, and Cybernetics **11** (1981) 61–70
4. Aknine, S., Pinson, S., Shakun, M.F.: An extended multi-agent negotiation protocol. Autonomous Agents and Multi-Agent Systems **8** (2004) 5–45
5. Fischer, K., Mller, J.P., Pischel, M.: Cooperative transportation scheduling: an application domain for dai. Journal of Applied Artificial Intelligence **10** (1995) 1–33
6. Armstrong, J.: Programming Erlang: Software for a Concurrent World. Pragmatic Bookshelf (2007)
7. Armstrong, J.: The development of Erlang. In: Proceedings of the second ACM SIGPLAN international conference on Functional programming. ICFP '97, New York, NY, USA, ACM (1997) 196–203
8. Kovacs, G., Haidegger, G.: Agent-based solutions to support car recycling. In: Mechatronics, 2006 IEEE International Conference on. (2006) 282 –287
9. Courdier, R., Guerrin, F., Andriamasinoro, F., Paillat, J.M.: Agent-based simulation of complex systems: Application to collective management of animal wastes. Journal of Artificial Societies and Social Simulation **5** (2002)
10. Karadimas, N.V., Rigopoulos, G., Bardis, N.: Coupling multiagent simulation and GIS - an application in waste management. WSEAS TRANSACTIONS on SYSTEMS **5** (2006) 2367 2371
11. Perugini, D., Lambert, D., Sterling, L., Pearce, A.: Provisional agreement protocol for global transportation scheduling. In Calisti, M., Walliser, M., Brantschen, S., Herbstritt, M., Klgl, F., Bazzan, A., Ossowski, S., eds.: Applications of Agent Technology in Traffic and Transportation. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhuser Basel (2005) 17–32
12. Stefano, A.D., Santoro, C.: eXAT: An experimental tool for programming multi-agent systems in Erlang. In: IN AI*IA/TABOO JOINT WORKSHOP ON OBJECTS AND AGENTS (WOA 2003), VILLASIMIUS. (2003)
13. Di Stefano, A., Santoro, C.: Designing collaborative agents with eXAT. In: Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004. 13th IEEE International Workshops on. (2004) 15 – 20
14. Varela, C., Abalde, C., Castro, L., Gulías, J.: On modelling agent systems with Erlang. In: Proceedings of the 2004 ACM SIGPLAN workshop on Erlang. ERLANG '04, New York, NY, USA, ACM (2004) 65–70