Towards robust multi-agent systems: Handling communication exceptions in double auctions

Simon Parsons
Dept of Computer and Information Science
Brooklyn College, City University of New York,
Brooklyn, NY 11210, USA.
parsons@sci.brooklyn.cuny.edu

Mark Klein
Center for Coordination Science
Massachusetts Institute of Technology
Cambridge, MA 02142, USA
m_klein@mit.edu

Abstract

This paper addresses an important question in the development of multi-agent systems—how can we create robust systems out of the often unreliable agents and infrastructures we can expect to find in an open systems' context? Here we examine an approach based on distinct exception handling services, and apply it to systems performing resource allocation by means of a double auction.

1. Introduction

This paper studies the following question. "How can we develop robust multi-agent systems from the kind of unreliable agents and infrastructures—whether buggy, malicious, or just dumb—we can expect to have to deal with in the context of open systems. This is an increasingly important question because of the emerging changes in the way that human organizations work.

One result of globalization, coupled with the increasing power and ubiquity of cheap telecommunications, is that organizations are under increasing pressure to re-configure within short time-frames. This can have the effect of bringing together partners who have never worked together before, and force these partners to make their infrastructure inter-operate in ways that it was never designed to. One way to deal with the challenge of enabling this interoperation is to build the infrastructure as a multi-agent system, and benefit from the ability of such systems to dynamically self-organize as their tasks and constituents change [3]. However, a critical problem remains.

Much of the work in multi-agent systems has considered *closed* systems in which well-behaved agents have run on reliable infrastructures in relatively simple domains [2]. Both agents and infrastructure have been developed for a specific multi-agent system, and have been engineered to work together. These assumptions do not hold for the *open* systems described above. For open contexts, we can expect to have to deal with unreliable infrastructure, non-compliant agents and emergent dysfunctions (for more discussion of these topics, see [4] and the full paper). These problems all give rise to *exceptions*, situations which fall outside the normal operating conditions of the multi-agent system.

2. Exception handling

One way to deal with exceptions is to elaborate the individual agents so that they are able to cope with all the exceptions that they might face. Most previous research on dealing with exceptions has taken this approach. This *survivalist* approach to exception handling faces a number of serious shortcomings.

First, developing survivalist agents greatly increases the burden on agent developers. For this to be an effective approach, all the agents have to include carefully coordinated and provided with potentially complex mechanisms for exception handling. Second, the survivalist approach can lead to poor exception handling. In open systems it is always possible that some agents won't have the necessary exception handling code, or may violate some of the assumptions built into the exception handling operated by others.

In order to overcome these limitations Klein *et al.* [4] suggested attaining robustness by off-loading exception handling to distinct domain-independent services. We refer to this as the *citizen* approach, by analogy with the way that exceptions are handled in human society. The key insight in the citizen approach is that highly reusable and *domain independent* exception handling expertise can be separated from the knowledge that agents use to achieve their main tasks.

Previous work on the citizen approach has found that every coordination protocol has its own set of domain-independent exceptions, and that these can be turned into domain-independent strategies for handling exceptions [4].

This paper extends this earlier work to a new set of coordination protocols—auction protocols—identifying a new set of exceptions and exception handling mechanisms. Due to the popularity of auctions in the agents community, we believe that these results will be interesting to a large number of agent developers.

3. Exception handling in double auctions

Double auctions are markets that include both buyers and sellers. A classic example of a double auction was the trading pit at the old Chicago Board of Trade. Here buyers and sellers, or rather human agents operating on their behalf, would call out offers, *bids*—offers to buy a good at a given price—or *asks*—offers to sell a good at a given price. Although such markets have long since become electronic, the same basic principles apply with buyers and sellers "gathering" in a virtual space in which bids and asks are broadcast. When a bid is greater than an ask, a trade is possible, and a price between the *bid price* and the *ask price* is decided on as the *trade price*. This is a *continuous* double auction in which a trade is possible after every offer, another *periodic* variant of the double auction collects bids and asks until some deadline and then finds possible trades [1].

To provide a citizen approach to exception handling we associate a *sentinel* with every agent through which all messages to and from an agent pass. These sentinels can then provide exception handling services. For example, sentinels can handle corrupted messages. Assuming that corruption is introduced stochastically on the link between the sentinels, a sentinel can identify a corrupted message and organise for it to be resent by the sentinel of the issuing agent.

The advantage of the citizen approach is that the mechanisms for detecting and resolving the exceptions, the exception *handlers*, are generic. Exactly the same mechanisms can be used for other classes of auction since (as described elsewhere [5]) the specific exceptions that are detected and resolved by for a double auction may be found across all kinds of auction, and so may be handled by the same mechanisms. Indeed, these kinds of exception—exceptions due to message delay, loss and corruption—will be common to all coordination mechanisms operating over unreliable infrastructures, and potentially the same handlers can be used for a wide range of multi-agent systems.

4. Results

The main contribution of the work reported here is that we have implemented and empirically evaluated the kind of exception handling mechanism described above in the context of a continuous double auction. The results are explored at length in the full paper—here we briefly sketch them.

We measured the impact of (1) message corruption and (2) the loss of messages telling agents to start bidding on the allocative efficiency[1] of, and the number of messages passed during, a small double auction with 10 buyers and 10 sellers. We performed this test for agents using a range of bidding mechanisms common in the literature, and found that (unsurprisingly) for all types the exceptions caused a loss of efficiency. We then performed the same test when suitable exception handlers were switched on, and found that most of the lost efficiency was restored at the cost of only a modest increase in the number of messages. This seems to indicate that our approach to exception handling is effective in a double auction setting.

5. Conclusions

This paper studied the question "How can we develop robust multi-agent systems from unreliable components?", and proposed the use of domain-independent exception handling services as a solution. In the context of multi-agent systems that implement double auctions, we showed empirically that the particular exception handling approach described here is able to provide this robustness.

Acknowledgments

This work was partially supported by NSF IIS-0329037, and by HP under the "Always on" grant.

References

[1] D. Friedman. The double auction institution: A survey. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 1, pages 3–25. Perseus Publishing, Cambridge, MA, 1993.

[2] S. Hägg. A sentinel approach to fault handling in multi-agent systems. In *Proceedings of the Second Australian Workshop on Distributed AI*, Cairns, Australia, 1996. Workshop held in conjunction with the Fourth Pacific Rim Conference on Artificial Intelligence.

[3] N. R. Jennings, K. P. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–36, 1998.

[4] M. Klein, J. A. Rodriguez-Aguilar, and C. Dellarocas. Using domain-independent exception handling services to enable robust open multi-agent systems: The case of agent death. *Journal of Autonomous Agents and Multi-Agent Systems*, 7(1/2), 2003.

[5] S. Parsons. Exception analysis for double auctions. Working paper, MIT Center for Coordination Science, Cambridge, MA, May 2002.

1 Allocative efficiency is a standard economic measure which estimates how much profit the auction extracts from agents.