Resilience for Autonomous Agents

Miranda Mowbray, Matthew M. Williamson
Internet Systems and Storage Laboratory
HP Laboratories Bristol
HPL-2003-210
October 17th , 2003*

E-mail: miranda.mowbray@hp.com, matthew.williamson@hp.com,

autonomous agents, resilience

In this paper we show how the resilience approach can give a generic solution to the problems of looping and high-bandwidth output in autonomous agents. A resilient approach to looping is for the agent to delay responding again to a source that has recently triggered a task. A resilient approach to high-bandwidth output is for the agent to delay output when the overall "noise" level in the environment is high. The conditions under which the delays are triggered may be determined by data on past system behaviour. Our generic approach allows agents to limit themselves, without requiring them to perform semantic analyses.

* Internal Accession Date Only
Approved for External Publication

Resilience for Autonomous Agents

Miranda Mowbray and Matthew Williamson

Hewlett-Packard Laboratories Bristol
Filton Road, Stoke Gifford, Bristol, BS34 8QZ, UK
miranda.mowbray@hp.com, matthew.williamson@hp.com

Abstract. In this paper we show how the resilience approach can give a generic solution to the problems of looping and high-bandwidth output in autonomous agents. A resilient approach to looping is for the agent to delay responding again to a source that has recently triggered a task. A resilient approach to high-bandwith output is for the agent to delay output when the overall "noise" level in the environment is high. The conditions under which the delays are triggered may be determined by data on past system behaviour. Our generic approach allows agents to limit themselves, without requiring them to perform semantic analyses.

1 Why We Need Resilient Agents

There is a general trend for computer systems to become larger, more complex, more heterogeneous, and more dynamically varying. As a result, central human control for many systems is becoming difficult or impossible.

There is a need to perform some operations performed automatically and autonomously, in a distributed fashion. Agents, or pieces of software capable of some autonomous behaviour, are already used for tasks such as searching for online information, monitoring systems, and e-shopping. (To see examples of these, visit agentland.com or botspot.com, and see [1] for an overview.) In HP's Utility Data Center, OpenView agents perform low-level tasks such as monitoring memory usage or trapping events [2].

For very large and complex systems, security and reliability are particularly important and problematic. It will not be possible to prevent all faults (whether arising from security oopholes or from failure of software or hardware), and it will be difficult to mobilize a human response to faults cheaply and easily.

The presence of agents can cause or amplify certain faults, and agents can be compromised by malicious users [4].

In summary, future systems are likely to use agents, and will therefore need ways to address problems caused by these agents without relying on swift human intervention.

2 Resilience – a General Approach

Traditionally, overall approaches to security and reliability have either sought to prevent security lapses and faults happening in the first place, and/or to mend them through a human response when they do happen.

It is generally impossible to prevent all faults, and prevention is expensive. Mending faults when they occur, on the other hand, tends to be slow, since human response time is slow compared with machine speed. Some types of fault can do considerable damage if not addressed quickly. At the height of the Code Red virus attack, Cisco's intrusion detection system was reporting 2.5 million events a day: a fast human response to each of these events would have been infeasible.

Our philosophy for reliability of complex systems, as outlined in [3], is to directly tackle the problem of damage cause by faults before a human response. The approach is to build resilient infrastructure that can hamper, mitigate and contain problems, so buying time for a human response. Since containing damage is simpler than finding the fault and deciding how to fix it, it makes sense to use automatic computer responses to contain problems, and humans to sort them out.

Creating resilient systems is original as a guiding philosophy, although there are many individual instances of resilient behaviour. This approach can be applied to a very broad range of applications (not just to agents). Matthew Williamson's *virus throttling*, which applies this approach, drastically slows the spread of computer viruses without noticeably affecting non-infected machines [3]. In some cases an automatic reaction might be enough to heal the fault entirely: see [7] for a discussion.

In this paper we discuss applying this approach to two classes of problem behaviour by agents, looping and high-bandwidth output.

2.1 Resilient Approach to the Looping Problem

One problem described in [4] is that an agent may get into an unwanted loop with its environment or with another agent. Such loops can occur for example in email mailing lists if two list members have wrongly-configured auto-reply agents that answer ach other.

Following our philosophy, a generic way of addressing the looping problem for agents is to programme the agent to delay before responding again to a source (and to perform other tasks in the meantime), if the source has recently triggered a task.

This slows down the loop, but does not completely close it down, so that benign loops still run – albeit slowly – and meanwhile disruption to other sources is minimized. Of course an alarm could be triggered if there is a prolonged series of rapid requests from a single source.

This generic approach to loops can be refined for specific examples by categorizing requests into different types, and only delaying responses to a request from a source if there had recently been a request from the same source in the same category. For example, in the case of email loops, it is enough to delay email messages if another message was recently received from the same source to the same recipient with close to the same length ("close to" rather than "equal to" because bounce messages may contain time stamps, for example).

Another refinement is for the meaning of "recently" to vary per source and/or per request type, with the associated time values derived from data on normal system behaviour.

2.2 Resilient Approach to the High-Bandwidth Output Problem

A second problem is that in some circumstances agents may output so many messages so quickly to a user's screen that the screen scrolls to quickly to read.

One example of this is the agent Cobot in the LambdaMOO Multi-user social environment [5, 9, 4]. This agent could be coerced by a malicious user into outputting so many messages that it drowned out any other conversations in the nline space. Another example is the use of IRC (Internet Relay Chat) agents to flood users' screens with text, in order to discourage them from joining a particular channel [6]. High-bandwidth output is not only a problem when the output is to users' screens. It is also a problem if the output is sent to software that cannot process it fast enough.

One way of addressing high-bandwidth output is for an agent to monitor the overall "noise" level – and hence the likely number of messages currently being output in a given time interval – and to delay messages (or drop unimportant ones) when the overall noise level is high.

This approach is self-adjusting. Users or administrators do not have to reprogramme the agent to hold back; it does so automatically when the noise level is high, whether the high noise level is a result of its own behaviour or external factors. The agent does not need to determine the reason why the noise level is high before taking action. High noise levels resulting from system faults, local agent faults, malicious attacks, and temporary high noise levels due to statistical fluctuations in normal operation, are all treated the same way, at least initially.

Prolonged periods of high noise level could trigger a human-readable alarm – or possibly an alarm to a higher-level agent authorized to take more drastic action – and a suggestion that there may be a fault.

Just as for the looping problem it was possible to refine the generic approach by categorizing messages into classes, and only delaying messages in the classes causing the problem, it is also possible to refine the generic approach to the high-bandwidth out problem in the same way. Mahajan et al [10] describe how this could be done for routers. Reasonable bandwidth will vary for different sources, and these numbers may be derived from historical data.

The resilient behaviour addressing the high-bandwidth output problem is one example of a class of resilient behaviour with more general applicability, in which the agent monitors the use of a scarce resource, and, when the resource is low, holds back on actions that would further deplete the resource and that can be delayed or cancelled without causing harm. In the examples mentioned above, the scarce resource is the space on a user's screen or in a queue to be processed by a piece of software. This resource is not monitored directly, but the agent estimates indirectly that the resource may be running low by measuring the noise level.

3 Other Generic Approaches to Reliability of Agents

Most work on the safety and reliability of agents has been *ad hoc*, trying to improve a specific agent.

Of the few more systematic approaches, some have been concerned with the standardization of agent languages (so as to avoid errors caused by incomprehension between agents or between an agent and its environment) and with limiting to a certain extent the actions that an agent can take: however there is ample scope within these limits for agents to cause problems.

Allen et al [8] consider designing agents which reason about the effects of their behaviour, or which examine their rules to see whether they will lead to problems. This approach requires the agent to have quite sophisticated abilities of semantic analysis and of knowledge about its environment, and, as [8] points out, may be computationally intractable.

A final approach taken by (among others) the programmers of Cobot [9] is to allow users to limit or stop the behaviour of an agent with which they are interacting.

Our approach, on the other hand, allows agents to limit themselves, without requiring them to perform semantic analyses.

We are currently applying these ideas to design an agent for a specific task that will be resilient to loops and high-bandwidth output, and will use past data to specify normal behaviour.

References

1. Feldman, S., Yu, E.: Intelligent Agents: A Primer. Searcher **7**:9 (1999). `http://www.infotoday.com/searcher/oct99/feldman+yu.htm`
2. Hewlett Packard: Press Release. Redefining the Evolution of the Data Center. November 20 2001. `http://www.hp.com/hpinfo/newsroom/feature_stories/datacenter-1.htm`
3. Williamson, M.: Resilient Infrastructure for Network Security. Complexity, in press. 2003. `http://www.hpl.hp.com/techreports/2002/HPL-2002-273.html`
4. Mowbray, M.: Ethics for Bots. In: Smit, I., Lasker, G.E. (eds.): Cognitive Emotive and Ethical Apects of Decision Making and Human Action. International Institute for Advanced Studies in Systems Research and Cybernetics (2002) 24–28
5. LambdaMOO online multi-user social space. *lambda.moo.mud.org:8888*
6. Meinel, C.P.: Flooding Attacks. Online Security article (1997) `http://onlinesecurity.virtualave.net/hacking/irc/flooding.htm`
7. Short, M.: Are you ready for enterprise systems that fix themselves? DevX enterprise art. **9891** (2002) `http://www.devx.com/enterprise/Article/9891`
8. Allen, C., Varner, G., Zinser, J.: Prolegomena to any future artificial moral agent. Journal of Experimental and Theoretical Artificial Intelligence **12** (1999) 251–261
9. Isbell, Jr, C.L., Kearns, M., Kormann, D., Singh, S., Stone, P.: Cobot in LambdaMOO: A Social Statistics Agent. In: Proc. AAAI 2000. AAAI Press/The MIT Press (2000) 36–41 http://cobot.research.att.com/papers/cobot.pdf
10. Mahajan, R., Bellovin, S.M., Floyd, S., Ioannidis, J., Paxson, V., Shenker, S.: Controlling High Bandwidth Aggregates in the Network. Computer Communications Review **32**:3 (2002) 62–73 `http://www.research.att.com/~smb/papers/pushback-CCR.pdf`