MIT Sloan School of Management

Handling Resource Oscillations Through Selective Misinformation

Mark Klein, Richard Metzler, Yaneer Bar-Yam

This paper also can be downloaded without charge from the
Social Science Research Network Electronic Paper Collection:
http://ssrn.com/abstract=1139893

Handling Resource Oscillations Through Selective Misinformation

Mark Klein
Massachusetts Institute of Technology
m_klein@mit.edu

Richard Metzler
New England Complex Systems Institute
richard@necsi.org

Yaneer Bar-Yam
New England Complex Systems Institute
yaneer@necsi.org

When resource consumers select among competing providers based on delayed information, inefficient oscillations in resource utilization can emerge. This paper describes an approach, based on selective stochastic resource request rejection, for dealing with this emergent dysfunction.

1. The Challenge

The convergence of ubiquitous electronic communications such as the Internet, electronic agents acting as proxies for human consumers, and web/grid service standards such as XML are rapidly ushering in a world where hordes of software agents, acting for humans, can rapidly select among multitudes of competing

providers offering almost every imaginable service. This is inherently an "open" world, a marketplace where the agents operate as peers, neither designed nor operated under central control. Such a world offers the potential for unprecedented speed and efficiency in getting work done.

In such open peer-to-peer systems we face, however, the potential of highly dysfunctional dynamics emerging as the result of many locally reasonable agent decisions [1]. Such "emergent dysfunctions" can take many forms, ranging from inefficient resource allocation [2] to chaotic inventory fluctuations [3] [4]. This problem is exacerbated by the fact that agent societies operate in a realm whose communication and computational costs and capabilities are radically different from those in human society, leading to collective behaviors with which we may have little previous experience. It has been argued, for example, that the 1987 stock crash was due in part to the action of computer-based "program traders" that were able to execute trade decisions at unprecedented speed and volume, leading to unprecedented stock market volatility [5].

Let us focus on one specific example of emergent dysfunctional behavior: resource use oscillation in request-based resource sharing. Imagine that we have a collection of consumer agents faced with a range of competing providers for a given resource (e.g. a piece of information such as a weather report, a sensor or effector, a communication link, a storage or computational capability, or some kind of data analysis). Typically, though not exclusively, the utility offered by a resource is inversely related to how many consumers are using it. Each agent strives to select the resource with the highest utility (e.g. response time or quality), and resources are allocated first-come first-served to those who request them. This is a peer-to-peer mechanism: there is no one 'in charge'. This kind of resource allocation is widely used in settings that include fixed-price markets, internet routing, and so on. It is simple to implement, makes minimal bandwidth requirements, and - in the absence of delays in resource status information – allows consumers to quickly converge to a near optimal distribution across resources (see figure 1 below).

Consumers, however, will often have a delayed picture of how busy each resource is. Agents could imaginably poll every resource before every request. This would cause, however, a N-fold increase in message traffic (for N servers), and does not eliminate the delays caused by the travel time for status messages. In a realistic open system context, moreover, consumers probably cannot fully rely on resource providers to accurately characterize the utility of their own offerings (in a way that is comparable, moreover, across providers). Resource providers may be self-interested and thus reluctant to release utilization information for fear of compromising their competitive advantage. In that case, agents will need to estimate resource utilization using other criteria such as their own previous experience, consulting reputation services, or watching what other consumers are doing. Such estimates are almost certain to lag at times behind the actual resource utility.

When status information is delayed in some way, we find that resource use oscillations emerge, potentially reducing the utility achieved by the consumer agents far below the optimal value predicted by an equilibrium analysis [6]. What happens is the following. Imagine for simplicity that we have just two equivalent resources, R1 and R2. We can expect that at some point one of the resources, say R1, will be

utilized less than the other due to the ebb and flow of demand. Consumer agents at that point will of course tend to select R1. The problem is that, since their image of resource utilization is delayed, they will continue to select R1 even after it is no longer the less utilized resource, leading to an "overshoot" in R1's utilization. When the agents finally realize that R2 is now the better choice, they will tend to select R2 with the same delay-induced overshoot. The net result is that the utilization of R1 and R2 will oscillate around the optimal equilibrium value. The range of the oscillations, moreover, increases with the delay, to the extent that all the consumers may at times select one server when the other is idle:
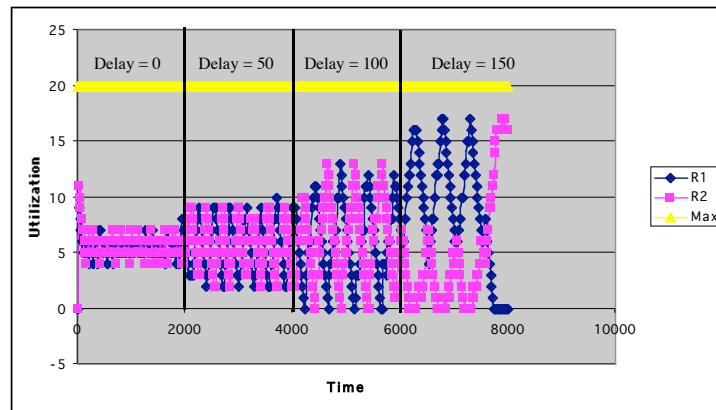

Figure 1: Utilization of two equivalent resources with and without information delays

Oscillations have several undesirable effects. One is that they can reduce the utility received by consumers below optimal values. This can occur, for example, when the oscillations are so severe that some of the resources go idle some of the time, reducing the effective resource availability. The other is that they can increase the variability of the utility achieved by the consumers, which may be significant in domains where consistency is important.

2. Our Approach: Stochastic Request Rejection

We have developed a technique for addressing delay-induced resource inspired by a scheme developed to improve the allocation of network router bandwidth [7]. We call our scheme 'stochastic request rejection', or SRR. Imagine that every resource stochastically rejects new requests with a probability proportional to its current load. This can be implemented by the resource itself, or by 'sentinel' agents that track the number of consumers each resource is currently serving, and stochastically intercept/reject consumer requests with a probability proportional to that load. When oscillations occur, we would predict that the increased level of rejections from the currently more heavily utilized resource will shift the requests to the less-utilized resource, thereby damping the oscillations and ameliorating their negative impact on the utility and consistency experienced by consumer agents.

3. Experimental Evaluation

Resource use oscillations will only occur if the utility of a resource to a consumer is a *negative* function of its utilization. If the utility of a resource to a consumer increases when the resource is more heavily utilized (e.g. imagine night-clubbers who want to select the club which has the most people there) then all consumers will eventually converge on a single resource. We can divide the remaining cases into two scenarios. One is the "grocery store scenario" where grocery store customers (consumers) choose from two checkout lines (resources). Their utility is inversely related to how long they have to wait, which is a *linear* function the length of a line. The second scenario is the "movie theater", where the utility of the resource is a *non-linear* function of how many people are already there (assuming that good seats are consumed before poorer ones). We did separate evaluations for these two scenarios.

The Grocery Checkout (Linear) Scenario: There were 20 consumers and 2 resources. Each consumer sends a 'request' message to the resource it believes has the smallest backlog, waits until it receives a 'job completed' message from the resource, and then after a randomized delay sends the next 'request' message. The consumers' estimate of a resources' utility may lag the correct value. Resources may either take on requests or reject them. If a consumer receives a 'reject' message, it sends the request to the other resource. Messages take 20 units of time to travel, resources require 20 units of time to perform each task, and consumers have a normally distributed delay at 40 ticks, with a standard deviation of 10, between receiving one result and submitting the next request. The aggregate results reported below represent averages over 100 simulation runs, each 4000 ticks long, and all the conclusions we make were statistically significant at $p < 0.01$.

The impact of applying SRR in this scenario can be visualized as follows:
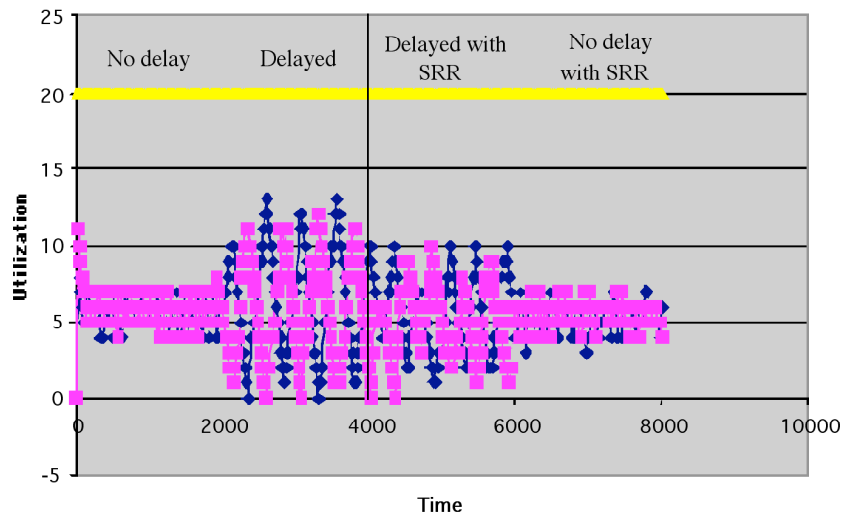

Figure 2: The impact of SRR on resource oscillations.

In this simulation run, the agents initially made their resource requests using current information on the length of each resources' backlog. As we can see, in this case the resource utilization clusters tightly around the optimal distribution of 50-50 across resources. At T = 2000, the backlog information provided to the consumers was made 100 time units out of date, rapidly leading to large resource use oscillations. At T = 4000, SRR was turned on, resulting in substantial damping in the magnitude of these oscillations. At T = 6000, the delay was removed but SRR was left on, whereupon the resource utilization returns to clustering tightly around the optimal distribution. The aggregate results confirm the patterns suggested by this example:

	Null	SRR
No delay	160 +/- 4 0%	160 +/- 6 33%
Short Delay (50)	160 +/- 7 0%	160 +/- 6 34%
Long Delay (100)	167 +/- 8 0%	161 +/- 6 35%

Table 1. Task completion times +/ 1 standard deviation, as well as reject rates, for different delays, with and without SRR.

As we would expect for the grocery scenario, the variability in task completion times without SRR increases with the delay in status information, and if the delay is long enough, the average task completion time can increase as well. If we turn on SRR, we find that it significantly reduces the variability in task completion times in the delayed cases, and almost eliminates the increase in task completion times in the long delay case. Rejecting some requests can thus, paradoxically, actually speed up task completion when delay-induced oscillations occur. But this does come at a cost. Message traffic is increased: roughly $1/3^{rd}$ of the consumer requests elicit a reject message and must be re-sent. The variability of task completion times in the no delay case is also increased by SRR. This is because many resource requests that would otherwise simply have been queued up incur the additional delay of being rejected and re-submitted. The absolute size of these effects, of course, can be expected to vary with the ratio of task and messaging times. Ideally, we would be able to enable SRR only when it is needed, so we can avoid incurring its costs in the no-oscillation contexts where it is not helpful. We will return to this point later.

The Movie Theater (Nonlinear) Scenario: The parameters for these simulations were the same as in the grocery store case, except for the following changes. Resources do not have a waiting line, but instead offer concurrent access to 15 different 'slots' with varying utility (the first slot has value 15, the second has value 14, and so on). Tasks take 160 ticks to perform. The aggregate results are as follows:

	Null	**SRR**
No delay	9.6 +/- 1.5 0% 331	9.7 +/- 1.2 59% 303
Short Delay (50)	9.1 +/- 1.9 0% 332	9.8 +/- 1.4 60% 303
Long Delay (100)	7.6 +/- 2.1 3% 331	9.6 +/- 1.4 66% 300

Table 2. Average quality +/- 1 standard deviation, as well as reject rates and number of completed requests, for different delays, with and without SRR.

As we can see, SRR is also effective in this scenario. Delay-induced oscillations cause consumers to often select the resource that is actually more heavily utilized and thus lower in quality, resulting in a reduction of the average achieved quality. Using SRR eliminates this problem, but with the cost of increasing message traffic, as well as reducing the rate of task completion (since every time a task is rejected a delay is incurred while the request is re-submitted). As in the "grocery checkout" case, we would ideally prefer to be able to apply SRR selectively, so we do not incur these costs when oscillations are not occurring. Can this be done?

4. Avoiding Needless Rejects Via Selective SRR

It is in fact straightforward to use spectral analysis to determine if persistent oscillations are occurring in resource utilization. In our implementation, each resource periodically (every 20 ticks) sampled its utilization and submitted the last 30 data points to a Fourier analysis. SRR was turned on if above-threshold values were encountered in the power spectrum so determined. The threshold was determined empirically. This approach proved to be successful. In the grocery checkout scenario, selective SRR was as effective as SRR in maintaining throughput and task duration consistency while avoiding increases in message traffic in the no-delay case:

	Null	**SRR**	**Selective SRR**
No delay	160 +/- 4 0%	160 +/- 6 33%	160 +/- 4 0%
Short Delay (50)	160 +/- 7 0%	160 +/- 6 34%	160 +/- 6 29%
Long Delay (100)	167 +/- 8 0%	161 +/- 6 35%	161 +/- 6 33%

Table 3. Task completion times +/ 1 standard deviation, as well as reject rates, for different delays, with and without [selective] SRR.

In the movie theatre scenario, selective SRR maintained task quality while almost eliminating increases in message traffic and task time in the no-delay case:

	Null	SRR	Selective SRR
No delay	9.6 +/- 1.5 0% 331	9.7 +/- 1.2 59% 303	9.5 +/- 1.4 6% 327
Short Delay (50)	9.1 +/- 1.9 0% 332	9.8 +/- 1.4 60% 303	9.6 +/- 1.5 41% 311
Long Delay (100)	7.6 +/- 2.1 3% 331	9.6 +/- 1.4 66% 300	9.3 +/- 1.6 54% 305

Table 4. Average quality +/- 1 standard deviation, as well as reject rates and number of completed requests, for different delays, with and without [selective] SRR.

This simple spectral analysis approach can be fooled, of course, into triggering SRR when resource use oscillations are due to variations in aggregate demand rather than status information delays. This problem, however, is easily addressed: whenever a resource detects significant usage oscillations, it analyzes the correlation of it's utilization with that of the other resource. Variations in aggregate demand will show a positive correlation, while delay-caused oscillations show a negative one. We have implemented this approach and found that it successfully avoids triggering SRR for aggregate demand variations while remaining effective in responding to delay-induced oscillations.

5. Contributions and Next Steps

The problem of resource use oscillation in request-based systems has been studied in some depth, most notably in the literature on "minority games" [8] [9]. This line of work has investigated how to design agents so that their local decisions no longer interact to produce substantial resource use oscillations. One example involves designing agents that make resource selection decisions using historical resource utilization values [6]. If the agents look an appropriate distance into the past, they will be looking at the resource state one oscillation back in time, which should be a good approximation of the current resource utilization. The agent's delay parameter is tuned using survival of the fittest: agents with a range of delay factors are created, and the ones that get the highest utility survive and reproduce, while others do not. With this in place the resource utilization, under some conditions, settles down to near-optimal values. Any such approach, however, predicated as it is on the careful design of agent resource selection strategies, faces a fundamental flaw in an open systems context. In open systems, we do not control the design or operation of the consumer agents and can not be assured that they will adopt strategies that avoid emergent dysfunctions. Our challenge, therefore, is to find an approach that moderates or eliminates oscillatory resource utilization dynamics without needing to control the design or operation of the consumer agents. This paper presents such an

approach, based on stochastic load-proportional rejection of resource requests, triggered selectively when spectral and cross-resource correlation analyses reveal that delay-induced oscillations are actually taking place.

Next steps for this work include evaluating the selective SRR approach when there are more than two resources. This research is part of the author's long-standing efforts to develop a systematic enumeration of the different multi-agent system exception types as well as how they can be addressed in open systems contexts [10] [11]. See http://cci.mit.edu/klein/ for further details.

Acknowledgements

Bibliography

[1] Jensen, D. and V. Lesser. *Social pathologies of adaptive agents*. in *Safe Learning Agents Workshop in the 2002 AAAI Spring Symposium*. 2002: AAAI Press.

[2] Chia, M.H., D.E. Neiman, and V.R. Lesser. *Poaching and distraction in asynchronous agent activities*. in *Proceedings of the Third International Conference on Multi-Agent Systems*. 1998. Paris, France.

[3] Youssefmir, M. and B. Huberman. *Resource contention in multi-agent systems*. in *First International Conference on Multi-Agent Systems (ICMAS-95)*. 1995. San Francisco, CA, USA: AAAI Press.

[4] Sterman, J.D., *Learning in and about complex systems*. 1994, Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts Institute of Technology. 51.

[5] Waldrop, M., *Computers amplify Black Monday.* Science, 1987. **238**: p. 602-604.

[6] Hogg, T., *Controlling chaos in distributed computational systems.* SMC'98 Conference Proceedings, 1998(98CH36218): p. 632-7.

[7] Braden, B., D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*. 1998, Network Working Group.

[8] Challet, D. and Y.-C. Zhang, *Emergence of Cooperation and Organization in an Evolutionary Game.* arXiv:adap-org/9708006, 1997. **2**(3).

[9] Zhang, Y.-C., *Modeling Market Mechanism with Evolutionary Games.* arXiv:cond-mat/9803308, 1998. **1**(25).

[10] Klein, M. and C. Dellarocas. *Exception Handling in Agent Systems*. in *Proceedings of the Third International Conference on AUTONOMOUS AGENTS (Agents '99)*. 1999. Seattle, Washington.

[11] Klein, M., 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.* Autonomous Agents and Multi-Agent Systems, 2003. **7**(1/2).