

Center for

**eBusiness@MIT**

<http://ebusiness.mit.edu>



A research and education initiative at the MIT Sloan School of Management

# Handling Resource Use Oscillation in Multi-Agent Markets

Paper 181

Mark Klein  
Yaneer Bar-Yam

July 2003

For more information,  
please visit our website at <http://ebusiness.mit.edu>  
or contact the Center directly at [ebusiness@mit.edu](mailto:ebusiness@mit.edu) or 617-253-7054



# Handling Resource Use Oscillation in Multi-Agent Markets

Mark Klein

Massachusetts Institute of Technology  
NE20-336  
Cambridge MA 02132  
+1 (617) 253-6796  
m\_klein@mit.edu

Yaneer Bar-Yam

New England Complex Systems Institute  
24 Mt. Auburn Street  
Cambridge, MA 02138  
+1 (617) 547-4100  
yaneer@necsi.org

## ABSTRACT

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.

## Keywords

Resource oscillations thrashing minority game

## 1. THE CHALLENGE

The convergence of ubiquitous electronic communications such as the Internet, software agents, and web/grid service standards such as XML is rapidly ushering in a world where hordes of agents, potentially 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.

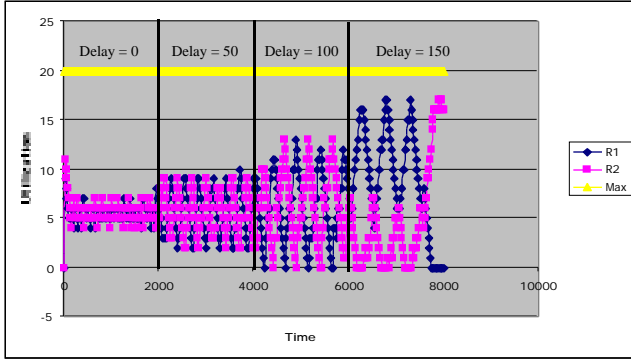
In such open markets we face, however, the potential of highly dysfunctional dynamics emerging as the result of many locally reasonable agent decisions [1]. Such “emergent dysfunctions” can take many forms, ranging from inefficient resource allocation [2] to chaotic inventory fluctuations [3] [4] to non-convergent collective decision processes [5]. 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 [6].

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).

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 number of required messages 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 [7]. 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 delays in status information.**

Oscillations have two undesirable effects. One is that they can reduce the utility received by consumers below optimal values. 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. The precise impact of oscillations is driven by the relationship between resource utilization and utility. Let us consider several scenarios to help make this more clear.

Imagine we have grocery store customers (consumers) choosing from two checkout lines (resources). Their utility is inversely related to how long they have to wait. The time needed to check out each customer is not affected by the length of a line, so the wait at each line is *negatively* and *linearly* impacted by its length. As long as both checkout lines remain busy, oscillations in the distribution of customers across lines do not affect average check out times. The longer waits faced by customers in one line are exactly offset by the shorter waits enjoyed by the customers in the other line. The *variance* of checkout times will, of course, be wider if there are oscillations. Note, also, that if the oscillations are so severe that one line goes idle while the other has customers, the average waiting times will increase because only one resource is active.

Now consider, by contrast, movie goers faced with a choice of theatres to attend. Each movie goer wants the best seat possible. The fuller the theatre, the poorer the seats that remain, so the utility of the seats is *negatively* and *non-linearly* impacted by the number of people using the theatre. There is one optimal distribution of movie goers across theatres (50-50 in this case), and oscillations in the actual distribution reduce the average utility because much of the time is spent at a non-optimal distribution. Oscillations increase the variance in this case as well.

A third scenarios involves TV viewers (consumers) with a choice of shows (resources) to watch. The utility of a show for a given consumer is *not* impacted by who else happens to be watching it. In this case we would expect no persistent oscillations to occur, because changing viewership does not impact the movement of viewers among shows. Any variations that do occur would, in any case, have no impact on the utility received by the consumers.

The last scenario involves club goers (consumers) with a choice of night clubs (resources) they can attend. The value of a club to them is given by how many other people are there. The resource utility is thus *positively* impacted by the number of consumers. In this scenario, there will be no persistent oscillations because

whatever club achieves an initial advantage in number of participants will eventually get all of the club goers.

We can therefore distinguish four scenarios for request-based resource sharing. All are potentially important in open agent system contexts. Network routers operate using the “Grocery Checkout” model. A broadcast operates using the “TV show” model. Auctions arguably fit into the “Nightclub” model. And providers of varying utility resources such as sensors, effectors, or storage capacity fit into the “Movie Theatre” model. These scenarios vary however in their susceptibility to the emergent dysfunction of resource use oscillation. The key factor concerns how the utility offered by a resource varies with its utilization:

**Table 1. The cases for how resource utility is affected by utilization.**

	<b>Linear Relationship</b>	<b>Non-linear Relationship</b>
<b>Positive Impact</b>	Nightclub	
<b>Zero Impact</b>	TV Show	
<b>Negative Impact</b>	Grocery Checkout	Movie Theatre

When resource utility is a constant or positive function of resource utilization, resource use oscillation will not occur and/or have no impact. In the remaining scenarios, however, oscillations can occur and negatively impact the consistency and/or average value of the utility achieved by consumers. We will confine our attention to these latter cases in the remainder of the paper.

## 2. PREVIOUS WORK

What can be done about resource use oscillation in request-based systems? This problem 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 [7]. 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 approach predicated on the careful design of agent resource selection strategies, however, 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.

### 3. OUR APPROACH

Our approach to this problem is inspired by a scheme developed to improve the allocation of network router bandwidth (the resource) to client computers (the consumers). Routers currently operate as follows. Clients send packets to routers, and routers then forward the packets on towards their final destination. Every router has a buffer where it stores the packets that arrive when the router is busy. If any packets arrive when the buffer is full, the routers send a ‘packet drop’ message to the originating clients, at which point they immediately drop their data send rate to some minimal value, and then gradually increase it. This scheme is prone to inefficient router use oscillations because clients can synchronize their data send rate changes. If several get a packet dropped message from the router at about the same time, they all slow down and potentially under-utilize the router until their data rates ramp up enough to overload it, at which point they all drop their rates again.

A scheme called “Random Early Detect” (RED) has been proposed to address this problem [10]. The idea is simple. Rather than dropping packets only when the buffer is full, the router drops them stochastically with a probability proportional to how full the buffer is (e.g. 50% full results in a 50% chance of a packet being dropped). RED is successful at damping oscillations in router utilization because it de-synchronizes the data send rate changes across clients. While it does increase client-side message traffic by increasing the number of reject messages, it has no negative impact on total throughput because it is very unlikely to cause the buffers to empty out and leave a router needlessly unutilized.

We have adapted this idea for the context of request-based resource sharing in open agent systems. We call our technique ‘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.

### 4. THE GROCERY CHECKOUT CASE

Our first set of tests studied the value of SRR when applied to “grocery checkout” scenario. The experimental setup was as follows. 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:

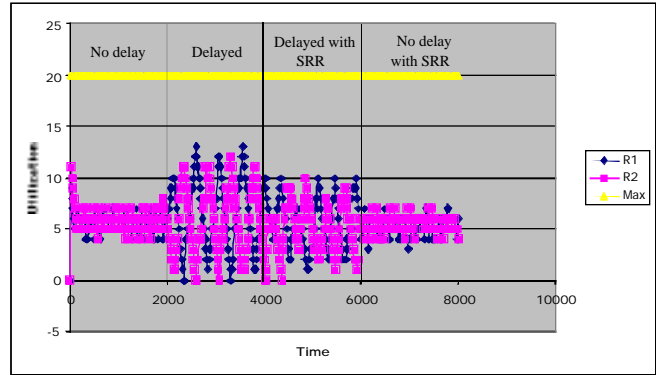


Figure 2. The impact of SRR on resource oscillations.

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:

Table 2. Task completion times +/- 1 standard deviation, as well as reject rates, for different delays, with and without SRR.

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

As we would expect for the grocery checkout 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<sup>rd</sup> 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.

## 5. THE MOVIE THEATRE CASE

The parameters for the movie theatre 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 for this case are as follows:

**Table 3. Average quality +/- 1 standard deviation, as well as reject rates and number of completed requests, for different delays, with and without SRR.**

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

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?

## 6. 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:

**Table 4. Task completion times +/- 1 standard deviation, as well as reject rates, for different delays, with and without [selective] SRR.**

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

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

**Table 5. Average quality +/- 1 standard deviation, as well as reject rates and number of completed requests, for different delays, with and without [selective] SRR.**

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

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 its 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.

## 7. CONTRIBUTIONS AND NEXT STEPS

We have presented a promising approach for mitigating the deleterious effects of delay-induced resource-use oscillations on request-based resource sharing. It differs from previous techniques in that it is designed to be appropriate for the important context of open agent systems, where we can not rely on being able to control the design or operation of the resource consumers. The key elements of this approach involve the 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

[11] [12]. See <http://ccs.mit.edu/klein/> for further details on this work.

## 8. ACKNOWLEDGEMENTS

This work was supported by the NSF Computational and Social Systems program as well as the DARPA Control of Agent-Based Systems program.

## 9. REFERENCES

1. Jensen, D. and V. Lesser. *Social pathologies of adaptive agents*. In the proceedings of *Safe Learning Agents Workshop in the 2002 AAI Spring Symposium*. 2002: AAAI Press.
2. Chia, M.H., D.E. Neiman, and V.R. Lesser. *Poaching and distraction in asynchronous agent activities*. In the proceedings of *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 the proceedings of *First International Conference on Multi-Agent Systems (ICMAS-95)*. 1995. San Francisco, CA, USA: AAAI Press.
4. Serman, J.D., *Learning in and about complex systems*. 1994, Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts Institute of Technology. 51.
5. Klein, M., H. Sayama, P. Faratin, and Y. Bar-Yam, *The Dynamics of Collaborative Design: Insights From Complex Systems and Negotiation Research*. Concurrent Engineering Research & Applications, 2003. **In press**.
6. Waldrop, M., *Computers amplify Black Monday*. Science, 1987. **238**: p. 602-604.
7. Hogg, T., *Controlling chaos in distributed computational systems*. SMC'98 Conference Proceedings, 1998(98CH36218): p. 632-7.
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. 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.
11. Klein, M. and C. Dellarocas. *Exception Handling in Agent Systems*. In the proceedings of *Proceedings of the Third International Conference on AUTONOMOUS AGENTS (Agents '99)*. 1999. Seattle, Washington.
12. 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).