Handling Emergent Resource Use Oscillations

Mark Klein, Richard Metzler, and Yaneer Bar-Yam

***Abstract*—Distributed computing systems are increasingly being created as self-organizing collections of many autonomous (human or software) agents cooperating as peers. Peer-to-peer coordination introduces, however, unique and potentially serious challenges. When there is no one "in charge," dysfunctions can emerge as the collective effect of locally reasonable decisions. In this paper, we consider the dysfunction wherein inefficient resource use oscillations occur due to delayed status information, and describe novel approaches, based on the selective use of misinformation, for dealing with this problem. A model of several servers offering equivalent service to independent clients is presented and studied numerically and analytically; the spreading of misinformation about the queue status is found to dampen oscillations and improve system performance for a wide range of parameters.**

***Index Terms*—Emergent dysfunctions, resource oscillations, selective misinformation.**

I. Introduction

BUSINESS and engineering systems are increasingly being created as self-organizing collections of autonomous and self-interested (human or software) agents cooperating as peers. The reasons are simple: the challenges we now face are simply too large, both in scale and complexity, to be handled by hierarchical control schemes. In many cases, moreover, political or other concerns exclude the possibility of centralized control even when technically feasible.

In such 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], [3] to chaotic inventory and price fluctuations [4]–[7] to nonconvergent and suboptimal collective decision processes [8]. The properties of these dysfunctions often appear paradoxical, and their solutions often require new kinds of thinking.

In this paper, we focus on one type of emergent dysfunction: delay-induced resource use oscillation in request-based resource sharing. We will sketch out the nature and importance of this problem, examine how previous approaches have not addressed the self-interested agent case, and propose several novel techniques, based on the selective use of misinformation, that are suited for this context. The strengths and weaknesses of these techniques are demonstrated using both empirical (simulation-based) and analytic techniques.

Manuscript received June 1, 2004; revised October 11, 2004, November 18, 2004, and November 20, 2004. This work was supported in part by the National Science Foundation Computational and Social Systems program and in part by the DARPA Control of Agent-Based Systems program. This paper was recommended by the Guest Editors.

M. Klein is with the Center for Coordination Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: m_klein@mit.edu; m.klein@mit.edu).

R. Metzler is with the New England Complex Systems Institute, Cambridge, MA 02138 USA, and the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: richard@necsi.org, metzler@necsi.org).

Y. Bar-Yam is with the New England Complex Systems Institute, Cambridge, MA 02138 USA (e-mail: yaneer@necsi.org).

Digital Object Identifier 10.1109/TSMCA.2005.846373

II. Challenge

Imagine that we have a collection of self-interested consumer agents faced with a range of competing providers for a given resource (e.g., a roadway, a piece of information, a sensor or effector, a communication link, a storage capability, or a web service). Typically, the utility of a resource is inversely related to how many consumers are using it. Each agent therefore strives to select the least-utilized resource. Such resource allocation is frequently carried out on a first-come first-served basis. This is a peer-to-peer mechanism—there is no one "in charge"—which is widely used in settings that include markets, internet routing, and so on. It is simple to implement, makes minimal bandwidth requirements, avoids centralized bottlenecks, and—in the absence of delays in resource status information—allows consumers to quickly converge to a near optimal distribution across resources. This is a kind of self-organization, caused not by direct consumer-consumer interactions, but rather by the mutual impacts they have on resource utilization.

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, an 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 [9], 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 accurate utilization information for fear of compromising their competitive advantage. In that case, agents will need to estimate resource utilization using other criteria such as extrapolating from their own previous experience with those resources, consulting reputation services, or watching what other consumers are doing. Such estimates often lag behind the actual resource utilization.

It can be shown analytically (see Section A of the Appendix) that when status delays occur, the resource queue lengths will oscillate, potentially reducing the utility achieved by the consumer agents far below optimum. This is also a kind of self-organization but, as we shall see, one with unfortunate effects. Imagine that we have two resources: $R1$ and $R2$. At some point, one of the resources, say $R1$, is utilized less than the other. Consumers at that point will of course tend to select $R1$. However, since their image of resource utilization is delayed, they will

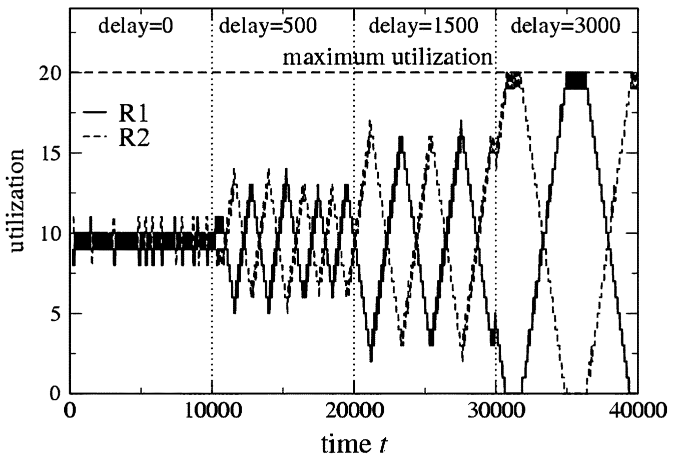


Fig. 1. Utilization of two equivalent resources with and without status info delays (simulation results; see Section IV for details).

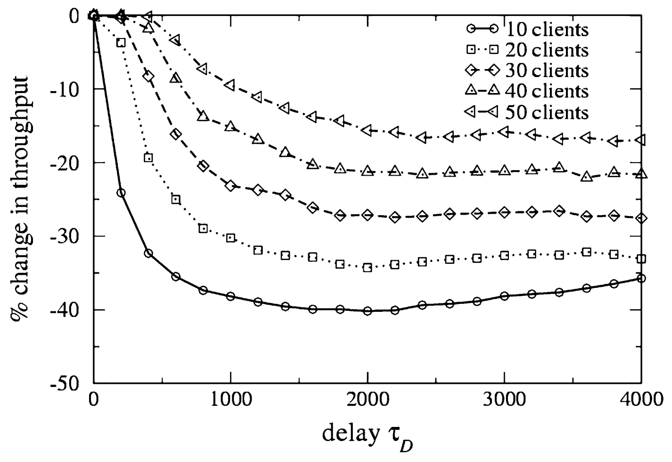


Fig. 2. Decline in throughput as a function of number of consumers for five resources (simulation results).

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 amplitude of the oscillations, increases with the delay, to the extent that all consumers may at times select one resource when the other is idle (Fig. 1).

Such oscillations have several undesirable effects. One is that they can reduce system throughput (and thus increase how long consumers have to wait for resources) because some resources may lay idle even when there are consumers not being served. Oscillations can increase the *variability* in how long consumers have to wait for a resource, which may be significant in domains where consistency, and thus predictability, is valued. Oscillations, finally, increase the maximum queue size needed by resources to avoid needless request rejections, which is an issue if queues, as is often the case, invoke some kind of cost.

This problem is influenced in seemingly paradoxical ways by changing the number of resources and consumers. Fig. 2 shows the decline in throughput for a system with five resources as a function of status delay and number of consumers. We see that reducing resource utilization actually *worsens* the decline in throughput, and causes throughput losses to occur at *lower* status delay values. The throughput reduction can be substantial, reaching as high as 40%.

Fig. 3 shows the decline in throughput for systems with differing numbers of resources, where the number of consumers per resource is fixed. We find that the throughput losses increase and come at shorter status delays as we increase the number of resources. The traditional "fix-all" of increasing system capacity thus actually makes this emergent dysfunction worse. Despite their apparently counter-intuitive nature, these results can be explained simply. When the utilization of a resource is low, even small amplitude oscillations can cause it to go idle. When all consumers shift to what they believe is the least-utilized resource, many resources can potentially go idle as a result of delay-induced oscillations.

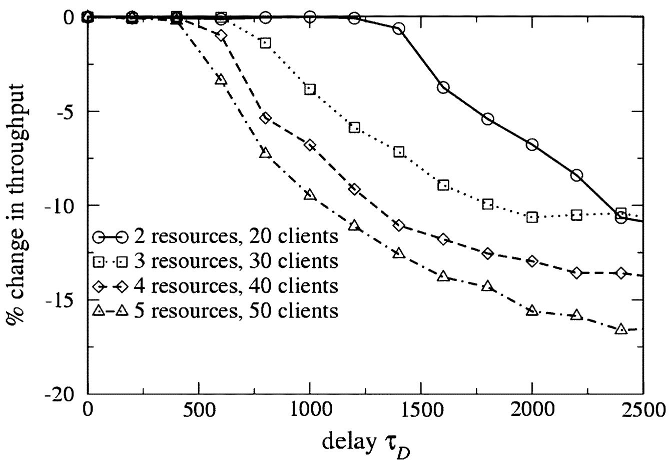


Fig. 3. Decline in throughput as a function of number of resources. Ten consumers per resource (simulation results)

TABLE I
DECLINE IN THROUGHPUT AS A FUNCTION OF VARIABILITY IN STATUS DELAY [FOR FIVE RESOURCES AND 50 CLIENTS (SIMULATION RESULTS)]

status delay	**throughput loss**
1000 +/- 0	39%
1000 +/- 500	29%
1000 +/- 1000	24%

It should also be noted that the detrimental impact of status information delays are remarkably robust in the face of variations in the size of these delays. This is illustrated in Table I, which shows the throughput loss, relative to the zero delay case, for five resources and ten consumers, when the status delay for each consumer is selected randomly from a normal distribution with the given mean and standard deviation. Even when the variability in status delays is quite high, throughput loss is substantial.

A final point is that we have considered only *linear* effects in assessing the impact of resource use oscillations. We have assumed that the time it takes before a request is handled is a linear function of the load on a resource. However, response times often increase *more* than linearly with load. Imagine, for example, that the resource is a web server. The more requests it is handling, the more likely the server is to experience virtual memory page faults that invoke additional page swapping overhead. Such nonlinear effects increase the negative impact of resource oscillations beyond what we have described above.

III. Previous Work

Resource use oscillations have been examined in the literature on "minority games" [14]–[16]. In the minority game, a large number of players decide each turn to take one of two alternative resources, and the players in the minority are rewarded. This work has investigated how to design agents so that they maximize the reward they receive. In the original formulation, each player is equipped with two strategies that prescribe an action based on the recent past, and uses the more successful of the two [17], [18]. Another version of the game has players repeating their previous action if they win, and switching with a given probability if they lose [19]. In one evolutionary variant of the model, players have an innate preference for one of the actions, and change their preference if their win/loss balance drops below a certain threshold [20]–[22]. Another evolutionary variant utilizes an ecology of agents that each look at resource status information with some characteristic additional delay. Those agents whose total status delay matches the resource use oscillation period will, in theory, do a superior job of estimating current utilization and will come to dominate the population [23]. Evolutionary approaches are best suited for contexts where the delay in status information changes slowly or not at all. If the status delay changes more quickly than the agent population can evolve, the population will tend to be dominated by agents with inappropriate strategies. While previous work has proven effective at ameliorating resource use oscillations under some conditions, it is *not* however suited for open systems contexts because it assumes that we can control the design or operation of the consumer agents.

Related literature has also emerged from work on queuing and control theory. Much of this work has focused on developing active queue management schemes that avoid congestion in network routers [24], [25], [41], but this addresses oscillations in the utilization of a *single* resource (i.e., a router), rather than the case considered here, which is usage oscillations across *multiple* resources. Some efforts, however, have addressed the multiple resource case, e.g., [26]–[30], but they have assumed closed systems with cooperative, rather than self-interested, resource consumers and providers. Network clients, for example, are assumed to respond cooperatively to the "dropped packet" messages they receive from routers by reducing their data send rate.

There has been a large body of work, finally, on using centralized mechanisms for load balancing in parallel and distributed computing systems, e.g., [31], [32]. This work is also aimed at closed systems, since centralized scheduling mechanisms are not suited to open peer-to-peer contexts.

IV. Our Approach: Efficiency Through Misinformation

As we have seen, emergent dysfunctions often have counterintuitive properties. The solutions for emergent dysfunctions can, similarly, grow out of behavior that seems locally suboptimal. This is the case with the techniques we have investigated. Our approach is predicated on resources (selectively) *misinforming* consumers about how busy they are. Paradoxically, this can lead, as we show below, to superior resource allocation performance, including greater throughput and reduced variability.

A. Assumptions

We assume, first of all, that there are multiple resource consumers and providers, and that they are *self-interested*. This means that these entities attempt to maximize their utility without regards to the impact their decisions have on the utility of other entities in the system. Resources allocate themselves by responding to consumer requests, as opposed to by using some kind of centrally controlled allocation mechanism. We also assume that the resource consumers have no direct information about what other consumers are doing or plan to do in terms of resource requests. Consumers only have access to estimated task completion times for the relevant resources. Finally, we assume that this status information can be delayed by an amount that is relatively large compared to the time a resource needs to complete a task.

These represent, we believe, realistic assumptions for at least some important domains. Human and computing systems of all types are increasingly *open* systems, which means that the component entities may come from diverse sources and cannot be expected to adhere in their design and behavior to any centralized authority. They can be assumed, instead, to be developed to pursue the interests of their owners/developers. Large-scale resource allocation, in such contexts as business markets, grid computing, and Internet bandwidth allocation, is request based because the needs of resource consumers are not knowable in advance by resource providers. Resource consumers, for the same reason, must operate largely in ignorance of which resources their peers may be interested in. Resource consumers, at least in the Internet environment, generally can access resources throughout the world. Finally, delayed status information is inevitable in a world where the time needed to perform computing tasks is decreasing exponentially according to Moore's law, but message transmission times cannot drop below the limit imposed by the speed of light. As anyone can verify using "ping," round-trip message times on the Internet can range from 15 to 500 ms or even more [33], while many computing tasks (such as serving static web pages) make take a server fractions of a millisecond to perform. Cross-resource usage oscillations thus appear inevitable and have indeed been observed in a variety of contexts [10]–[13], [34], [35].

B. Scenario

All of the solution approaches proposed herein were evaluated using a specially written discrete event simulator running under Macintosh Lisp 5.0 on a 733-MHz PowerMac G4. The simulation scenario considered multiple (from 20–50) consumers and multiple (2–5) resources. Consumers submit requests to the resource that they judge is the least heavily utilized. Resources can differ in how quickly they can complete requests. When a request is received by a resource, it is placed on a queue and, once it reaches the front of the queue, the resource is allocated to the consumer for a length of time inversely proportional to the speed of the resource. When that period is over, a notification message is sent to the consumer. Messages take a fixed amount of time to travel from sender to receiver. Consumers wait a randomly selected amount of time, after a notification is received, before submitting a new request. The value of the service provided by a resource (though not the time it takes to access the resource) is independent of the resource's utilization. The case where utilization does affect resource value is considered in [36]. The two metrics of interest to consumers in this scenario include 1) the aggregate throughput of the system, in terms of requests processed per time unit and 2) the variability in request processing times. In our simulations, messages took 20 time units to propagate, the time gap between receiving a completion notification and sending a subsequent request was normally distributed with an average of 40 and a standard deviation of 10, one server took 80 time units to service a request, and the other took 160 time units. Each simulation run was 10 000 time units long.

All of the experimental (i.e., simulation-based) conclusions reported in this paper were confirmed by performing a T test on the results of 100 simulator runs per condition. These conclusions were all statistically significant at $p < 0.01$.

C. Status Misinformation

Let us assume that the resources have control over the status information that the consumers are given when they decide which resource to request (this is a rather strong assumption, of course, which will be relaxed later). The resources use this control to lead consumers to select the "wrong" (more heavily utilized) resource with probability p. The notion that agents can have somewhat "corrupted" status information was broached in [23], but that work did not investigate how status misinformation can be beneficial by dampening delay-induced oscillations. Oscillations are damped because misinformation-caused requests are spread to some extent to both resources, regardless of which one is actually less utilized. In Fig. 4, for example, we can see how the oscillations in resource queue lengths were substantially reduced when resource status misinformation ($p = 0.25$) was introduced at time = 10 000.

If the status misinformation probability is made large enough, however, consumers get less and less "real" information and begin to choose resources without regards to their actual utilization. Resource utilization then increasingly performs a "random walk" [37], once again increasing the variability in queue lengths. There is thus a tradeoff involved (Fig. 5).

These changes in queue length variance can be expected to have a direct impact on the variability of request processing times, and also raise the possibility that the queue for one of the resources will empty out, thereby reducing throughput. This is confirmed by simulations [see Fig. 6(a) and (b)]. When $\boldsymbol{p} = 0$, we find that the variability in how long a consumer must wait for a resource increases, as we would expect, with the status information delay, due to periodic oscillations. When the delays get large enough to cause queue emptying, throughput drops. For intermediate values of $\boldsymbol{p}$, throughput is returned to near-optimal levels even with large delays, but variability is high. As p approaches $1/2$, throughput drops off again (due to queue

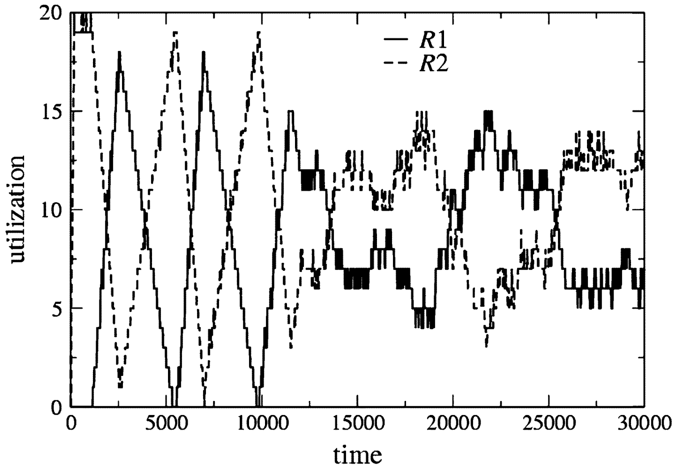


Fig. 4. Effect of introducing status misinformation with delay-induced resource use oscillations: simulation results, two resources, 20 consumers.

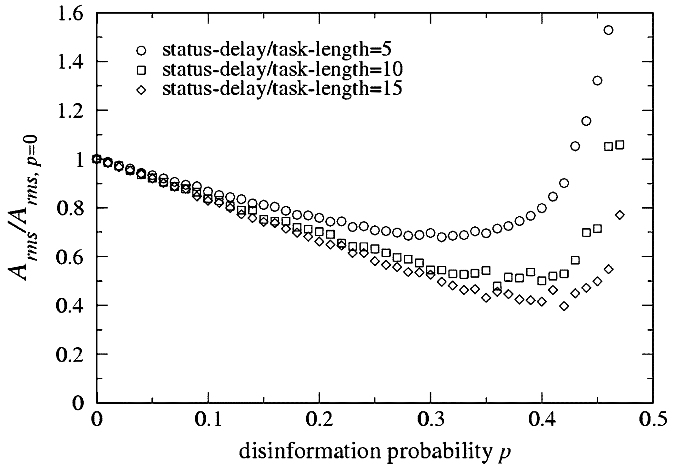
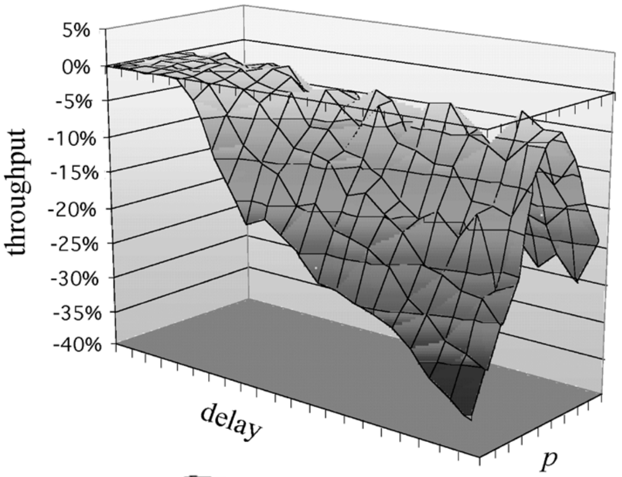


Fig. 5. Effect of varying status misinformation probability on queue length variability. The y axis shows the root mean square $(A_{\rm rms})$ of the difference in queue lengths, divided by the $A_{\rm rms}$ value for $\boldsymbol{p} = 0$ (simulation results).

emptying caused by random walk fluctuations) and variability becomes higher yet. In our example system, throughput is maximized when p is about 0.35; remarkably, performance is improved by imposing substantial misinformation.

This approach faces some serious disadvantages, however, when applied in an open systems context. We either must control the status information that consumers use to select resources, or else the consumers themselves must impose a stochastic element upon their resource selection decisions. Both assumptions are problematic. There is no guarantee that consumers will rely on an external, and thus potentially manipulable, information source to estimate resource availability. They may, for example, rely solely on their own recent experience with the different resources. It may also be unrealistic to expect consumers to add stochastic "noise" to their resource selection decisions. While adding noise is an effective strategy when delay-induced oscillations are occurring, it is *against* a consumer's interest to use stochastic noise when substantial delay-induced oscillations are *not* present, as this will only lead the consumers to select resources that offer slower task completion times. This insight is confirmed by simulations where consumers that either do or



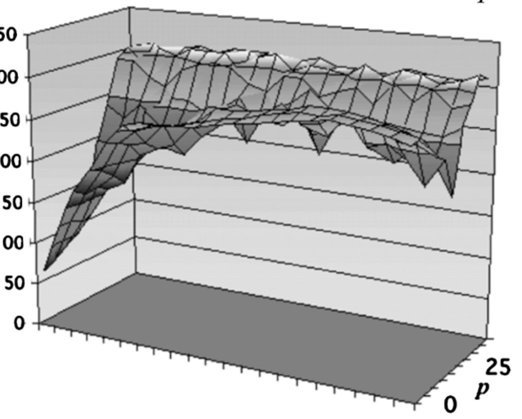


Fig. 6(a). Throughput as a function of delay and misinformation probability (simulation results). (b) Completion time variability as a function of delay and misinformation probability (simulation results).

TABLE II(A)
PAYOFF MATRICES SHOWING TASK COMPLETIONS PER CONSUMER IN THE $\text{delay} = 1000$ CONDITION

	no noise	noise
no noise	23/23	26/26
noise	26/26	28/28

TABLE II(B)
PAYOFF MATRICES SHOWING TASK COMPLETIONS PER CONSUMER IN THE DELAY $= 0$ CONDITION. DIAGONALS SHOW THE CASE WHERE HALF THE CONSUMERS USE NOISE, AND THE OTHER HALF DO NOT. SIMULATION RESULTS FOR TEN CONSUMERS AND FIVE RESOURCES

	no noise	noise
no noise	38/38	21/17
noise	17/21	33/33

do not use noise compete against each other to get resources (Table II).

When significant status delays are present, consumers that use noise enjoy a higher task-completion rate. When status delays are *not* present, however, the consumers that use noise fare *worse* than the consumers that do not. This introduces a dilemma, because in an open system consumers are unlikely to be able to determine if delay-induced oscillations are occurring or not. This requires either knowing the status delays for the other consumers, or (as we shall see below) having detailed utilization information for all the competing resources. Both alternatives are excluded by the open system assumptions we delineated earlier in the paper.

D. Stochastic Request Rejection

These concerns motivated us to explore an alternative approach for alleviating delay-induced resource use oscillations. The idea is simple: some fixed fraction of resource requests are rejected, at random, by resources. When a consumer receives a rejection message, it is (reasonably) assumed to send its request to some other server instead. It is in the consumer's interest to do so because, as far as a consumer knows, the reject implies that the resource is busy, and thus that some other resource is likely to be less loaded. The net effect of such stochastic rejection is the same as with the previous approach in that, for some constant fraction of requests, consumers are misled about which resource is the least utilized. In the scenario we studied, throughput was maximized when $1/2$ of all requests were stochastically rejected.

The stochastic request rejection approach can, however, reduce throughput if resource demands are low enough that the resource queues are forced to empty out due a request rejection. It also increases message traffic due to the addition of reject messages. When the rejection probability is 0.5, the average number of rejections for a request is 1, so an average of two requests will be needed to access a resource, increasing total required message traffic from 2 (one request and one notification) to 4 (two requests, one reject, and one notification). (See Appendix Part C for an analytic treatment).

E. Load-Dependent Rejection

Both of these disadvantages can be substantially ameliorated by adopting a load-dependent rejection scheme, inspired by the "random early drop" technique proposed for avoiding send-rate synchronization among the clients sharing a single network router [38]. Instead of using a fixed request rejection frequency, resources reject requests with a frequency proportional to how full their queue is. The number of rejection messages generated is less (because high rejection rates are only incurred at the utilization peaks) and very few rejections occur when the resources are underutilized, making it unlikely that throughput will be reduced because a request was rejected when a resource was available. Load-dependent rejection also offers the bonus of somewhat higher throughout than fixed-rate rejection because the rejection rate (and thus the degree of damping) increases with the amplitude, the oscillations have a rounded shape that results in a smaller peak amplitude.

The average rate of rejection needs to be tuned to the current average load. There is a tradeoff involved. If the rejection regime is too aggressive, we incur excessive reject message traffic, and the possibility of causing queue emptying by rejecting requests when a resource is lightly utilized. If the rejection regime is not aggressive enough, however, there will be insufficient damping, which can also led to queue emptying and throughput loss. The following figure shows a typical tradeoff (Fig. 7).

Each point on the curve represents a different level of load-dependent rejection: a relatively mild rejection regime (i.e., where

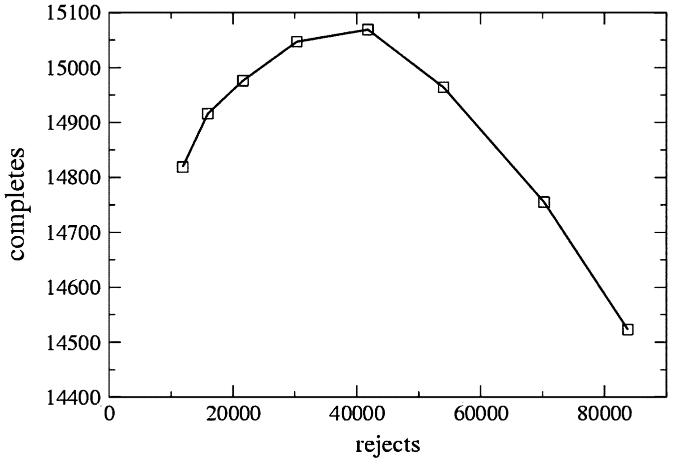


Fig. 7. Throughput (completes) versus reject messages for different levels of load-dependent rejection: simulation results for five resources, 50 consumers, and status delay 1000.

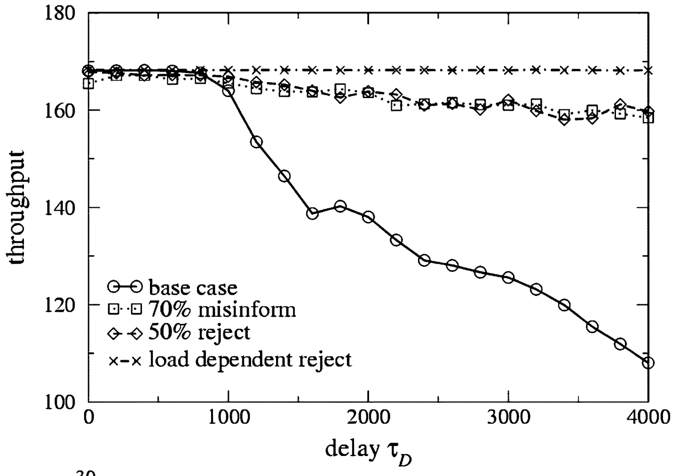
the rejection rate increases slowly with load) on the left, and increasingly aggressive rejection to the right. As we can see, there is an optimum rejection "strength," beyond which throughout begins to decrease.

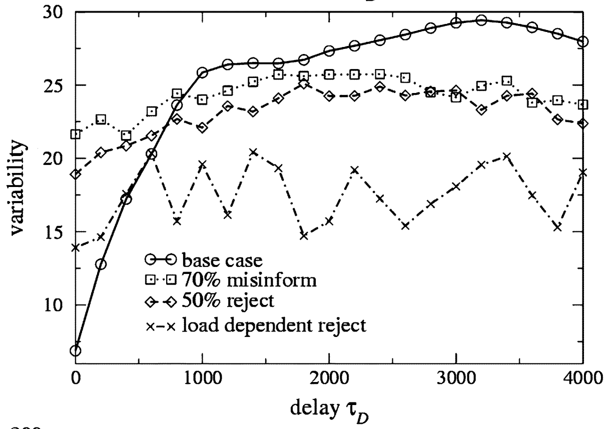
The impact of the schemes we have discussed can be summarized and contrasted as follows [Fig. 8(a)–(c)].

Misinformation-based techniques substantially increase throughput and reduce the variability in the time it takes to get a consumer request satisfied, for a wide range of delays, relative to the base case where these techniques were not used. Load-based rejection is the best technique in terms of throughput and variability, with the additional advantage of not assuming we can control the status information received by consumer agents, but incurs increased message traffic.

One final refinement involves the realization that there is no point in incurring the increased message traffic caused by request rejection if there are no resource use oscillations, or if the oscillations are caused by variations in *aggregate* consumer demand rather than by status delays. This challenge, fortunately, is easy to address. Stochastic request rejection should only be activated if (1) there are significant periodic oscillations in resource utilization (determined by looking for above-threshold values in the power spectrum derived by a fast Fourier transform), and (2) the resource utilization across servers is *negatively* correlated (positive correlation would imply that aggregate demand is varying). We have implemented this approach and found that it successfully avoids being triggered by aggregate demand variations while remaining effective in responding to delay-induced oscillations.

Are self-interested resources likely to implement a load-dependent rejection (LDR) scheme? The incentives are mixed. We can expect that most resources are motivated by two concerns: 1) increasing the number of requests they service (e.g., because they make a profit for each task completion) and 2) reducing maximum queue sizes (because long queues typically invoke some kind of storage cost). As we have seen, LDR does reduce maximum queue sizes, and increases overall throughput. Our simulations show that the more resources implement LDR, the higher the overall throughput will be. When only a subset of the





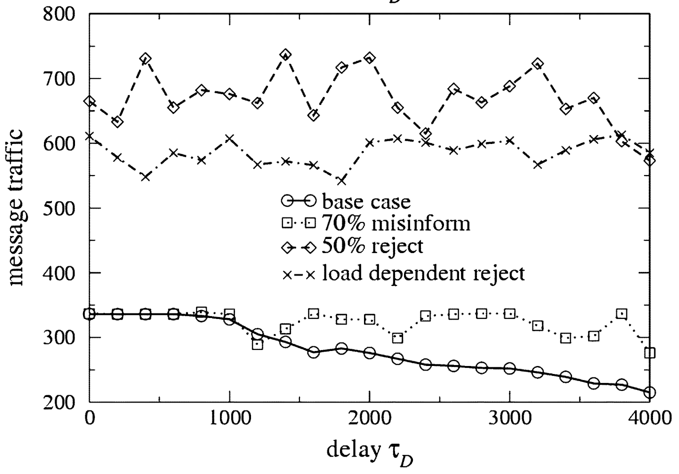


Fig. 8(a). Throughput for different oscillation remediation schemes. Simulation results for two resources and 20 consumers. (b) Variability for different oscillation remediation schemes. Simulation results for two resources and 20 consumers. (c) Message traffic for different oscillation remediation schemes. Simulation results for two resources and 20 consumers.

resources implement LDR, however, the resources that do use it fare *worse* (in terms of tasks completed) than the resources that do not. This is because the non-LDR resources do not "cooperate" by redistributing some of the requests they receive to other resources with lower utilizations. This is thus a kind of prisoner's dilemma: all benefit if all use LDR, but individual resources are incented to "defect" and abandon LDR. However, if all defect, then all lose. One can imagine several solutions to this problem. One is to impose penalties on resources that do not use LDR. Another is for resources to implement a "tit-for-tat" scheme, wherein resources begin using LDR, but abandon it if the others do. It has been shown that tit-for-tat strategies incent individuals to cooperate in prisoner's dilemma contexts [39]. A final possibility is to wrap resources in "sentinels" that monitor and modify the requests and responses passing in and out of these resources, sending "reject" messages to consumers when the resources they monitor have large queues [40].

V. Contributions and Future Work

We have presented a novel and promising approach for mitigating the deleterious effects of delay-induced resource-use oscillations on request-based resource sharing, by exploiting the paradoxical power of selectively (and stochastically) misinforming consumers. While stochastic approaches have been applied to such challenges as nonlinear optimization [43], [44], this is the first instance we are aware of where stochastic misinformation has been applied to resource use oscillations.

Our approach is designed to be appropriate for the important context of open distributed systems, where we can not rely on being able to control the design or operation of the resources or consumers. Our work therefore addresses an important gap in the literature, since previous efforts have focused almost exclusively on closed-system solutions that involve centralized scheduling, cooperative resources and consumers, or both.

Our future efforts will include empirical and analytic components. We will extend our analytic work to address more than two resources as well as asymmetric resources. Our simulations to date have shown that load-dependent rejection is effective at ameliorating status delay-induced oscillations in this context, but this needs to be backed up with an analytic treatment. We are attempting to derive an analytic way of determining the correct rejection regime for different contexts; we have done this empirically to date. We also plan to use our models to predict the degree of resource oscillation, as well as the potential benefits of selective misinformation, for real-world resources such as competing web sites.

Appendix
Analytic Treatment

We have been able to formally verify many of the conclusions described above. Our analysis results to date are restricted to the case of two servers R_1 and R_2, which offer the same service to a number N of clients (note that the simulations described above also consider cases where there are more than two resources). While previous work has derived the delay-differential equations that underlie resource use oscillation (see, for example, [4] and [23]), we are unaware of any successful previous efforts to formally model resource sharing dynamics with stochastic misinformation and request rejection.

In our analytic treatment, clients send data packets to one of the servers. After a travel time τ_T, the packets arrive at the server, and are added to a queue. Servers need a time τ_P to process each request. We chose the time scale such that $\tau_P = 1$ (i.e., all times are given in units of τ_P, and processing rates in units of requests/ processing time). When a client's request is completed, the server sends a "done" message (which takes another τ_T to arrive) to the client. The client is then idle for a time τ_I, after which it sends a new packet. Clients send their request to the server reputed to have the shorter queue length; however, the information they receive is obsolete—they only know the length of the queues a delay time τ_D ago.

A. Dynamics Without Disinformation

We first study the simple case where both servers accept all incoming requests and demand is distributed uniformly enough, such that both servers are busy at all times. The only relevant variables are $N_1(t)$ and $N_2(t)$, the number of clients whose data is in the queue or being processed by R_1 and R_2, respectively, at time t; we treat them as continuous variables. Idle clients do not have to be taken into account explicitly; neither do clients who are waiting for their "done" message from the server—for our purposes, they are the same as idle agents, and we simply do not count them in the number N of active clients. We will first solve the problem neglecting agents whose message is traveling to the server, then include nonvanishing travel times.

There are only two processes which change the length of the queues. 1) Due to processed requests, both N_1 and N_2 decrease by 1 per time unit. 2) In the same time span, two clients (whose data was processed by R_1 and R_2 a time $\tau_T + \tau_I$ ago) compare the obsolete values $N_1(t - \tau_D)$ and $N_2(t - \tau_D)$ and add their requests to the queue according to this information.

We write delay-differential equations for N_1 and N_2

$$\begin{aligned}\frac{dN_1}{dt} &= 2\Theta\big(N_2(t-\tau_D) - N_1(t-\tau_D)\big) - 1\\ \frac{dN_2}{dt} &= 2\Theta\big(N_1(t-\tau_D) - N_2(t-\tau_D)\big) - 1\end{aligned} \tag{1}$$

where Θ stands for the step function, $\Theta(x) = 0$ for $x < 0$, and $\Theta(x) = 1$ for $x \geq 0$. This can be simplified by introducing the difference in queue lengths $A(t) = N_1(t) - N_2(t)$

$$\frac{dA}{dt} = -2\mathrm{sign}\big(A(t-\tau_D)\big). \tag{2}$$

The stationary solution (i.e., the oscillation that will result after a transient period) for this equation is

$$A(t) = 2\tau_D \mathrm{tri}\left(\frac{t}{4\tau_D} + \phi\right) \tag{3}$$

where $\mathrm{tri}(x)$ is the triangle function

$$\mathrm{tri}(x) = \begin{cases} 4x - 1, & \text{for } 0 < x < 1/2 \\ -4(x - 1/2) + 1, & \text{for } 1/2 \leq x < 1, \text{periodic in } 1 \end{cases} \tag{4}$$

and ϕ is a phase determined by initial conditions. The frequency of the resulting oscillation is only determined by the delay, and the amplitude by the ratio of delay time to processing time—the total number of clients does not play a role. Clients typically spend much of their time with their request in the queue, and adding more clients only makes both queues longer. Also, if the delay goes to zero, so does the amplitude of oscillations.

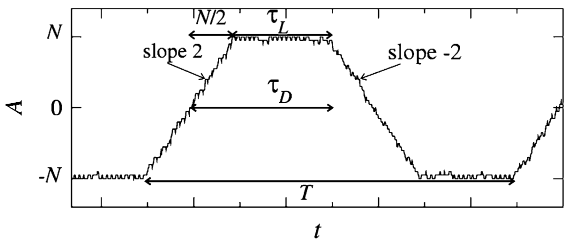


Fig. 9. Illustration of $A(t)$ in the regime where servers empty out periodically. When A reaches $+N$ or $-N$, throughput drops from 2 to 1.

Introducing a nonvanishing travel time τ_T in this simple case has the same effect on $A\,(t)$ as increasing the delay time: it leads to the delay-differential equation

$$\frac{dA}{dt}(t+\tau_T) = -2\text{sign}\big(A(t-\tau_D)\big). \tag{5}$$

The solution is given by (3) with τ_D replaced by $\tau_D + \tau_T$.

1) The Impact of Idle Servers: The case where oscillations become so strong that servers go idle periodically $(2\tau_D > N)$ can be treated in a similar framework, for $\tau_\text{T} = 0$: Once the difference in queue lengths reaches the value $+N$ or $-N$, one queue ceases to process requests. Hence, the rate of requests at the other server drops from 2 to 1—exactly the rate at which it keeps processing them. The queue length at the active server therefore stays constant for some time τ_L. An example of the resulting curve can be seen in Fig. 9. Starting from the time where $A(t)$ crosses the zero line, it will take a time τ_D for clients to realize that they are using the "wrong" server, so $\tau_D = \tau_L + N/2$, or $\tau_L = \tau_D - N/2$. The period T of the oscillations is then $T = 2\tau_L + 2N = 2\tau_D + N$, which is smaller than $4\tau_D$. Data throughput of the system drops from 2 to $(3 + 2\tau_D/N)/(1 + 2\tau_D/N)$ (in units of τ_P).

B. Impact of Global Disinformation

1) Idealized Case: No Fluctuations: We now introduce global disinformation: Clients have probability p of receiving the wrong answer, and accordingly choose the "wrong" server. The update equations are

$$\begin{aligned}\frac{dN_1}{dt} &= 2[(1-p)\Theta\big(-A(t-\tau_D)\big) + p\Theta\big(A(t-\tau_D)\big)] - 1\\ \frac{dN_2}{dt} &= 2[(1-p)\Theta\big(A(t-\tau_D)\big) + p\Theta\big(-A(t-\tau_D)\big)] - 1\end{aligned} \tag{6}$$

leading to

$$\frac{dA}{dt} = -2(1-2p)\text{sign}\big(A(t-\tau_D)\big). \tag{7}$$

This equation has the form of (3) with a prefactor of 1–2 p, and has a steady-state solution

$$A(t) = 2\tau_D(1-2p)\text{tri}\left(\frac{t}{4\tau_D} + \phi\right) \tag{8}$$

for $p < 1/2$. At $p = 1/2$, no information is available: clients' decisions are random, and queue lengths perform a random walk, whose fluctuations are not captured by the deterministic framework we are using. Even for values $p < 1/2$, fluctuations may become larger than the typical amplitude of oscillations, and thus dominate the dynamics. For $p > 1/2$, users migrate systematically from the less busy to the busier server, until one is idle much of the time, and the other has almost all clients in its queue.

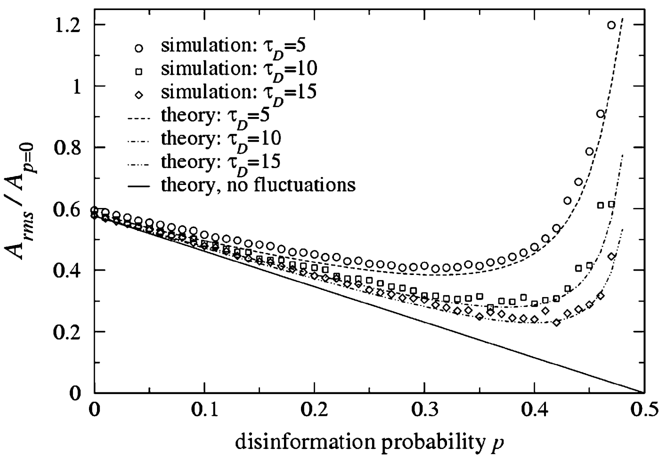


Fig. 10. Global disinformation: The y axis shows the root mean square (RMS) of the difference in queue lengths, divided by the RMS one gets without disinformation. The theory for negligible fluctuations (solid line) is given by (8); agreement with it improves as the amplitude of oscillations increases. The effect of fluctuations can be modeled using a discrete random walk (dashed lines).

2) Effects and Treatment of Fluctuations: The tradeoff between reduced oscillations and increased fluctuations can be seen in Fig. 10. Rather than measuring the amplitude of oscillations, the root mean square $A_\text{rms} = \langle A^2\rangle^{1/2}$ of $A(t)$ is shown. For a pure triangle function of amplitude a, one gets $A_\text{rms} = a/\sqrt{3}$. For small p, the amplitude is reduced linearly; for larger p, fluctuations increase, dominating the dampened oscillations. When the amplitude of the undisturbed system is small, fluctuations have a large impact. As the amplitude of oscillations gets larger, the impact of fluctuations becomes smaller, and the value of p where fluctuations dominate moves closer to $1/2$.

Under the influence of randomness, A performs a biased random walk. Let us assume that server R_2 is currently preferred. In each unit of time, $A/4$ increases by 1 with probability p^2 (the two clients processed both go to R_1), stays constant with probability $p(1-p)$, and decreases by 1 with probability $(1-p)^2$ (both clients move from R_1 to R_2). To reproduce quantitatively the effects of fluctuations, one can numerically average A^2 over such a random walk that takes place in two phases: the first phase lasts until $A = 0$ is reached; the second takes another τ_D steps until the direction of the bias is reversed. The probability distribution of A at the beginning of the half-period has to be chosen self-consistently such that it is a mirror-image of the probability distribution at the end; the proper choice for $A/4$ is a Gaussian restricted to positive values with mean (1–2p)τ_D and variance $2p(1-p)\tau_D$.

These analytic results are, as Fig. 10 shows, in excellent agreement with our simulation results.

C. Request Rejection

Assuming that servers cannot influence the public information on queue status, they can still influence the behavior of clients directly: they claim they are not capable of processing a request, and reject it—in the simplest case, with a constant probability r. Compared to global disinformation, a new possibility arises that a request bounces back and forth several times between the servers, but that adds nothing new in principle: the fraction of requests that end up at the server that they tried at first is $(1-r)+r^2(1-r)+r^4(1-r)+\cdots=1/(r+1)$, whereas a fraction $r/(r+1)$ will be processed at the other server. This is equivalent to setting $p=r/(1+r)$ in the "global disinformation" scheme [see (8)], and gives equivalent results.

Choosing r close enough to 1 reduces the amplitude of oscillations dramatically; however, each message is rejected a large number of times on average, generating large amounts of extra traffic.

D. Load-Dependent Request Rejection

As a more general case of individual rejection, servers can have rejection probabilities that depend on their current queue length. For example, let us consider the case where $r_i = cN_i$ if $cN_i < 1$, and 1 otherwise, with some appropriately chosen constant c.

The analysis from the "individual rejection" section can be repeated with the additional slight complication of two different rejection rates r_1 and r_2. A fraction $(1-r_1)/(1-r_1r_2)$ of agents who initially try server R_1 ends up being accepted by it, whereas a fraction $r_1(1-r_2)/(1-r_1r_2)$ eventually winds up at server R_2, and *vice versa* for clients who attempted 2 first. Combining the resulting delay-differential equations for N_1 and N_2 into one for A, one obtains

$$\frac{dA}{dt}=\frac{2}{1-r_1r_2}\big(\Theta(-A(t-\tau_D))(1-2r_1+r_1r_2) - \Theta\big(A(t-\tau_D)\big)(1-2r_2+r_1r_2)\big). \quad (9)$$

We can now substitute the load-dependent rates. We write them as follows: $r_1=r_{\rm avg}+c'A, r_2=r_{\rm avg}-c'A$, with $r_{\rm avg}=(r_1+r_2)/2$ and $c'=c/2$. For small amplitudes A relative to the total number of players N, the deviation from $r_{\rm avg}$ does not play a significant role, and it is $r_{\rm avg}$ that determines behavior, yielding the same results as a constant rejection rate. For larger relative amplitudes, the oscillations are no longer pure triangle waves, but have a more curved tip, effectively reducing the maximum values that A takes. These nonlinear effects make LDR more efficient at avoiding queue emptying. They also provide a restoring force that suppresses fluctuations effectively. Further details are available in [41].

References

[1] D. Jensen and V. Lesser, "Social pathologies of adaptive agents," in *Proc. Safe Learning Agents Workshop, 2002 AAAI Spring Symp.*, 2002, pp. 13–19.

[2] M. H. Chia, D. E. Neiman, and V. R. Lesser, "Poaching and distraction in asynchronous agent activities," in *Proc. 3rd Int. Conf. Multiagent Syst.*, Paris, France, 1998, pp. 88–95.

[3] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, pp. 1243–1248, 1968.

[4] M. Youssefmir and B. Huberman, "Resource contention in multiagent systems," in *Proc. 1st Int. Conf. Multiagent Syst.*, San Francisco, CA, 1995, pp. 398–403.

[5] J. D. Sterman, *Learning in and About Complex Systems*. Cambridge, MA: MIT Press, 1994.

[6] J. O. Kephart, J. E. Hanson, and A. R. Greenwald, "Dynamic pricing by software agents," *Comput. Netw.: Int. J. Distributed Inform.*, vol. 32, no. 6, pp. 731–52, 2003.

[7] P. Ranjan *et al.*, "Decision making in logistics: A chaos theory based analysis," in *Proc. AAAI Spring Symp. Diagnosis, Prognosis, Decision Making*, 2002.

[8] M. Klein *et al.*, "The dynamics of collaborative design: Insights from complex systems and negotiation research," *Concurrent Eng. Res. Applicat.*, vol. 11, no. 3, pp. 201–210, 2003.

[9] C. Hewitt and P. D. Jong, *Open Systems*. Cambridge, MA: MIT Press, 1982.

[10] M. Mitzenmacher, "How useful is old information?," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 1, pp. 6–20, Jan. 2002.

[11] J. Wang, J. Liu, and X. Jin, "Modeling agent-based load balancing with time delays," presented at the IEEE Int. Conf. Intell. Agent Technol., 2003.

[12] Y. Jiang *et al.*, "Inter-domain routing stability measurement," presented at the Workshop High-Performance Switching Routing, 2002.

[13] A. Khanna, "The revised ARPANET routing metric," presented at the SIGCOMM, Sep. 1989.

[14] Emergence of Cooperation and Organization in an Evolutionary Game, D. Challet and Y.-C. Zhang. (1997). [Online]. Available: http://arxiv.org/abs/adap-org/9 708 006

[15] Modeling Market Mechanism with Evolutionary Games, Y.-C. Zhang. (1998). [Online]. Available: http://arxiv.org/abs/cond-mat/9 803 308

[16] Minority Game Homepage [Online]. Available: http://www.unifr.ch/econophysics/minority

[17] Trading Behavior and Excess Volatility in Toy Markets, M. Marsili and D. Challet. (2000). [Online]. Available: http://arxiv.org/abs/cond-mat/0 004 376

[18] R. Savit, R. Manuca, and R. Riolo, "Adaptive competition, market efficiency, and phase transitions," *Phys. Rev. Lett.*, vol. 82, no. 10, pp. 2203–2206, 1999.

[19] G. Reents, R. Metzler, and W. Kinzel, "A stochastic strategy for the minority game," *Phys. A: Stat. Mech. Applicat.*, vol. 299, no. 1–2, pp. 253–261, 2001.

[20] N. F. Johnson *et al.*, "Self-organized segregation within an evolving population," *Phys. Rev. Lett.*, vol. 82, no. 16, pp. 3360–3363, 1999.

[21] S. Hod and E. Nakar, "Self-segregation versus clustering in the evolutionary minority game," *Phys. Rev. Lett.*, vol. 88, no. 23, 2002.

[22] E. Nakar and S. Hod, "Temporal oscillations and phase transitions in the evolutionary minority game," *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, vol. 67, no. 1, p. 016109, 2003.

[23] T. Hogg and B. Huberman, "Controlling chaos in distributed system," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 6, pp. 1325–1332, Nov./Dec. 1991.

[24] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proc. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 1999, pp. 1346–1355.

[25] R. J. La, P. Ranjan, and E. H. Abed, "Global stability conditions for rate control of discretized model with communication delays," presented at the IEEE Globecom, 2004.

[26] C. M. Lagoa, H. Che, and B. A. Movsichoff, "Adaptive control algorithms for decentralized optimal traffic engineering in the internet," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, p. 415, 2004.

[27] Z. Wang and J. Crowcroft, "Shortest path first with emergency exits," presented at the ACM Symp. Commun. Archit. Protocols, 1990.

[28] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Proc. Conf. Applicat., Technol., Archit., Protocols Comput. Commun.*, 1999, p. 227.

[29] R. Johari and D. K. H. Tan, "End-to-end congestion control for the internet: Delays and stability," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, 2001.

[30] D. Awduche *et al.*, "Overview and Principles of Internet Traffic Engineering," Working Report 3272, 2002.

[31] H. Y. Sit *et al.*, "An adaptive clustering approach to dynamic load balancing," in *Proc. 7th Int. Symp. Parallel Archit., Algorithms, Netw.*, 2004, pp. 415–420.

[32] M. H. Willebeek-LeMair and A. P. Reeves, "Strategies for dynamic load balancing on highly parallel computers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 9, pp. 979–993, Sep. 1993.

[33] M. Allman, "A web server's view of the transport layer," *ACM Comput. Commun. Rev.*, vol. 30, no. 5, 2000.

[34] Y. Wang, J. Liu, and X. Jin, "Modeling agent-based load balancing with time delays," in *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol.*, 2003, pp. 189–195.

[35] A. Erramilli and L. J. Forys, "Oscillations and chaos in a flow model of a switching system," *IEEE J. Select. Areas Commun.*, vol. 9, no. 2, pp. 171–178, Feb. 1991.

[36] M. Klein and Y. Bar-Yam, "Handling resource use oscillation in multiagent markets," in *Proc. AAMAS Workshop Agent-Mediated Electron. Commerce V*, Melbourne, Australia, 2003.

[37] Y. Bar-Yam, *Dynamics of Complex Systems*. Reading, MA: Addison-Wesley, 1997, ch. xvi, p. 848.

[38] B. Braden *et al.*, "Recommendations on Queue Management and Congestion Avoidance in the Internet," Network Working Group, Working Report Request for Comments: 2309, 1998.

[39] R. Axelrod, *The Evolution of Cooperation*. New York: Basic, 1984.

[40] M. Klein and C. Dellarocas, "Exception handling in agent systems," presented at the 3rd Int. Conf. Autonomous Agents, Seattle, WA, 1999.

[41] Efficiency Through Disinformation, R. Metzler, M. Klein, and Y. Bar-Yam. (2004). [Online]. Available: http://www.arxiv.org/abs/cond-mat/0312266

[42] S. H. Low *et al.*, "Linear stability of TCP/RED and a scalable control," *Comput. Netw. J.*, vol. 43, no. 5, pp. 633–647, 2003.

[43] S. A. Kauffman, *At Home in the Universe: The Search for Laws of Self-Organization and Complexity*. New York: Oxford Univ. Press, 1996.

[44] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

Mark Klein received the Ph.D. degree in computer science from the University of Illinois, Urbana–Champaign.

He is a Principal Research Scientist at the Center for Coordination Science, Massachusetts Institute of Technology (MIT), Cambridge, and an affiliate at the MIT Artificial Intelligence Laboratory and the New England Complex Systems Institute, Cambridge, MA. His research focuses on better understanding coordination and how it can help solve important problems concerning the design and management of human organizations and software systems.

Richard Metzler received the Ph.D. degree in theoretical physics from the University of Wuerzburg, Wuerzburg, Germany, in 2002.

He is a Post-Doctoral Associate with the New England Complex Systems Institute, Cambridge, MA and the Massachusetts Institute of Technology, Cambridge. He has performed research on neural networks, the minority game, and other subjects. Currently, he is working on problems in econophysics and basic topics of complexity.

Yaneer Bar-Yam received the Ph.D. degree in physics from the Massachusetts Institute of Technology, Cambridge.

He is the President of the New England Complex Systems Institute, Cambridge. He is the author of *Dynamics of Complex Systems* (New York: Perseus, 1997). He studies patterns of collective behavior, evolution, complexity, and multiscale representations in application to physical, biological, social, and engineered systems.

Dr. Bar-Yam is the Chairman of the International Conference on Complex Systems. He is the Managing Editor of *InterJournal*.