Assessment of Vulnerability of Content Adaptation Mechanisms to RoQ Attacks [†]

MINA GUIRGUIS JOSHUA THARP
{msg,jt1238}@txstate.edu
Computer Science Department
Texas State University
San Marocs, TX 78666, USA

AZER BESTAVROS IBRAHIM MATTA
{best,matta}@cs.bu.edu
Computer Science Department
Boston University
Boston, MA 02215, USA

Abstract— **Current computing systems employ different mechanisms to deal with overload conditions. Of those widely deployed are content adaptation mechanisms whereby the quality level of the content is adapted dynamically to mitigate overload conditions. Serving degraded content reduces strain on resources and enables them to cater for a larger set of clients. To that end, this paper studies adversarial exploits of dynamic content adaptation mechanisms to a new instantiation of Reduction of Quality (RoQ) attacks. The RoQ attack pattern is orchestrated to cause different forms of damage such as longer response time for legitimate clients, degraded content being served and under-utilization of resources. We assess the impact of RoQ attacks via the potency metric which reflects the tradeoffs between the damage inflicted and the cost in mounting the attack. We validate our results through numerical analysis as well as real Internet experiments.**

I. INTRODUCTION

Internet server farms and web service applications often employ sophisticated adaptation mechanisms that enable them to deal with overload conditions. Of those widely employed mechanisms are admission controllers, load balancers and content adaptation controllers. And due to their normal operation in open-access environments, these mechanisms may exhibit complex dynamics that result from many factors interacting together. Such dynamics are hard to analyze or even capture (sometimes). This, in return, poses significant challenges on how to ensure that the resulting dynamics are safe and that they could not be exploited by an adversary. Recent work of ours have exposed dynamic exploits on two of the above three mechanisms; admission controllers [1] and load balancers [2]. This work focuses primarily on security implications that arise from employing dynamic content adaptation mechanisms to mitigate overload.[1].

In a content adaptation setting, the content adaptation controller decides the quality of the content served, based on the load measured on the server(s) [3], [4], [5]. Serving degraded content requires fewer resources to handle, and thus enables catering for a larger set of clients without having to drop or refuse the admission of new clients. Content adaptation takes different forms depending on the content itself. Static content, for example, is typically prepared off-line and only used during the periods when the system is overloaded. Dynamic content, on the other hand, can be degraded online through embedding less objects, making less-intensive database calls and returning fewer results, for example.

Current data centers consume significant amounts of energy to carry on with their operations [6]. Virtualization has been one of the fruitful directions adopted to reduce the number of powered-on servers, while increasing their utilization [7]. This, in return, causes them to operate close to their maximum capacities. Content adaptation mechanisms (as well as other overload mitigation mechanism) are envisioned to be used in such systems to protect them from being pushed into overload. Therefore, it is very important to study their vulnerabilities against new instants of attacks.

The operation of content adaptation controllers while ensures optimized performance during overload conditions, it opens a door for a new class of Reduction of Quality (RoQ) attacks, to be mounted [8]. Unlike traditional Denial of Service (DoS) attacks, where an attacker saturates a given system with a sustained load, RoQ attacks cause constant oscillations between overload and underload states, without sustaining the attack traffic. This is achieved through orchestrating the attack traffic in a manner that exploits the transient dynamics of the system. The example below illustrates this point.

An illustrative Example: Consider a content adaptation controller that sets the quality level of the content served based on the measured load from a server. Now, consider a point in time when the offered load is low enough for the server to serve full content to all requests. At this point, an adversarial burst in demand arriving in a very short time would push the server into overload. This, in turn, would result in the content adaptation controller reducing the quality of the content being served, to bring the server out of this overload state. Due to thrashing in the overload state, legitimate clients observe an increase in their response times and some receive degraded content. Recovering from such conditions takes a long period of time, but once the server stabilizes, the attacker would repeat this

[†] This work was partially supported by NSF research grants CISE CSR Award #0720604, ENG EFRI Award #0735974, CISE CNS Award #0524477, and CISE CNS Award #0520166.

[1]Content adaptation is also employed to support a heterogenous set of clients accessing a service via different devices, which is outside the scope of this work.

process by subjecting the server to a new adversarial burst.

Paper Outline: In Section II, we present a discrete-time model that studies the effect of RoQ attacks on a system employing a content adaptation controller. In Section III, we present results from our experimental implementation. We discuss related work in Section IV and we conclude the paper with a summary in Section V.

II. Model

In this section, we present a discrete-time model that studies the effect of RoQ attacks on a system employing a content adaptation controller.

A. Model Derivations

A system employing content adaptation can be modelled using three components: the content adaptation controller, the server and the feedback monitor. The content adaptation controller decides the quality level of the served content based on the overload conditions the server is experiencing. Deciding the quality level is typically a discrete process whereby the controller chooses between l different levels of quality. For example, the simplest content adaptation controller chooses between two levels, "Full" versus "Degraded" content. These different levels of content are prepared off-line to save the resources needed in having them prepared on-line and specially during overload conditions. The server simply serves requests based on the degradation level. The monitor measures the load on the server and reports this value back to the content adaptation to effect its further decisions.

Parameter	Description
ρ_i	Server utilization at time step i
$\bar{\rho}_i$	Average server utilization at time step i
λ_i	Legitimate arrivals at time step i
r_i	Total resource units utilized at time step i
α_i	Percentage of units used at time step i
μ_i	Service rate at time step i
U	Units used to serve a request fully
R	Total resource units available
C	System's capacity (maximum service rate)
ρ_T	Thrashing threshold
ω	Thrashing index
$\bar{\rho}_O$	Utilization operating threshold
δ	Attack rate
τ	Attack duration
T	Attack period
M	Attack magnitude

TABLE I
MODEL PARAMETERS.

Table I summarizes the notation and the description of the parameters used in our model described below.

We consider a discrete-time model where time is divided into equally-sized slots, indexed by the parameter i. During each time slot, requests arrive with a rate λ_i. We assume that all requests are identical and each requires U units of a particular resource to be fully served. However, during overload periods, only a percentage α_i of those units would be allocated to an incoming request. Thus, the resource utilization ρ_i at time slot i is given by:

$$\rho_i = \frac{r_i}{R} \quad (1)$$

where r_i is the total units utilized at time i by all requests inside the system and R is the total number of resource units available.

The total number of resource units, r_i, evolves according to the following equation:

$$r_i = r_{i-1} + \lambda_i \times (\alpha_i U) - \mu_{i-1} \quad (2)$$

where the first term indicates the units still used from the previous time slot, the second term indicates the total new units required by incoming requests, based on the level of degradation α_i and the last term indicates how many units are freed according to the service completion rate μ_i.[2]

The service completion rate μ_i is given by:

$$\mu_i = \begin{cases} C & \rho_i < \rho_T \\ C - \omega(\rho_i - \rho_T) & \text{Otherwise} \end{cases} \quad (3)$$

Equation 3 indicates that as long as ρ_i is less than some thrashing threshold ρ_T, the system operates at its rated capacity C. Once ρ_i exceeds ρ_T, the effective capacity of the system is reduced. The *thrashing index* ω is a constant that represents the severity of degradation in service rate as ρ_i increases beyond ρ_T.

Due to the bursty nature of incoming requests coupled with the variability in the number of resource units used, ρ_i exhibits short-time variations that prevents it from being used directly as a signal for control. Thus, we rely on an Estimated Weighted Moving Average (EWMA) value for a smoother control. Thus, the average utilization $\bar{\rho}_i$ evolves according to the following equation:

$$\bar{\rho}_i = (1 - \beta)\bar{\rho}_{i-1} + \beta\rho_i \quad (4)$$

where β is the EWMA smoothing parameter chosen between 0 and 1. β decides the weights given to memory versus current conditions. When β is close to 1, current utilization dominates the average utilization. However, when β is close to 0, the history of the average value dominates the new average utilization. Based on the average utilization, the content adaptation controller decides the degradation ratio α_i. One simple 2-level content adaptation controller would set the degradation ratio α_i according to the following equation:

[2]Notice that we measure the service completion rate in resource units per time as opposed to requests per time since requests utilize different number of resource units based on the degradation value.

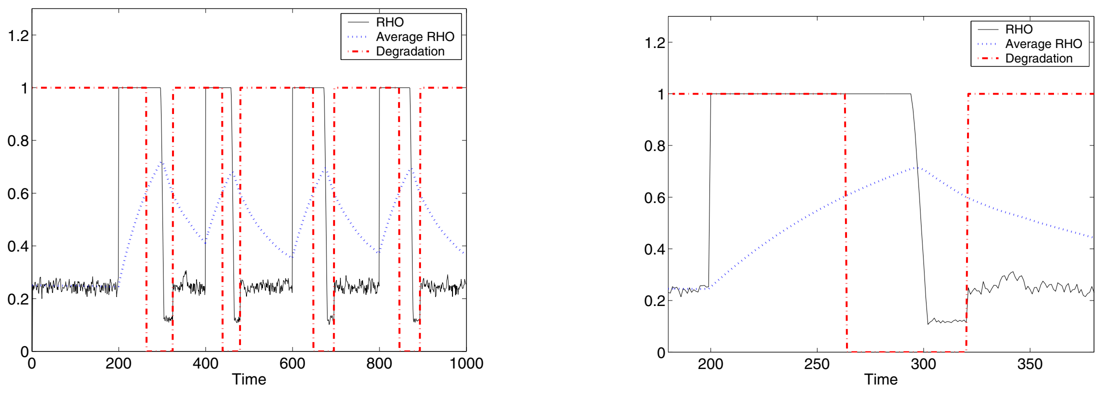

Fig. 1. Numerical results: assessment of vulnerability to RoQ attacks. Left plot shows the utilization, average utilization and degradation percentages on the Y-axis. Right plot zooms into a single attack burst.

$$\alpha_i = \begin{cases} 1 & \bar{\rho}_i < \bar{\rho}_O \\ d & \text{Otherwise} \end{cases} \tag{5}$$

where d is the fraction of resource units granted for an incoming request that receive degraded content during overload. Equation 5 states that an incoming request will be granted full resource units when $\bar{\rho}_i$ is less than a given operating threshold $\bar{\rho}_O$. Otherwise, an incoming request will be granted a degraded fraction d of the total resource units requested.

B. *RoQ Attack Definition and Goal*

We consider a RoQ attack on the system outlined above. The attack is comprised of M requests sent at the rate of δ requests per second over a short period of time τ, where $M = \delta\tau$. This process is repeated every T units of time. We call M the *magnitude* of the attack, δ the *amplitude* of the attack, τ the *duration* of the attack, and T the *period* of the attack. We refer the reader to some of our previous work on the theoretical backgrounds of RoQ attacks [8], [1].

For the above RoQ attack, we define Π, the *attack potency*, to be the ratio between the *damage* caused by that attack and the *cost* of mounting such an attack. Clearly, an attacker would be interested in maximizing the damage per unit cost—i.e., maximizing the attack potency.[3]

$$\text{Potency} = \Pi = \frac{\text{Damage}}{\text{Cost}^{\frac{1}{\Omega}}} \tag{6}$$

The Potency definition given by equation 6 allows us to consider different instantiations of "damage" and "cost". In this paper, we consider two instantiations for potency. We define Π_1 to be:

$$\Pi_1 = \frac{L_D}{M} \tag{7}$$

[3]The Potency definition in equation 6 allows for a parameter Ω to model the aggressiveness of the attacker [8]. We take Ω to be 1.

where L_D is the number of legitimate requests that received degraded content (damage) due to single attack burst and M is the magnitude of a single attack burst (cost). Thus, potency Π_1 reflects the impact of a single attack request on degrading the content served to a number of legitimate requests.

We also define Π_2 to be:

$$\Pi_2 = \frac{L_R}{M \times T_s} \tag{8}$$

where L_R is the response time experienced by the legitimate requests as a result of a single attack burst. T_s is the service time for a request. $M \times T_s$ represents the time spent by the server in serving the attacking burst. Thus, potency Π_2 reflects the relative delay caused to legitimate clients per unit time spent by the server serving the M requests.

C. *Numerical Solution*

We numerically instantiate a model from the equations above and we solve them numerically. We assume that legitimate requests arrive at a rate λ of 290 requests per second (chosen to be less than the effective service rate of 300). Each request requires 10 resource units to be served fully. The total number of resource units available R at any point in time is 12000. According to Equation 3, we set C to be 3000, ω to be 0.00033 and thrashing threshold ρ_T to 0.8. The average utilization is computed based on a value of β of 0.01. According to Equation 5, we choose $\bar{\rho}_O$ to be 0.6 and d to be 0.5.

Figure 1(left) shows the results obtained under a RoQ attack. The attack is compromised of bursts, each of 1000 requests in magnitude injected in 1 second, and repeated every 200 seconds. Figure 1(right) zooms into the first attack burst occurring at time 200. The arriving burst pushes the server instantly into overload as indicated by its utilization hitting 1. From that point on, the average utilization starts catching up with the current utilization until it reaches 0.6 (happens around time 260). At this point in time, the content adaptation adjusts its degradation value and the server starts serving degraded content for new arrivals (indicated by the degradation line hitting 0). This, however, does not remove the strain on the server resources instantly since the server has to get rid of the burst. Notice that this takes longer time due to thrashing. Only at time 300, when the server serves all accumulated requests as indicated by its utilization dropping to its lowest value. Again due to the time it takes the average utilization to catch up with the current utilization, degraded content is still being served. Only at time 320 does the server completely recovers.

The attack burst has resulted in three different forms of damage:

1) An increase in the response time for legitimate requests. In particular, from time 200 till 300, legitimate requests experience an increase in their response time as indicated by the utilization at its maximum value while the system is thrashing.
2) Degraded content served for legitimate clients. Requests arriving from time 270 till 320 are all being served degraded content, leading to a potency π_1 of 20.
3) Under utilization of resources on the server. Notice that from time 300 till 320, the utilization is actually below its normal value, indicating a waste of resources that could have been used in serving full content as opposed to degraded content.

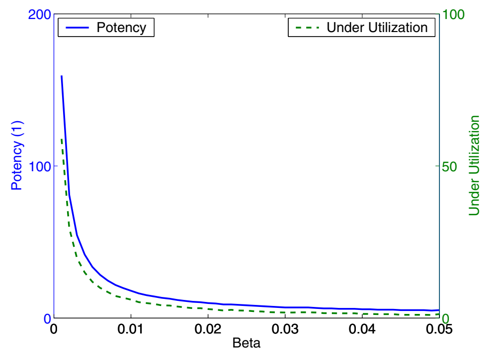

Fig. 2. Impact of β on Potency.

Clearly, the behavior of the system depends on our choice of parameters. β plays an important role since it dictates how aggressively the average utilization value is adjusted based on the the current utilization. Figure 2shows the effect of β on the potency (Π_1) (left Y-axis) and on the under-utilization (right Y-axis). Under-utilization is measured as the time wasted in serving degraded content, when it could have been utilized to serve full content. One can observe that a larger value of β reduces the impact of RoQ attacks. This is indeed the case, since the system reacts more aggressively to any changes. However, this comes at a price. In particular, a larger value of β also implies that the system will react swiftly to minor changes in its workload. Under normal operations, this is quite undesirable as it compromises stability. Notice that the model we adopted in this section does not capture the effect of long feedback delays as it assumes a feedback delay of unit value. In a practical setting, this feedback delay will slow the system's reaction making it even more vulnerable.

Notice that due to the fluid-like nature of our model, we could not track the response time for each individual request, and hence we could not compute Π_2 precisely. This, however, will be assessed by our Internet experiments, since one can track the response time for each individual request.

III. Internet Experiments

In this section, we evaluate the impact of RoQ attacks on content adaptation mechanisms through experiments performed in our lab.

The experimental setup: The experimental setup consists of three Dell Optiplex GX260 machines, with an Intel Pentium 4 CPU (2.2GHz) and 1 GB of memory. All machines run Red Hat Linux 2.6.9-11.EL. The first machine is a server running Minihttpd [9] web server. We have modified Minihttpd to employ a content adaptation controller within. A monitor application measures the load on the memory (or an average function of it) and reports this value back to the controller via shared memory. One of the other two machines is used to generate a legitimate workload whereas the other one is used to generate a malicious workload. Traffic workload generation is done using Httperf [10].

We assume that all requests are identical, however, the exact response from the server depends on the operation of the content adaptation controller. To serve a request in full, the server allocates 6 MBs of memory and writes data into them. To serve a content-degraded request, the server allocates 3 MBs of memory and writes data into them. The time it takes to serve a full-content and a degraded-content request is 224 msec and 122 msec, respectively. To smooth out the transient dynamics of the memory utilization, we use an EWMA function with parameter β (Equation (4)). The operating threshold $\bar{\rho}_O$ is chosen to be 0.65. We assume that legitimate traffic arrives to the server at a rate of 4 requests per second and each request has a timeout value of 50 seconds. Each experiment lasts between 5 and 9 minutes.

Impact of RoQ attacks: Figure 3 illustrates the results obtained from a single attack burst of magnitude 200 injected at a rate of 10 requests/sec (around time 60). The averaging parameter β is chosen to be 0.01. One can see the impact of the burst on the memory utilization, average memory utilization and the degradation. We also plot the swap utilization, since once the memory utilization hits 1, the system starts paging to/from disk. Clearly, this reduces the effective capacity of the system due to the time wasted in paging. Notice the resemblance between the experimental results in Figure 3 and those obtained from the model in Figure 1.

The attack burst above has resulted in 288 legitimate requests receiving degraded content, leading to a potency π_1 of 1.44 (Equation 7). Over this attack period, 334 legitimate requests observed a response time of 21.9 seconds, on average. Given that the total time spent by malicious requests getting

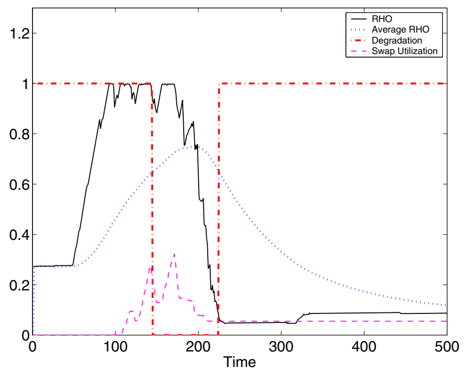

Fig. 3. Experimental results for a single attack burst.

service is 44 seconds, we compute the potency π_2 to be 166 (Equation 8).[4] That is for every 1 sec spent by the system serving malicious requests, 166 seconds get observed as a response time for legitimate requests. The RoQ attack above has also resulted in 401 legitimate requests timing out (their response time has exceeded the timeout value). Additionally, the RoQ attack has resulted in 25 seconds of under-utilization. This is the time wasted in serving degraded content, when full content could have been served.

Impact of the parameter β: To assess the impact of β on different metrics, we repeat the above experiment with different values of β. Table II depicts the results.

β	π_1	π_2	Timeouts	Under-utilization
0.01	1.4	166	401	25
0.05	1.4	171	138	11
0.1	1.3	177	114	7
0.5	1.2	189	90	1
0.9	1.2	241	87	0

TABLE II
IMPACT OF β ON DIFFERENT PERFORMANCE METRICS.

Except for π_2, one can observe that the higher the β, the less the impact of the RoQ attack (similar to our numerical results in Figure 1 (right)). The case of π_2 is understood once is put in context with the timeout values. The reason of the lower value of π_2 is due to many requests timing out (at lower values of β) and their response time could not be included in the calculation of π_2. While this may suggest using a higher value of β to minimize the impact of RoQ attacks, using a higher value of β makes the system less stable, since any perturbations in the memory utilization will reflect in the average and hence would cause unnecessarily degradation. This trade-off is important to highlight.

Impact of different attack parameters: In this set of experiments, we vary the attacking rate δ, keeping the Magnitude of the attack constant at 200 requests. Since $M = \delta \times \tau$, this also corresponds to adjusting the attack duration accordingly. β is chosen to be 0.1. Figure 4 shows π_1 and π_2 for different attacking rates.

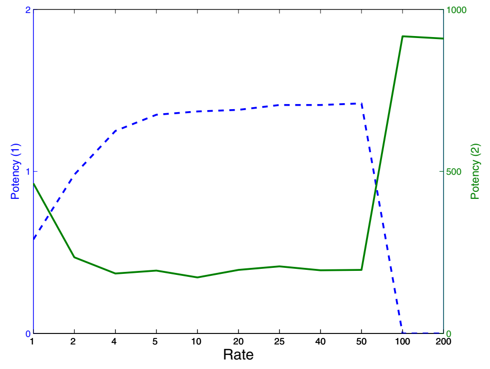

Fig. 4. Impact of different attack rates (δ).

At low attacking rates, the server observes a slight increase in the load over time. This extra load consumes memory at a lower pace and as a result, the server would only need to degrade the content occasionally to keep the utilization below its operating threshold. This would result in a low attack potency π_1. Due to the small level of degradation, the response time is still high since degradation typically reduces the response time.

As the attacking rates increase, more malicious requests get into the system in a shorter period of time. Their number is still small enough for Linux to handle them all together. This consumes memory very fast and as a result causes degradation. That is why we observe higher potency π_1 yet a lower potency π_2, due to degradation improving the response time for legitimate requests. Notice the existence of a range of rates that lead to a diminishing return effect, where varying the rate does not impact the potency values much.

At very high attacking rates, Linux cannot handle those requests and starts killing threads and putting others to sleep. This causes the memory utilization to remain constant, which in return prevents the content adaptation controller from taking over. This leads to 0 potency π_1, at the price of a very high response time (high potency π_2), since most the time spent by Linux is in context switching between the current running threads.

The above experiments highlight the trade-offs the attacker has on inflicting different forms and levels of damage, depending on its choice of attacking parameters. Such choices can also be manipulated in a smart manner to evade being detected by intrusion detection systems. For example, by keeping the attacking rate just low enough such that the attack does not register with the intrusion detection system.

[4]Notice the we do not consider the queuing delay for the malicious requests, since the attacker would not be waiting for the service.

IV. Related Work

The work presented in this paper relates to two main areas of research: (1) Server overload and (2) Security.

Server Overload: There is a large body of work that investigated the server overload problem and proposed different approaches to mitigate it and to decrease the response time experienced by clients. Such approaches can be broadly classified along two dimensions, static and dynamic. Static polices aim to increase the capacity of the system by over-provisioning the resources needed. Clearly, this wastes resources when they are not used. Dynamic policies, on the other hand, rely on adapting the traffic workload or the operation of the resource, based on the level of overload. These polices are typically employed via load-balancers [11], [12], admission controllers [13], [14], [15], [16], schedulers [17], [18] and content adaptation controllers [3], [4], [5]. Load balancers and schedules typically do not mitigate overload directly but aim to improve the response time for clients by making judicious decisions. Admission controllers and content adaptation controllers, on the other hand, reject or degrade the quality of responses to mitigate overload, respectively.

Security: DoS attacks [19], [20] aim to deny access to legitimate clients from accessing resources through targeting the capacity of those resources. In contrast to DoS attacks, RoQ attacks aim to maximize the attack potency without sustaining the attack traffic. Such stealthiness makes them hard to be detected and defended against. We have exposed other instances of RoQ attacks on admission controllers [1], load balancers [2] and active queue management schemes [8]. Other work in the area of stealthy attacks include [21], [22]. In [21], the Shrew attack is exposed that targets the timeout mechanism of TCP. The work in [22] shows how attackers are switch from simple flooding to other forms of attacks that mimic web browsing for a larger subset of clients in order to evade detection.

V. Conclusion

This paper demonstrated that content adaptation mechanisms, while mainly employed to mitigate overload, are in fact vulnerable to a new instantiation of RoQ attacks. Through exploiting dynamics, this new instantiation of RoQ attacks is capable of inflicting different forms of damage on legitimate clients (such as an increase in their response time and degraded content being served to them) and on the server (under utilization of resources). We were able to identify the key parameters that affect the impact of RoQ attacks as well as expose the tradeoffs between efficiency and susceptibility to attacks. Our assessment was performed through different instantiations of the potency metric, since it reflects the best interest for an attacker. We have confirmed our results with Internet experiments performed in our lab.

References

[1] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Reduction of Quality (RoQ) Attacks on Internet End Systems," in *Proceedings of IEEE INFOCOM*, Miami, Flroida, March 2005.

[2] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Reduction of Quality (RoQ) Attacks on Dynamic Load Balancers: Vulnerability Assessment and Design Tradeoffs," in *Proceedings of the 26th IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[3] T. F. Abdelzaher and N. Bhatti, "Web Content Adaptation to Improve Server Overload Behavior," *Computer Networks*, vol. 31, no. 11–16, pp. 1563–1577, 1999.

[4] M. Andersson, M. Host, Jianhua Cao, C. Nyberg, and M. Kihl, "Design and Evaluation of an Overload Control System for Crisis-Related Web Server Systems," in *Proceedings of the International Conference on Internet Surveillance and Protection (ICISP)*, Cote d'Azur, France, August 2006.

[5] J. Cho, S. Lee, and E. Lee, "Multi-agent Based Hybrid System for Dynamic Web-Content Adaptation," in *Proceedings of the 7th International Conference on Intelligent Data Engineering and Automated Learning*, Burgos, Spain, September 2006.

[6] U.S. Department of Energy (DOE) and U.S. Environmental Protection Agency (EPA), "Fact Sheet on National Data Center Energy Efficiency Information Program," March 2008.

[7] IDC Press Release, "Demand for Increased IT Efficiency Drives Worldwide Server Virtualization Adoption, IDC Says," January 2007.

[8] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources," in *Proceedings of the International Conference on Networking Protocols (ICNP)*, Berlin, Germany, October 2004.

[9] "mini_httpd: small HTTP server," http://www.acme.com/software/mini_httpd.

[10] D. Mosberger and T. Jin, "Httperf: a tool for measuring web server performance," in *Proceedings of the First workshop on Internet Server Performance*, Madison, WI, June 1998.

[11] V. Cardellini, M. Colajanni, and P. Yu, "Dynamic load balancing on web-server systems," *IEEE Internet Computing*, 1999.

[12] J. Watts and S. Taylor, "A Practical Approach to Dynamic Load Balancing," *IEEE Trans. Parallel and Distributed Systems*, 1998.

[13] M. Andersson, M. Kihl, and A. Robertsson, "Modelling and Design of Admission Control Mechanisms for Web Servers using Non-linear Control Theory," in *Proceedings of ITCom*, Orlando, FL, September 2003.

[14] A. Robertsson, B. Wittenmark, and M. Kihl, "Analysis and Design of Admission Control Systems in Web-server Systems," in *Proceedings of American Control Conference*, Denver, CO, June 2003.

[15] S. Lim, C. Lee, C. Ahn, C. Lee, and K. Park, "An Adaptive Admission Control Mechanism for a Cluster-Based Web Server System," in *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, Fort Lauderdale, Florida, April 2002.

[16] J. Carlstrom and R. Rom, "Application-aware Admission Control and Scheduling in Web Servers," in *Proceedings of Infocom*, New York City, NY, 2002.

[17] B. Schroeder and M. Harchol-Balter, "Web servers under overload: How scheduling can help," *ACM Transactions on Internet Technology (TOIT)*, February 2006.

[18] D. Andresen and T. Yang, "Multiprocessor scheduling with client resources to improve the response time of WWW applications," in *Proceedings of the 11th ACM/SIGARCH Conference on Supercomputing (ICS)*, Vienna, Austria, July 1997.

[19] CERT Coordination Center, "Denial of Service Attacks," http://www.cert.org/tech_tips/denial_of_service.html.

[20] CERT Coordination Center, "Trends in Denial of Service Attack Technology - October 2001," http://www.cert.org/archive/pdf/DoS_trends.pdf.

[21] A. Kuzmanovic and E. Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants)," in *Proceedings ACM SIGCOMM'03*, karlsruhe, Germany, August 2003.

[22] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds," in *Proceedings of NSDI*, Boston, MA, May 2005.