From Detection to Remediation: A Self-Organized System for Addressing Flash Crowd Problems

Linlin Xie, Paul Smith, and David Hutchison
Computing Department, InfoLab21[1]
Lancaster University,
Lancaster, UK
{linlin.xie,p.smith,dh}@comp.lancs.ac.uk

Mark Banfield and Helmut Leopold
Telekom Austria AG[2]
Lassallestraße 9
A-1020, Vienna, Austria
{mark.banfield, helmut.leopold}@telekom.at

Abdul Jabbar and James P.G. Sterbenz[1,3]
Dept. of Electrical Engr. and Computer Science[3]
Information & Telecommunication Tech. Center
The University of Kansas, USA
{jabbar, jpgs}@ittc.ku.edu

***Abstract*—A flash crowd event can be characterised by a dramatic increase in requests for a service over a relatively short period of time. Often, these events lead to a loss of service because of the saturation of the target server and associated network resources. This paper presents a set of mechanisms that can be used to make Web servers and associated resources more resilient to flash crowd events. Specifically, we present a novel admission control mechanism that uses a detection mechanism we developed in earlier work to adjust the admission rate of HTTP requests to a Web server. We demonstrate, via simulations, that the admission control mechanism can be used to protect a Web server from the effects of a flash crowd event, protect the traffic of other services that are hosted on the same network as a targeted Web server, and in combination with a push-back mechanism reduce the effect of flash crowd traffic on an ISP's network that is serving the Web server. The mechanisms presented here are exemplars that fit within a resilience strategy we are developing – D^2R^2+DR – which is summarised here.**

I. Introduction

The objective of our research, in general, is to develop resilient network systems, where resilience is defined as the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operations. In this paper, we target one such challenge to normal operations of a Web server: flash crowds.

A flash crowd event can be characterised by a dramatic increase in requests for a service over a relatively short period of time. A Web-based flash crowd event may cause three major problems: the saturation of the server-side resources, which is comprised of the computational and memory resources of the Web server and its associated access link; the exhaustion of the resources in the network that carries the request traffic to the server, in which only a reduced number of requests can be serviced; and the adverse effect on the performance of the cross traffic that shares the same links with the unusually large volumes of response traffic, or on the traffic distribution balance within the network.

In order to be resilient a network service must have a specific strategy to address various challenges to normal operations. We propose a two-phase strategy represented as D^2R^2+DR [8]. Briefly stated, the first phase of resilience strategy (D^2R^2) involves defence against and detection of an adverse event or condition, remediation of the effects after the occurrence to minimize the impact on system performance and finally, recovery to normal operations at the end of event duration. The second phase of resilience (DR) is a two-step background process of diagnosing the root cause of faults and refining the system behaviour based on previous D^2R^2 cycles. In the following sections, we will discuss specific mechanisms that we have developed for each step of the resilience strategy. Since the traffic involved in flash crowds is in fact legitimate, we cannot devise effective defence mechanisms against flash crowds without significant over-provisioning of the ISP network and access links.

In previous work [9], we developed a mechanism (summarised here in Section II-A) that can be used to detect the onset of a Web-based flash crowd event or DDoS attack and its associated impact on a target server. In this paper, we propose a novel admission control mechanism (described in Section II-B) that can be used to remediate against the effects of a flash crowd event. In summary, the flash crowd detection mechanism produces ratio values that are used as a parameter for triggering and adjusting the aggressiveness of the admission control mechanism. This is carried out in an automatic manner, which is essential in this scenario because of the unpredictable and rapid nature of flash crowds.

We demonstrate, via simulations that are presented in Section III, that the admission control mechanism can be used to protect a Web server from the effects of a flash crowd event, protect the traffic of other services that are hosted on the same network as a targeted Web server, and in combination with a push-back mechanism reduce the effect of flash crowd traffic on an ISP's network that is serving the Web server. We propose to use programmable networking technology [1] to enable both detection and remediation, and support the implementation and deployment of these services.

The final step is to recover the system after the adverse event has passed. In this case, the proposed system detects the cessation of flash crowds and restores the normal operation of the Web server. In this paper, we primarily focus on the first phase (D^2R^2) of the strategy. The second phase (DR) will be dealt with in future work, which is discussed in Section V.

II. Mechanisms

In this section we summarise our flash crowd detection mechanism and describe how it can be used in conjunction with a simple admission control mechanism to remediate against the effects of a flash crowd event.

A. Flash Crowd Detection Mechanism

The detection mechanism used in [9] makes use of the power-law distribution relationship between request number rate and response volume rate. From an observed incoming request rate, it predicts the corresponding response rate from a server. The inconformity of the relationship indicates the saturation of a server's resources, and is decided by a specific confidence range $[LowerLimit, UpperLimit]$ of the distribution. Two continuous ratios below the $LowerLimit$ value confirm the onset of problems caused by a flash crowd event. For example, if at time t_{i-1} the ratio is still within the confidence range, but at t_i and t_{i+1} the ratio values are both below the $LowerLimit$, at t_{i+1} a problem is confirmed and remediation mechanisms can be initiated. The detection time interval is determined based upon a trade-off between the resources needed to calculate the ratio values and timeliness. We envisage this mechanism executing on the edge routing device(s) of a Web server.

B. Admission Control of HTTP Requests

As mentioned before, a resilient system must ensure a certain level of service during a flash crowd event. The aim of the admission control mechanism is to protect the resources associated with a server and be able to service a percent of the HTTP requests without shutting down. We achieve this by admitting a limited number of HTTP requests to a server, the rate of which is governed by the detection mechanism described in Section II-A. Additional requests can either be dropped (as is done in the simulations presented here) or redirected to Web caches, client caches, or CDNs, for example.

There are two forms of shaping that can be used to manage a request flood – dropping HTTP requests or TCP SYN packets. Dropping TCP SYN packets may not lead to an immediate reduction of server load since existing connections continue to issue service requests. However, this approach ensures that existing connections receive a better service as compared to random HTTP request drops. The latter strategy could result in requests being resent by clients thereby aggravating the problem leading to longer delays for successful requests. Another approach, especially useful in differentiated service model, is to drop requests preferentially (instead of random drops). This, however, requires application-level information, which may be computationally expensive to gain.

An important parameter in traffic shaping is the dropping rate or its complement, the admission rate. Based upon the example presented in Section II-A, at the first onset of flash crowd event the admission rate is set to the rate of HTTP requests (or TCP SYN packets) at t_{i-1} – the last known rate that did not cause degradation in server performance. From there on, the rate is adjusted according to the subsequent ratio values as calculated using the mechanism described in Section II-A.

In this simple rate adjustment mechanism, when two consecutive request to response ratios (R_{curr} and R_{prev}) are below the $LowerLimit$, indicating degradation of server performance, the admission rate is decreased by a fixed percentage (e.g., 10% of the current rate); this is termed as a *degrade*. When both R_{curr} and R_{prev} are within the confidence range, the bandwidth utilisation of the access link from the Web server to the detection module decides the next action. The bandwidth utilisation, U, is defined as the ratio of the used bandwidth (R_{used}) over the provisioned bandwidth (R_{prov}), i.e., $U = R_{\text{used}}/R_{\text{prov}}$. If utilisation is below a certain threshold (say 90%), meaning the requests are under-admitted, then the admission rate is increased (e.g., 10% of the current rate); this is termed as an *upgrade*.

To dampen fluctuations in the admission rate and avoid flapping between upgrade and degrade actions, three states are used: *Unknown*, *Trial* and *Stable*. The state transitions are illustrated in Fig 1. Stable refers to the state of normal operations of the system. In this state if a flash crowd event is detected (by two consecutive ratio values below the *LowerLimit*), admission rate is degraded and the system moves to the *Unknown* state. Once in the unknown state, several degrades are performed to remediate server performance (to a value within confidence range). When the system in unknown state satisfies the upgrade condition (i.e., R_{curr} and R_{prev} are both within the confidence range and utilisation is below a defined threshold), indicating a letup in flash crowd, an upgrade is performed and the state migrates to *Trial*. In trial state, the admission rate is upgraded until the number of admitted service requests is maximized. The following upgrade results in degrade condition being satisfied, indicating the maximum allowable limit has reached. Hence, the admission rate is degraded once and the system migrates to stable state indicating a service recovery. Lastly, if an upgrade condition were to be met during the stable state, indicating excess available bandwidth, no upgrade is performed; instead the system migrates to unknown state, where it is upgraded in the following time step.

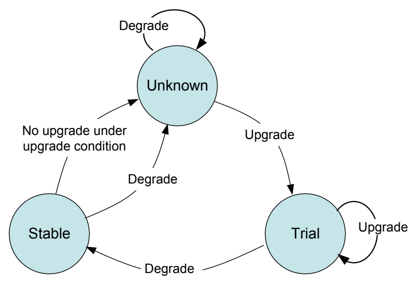

Fig. 1. State Transitions

When the incoming rate is lower than the specified accepted rate for a given period of time (denoting the end of flash crowd event), the rate-limiter and other remediation mechanisms can be terminated and the system can recover to a normal state.

III. Experimentation

Ns-2 version 2.29 is used to simulate the flash crowd traffic and the proposed mechanisms for detection and remediation. The flash crowd simulation parameters are presented in Table I. The results presented in Section III-A are representative of those from multiple runs of simulations; we do not show average values as they impede explanation of the results. All other simulation results are averages of three runs of the simulator. Three sets of simulations are performed: the first simulation evaluates the effectiveness of the dynamic rate limiting mechanism described in Section IIB; the second shows the effect of adjustment mechanism on cross traffic performance; and the third demonstrates the effects of bandwidth resource protection using a push-back mechanism.

TABLE I
Simulation parameters

Traffic Type	Session Number	Inter-Session Time(s)	Session Size	Inter-Page Time(s)	Page Size	Inter-Object Time(s)	Object Size
Background	10^3	1	15	1	10	0.01	Avg:12 Shape: 1.2
Flash Crowd	20000	0.025	10	1	10	0.01	Avg:12 Shape: 1.2

A. *Dynamic and Automatic Adjustment of Admission Rate*

The topology used to evaluate the effectiveness of the mechanism used to control the admission rate of requests to the server (described in Section II-B) is shown in Fig. 2.

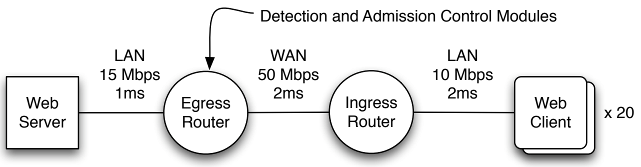

Fig. 2. Topology configuration for admission rate simulations

The detection module (described in Section II-A) is located at the egress (edge) router along with the admission control module. The detection epoch, where a new ratio value is calculated, is 30 seconds, U is set to 90% of R_{prov} (in this simulation 14.25 Mbps) and the upgrade and downgrade amount is set to 10% of the current request rate. For these simulations we performed random HTTP request dropping to manage the admitted rate of requests. A flash crowd event was started 500 seconds into the simulation.

Fig. 3 shows the request to response ratio values that are calculated over time by the detection module (each point on the plot represents an epoch where the ratio is calculated). We define the confidence range as $[0.62205216, 1.42968584]$, as discussed in [9]. Two consecutive ratio values outside this range at 510 (t_i) and 540 (t_{i+1}) seconds initiates the dropping mechanism, as discussed earlier.

Fig. 4 shows how the accepted request rate is managed by the detection and dropping mechanism. At 540 seconds (t_{i+1}) into the simulation (at the point of detection) the random dropping of HTTP requests is started. The admitted rate of HTTP requests is set to the rate at 480 seconds (t_{i-1}) $\sim$190 requests/second. At the next detection interval (570 seconds, as shown in Fig. 3) the ratio returns within the confidence range and the admitted request rate is shaped at 630 seconds to the admission rate at t_{i-1}.

Fig. 5 shows the effect of randomly dropping HTTP requests on the incoming request rate. It can be seen that this strategy results in a dramatic increase in HTTP requests being retransmitted; this is clearly an undesirable side effect of the strategy we adopt in these simulations. In Section III-C, we present simulations that show how this problem can be ameliorated on the network that serves the Web server by pushing back the admission control module to the network's ingress edge. As mentioned earlier, a better solution to dropping HTTP requests would be to redirect them to caches, for example. This is a matter for future work.

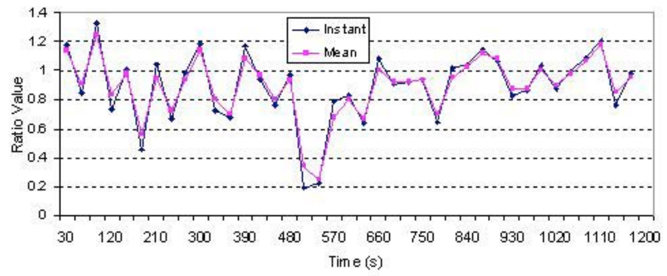

Fig. 3. Ratio value variation with dropping HTTP requests

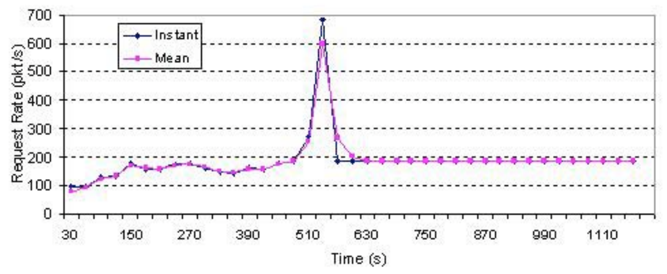

Fig. 4. Admission rate variation when dropping HTTP requests

B. *Cross Traffic Protection*

In the simulation we have presented thus far a single service is being provided on the server network – a HTTP Web service. It is possible that more than one service will exist on the server network that is the target of a flash crowd event. In these simulations we show, with some minor modifications, how our detection and admission control mechanisms can be used to ensure that traffic from other services can remain unaffected by a flash crowd event. Without performing this task the traffic associated with the other services would be adversely affected by the response traffic from the Web server.

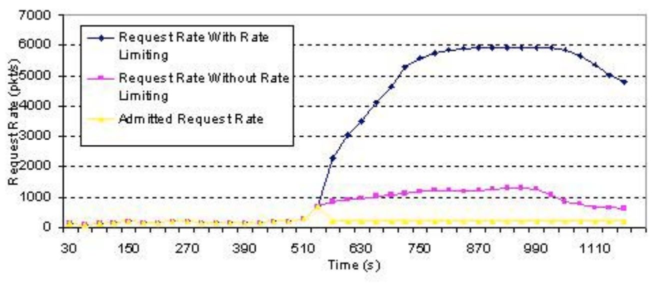

Fig. 5. Request rates with and without rate limiting

To protect traffic from other services, a modification is made to the way the utilisation of the link is calculated. Recall that this value is used to control whether an adjustment is made to the admission rate when two consecutive ratio values are both within the confidence range. If utilisation is greater than 90%, for example, the rate is decreased otherwise it is increased. To provision for other services, one simply removes the amount of bandwidth that is to be allocated to it from the value of R_{prov} in the following way: $U = R_{\text{used}}/(R_{\text{prov}} - R_{\text{resv}})$, where R_{resv} is the amount of bandwidth to be allocated to other services.

To determine the effectiveness of this approach at protecting traffic from other services from the effects of a flash crowd event we conducted some simulations. Fig. 6 shows the topology we used to achieve this. A new FTP service is introduced on the server network whose traffic we wish to protect. The FTP service is allocated 5 Mbps of bandwidth; our mechanism should ensure in light of the flash crowd event it is able to continue sending at approximately this rate while the Web service uses what remains of the available bandwidth. In these simulations the FTP clients commence operation at the start of the simulation, the Web clients begin at 200 seconds, and the flash crowd is initiated at 600 seconds into the simulation

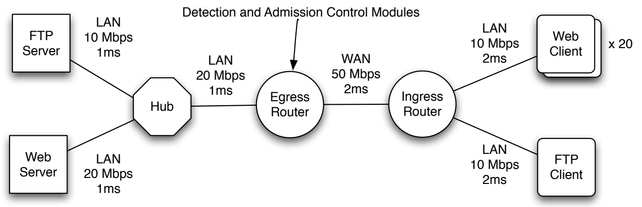

Fig. 6. Topology configuration for simulations with cross FTP traffic

The admitted HTTP request rate for this simulation is shown in Fig. 7. It can be seen that the flash crowd event is initiated at 600 seconds into the simulation. After two detection intervals at 630 and 660 seconds the admission control module begins to reduce the admitted request rate to acceptable levels, which converges to below 200 packets/second.

The relationship between the rates of the Web and FTP response traffic is shown in Fig. 8. Initially, when there is no Web traffic, the FTP traffic is being sent at the maximum allowed by the topology at 10 Mbps. When the Web service is started at 200 seconds, the FTP traffic is restricted to 5 Mbps. The effect of the flash crowd on FTP traffic can be seen at 630 seconds (before the flash crowd event is detected) – very little FTP traffic is being sent. After two more (30 second) intervals, the rate of the FTP traffic returns to approximately 5 Mbps with some variation. These simulations suggest that our admission control mechanism, with a minor adjustment, can be used to protect the traffic of co-located services of a flash crowd target.

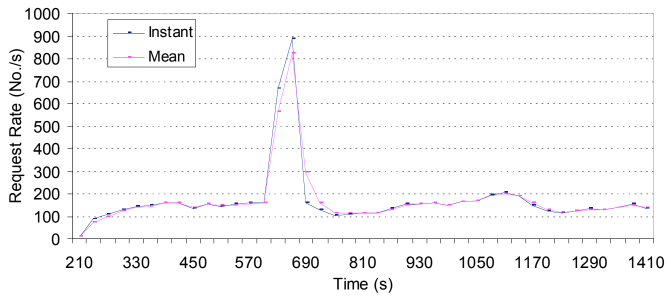

Fig. 7. HTTP request rate variations with cross traffic protection

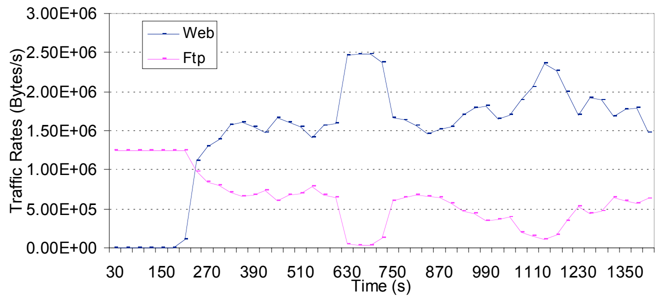

Fig. 8. Variations of Web and FTP response rates with cross traffic protection

C. *Pushing Back Request Traffic*

As the simulations presented in Section III-A demonstrated, as a consequence of dropping HTTP requests a significant number of retransmissions can be generated (this is shown in Fig. 5). As such, there is an incentive to migrate the admission control mechanism to the ingress edge of an ISP's network that is serving the target of the flash crowd.

We simulated the benefit of adopting this approach. To achieve this we modified the mechanism described in [5] and [3] that is used to push back traffic in light of a DDoS attack. (A simulated implementation of this is available in ∼ns/pushback/ of the ns-2.29 distribution.) We modified the push-back mechanism in the following ways: the push-back procedure is not triggered by network congestion, but by our detection module; there is no need to identify the High Bandwidth Aggregates (HBA) since we know the traffic type

(HTTP requests destined for the Web server) and the admission rate is determined by our algorithm; and we do not require rate limiters on routers inside the network.

The simulation topology configuration is presented in Fig. 9. The network in the oval represents a simple ISP network. The detection module, as before, is located at the egress router. Nineteen Web clients are used to generate the flash crowd traffic, and the twentieth Web client sends background traffic. The push-back process has to identify the real source of heavy flash crowd requests, and decide where and how much to drop the flash crowd requests.

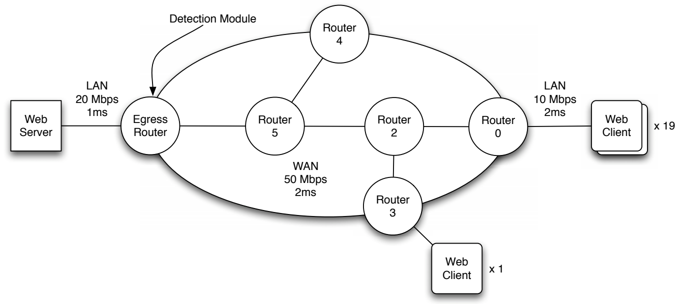

Fig. 9. Topology configuration for push-back simulations

Fig. 10 shows that the push-back mechanism was able to successfully determine the flash crowd origin at router zero and control the admitted request rate. No shaping is carried out at router three where there is little contribution being made to the flash crowd event. The motivation for pushing back request traffic is to reduce the effect of the flash crowd of the ISP's network serving the Web server; the ability of our approach to do this is shown in Fig. 11. Shown is the link utilisation (calculated as described in Section II-B) between router five and the egress router. It can be seen that without the push-back mechanism in place the link is fully utilised at periods when the flash crowd is at its peak, whereas the mechanism controls this problem. This would be an important feature to enable other services to operate on the ISP's network during a flash crowd event.

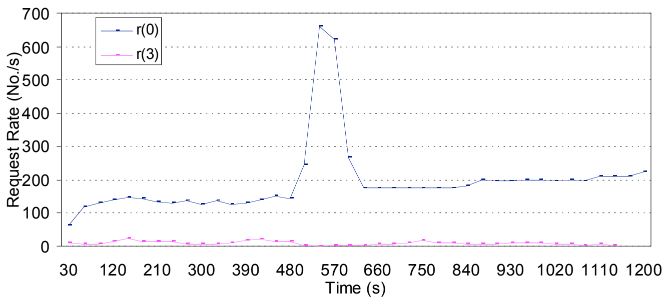

Fig. 10. Admission rate of HTTP request at ingress routers with push-back

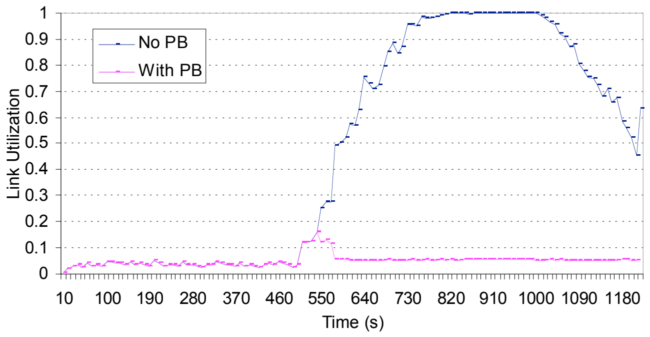

Fig. 11. Link utilisation with and without push-back

IV. Related Work

Much research has been carried out on remediating the effects of flash crowd events. These include Web caching [6], peer-to-peer caching [7], and Web-based CDN [4] techniques. As discussed earlier, these are complementary to our work – instead of performing simple random dropping as we do here, caching or CDNs could be used. A rate-limiting algorithm is proposed in [2] with the aim of mitigating the flash crowd effects on a Web server. We have shown we can achieve this with our approach, and protect other services on the server network and, by implementing a push-back mechanism, manage the problem on an ISP's network.

V. Conclusion

We have presented a novel admission control mechanism that can be used to remediate against the effects of a flash crowd event. We have shown in simulations that the approach we adopt has the potential to protect a Web server, services co-located with a targeted server, and an ISP's network from the effects of a flash crowd event. These are relatively early results. We need to perform further evaluation of the mechanism to determine its utility in a real-world context. For example, it is not clear what effect the detection interval has on the time to effectively remediate under different flash crowd conditions. Furthermore, it is not clear if our mechanism would work effectively in a multi-homed environment, for example, independently operating admission control modules could cause oscillations of admitted request rates and fail to converge.

The results presented here are part of ongoing research into resilient networked systems. The mechanisms we proposed are exemplars that fit into the D^2R^2 phase of our resilience strategy. Longer-term future work will investigate the diagnosis and refinement stages as ways of improving the effectiveness of the mechanisms through self-learning techniques.

Acknowledgements

Linlin Xie's PhD study and Paul Smith's research work are both funded by Telekom Austria, for which they are extremely grateful. Abdul Jabbar is supported in part by the US National Science Foundation.

References

[1] A. T. Campbell, H. G. DeMeer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela. A survey of programmable networks. *ACM SIGCOMM Computer Communication Review*, 29(2):7–23, April 1999.

[2] X. Chen and J. Heidemann. Flash Crowd Mitigation via Adaptive Admission Control Based on Application-Level Observation. *ACM Transactions on Internet Technology*, 5(3):532–562, August 2005.

[3] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. Technical report, AT&T Center for Internet Research at ICSI (ACIRI) and AT&T Labs Research, 2001.

[4] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *ACM WWW 2002*, Hawaii, USA, May 2002.

[5] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network (Extended Version). Technical report, AT&T Center for Internet Research at ICSI (ACIRI) and AT&T Labs Research, July 2001.

[6] J. A. Patel and I. Gupta. Overhaul: Extending HTTP to Combat Flash Crowds. In *International Workshop on Web Content Caching and Distribution (WCW2004)*, Beijing, China, October 2004.

[7] T. Stading, P. Maniatis, and M. Baker. Peer-to-Peer Caching Schemes to Address Flash Crowds. In *International Workshop on Peer-to-Peer Systems (IPTPS2002)*, Cambridge, MA, USA, March 2002.

[8] James P.G. Sterbenz and David Hutchison. ResiliNets: Multilevel resilient and survivable networking initiative web page. http://www.ittc.ku.edu/resilinets/index.html, 2007.

[9] L. Xie, P. Smith, M. Banfield, H. Leopold, J. P.G. Sterbenz, and D. Hutchison. Towards Resilient Networks using Programmable Networking Technologies. In *International Workshop on Active Networtks (IWAN) 2005*, Sophia Antipolis, Nice, France, November 2005.