Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites

Jaeyeon Jung
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
jyjung@lcs.mit.edu

Balachander Krishnamurthy
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932
bala@research.att.com

Michael Rabinovich
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932
misha@research.att.com

ABSTRACT

The paper studies two types of events that often overload Web sites to a point when their services are degraded or disrupted entirely - flash events (FEs) and denial of service attacks (DoS). The former are created by legitimate requests and the latter contain malicious requests whose goal is to subvert the normal operation of the site. We study the properties of both types of events with a special attention to characteristics that distinguish the two. Identifying these characteristics allows a formulation of a strategy for Web sites to quickly discard malicious requests. We also show that some content distribution networks (CDNs) may not provide the desired level of protection to Web sites against flash events. We therefore propose an enhancement to CDNs that offers better protection and use trace-driven simulations to study the effect of our enhancement on CDNs and Web sites.

Keywords

Web Workload Characterization, Flash Crowd, Denial of Service Attack, Content Distribution Network Performance

1. INTRODUCTION

Coined in 1971 in a science fiction short story [24], the term *flash crowd* referred to the situation when thousands of people went back in time to see historical events anew. The ease of teleportation enabled this situation. On the Web, the ubiquitous access of browsers and rapid spread of news about an event, leads to a similar situation when a very large number of users simultaneous access a popular Web site. Common examples of events include widely known ones as the release of Ken Starr's report on a few Web sites in 1999, popular webcasts like that of Victoria's Secret company, and sports events like the Olympics. In some cases, information about the occurrence of the events is known in advance. However, there have been several flash crowds with no advance warning. Often this is due to a catastrophic event, such as the September 2001 terrorist attack in the United States. Popular news sites such as `www.cnn.com` noticed a dramatic increase in the number of requests and many sites became unavailable.

Unlike science fiction, the effect of flash crowds on the servers operating at the Web sites and the network infrastructure is real and can be acute. Congestion at the network layer may even prevent some of the requests from reaching the servers, and if they do, it can be after significant delays caused by packet loss and retransmission attempts. Servers are unable to handle the volume of requests that actually reach it. Finally, users who are trying to obtain information during a flash crowd are often frustrated due to the resulting long delays or outright failures. In some cases, the Web server may have to be restarted due to the effects of the flash crowd [21]. Given the increase in the frequency of flash crowds and their overall unpredictability, it is important that we have a broad understanding of flash crowds.

Our contributions in this paper are threefold:

- We define the term *flash event* (an event that causes flash crowds) and provide a taxonomy to better understand its impact on Web servers.
- Before studying the impact of flash events on Web servers, we need to identify and separate a related but distinct phenomenon—that of denial of service attacks (DoS [13]). DoS attacks have been occurring fairly frequently in the last few years on the Internet. A DoS attack shares several characteristics with that of flash events but is not a flash event. Our taxonomy of flash event is used to clearly identify and separate DoS attacks from flash events.
- After separating DoS, we examine current solutions by Web servers for dealing with flash events. Specifically, we examine what roles Content Distribution Networks (CDNs) play and we offer some solutions of our own in this regard.

A flash event at a Web site can be predictable when a site is aware of the possibility of its occurrence. A common example is the on-line play-along Web site for a popular television program. The Web site typically does not receive many requests during the time the television show is not being broadcast and experiences a significant surge for the duration of the broadcast. A Web site can provision in advance for such flash events and handle them better. Another looser kind of predictable flash event is one where the content creator knows a priori that traffic volume is going to be significantly higher. For example, URLs that are advertised specifically during an event (such as widely-followed football games) can lead to expected high levels of traffic.

Copyright is held by the author/owner(s).
WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

The unpredictable category of flash events often arise due to the sudden prominence of a Web site unanticipated by the content owners. Medical Web sites may experience a sudden surge as a result of public concern over an epidemic. Alternately, a Web site that is discussed in another popular Web site may experience a flash event.

While Web sites can try to provision for the predictable flash events, they often fail to correctly predict the demand, as several well planned events (such as the Victoria Secret webcast) could attest [27]. Even when a Web site has prior knowledge that its site is going to experience a surge in demand, it may not have enough time to react and provision the necessary resources. The issue is complicated by competitive pressure where various sites are planning to display similar content (such as a major news story). Both predictable and unpredictable flash events can thus pose a serious risk to a Web site.

DoS attacks and flash events can both overload the server or the server's Internet connection and result in partial or complete failure. Unlike DoS attacks, which are simply malicious requests that *do not* have to be handled by a Web site, flash events consist of *legitimate* requests. A Web server has the responsibility to try and handle as many of the requests as possible during a flash event. By doing so, the site can increase its overall profile on the Web resulting in possible additional revenue. If a DoS attack occurs during a flash event, a Web server should aim to ignore DoS requests and handle the legitimate requests. This requires the Web site to be able to distinguish between the two sets of requests.

Content Distribution Networks (CDNs), have been in use for a few years primarily to help offload the request volume on Web servers during *normal* Web activity. A CDN typically serves a subset of resources (often static images) on behalf of a set of origin servers via an overlay network. Besides being used to improve routine access to Web sites, CDNs can also serve as insurance protection against flash events so that Web sites can continue to function normally. However, some CDNs are not always capable of dealing with flash events and we present circumstances under which a flash event can cause a Web site to fail even if it uses a CDN. The primary reason is the surge of cache misses in the beginning of flash events. In this paper, we thus propose an alternate approach, called an *adaptive CDN*, which addresses this problem.

The rest of the paper is divided as follows: Section 2 defines flash events and DoS attacks and discusses metrics that are useful to characterize them. Section 2.1 discusses the somewhat limited amount of related work in this area. Sections 3 and 4 present characteristics of flash events and DoS attacks, respectively, with a view towards distinguishing between them. Section 4.3 summarizes the behavioral differences of flash events and DoS attacks and develops some recommendations for Web servers based on these differences. Section 5 discusses the difficulties some CDNs face due to flash events and presents our notion of an adaptive CDN. We conclude with a look at ongoing and future work.

2. CHARACTERIZING FLASH EVENTS AND DOS ATTACKS

A flash event (FE) is a large surge in traffic to a particular Web site causing a dramatic increase in server load and putting severe strain on the network links leading to the server, which results in considerable increase in packet loss and congestion. A denial of service attack (DoS) is "an explicit attempt by attackers to prevent legitimate users of a service from using that service" [2]. We interpret this definition broadly—we consider any attempt to undermine a Web site to be a denial of service attack. Examples of attacks include TCP SYN flooding [1], HTTP request flooding including an attack to crack down password protected web pages, or attempting to crash a Web server such as the recent `Code Red` attack [23].

The key semantic difference between an FE and DoS is that the former represents legitimate access of the Web site while the latter does not. However, this does not help in distinguishing between the two automatically. One needs to develop behavioral differences between the two phenomena after understanding their individual properties. We characterize FE and DoS along the following dimensions:

- **Traffic patterns:** Traffic patterns as seen by the Web site are important for several reasons. Overall traffic volume determines how much a server should provision resources to keep the site operational up to a certain level. If server load exceeds its maximum tolerance level which is pre-defined by its capacity, the server begins to slow down and can be driven to a shutdown. Thus, watching traffic patterns allows us to articulate the period when an unusually large number of clients can overwhelm a site and how much time the server has from the start of an FE or DoS to take defensive measures.

- **Client characteristics:** Understanding client characteristics can help identify malicious attackers who intentionally stress a server pretending to be legitimate clients. When a server faces an unexpectedly heavy load, detecting DoS clients quickly would allow the server to reject their requests and concentrate resources on serving legitimate clients. In particular, we use a network-aware clustering technique [20] to determine the topological distribution of clients in FE and DoS. Client clustering allows one to aggregate individual clients into groups belonging to the same administrative domain. Clustering uses a large collection of unique network prefixes assembled from a wide set of BGP routing tables. The various client IP addresses are grouped into clusters based on longest prefix matching.

- **File reference characteristics:** File reference characteristics could potentially provide additional differentiators to rule out suspicious activities. Once it is clear that all requests come from legitimate clients, we can utilize file referencing behavior to deal with flash crowds. Locality of reference enables a reduction of server load through caching. We exploit these characteristics in designing an adaptive CDN. We consider aggregate file references as seen by the server, reference patterns of individual clients, and reference patterns of client clusters.

2.1 Related Work

There have been a number of studies that attempt to characterize the workload of Web servers. Ten invariants that applied across all six data sets studied were identified in [6].

The SURGE [7] tool was developed to generate Web workload matching empirical measurements of several aspects of Web server usage. Some studies focused on improving the utility of caching for Web clients based on traffic patterns observed by a proxy or web server [3, 11]. These efforts of Web workload characterization are aimed at understanding typical traffic behavior.

However, there has been little published on Web workload analysis when traffic is very high. The World Cup Web site study [5] presents a peak workload analysis pointing out that file referencing is more concentrated on a few extremely popular pages. Also many clients repeatedly visited the site with shorter inter-session time. A study of the 1998 Olympic Games Web site [16] analyzed the workload and developed traffic models that could be used to predict seasonal traffic variations and peak request rates at a Web site during its normal operation. Neither study is however, concerned with DoS attacks or implications on CDNs. A recent study [8] examined IP-flow-level traffic at the border router of the university network and presented flow characteristics of associated DoS attacks and flash crowd behavior. They found that flash crowd behavior was identified by rapid rise in traffic to a particular flow followed by a gradual drop-off over time. One incident of a DoS attack, on the other hand, was detected by a very sharp increase and decrease of the number of outgoing flows to a target server because each attempt was from a distinct source address and port number which appeared as a separate flow in the analysis. Our work focuses on HTTP-level traffic measurement and shows much detailed analysis of HTTP traces recorded at several Web sites.

Crovella *et al.* showed that under the high workload, shortest connection first scheduling algorithm improved the mean response time of Web servers by the factor of four or five [10]. A study of the effect of Ken Starr's report on NLANR Web caches by Wessels pointed out that inappropriate settings of *Expires* and *Last-Modified* header fields prevented ordinary caches from dealing effectively with flash events [28]. In contrast, our study presents an approach for improved flash crowd protection using CDNs assuming existing Web site behavior.

There have been a few suggestions, such as the pushback technique [12] and route-based packet filtering [25] for dealing with DoS attacks at the router level but our technique does not rely on changing router architecture and is complementary to anything that might bypass such a filter that may be deployed.

Requests from spiders are another type of HTTP requests that may inconvenience a Web site and, while usually legitimate, might be good candidates for being dropped by an overloaded site. Previous work on identifying these requests appeared in [20, 4] and is orthogonal to our study.

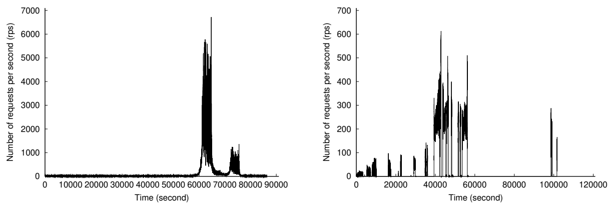
3. FLASH EVENTS

3.1 Data

To explore properties of flash events, we analyzed HTTP traces collected from two busy Web servers, one from the play-along Web site for a popular TV show (referred to as the `Play-along` site for short) and the other from the Chilean election site (referred to as the `Chile` site). Both traces reflect flash events. The `Play-along` site experienced such an event during the show time on TV. The `Chile` Web site hosted the continuously updated results of the 1999 presidential election in Chile.

Table 1 summarizes statistics of the two traces separately for the entire trace and for the FE segment, including the number of requests, documents and clients found in the traces. The table also lists the number of client clusters obtained by the network-aware client clustering technique [20].

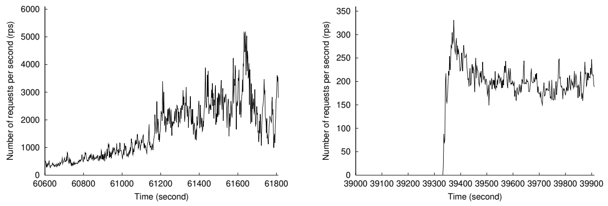
3.2 Traffic Volume



(a) `Play-along` (b) `Chile`

Figure 1: Traffic volumes (averaged over successive 2 seconds intervals)

Figure 1 shows the request rate experienced by the two sites over time. According to our definition of a flash event (FE) as a sudden growth in the request rate, we classify noticeable peaks as FE in each trace. The figure shows that the request rate grows dramatically during an FE but that the duration of the FE is relatively short. So, when provisioned to handle the peak load, servers would stay practically idle most of the time.



(a) `Play-along` (b) `Chile`

Figure 2: Request rate at the beginning of FE

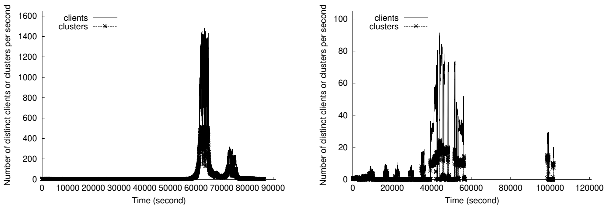
To see how much time a site has to react in the face of an unfolding FE, Figure 2 shows the traffic pattern when we zoom in on the first 15 minutes of the FE. Both figures show that request rate increase, while rapid, is far from instantaneous. In the `Play-along` case, the rate increase occurs gradually over the 15-minute interval from the time index of 60,600 second and reaches 6,719 req/sec at 64,700 second. The `Chile` trace has a much steeper rate increase, but even here the increase occurs over a few tens of seconds (from 39,334 second to 39,374 second) and reaches only half the peak rate: the peak rate of about 610 req/sec is reached only later, outside the time window shown in Figure 2(b). We use this property in Section 5 to let a CDN adapt to

Trace	Data size (Mbyte)		Duration	Requests	Documents	Clients	Clusters
Play-along	112.1	Total	02/05/00 (24h)	13,018,385	7,084	53,745	14,100
		FE	100 minutes	9,245,399 (71.0%)	4,887 (68.9%)	34,349 (63.9%)	8,689 (61.6%)
Chile	20.5	Total	12/12/99 (32h30m)	2,634,567	10,302	20,532	1,737
		FE	282 minutes	2,385,811 (88.2%)	9,388 (90.2%)	18,281 (89.0%)	1,507 (86.8%)

Table 1: Basic statistics for `Play-along` and `Chile` traces. Data size is counted over the traces compressed with a `gzip` utility

the unfolding FE so that it offers better protection to origin sites against overload.

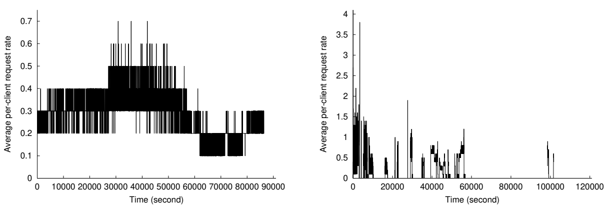
3.3 Characterizing Clients



(a) `Play-along` (b) `Chile`

Figure 3: Distribution of clients and clusters over time

Figure 3 shows how the number of distinct clients and client clusters accessing the site in 10-second intervals change over time. We see that the spikes in request volumes during an FE correspond closely with the spikes in the number of clients accessing the site. Thus, the number of clients in a flash event is commensurate with the request rate.



(a) `Play-along` (b) `Chile`

Figure 4: Average number of requests per client in a second

To verify this claim more directly, Figure 4 shows how the average request rate generated by a single client changes during and outside the flash event. The figure dos not show any clear increase in per client request rate during the flash event, at least for the two FEs we consider. The per-client rates rather drop and remain lower during the flash events than any other time slots. Thus, the increase in traffic volume occurs largely because of the increase in the number of clients.

The fairly stable per-client request rate can also be used to identify robots or proxies that automatically retrieve documents from the Web, which would behave more aggressively. We picked out several clients in this way from the `Chile` trace that turned out to be proxies that had sporadically sent tens of requests in a row.

Figure 3 also shows that the number of distinct clusters during the FE is much smaller than the number of distinct clients. This is because that cluster size distribution is very skewed – few clusters account for a large fraction of hosts – and therefore new clients come disproportionately from existing clusters, which is more apparent during the flash events. We will see that this is not true in the case of DoS, and is one of the strongest differentiator between FE and DoS that we were able to identify.

To better understand the dynamics of client clusters accessing each Web site, we divided the trace into before and during an FE and marked an incoming cluster during the FE as *old* if it did appear before it. We then calculate the percentage of old clusters as the ratio of old clusters over the number of distinct clusters during the FE. Surprisingly, we found that a large number of clusters active during an FE had also visited the sites before the event. The amount of the cluster overlap is 42.7% in the `Play-along` trace and 82.9% in the `Chile` trace. A possible explanation is that big clusters are responsible for most of requests to the Web site. With a high overall request rate, most of these big clusters would have at least one client that accessed the site already, so this cluster will not be new no matter how many more clients from it access the site.

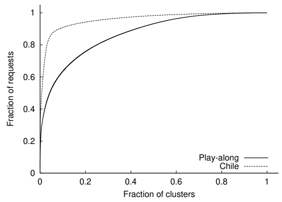


Figure 5: Cluster contribution to requests

To verify this explanation, Figure 5 shows the cumulative distribution of the requests contributed by clusters that visited the site. It shows that 10% of clusters contributed a majority of requests—60% to the `Play-along` site and 90%

to the `Chile` site. The distribution of requests across clusters is indeed highly skewed and reflects the skewed distribution of cluster sizes [20]. This suggests that client distribution in an FE follows user distribution across networks.

3.4 Characterizing File Reference

Understanding file reference patterns before and during a flash event is important for devising methods to handle FE effectively. With two traces described above, we observed the following file referencing behaviors in flash events.

(a) Over 60% (61% and 82% for `Play-along` and `Chile`, respectively) of documents are accessed only during flash events. This observation forms the basis of our insight of Section 5 that some current CDNs may not provide the desired level of protection against FE. Indeed, most CDN caches will not have these documents at the beginning of the FE; hence most of the requests at the beginning of the event would miss in the caches and be forwarded to the origin server. Although subsequent requests would be served from the caches, a CDN with many caches can send a large number of initial misses to the origin server in a short time.

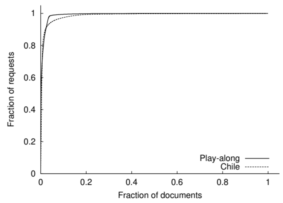
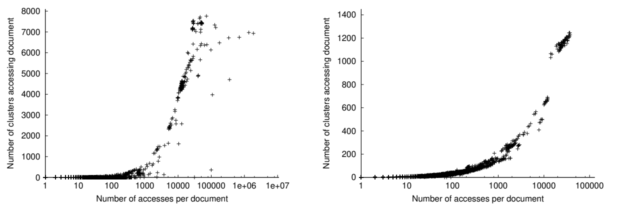


Figure 6: Cumulative distribution of document requests

(b) To deduce the popularity of documents during flash events in our traces, we plot the cumulative fraction of the requests accounted for by distinct documents, most popular first, in Figure 6. For both traces, less than 10% of most popular documents account for more than 90% of requests. This is a promising result, since it indicates that in principle, intelligent caching of these documents might be able to address the FE problem.



(a) `Play-along` (b) `Chile`

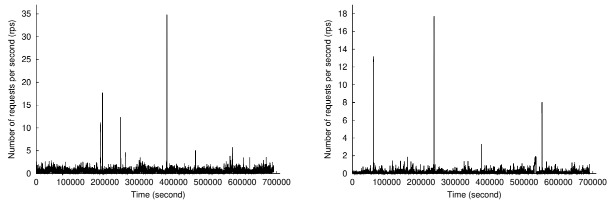
Figure 7: Correlation between document popularity and the number of clusters accessing the document

(c) Requests for popular documents come from a large number of clusters as shown in Figure 7. This again has implications for current CDNs with a large number of servers because requests from a large number of different clusters are likely to arrive at a large number of different CDN servers, and these requests will therefore generate a large number of misses that will seep through to origin servers.

4. DOS ATTACKS

4.1 Password Cracking

While attempts to crack passwords technically do not fall under the rubric of DoS attacks, our proposed strategy treats them similarly, namely, to identify and discard these requests as early as possible (see Section 4.3. Thus, it is appropriate to study these attacks together.



(a) The number of requests per second (`esg`) (b) The number of requests per second (`ol`)

Figure 8: Request rates in password cracking DoS attacks to `esg` and `ol`

We analyzed two log files, `esg` and `ol`, which recorded more than 1 million requests within 60 days. Table 2 summarizes high-level statistics for those traces. As shown in Figure 8(a) and 8(b), both recorded several peaks throughout the period. On closer examination, it turned out that those were attempts to access resources that required a password authorization. Interestingly, each peak was made by the same IP address which sent from 600 to 1,000 *HEAD* requests in a minute in most cases. The pattern resulting from this activity is two-fold:

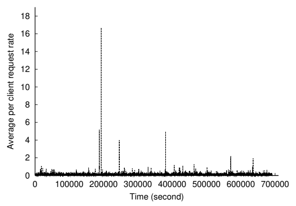
(a) The surge in request rate during the attack occurs due to the increase in a per-client request rate. Comparisons of Figures 8(a) and 9(a) for the `esg` site and Figures 8(b) and 9(b) for the `ol` site show that peaks in the request rates match those in the per-client average request rates. This is a deviation from the behavior we observed in flash events.

(b) An overwhelming majority of client clusters that generated requests during the attack were new clusters not seen by the site before the attack. Only 0.6% of the clusters seen at the `esg` site during the attack were also seen before, and the percentage of these clusters drops to 0.1% for the `ol` site.

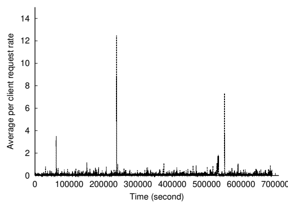
With respect to file referencing, most of the malicious requests were toward one of three resources on either site. There was one exceptional client whose requests were across tens of documents all of which yielded a 401 response indicating an unauthorized request. That exception aside, the

Trace	Data size (Mbyte)		Duration	Requests	Documents	Clients	Clusters
esg	30.5	Total	01/06/01:00:00:06 - 26/07/01:00:00:00	3,501,468	19,456	88,078	17,043
		401		439,585	2	241	185
ol	9.3	Total	01/06/01:00:04:50 - 25/07/01:23:59:41	1,162,675	25,083	39,799	10,259
		401		377,330	3	16	14

Table 2: Basic statistics for esg and ol. Requests with the response code 401 Unauthorized are considered password-cracking attempts and listed in the 401 row.

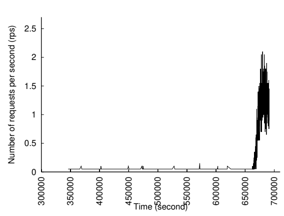


(a) The number of requests per clients in a second (esg)

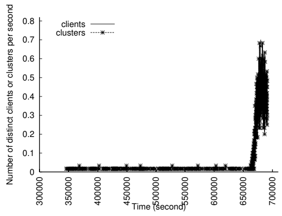


(b) The number of requests per clients in a second (ol)

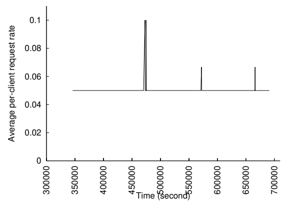
Figure 9: The average per-client request rates to esg and ol



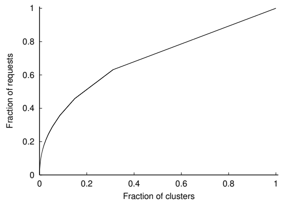
(a) The number of requests per second



(b) The number of clients vs. clusters in a second



(c) The number of requests per client in a second



(d) Cluster contribution to requests

Figure 10: Characteristics of a DoS attack to bit.nl

concentration of malicious requests to just three documents was unique to this type of attack.

4.2 Web Server Disabling

The Code Red worm utilizing a previously disclosed buffer overflow flaw [19] disables vulnerable Microsoft IIS Web servers by sending a very large Request-URI (beginning with default.ida?...). We analyze 5 traces where each log file recorded a number of accesses from infected computers. Table 3 lists basic statistics for each trace. The trace for bit.nl contained only accesses by infected computers so we could not compute cluster overlap for this site.

Figure 10 plots characteristics of the bit.nl trace, the largest among the traces we considered. As Figure 10(a) shows, the attack started with few requests followed by a surge of requests later. However, one can observe from Figure 10(b) that, this surge occurred because of new clusters joining in the attack. This is different from FEs which did not exhibit any surge in new cluster arrivals (see Figure 3). Figure 10(c) confirms this observation by showing that the request rate per cluster did not change during the surge in the request rate. The per-client request rate is three for all cases except for bit.nl where the ratio is three or six.

With respect to client distribution, we found that DoS requests came from clients widely distributed across clusters in the Internet. In fact, each client came from a unique cluster in the fullnote and creighton traces. The percentage of clusters with just one DoS client was 72% for the bit.nl trace, 84% for the rellim trace and 79% for the spccctxus trace. The distribution of DoS attackers is more broad than in the case of flash events. Two corresponding figures, Figure 10(b) and Figure 3(a) highlight the difference: while Figure 3(a) shows many fewer clusters than clients, the cluster and client curves in Figure 10(b) almost match. It is also noticeable that the distribution of requests among clusters are more spread across a number of clusters as shown in Figure 10(d).

Interestingly, for traces that contained both DoS and non-DoS accesses, not a single DoS client had previously accessed the Web site. The percentage of overlap increases slightly for clusters but still remains far below what we have seen for FEs. Among the traces where we could calculate the cluster overlap, its value was 0.6% in the creighton site, 0% in the fullnote site, 1.8% in the spccctxus site and 14.3% rellim site. With the exception of the rellim site, these overlaps are are similar to the password-cracking attacks.

4.3 Comparative Characterization and Server Guidelines

We analyzed the behavior of flash events and DoS attacks

Trace	Data size (Kbyte)		Duration	Requests	Documents	Clients	Clusters
`bit.nl`	764	DoS	15/07/01:18:06:43 - 20/07/01:05:19:11	35,657	1	11,092	6,155
`creighton`	44	Total	12/07/01:01:00:41 - 20/07/01:14:42:25	3,925	565	394	225
		DoS	19/07/01:11:48:18 - 19/07/01:18:57:03	51	1	51	51
`fullnote`	36	Total	01/05/01:17:28:29 19/07/01:19:55:50	3,935	462	175	117
		DoS	19/07/01:11:45:44 - 19/07/01:19:55:50	28	1	28	28
`rellim`	12	Total	12/07/01:14:56:24 - 20/07/01:13:02:30	474	79	172	73
		DoS	19/07/01:12:13:14 - 20/07/01:08:45:31	143	1	143	67
`spcctxus`	3,360	Total	11/07/01:20:00:28 - 19/07/01:21:23:43	390,683	17,810	5,269	1,687
		DoS	19/07/01:10:39:55 - 20/07/01:19:56:58	192	1	192	186

Table 3: Basic statistics for the traces containing a `Code Red` attack. A request is classified as DoS if it requests the URL generated by an infected host when `Code Red` spreads

in the previous sections and observed several differences between FEs and DoS events. These observations lead to a better understanding of the nature of FE and DoS. Table 4 summarizes properties of FE and DoS and their differences in broad terms.

An important difference is that the Web site sees a much larger number of previously seen clusters during a flash event than during a DoS attack. For the `Play-along` and `Chile` events, 42.7% and 82.9% of all clusters that sent requests during the flash event had also sent requests before the flash event. This contrasts with only 0.6-14% of such *old* clusters seen in the `Code Red` attack. We should note that clustering is essential to draw this distinction because the percentage of previously seen individual clients was small in all events. This difference can be intuitively explained by the fact that legitimate requests are more likely to come from clusters where at least some clients had been interested in the Web site even before the flash event, whereas DoS attacks rely on random machines that the attacker was able to hijack.

Another difference is that in DoS attacks that involve a large number of clients (such as the `Code Red` attack), distribution of clients among clusters is much more uniform than in a flash event. The highly skewed distribution of clients among clusters in a flash event reflects the highly skewed distribution of cluster sizes [20]. However, an attack does not reflect the cluster size distribution. While password cracking has too few clients to infer the distribution, server disabling exhibits a more even distribution of clients over clusters than FEs. The same aspect can be seen when considering the distribution of requests among clusters: legitimate requests are more concentrated in a few most active clusters while DoS requests are more scattered across numerous clusters.

A third difference is in the regularity of request rate from individual clients in a DoS attack. In the `Code Red` attack each cluster produces exactly either 3 or 6 requests per minute. In the password-cracking attack, a single client generates a huge number of requests during a short period of time. On the other hand, the per-client request rates get lower during an FE than usual, which suggests that the time gap between successive requests from a client is larger due to a increased server processing time and a possible transmission delay by network congestion.

A fourth difference is in the pattern of requested files. In a flash event, unlike that of DoS attacks, the requested file distribution is Zipf-like. In the attacks we considered, all malicious requests were to a small set of files - three files in the password-cracking attack and one file in the server-disabling attack.

4.4 Server Strategy

A server approaching its capacity of handling requests tends to discard some requests. An intelligent strategy should select the requests that are to be discarded. By exploiting the differences between legitimate and malicious requests, the server can discard requests that are more likely to be malicious. Some legitimate requests may also be discarded but since the server cannot process all requests anyway, the overall behavior will be more desirable.

The server can exploit the differences that we have identified so far. Cluster overlap and per-client request rates are the two characteristics that offer especially strong differentiation between an FE and DoS and are also easy to monitor. Consequently, a server can adopt the following simple procedure for choosing requests to ignore:

- Monitor the clients that access the site and their request rate.
- Periodically perform network aware clustering over the client set accumulated over the past period without flash or DoS events. We will call these clusters "old clusters".
- When performance degrades to an unacceptable level, discard packets (including initial TCP SYNs) that come from clients that do not belong to old clusters as well as from those nonproxy clients whose request rate deviates significantly from average.

Implementation of the strategy can be done without requiring too many CPU cycles or modifying TCP stacks on

Characteristic	Flash Events	DoS Attacks
Traffic volume	Both have a noticeable increase in terms of the number of requests. The length of peaks can be large or small depending on the episode.	
Number of clients and their distribution	Caused mostly by an increase in the number of clients accessing the site. Client distribution can be expected to follow population distribution among ISPs and networks.	Caused either by an increase in the number of clients or a particular client sending requests in a high rate. Client distribution across ISPs and networks does not follow population distribution.
Cluster overlap	Significant overlap between clusters a site sees before and during the FE.	Cluster overlap is very small.
Per-client request rates	Because a server usually gets slower during the FE, per-client request rates are lower during the FE than usual. This indicates that legitimate clients are responsive to the performance of a server unlike DoS attackers who generate requests by a pre-determined time distribution.	Some DoS attacks involve a few clients emitting very high request rates and some a large number of clients generating a low request rate, but in both cases per-client request rate is stable during the attack and significantly deviate from normal.
Requested files	File popularity follows Zipf-like distribution.	The distribution of files used by attackers is unlikely to be Zipf-like. In many cases, a large number of compromised hosts stress a Web server by requesting the same small set of files. It is also possible that a particular host repeatedly requests a large set of files (which may not even exist on the server). Both behaviors result in distributions that are not Zipf-like.

Table 4: Comparative characteristics of flash events vs. DoS attacks

a web server. The cluster to which an IP address belongs can be located via our clustering library that takes three orders of magnitude less time than the time for a typical HTTP request processing. Thus the overhead on the server is negligible.

There are well known techniques to use dynamic shared libraries that function as system call filters [18, 17] that can be used selectively when the load on the Web server exceeds a certain threshold. The system call filter can use a modified *accept()* which decides if a requesting IP remote address meets the criteria for being dropped.

However, it is impossible to anticipate and defeat all DoS attacks and this is not our goal. DoS attacks often flood the links and rarely make it to the application level but if they do, we believe we have a way of dealing with them. We also realize disclosing techniques help attackers in finding ways to circumvent it. We believe that increasing the cost to DoS attackers without a corresponding increase on the server side is a potential alternative. Since a Web site does not make available the list of IP addresses (and thus clusters) of its frequent visitors, the DoS attackers would potentially have to send requests from a very large number of clusters. Additionally, since many of the newer DoS attackers don't bother to spoof addresses [13], we believe that our technique becomes advantageous.

This strategy is unfortunately biased against clients in small clusters: these clusters are less likely to be seen by the site before the FE than large clusters. Thus, the site is more likely to discard requests by clients from small clusters. Still, the strategy is beneficial overall because it allows to service more legitimate requests under overload conditions.

5. FLASH CROWDS AND CDNS

So far we have considered characteristics of flash events and denial of service attacks and how the two could be distinguished by Web servers. Let us now assume that a Web site is able to deal with the denial of service attacks, perhaps using our approach to discard DoS requests early. To deal with flash events, the site might turn to CDNs, which cite the protection against a flash crowd as one of their main benefits. This section analyzes CDN behavior under a flash event and shows that the protection some current CDNs offer might be weaker than claimed. We then propose an approach for improved flash crowd protection, which we call an *adaptive CDN*.

5.1 The Behavior of Current CDNs during a Flash Event

We observed in Section 2 that a small number of objects are responsible for a large percentage of request during a flash event. This suggests that caching would be an effective way to deal with a flash event. However, many of these objects are new to the flash event, that is, they were not recently accessed before the flash event. In the initial phase of the flash event, requests for these objects will miss in most caches and be forwarded to the origin server. This section analyzes the effect of a CDN on the origin site during a flash event.

We perform our analysis by means of trace-driven simulation using the two traces described earlier, `Play-along` and `Chile`. We assume an infinite lifetime for cached objects. This makes our results conservative in terms of the request rate at the origin server because this request rate would only grow if objects were allowed to expire in the cache.

Since clusters group topologically close clients, we assume that CDN caches are assigned to clients at the granularity of clusters: all clients in the same cluster are assigned to the same cache. In our simulations, each client cluster is randomly assigned to one of the caches as soon as it generates the first request and uses that cache afterward. In this subsection, our focus is on cache misses that reach the origin server and hence a different cache assignment will not change our results in any noticeable way.[1] In Section 5.2.1, where we consider load balancing among caches, more intelligent cache assignment would further improve the performance of our scheme.

# of caches (limit)	# of caches (actual)	Average # of documents	Average # of clusters
1	1	7084.00	14100.00
10	10	1479.60	1410.00
100	100	470.31	141.00
500	500	312.70	28.20
1000	1000	270.20	14.10
5000	4725	163.92	2.98
10000	7589	130.34	1.86
15000	9102	118.80	1.55
cache per cluster	14100	94.03	1.00

Table 5: Cache configurations for the `Play-along` trace

# of caches (limit)	# of caches (actual)	Average # of documents	Average # of clusters
1	1	10303.00	1737.00
10	10	3031.30	173.70
100	100	834.84	17.37
500	482	293.66	3.60
1000	823	203.90	2.11
5000	1490	135.77	1.17
10000	1606	128.88	1.08
15000	1643	127.00	1.06
cache per cluster	1737	122.80	1.00

Table 6: Cache configurations for the `Chile` trace

Tables 5 and 6 show cache configurations considered for each trace. The first column indicates the number of caches from which the system selects caches to assign to clusters. The second column gives the number of caches that were actually assigned. The third column provides the average number of objects each cache contains at the end of the run and the last column shows the average number of clusters assigned to a cache. The last row in the tables corresponds to the case when each cluster has a designated cache.

Figure 11(a) and 11(b) show the peak request rate at the origin server during the run for different numbers of caches in the system. It is this characteristic that determines the ability of the Web site to withstand the flash event in the presence of a CDN. This request rate increases rapidly with the number of caches because each cache must obtain an object from the server when the cache receives the first request for the object. The peak occurs in the beginning of FE, when clients request objects which are not yet cached, and reaches about 450 requests per second for thousand caches in the case of `Play-along`. Depending on the complexity of the processing involved in servicing a request, this rate can easily pose a problem for a server. Also note that the FEs we studied were rather modest (due to the unavailability of larger logs), with peak request rates in single thousands. For more significant FEs, the peak request rate at the origin in the presence of a thousand caches can be expected to be higher. Thus, we can conclude that a CDN with a thousand caches might not be able to provide an absolute protection against an FE.

One could keep the request rate at the origin server manageable by limiting the number of caches but that increases the load on individual caches, as Figures 11(c) and 11(d) illustrates.[2]

This situation calls for an approach that would lower the peak rate at both the origin server and at caches. One technique is cooperative caching, where caches forward missed requests to other caches rather than to the origin server (see, e.g., [9, 22]). This imposes an extra overhead for missed requests even when the origin server is not overloaded. A rough indication of this overhead is provided in [26], which reported that a Squid campus proxy cache took up to 150 ms to send cached responses to clients. (Although CDNs are more likely to use commercial cache products, we are not aware of similar data for commercial caches.) We therefore propose a different technique that allows a CDN to break the above tradeoff between the request rate at the cache and the origin server without adding the extra overhead of cache cooperation when it is not warranted.

5.2 Adaptive CDN: Improved Protection of Origin Server

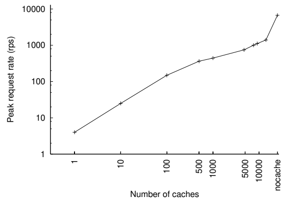
Figure 12 illustrates our proposed adaptive CDN architecture. Like all existing CDNs we are aware of, our architecture uses its Domain Name Server (DNS) to distribute client requests among CDN caches. As in most existing CDN architectures, we assume that there is a control component which collects feedback from caches and updates corresponding entries in the CDN's DNS server[3]. Many commercial DNS implementations provide the functionality to to support frequent changes of an entry in a seamless way, although this functionality would have to be added to the prevalent open-source DNS implementation – BIND [15]. Caches are organized into groups, with one cache within a group selected to be the *primary cache* for a given Web site. Presumably, groups comprise topologically close caches, such as those on a LAN in a data center. To spread the load, different caches in the group serve as primaries for different sites.

Normally, a CDN's DNS server distributes client requests

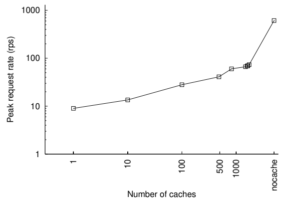
[1]Indeed, with infinite cached object lifetime and infinite capacity, each cache has a miss only once per object for the entire run. Because each cache is likely to see requests for most FE objects at the start of the flash event regardless of the cache assignment, cache assignment will not have much effect on the miss rate at the origin server.

[2]Given the small scale of `Play-along` FE, the absolute levels of per-cache request rate these FEs generate would not overload modern caches: these rates only reach about 250 req/sec without load balancing and 65 req/sec with perfect load balancing across 100 caches. However, the tradeoff in question that our traces illustrate can cause cache overload in a higher-scale FE.

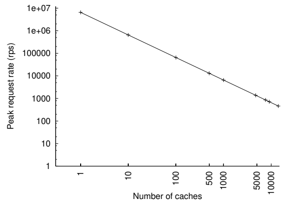
[3]Although CDNs can have multiple DNS servers controlling their network, we use a single DNS server in the rest of the paper without loss of generality.



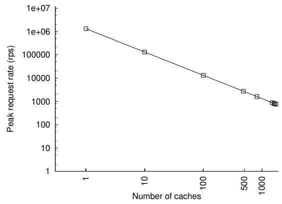
(a) Peak request rate forwarded to the origin (`Play-along`)



(b) Peak request rate forwarded to the origin (`Chile`)



(c) Average number of requests to each cache (`Play-along`)



(d) Average number of requests to each cache (`Chile`)

Figure 11: Simulated behavior of current CDN during a flash event. We assume a CDN that deploys caches in some of the clusters.

only among primary caches for respective Web sites. When the load on a primary cache reaches a dangerous level, the primary requests the DNS to start distributing requests among other members of the group called *delegates*. However, when a delegate receives a missed request, it forwards the request not to the origin server but to the delegate's primary. In this way, the number of caches that can forward misses to the origin server remains limited to the primaries and at the same time the load is spread among both primaries and delegates. When the flash event ends, the primary sends another report to the DNS server asking it to stop using delegates for future requests. When delegates are engaged, the system behaves similar to cooperative caching. However, such behavior occurs only during a flash event, so that the cooperative caching overhead is not incurred during normal operation. We call this technique *dynamic delegation*.

Note that a delegate for one Web site could be a primary cache for another Web site. Thus, with an appropriate allocation of primaries among Web sites, an adaptive CDN utilizes all available caches even when no delegates are engaged. Also, for efficiency, it makes sense for a primary to allocate delegates that are topologically close to the primary, e.g., on the same local area network (LAN).

The simple idea behind dynamic delegation comes from the two factors. First, flash crowd traffic are generated for a small set of hot documents from many individual clients scattered over the Internet. Secondly, a hierarchical caching absorbs a number of requests at the lower level of caches and only missed requests are forwarded to the caches in the upper level. In the case of a perfect hierarchical caching where trees are balanced with one top level caches, we can

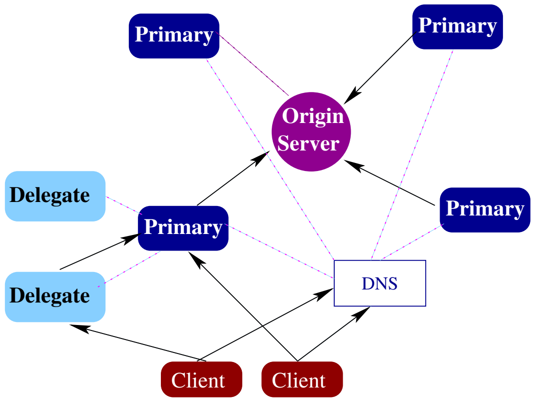


Figure 12: Architecture of an adaptive CDN

reduce the number of requests to be served from an origin server to the number of hot documents. In our architecture, a number of cache trees are dynamically constructed per server basis assuming that not every server becomes busy at the same time and some caches are available and take a role as a delegate to the other busy server.

There are a variety of algorithms to implement dynamic delegation. As a first approach, we simulated an algorithm where a primary decides whether to add a delegate at the time of request processing. This involves executing a lightweight protocol similar to ICP [14] except that our protocol is performed among topologically close caches (on a LAN) and it does not wait for the slowest response from a group of caches as ICP does on a miss. Equally possible is an asynchronous approach where each primary periodically considers its load and the load of its delegates and decides if adding or releasing delegates is warranted. We leave exploring this and other possible dynamic delegation algorithms for future work.

Specifically, our current dynamic delegation algorithm is as follows. Each cache retains load information during last δ seconds. The other two parameters of the algorithm are high and low load watermarks, l_h and l_l. Initially every primary p is in *underloaded* state. When a request arrives at a primary p, the following cases are possible:

1. If p is in *underloaded* state and the load on p has not exceeded l_h in the last δ seconds, p processes the request.

2. If p's load has exceeded l_h in the past δ seconds, p queries all members of its group that are not its delegates already for their load and adds the first cache d that responds with load below l_l as a new delegate. To this end, p informs the DNS server to add d to the set of current delegates of p. In addition, p responds to the current request with HTTP redirect to new delegate d.[4] Finally, p enters the *overloaded* state. When there is no cache available to perform as a delegate, the request is dropped with a `503 Service Unavailable` response.

[4]The idea is that such a response would be generated entirely from main memory and take less processing than responding to the request in a normal way.

3. If p is in *overloaded* state and its load is between l_h and l_l, then p selects an existing delegate d to redirect the request to, queries the delegate for its load and redirects the request to d unless d's load exceeds l_h in which case p adds a new delegate and redirects the request to it.

4. If p's load has not reached l_l in the past δ seconds, p processes the request, releases all delegates (that is, informs the DNS server to stop using delegates for future requests), and enters the *underloaded* state.

5.2.1 Simulation Results

This section presents our preliminary results from the trace-driven simulation that evaluates our adaptive CDN. In our simulations, there are N total number of caches. Among them, n are primary caches which can have up to $N-n$ delegates from cache pools. Clients are redirected by the DNS server to one of n primary caches at the granularity of clusters, assigning all clients from the same cluster to the same cache. The DNS server randomly assigns a cluster to a cache when the first client from this cluster visits the site for the first time. After that, all clients from this cluster use this cache for the *time-to-live (ttl)* period of the DNS assignment, which is set to 10 seconds in our experiments. After that, the DNS will use the same primary cache for the cluster unless delegates have been engaged for this primary. In the latter case, the DNS assigns to the cluster the delegate that has the lowest load at the time of the request. Again, all clients in the cluster will use this delegate for the *ttl* period.

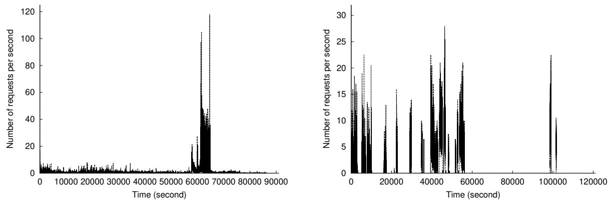
N	n	*ttl*	δ	l_h	l_l
300	100	10 sec	30 sec	50	25

Table 7: Parameter settings for the simulation.

Table 7 shows the simulation configuration for each trace. Load on caches is computed as requests per second averaged over two-second intervals.

Considering the request rate at the origin server, the adaptive CDN with n primary caches generates exactly the same request rate as that of the conventional CDN with n caches (see Figures 11(a) and 11(b)). However, the conventional CDN with n caches will place much higher load on individual caches. To precisely compare the load on individual caches placed by conventional and adaptive CDNs would require knowing the exact load balancing algorithm used by conventional CDNs—information that is not publicly available. The simulation shows that with 100 primary caches, the number of delegates used is 116 for the `Play-along` trace and 6 for the `Chile` trace. The number of primary caches that ever had a delegate is 65 for the `Play-along`, while only two primary caches reached l_h and actively engaged delegates throughout FE for the `Chile` trace. This suggests that request sources are more skewed to a few busy clusters for the `Chile` case. It is also noticeable that none of the caches including delegates experienced the load above l_h .

Figure 13 shows the simulation results plotting the number of requests per second at the origin server when there are 100 primary caches. Comparing this with Figure 1, which shows the request rate at the origin server without a CDN, we see that peak request rates are reduced by a factor of 50 for the `Play-along` trace and 20 for the `Chile` trace. There are still peaks in the beginning of FE but they are as small as 150 requests per second for the `Play-along` trace and 30 requests per second for the `Chile` trace. This suggests that an adaptive CDN effectively lowers peak request rates of an origin server in the case of FE. At the same time, by using dynamic delegation, the adaptive CDN also ensures that load on each cache remains low and that proximity-based cache selection is not compromised.



(a) `Play-along` (dynamic delegation) (b) `Chile` (dynamic delegation)

Figure 13: Request rate at the origin server with adaptive CDN

6. CONCLUSIONS

In this paper we have defined a flash event, a phenomenon that can severely cripple a Web server. Even in cases where servers may anticipate significantly increased load, they need help in proper provisioning to handle a flash crowd. Both flash crowds and denial of service attacks have the potential to have similar impact on Web servers. We demonstrate a way to distinguish between them using network aware clustering, so that Web servers can attempt to serve normal clients and drop requests from clients involved in DoS attacks.

In the second half of the paper we explore the role of CDNs that claim to offer protection against flash crowds. We show that some CDNs as they are currently structured may not offer maximum protection and we propose an alternative architecture that uses adaptive techniques. In addition, we present trace-driven simulation results evaluating our proposed adaptive CDN architecture. By contrasting against an idealized allocation mechanism we show that adaptive CDNs can help handle extreme situations such as flash crowds. As future work, we would like to obtain larger flash crowd logs from diverse places and experiment against instrumented servers. In our proposed adaptive CDNs, we plan to prepopulate caches and experiment with alternate dynamic delegation algorithms. The dynamic delegation algorithm could also use alternate redirection mechanisms.

7. ACKNOWLEDGMENTS

We would like to thank all those who gave us access to their logs without which such research would be impossible. Included among them are the University of Chile, Sabri Berisha, Mike Hendrickson, Dan Klein, Brian Miller, and Gary Miller, and Tim Winders. We would like to thank Lee Breslau and Fred Douglis for their comments on an earlier version of the paper. We thank Shubho Sen for his detailed

comments.

8. REFERENCES

[1] CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks. http://www.cert.org/advisories/CA-1996-21.html, Sept. 1996.

[2] Denial of Service Attacks. http://www.cert.org/tech_tips/denial_of_service.html, 1999. CERT Coordination Center.

[3] G. Abdulla, E. A. Fox, M. Abrams, and S. Williams. WWW Proxy Traffic Characterization with Application to Caching. Technical Report TR-97-03, Computer Science Dept., Virginia Tech, Mar. 1997.

[4] V. Almeida, D. Menasce, R. Reidi, F. Peligrinelli, R. Fonseca, and W. M. Jr. Analyzing Web Robots and their Impact on Caching. In *Proceedings of the 6th Web Caching and Content Delivery Workshop*, June 2001.

[5] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. HPL-1999-35R1.

[6] M. F. Arlitt and C. L. Williamson. Internet Web servers: workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.

[7] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Measurement and Modeling of Computer Systems*, pages 151–160, 1998.

[8] P. Barford and D. Plonka. Characteristics of Network Traffic Flow Anomalies. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.

[9] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A Hierarchical Internet Object Cache. In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996.

[10] M. E. Crovella, R. Frangioso, and M. Harchol-Balter. Connection Scheduling in Web Servers. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'99)*, Oct. 1999.

[11] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. C. Mogul. Rate of Change and other Metrics: a Live Study of the World Wide Web. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[12] S. Floyd, S. Bellovin, J. Ioannidis, K. Kompella, R. Mahajan, and V. Paxson. Pushback Messages for Controlling Aggregates in the Network. http://search.ietf.org/internet-drafts/draft-floyd-pushback-messages-00.txt.

[13] K. J. Houle, G. M. Weaver, N. Long, and R. Thomas. Trends in Denial of Service Attack Technology. http://www.cert.org/archive/pdf/DoS_trends.pdf.

[14] Internet Cache Protocol (ICP), version 2. RFC 2186, Sept. 1997.

[15] Internet Software Consortium. The Berkeley Internet Name Daemon. http://www.isc.org/products/BIND/.

[16] A. K. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale Web server access patterns and performance. *World Wide Web*, June 1999.

[17] M. B. Jones. Interposition Agents: Transparently Interposing User Code at the System Interface. In *Symposium on Operating Systems Principles*, pages 80–93, 1993. http://www.research.microsoft.com/~mbj/papers/sosp93.ps.

[18] E. Krell and B. Krishnamurthy. COLA: Customized Overlaying. In *Proceedings of the USENIX San Francisco Winter 1992 Conference*, pages 3–7, 1992.

[19] B. Krishnamurthy and M. Arlitt. PRO-COW: Protocol Compliance on the Web, Nov. 1999. Invited plenary session talk at 46th IETF meeting, Washington D.C. http://www.research.att.com/~bala/papers/ietf99.ps.

[20] B. Krishnamurthy and J. Wang. On Network-Aware Clustering of Web Clients. In *Proceedings of the ACM SIGCOMM*, Aug. 2000.

[21] S. Lorenz. Is your Web site ready for the flash crowd? http://www.serverworldmagazine.com/sunserver/2000/11/flash.shtml.

[22] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive Web caching: towards a new global caching architecture. *Computer Networks And ISDN Systems*, 30(22-23):2169–2177, Nov. 1998.

[23] D. Moore. The Spread of the Code-Red Worm (CRv2). http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml, Aug. 2001.

[24] L. Niven. Flash crowd. In *The Flight of the Horse*. Ballantine Books, 1971.

[25] K. Park and H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In *Proceedings of the ACM SIGCOMM*, Aug. 2001.

[26] A. Rousskov and V. Soloviev. A Performance Study of the Squid Proxy on HTTP/1.0. *World Wide Web*, pages 47–67, June 1999.

[27] B. Trott. Victoria's Secret for Webcasts is IP multicasting, August 1999. http://www.infoworld.com/articles/hn/xml/99/08/16/990816hnmentors.xml.

[28] D. Wessels. Report on the effect of the Independent Council Report on the NLANR Web Caches. http://www.ircache.net/Statistics/ICreport/.