Dynamic CDN against Flash Crowds

Norihiko Yoshida

1 Introduction

With the rapid spread of information and ubiquitous accesses of browsers, new congestion phenomena on the Internet, *flash crowds*, makes traditional techniques fail to solve. Flash crowds are sudden, unanticipated surges in traffic volume of request rates towards particular Web sites. Differing from the consistent Internet congestions, flash crowds produce short-term congestions. It makes Web sites over-provisioned and the hosting Content Delivery Network (CDN) inefficient and uneconomical. Thus, they pose new challenges to the today's Internet.

The term "flash crowd" was coined in 1973 by a science fiction writer Larry Niven in his short novel "Flash Crowd" [31]. In the novel, cheap and easy teleportation enabled tens of thousands of people worldwide to flock to the scene of anything interesting almost instantly, incurring disorder and confusion.

The term was then applied to similar phenomena on the Internet in the late 1990's. When a Web site catches the attention of a large number of people, it gets an unexpected and overwhelming surge in traffic, usually causing network saturation and server malfunction, and consequently making the site temporarily unreachable. This is the "flash crowd" phenomenon on the Internet, which is also sometimes referred to as the "SlashDot effect" [2] or a "Web hotspot" [50]. An example of a flash crowd is shown in Fig. 1 [34], which occurred on the "LIVE! ECLIPSE" Web site [26] on November 3rd, 2005.

Flash Crowds are not frequent phenomena. They differ from those workloads that vary over time, such as time-of-day effects [13], e.g. more people enjoy the Web during lunch hours, where long-term periodic trends can be predicted. However, they are triggered relatively easily, in that even the mere mention of a popular Web

Norihiko Yoshida
Division of Mathematics, Electronics and Informatics, Saitama University, Saitama 338-8570, Japan, e-mail: yoshida@ics.saitama-u.ac.jp

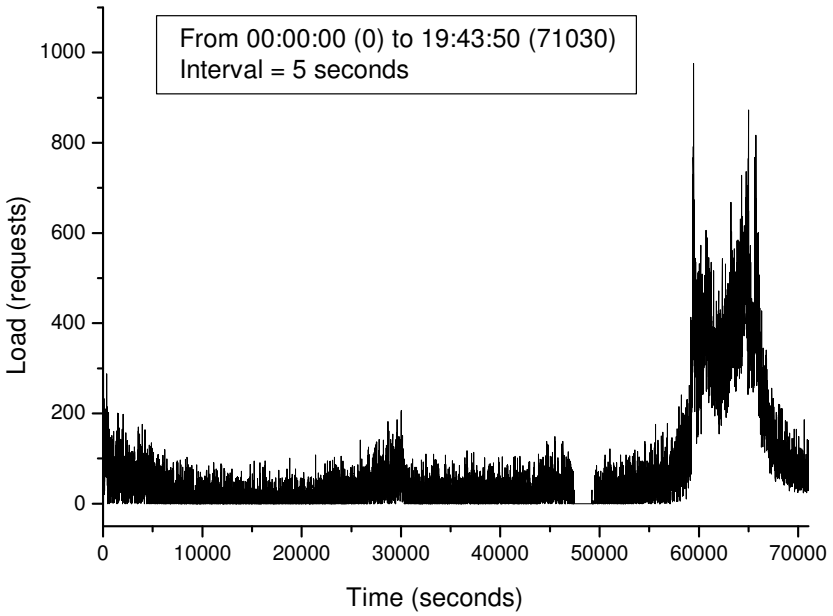


Fig. 1 Flash crowd to the "LIVE! ECLIPSE" site on Nov. 3, 2005

site will produce one. Due to an increase in the frequency of flash crowds and their overall unpredictability, flash crowds have now become a bane of most Web sites.

A conventional CDN works well if the request load is relatively constant. However, it is static in the sense that it uses a fixed number of surrogates all the time, and it is permanently prepared for the congested state. Considering the situation of flash crowds, resorting to a high level of over-provision suffers from low efficiency. Due to the infrequency of the high load, static CDNs lead to under-utilization of resources, and the surrogate servers will remain idle most of the time.

Moreover, Web requests can be bursty. It is not easy to predict the peak load of a flash crowd. Even a very well-configured CDN site may be crippled due to the demand unpredictability associated with a traffic surge.

Some solutions have already been proposed to address the problem of flash crowds. Promising solutions should incorporate a certain principle: changing the static CDN into an adaptive dynamic CDN. The network adaptively changes its architecture to reach the best optimum according to the observed traffic load. When the load is under control, the normal client/server (C/S) configuration manages the requests well. When the load exceeds a threshold or fulfills a certain condition, then the set of surrogate servers kicks in to absorb the huge number of requests. In this way, the network is supposed to be more efficient as regards resource utilization, practical in its ability to address flash crowds and affordable by more Web sites.

There are three main challenges in addressing this issue:

1. How to organize a temporary overlay of surrogate servers quickly. The surrogate servers should be utilized efficiently and need to cooperate with each other al-

most immediately when faced with a flash crowd. When the flash crowd departs, everything should go back to normal operation without involving much overhead. The impact of being a potential surrogate during the normal period of operation should be controlled, so that it is minimized as much as possible.
2. How to detect the arrival and departure of a load spike properly. Flash crowds are different from the normal workloads, whose magnitude and duration depend on the people's interest toward some triggering events, and it is difficult to make long-term predictions in advance. Thus, the network must be reactive to the arrival of a flash crowd by relying on short-term quick predictions. The detection must be careful because any improper detection may result in a waste of resources or oscillations of the network architecture.
3. How to redirect client requests transparently to the temporary overlay. Once the dynamic CDN is ready for the flash crowds, the flooded client requests must be redirected to any of the surrogates, and they should preferably be redirected in a load-balancing manner. Different from single site schemes where a local load balancer works, this redirection must be performed within a wide-area temporary environment.

We advocate FCAN (Flash Crowds Alleviation Network), a dynamic CDN network that adaptively optimizes the network architecture between C/S and CDN configurations. We utilize an Internet infrastructure of cache proxies to organize a temporary overlay of surrogate servers. This mode is invoked on the fly when a flash crowd comes, but pulled out of action when the normal C/S configuration works adequately [7, 33].

This chapter is organized as follows: Sect. 2 provides a brief overview of flash crowds and analyzes their triggering, types, and characteristics. It also discusses how to distinguish flash crowds from other similar Internet phenomena, Denial of Service (DoS) and Distributed DoS (DDoS) attacks. It then examines state-of-the-art research works. By analyzing and comparing several related solutions, it clarifies their advantages and disadvantages. Sect. 3 presents FCAN. It explains how the network reacts to the beginning and ending of a flash crowd, how the temporary surrogates are organized and cooperate with each other, and how redirection based on DNS (Domain Name System) offloads the burden on the origin Web site. It then exhibits simulation-based evaluations using real trace workloads. Section 4 summarizes some visionary thoughts for practitioners. Section 5 presents the future research directions with discussions on open issues. Section 6 comprises the conclusion.

2 Background and Related Work

We first study the characteristics of flash crowds. We show that network bandwidth is the most serious bottleneck, and a small number of objects is responsible for a greater percentage of requests (i.e. heavy-tailed behavior). These observations imply that flooded requests must be redirected away from the server and caching these

flash-crowd objects could be a possible solution. Study of related work in this context show that there is still room for improvement to the solutions for handling the problem of flash crowds.

2.1 Flash Crowds

Usually, sudden events of great interest trigger flash crowds, whether planned or unplanned. Some well-analyzed ones include: World Cup 1998 [6], RedHat Linux image distribution [8], Play-along TV show 2000 and Chilean presidential election broadcast 1999 [21], and CNN broadcast on the terrorist attacks of September 11, 2001 [13]. In addition, a Web site which is referred to on a popular site, news or blog often experiences an unusual amount of accesses unexpectedly.

Due to resource limits and/or the network bandwidth, the servers are unable to handle the high volume of requests. As a result, most users perceive unacceptably poor performance. Moreover, flash crowds unintentionally deny service for other users who either share common paths with the flash crowd traffic or who try to retrieve unrelated information from the same servers [30, 51].

Through analyses of such real traces as mentioned above and other research efforts [19, 27], some significant characteristics can be concluded, as stated below. These observations allow us to tell when a flash crowd arrives; how long (or short) a time we have to take defensive action; how different it is from a malicious attack; how we can utilize the locality of reference; and more.

1. The increase in the request rate is dramatic, but relatively short in duration. A flash crowd lasts as long as the attention span of the concerned audience, from hours to days, which is relatively short compared to the life span of a Web application. Therefore, if we make an over-provision or switch to the conventional CDN, the results can lead to under-utilization of resources during the normal operational period, especially for small or personal Web sites, which might experience flash crowds only once or twice in their lifetime.
2. The increase in the requests is rapid but not instantaneous. In the case of the Play-along TV show, the rate increase continued for 15 min. before it reached its peak. Another case, the September 11, 2001 event, resulted in a massive load on the CNN Web site which doubled every 7 min., finally reaching a peak of 20 times higher than the normal load [24]. This property suggests that we still have adequate time to detect a flash crowd and react.
3. Network bandwidth is the primary constraint bottleneck. CPU may be a bottleneck if the server is serving dynamically generated contents. For instance, on the morning of September 11, dynamic pages on the MSNBC news Web site consumed 49.4% of "500" (server busy) error codes [32]. However, MSNBC quickly switched to serving static HTML pages, and the percentage of the error status codes dropped to 6.7%. Observations also revealed that network bandwidth became the primary constraint bottleneck, and the closer paths are to the server, the worse they are affected [32]. It is reported that modern PCs could sustain more

network throughput than 1 Gbps when serving static files [20], while the network bandwidth of a Web site is typically much lower [40]. Accordingly, we should focus on alleviating the bandwidth bottleneck around the servers.

4. A small number of contents, less than 10%, is responsible for a large percentage, more than 90%, of requests. For instance, the MSNBC traces from September 11 showed that 141 files (0.37%) accounted for 90% of the access, and 1086 files (2.87%) for 99% of the access [32]. Moreover, the set of hot contents during a flash crowd tends to be small to fit in a cache. This is a promising result implying that the caching of these 10% contents can be a solution to flash crowds. We also observe that this "10/90" rule of reference follows the Zipf-like distribution, in which the relative probability of a request for the i'th most popular content is proportional to $1/i^a$ [10]. This property distinguishes flash crowds from attack traffic which is generated automatically by "bots".
5. More than 60% of contents are accessed only during a flash crowd. In addition, among the 10% hot contents, more than 60% are new to being cached. For instance, 61% of contents were uncached in the Play-along case, and 82% in the Chile case [21]. This implies usual Web caches may not provide the desired level of protection. Most cache proxies on the Internet will not have the requested contents at the beginning of a flash crowd. Therefore, most requests would miss in the caches, and be forwarded to the origin server. Although subsequent requests would be served from the caches, a large number of initial cache misses will be generated to the origin server within a short period of time.
6. The number of clients in a flash crowd is commensurate with the request rate. This feature can be used to rule out malicious requests. During a flash crowd, spikes in requested volumes correspond closely with spikes in the number of clients accessing the site. The increase in traffic volume occurs largely because of the increase in the number of clients, and most requests come from a large number of client clusters. However, because a server usually slows down during a flash crowd, per-client request rates are lower than usual. This indicates that legitimate clients are responsible for the performance of a server.

While studying the behavior of flash crowds, we need to identify and distinguish related but distinct phenomena, DoS attacks. A DoS attack is "an explicit attempt by attackers to prevent legitimate users of a service from using that server" [12]. It overwhelms a target server with a huge amount of packets in primarily a brute force manner, so as to saturate the target's connection bandwidth or deplete the system resources to subvert the normal operation. Some well-known DoS attacks include: SYN attack [11], Code Red attack [29], and Password Cracking [18]. Recently, DDoS attacks, which employ a large number of "bots" emitting requests to the target, have also been frequently reported [22].

DoS attacks share several characteristics with flash crowds. They both overload a server's Internet connection and result in partial or complete failure. However, the server should ignore DoS attacks during flash crowd protection, and handle legitimate requests only. There are some ways to distinguish DoS attacks from flash crowds [21] : (1) Client distribution across ISPs and networks does not follow population distribution; (2) Cluster overlap which a site sees before and during the attack

is very small; (3) Per-client request rate is stable during the attack and deviates significantly from normal; and (4) The distribution of files (which may not even exist on the server) targeted by attackers is unlikely to be Zipf-like.

By exploiting these differences, a server may take a strategy for distinguishing DoS attacks from flash crowds, and discard these malicious requests as early as possible. It may monitor clients that access the site and their request rates, and perform some checks on the content of packets, HTTP headers, and arrival rates.

More details and implementations on how to distinguish malicious requests from legitimate ones are beyond the scope here, as exclusive coverage of works in this respect can be found in literature [21, 22, 35]. This chapter assumes that servers have already ruled out malicious requests of DoS attacks by using some mechanisms.

2.2 Possible Solutions

Solutions proposed so far for addressing flash crowds are classified into three categories according to the typical architecture of networks: server-layer, intermediate-layer and client-layer solutions. Figure 2 shows their schematic overviews.

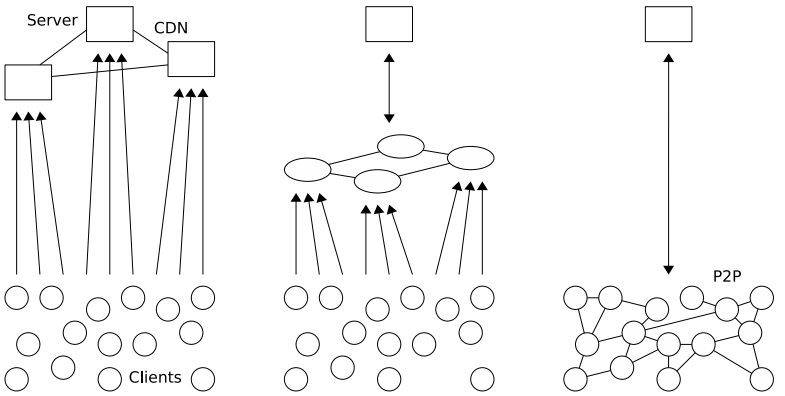


Fig. 2 Three solutions

2.2.1 Server-Layer Solutions

As mentioned above, traditional over-provisioning and use of static CDN [3, 25, 43] on the server-side are straightforward but costly approaches. They are inefficient and difficult to deal with short-term Internet congestion. Due to the unpredictability of flash crowds, any imperfectly provisioned system is likely to fail under sustained overload conditions.

CDN with Dynamic Delegation J. Jung, et al. [21] proposed an adaptive CDN using dynamic delegation to improve the protection of origin servers under heavy load conditions. They organize surrogate servers into groups, with one surrogate within a group selected to be the primary surrogate. Usually, a DNS server for the CDN assigns client requests only to primary surrogates. When the load on the primary surrogate reaches a alarming level, the primary surrogates asks DNS to reassign requests to other members in the group called delegates. When a delegate receives a missing request, it forwards the request not to the origin server but to the delegate's primary. This mechanism is called "dynamic delegation". When delegates are engaged, the system behaves like cooperative caching.

Dynamic delegation takes 60% uncached objects into consideration at the beginning of flash crowds, as mentioned above, and improves the efficiency of the system. It pulls the popular objects from the origin server, and absorbs the cache-miss requests by hierarchical caching. However, the surrogate groups with primaries should be configured manually and permanently even during peaceful periods.

DotSlash DotSlash [50] allows different Web sites to form a mutual-aid community and to use spare capacity within the community so as to relieve flash crowds experienced by any individual site. As a rescue system, DotSlash continuously monitors the workload at each Web server; when a server becomes heavily loaded, rescue services are activated, and once the server's load returns to normal, the rescue services cease. As a result, a Web site has a dynamic server set which includes a single or a cluster of fixed origin servers, and a changing set of rescue servers.

Different from most other systems mentioned here and in 2.2.2, which use permanent and fixed resources, DotSlash triggers its rescue system on a temporary mutual-aid community. However, DotSlash needs clients to connect with the origin server first, and then issues a redirected URI containing the virtual host name for DNS redirection. Consequently, there is a risk that the bandwidth and processing needed to send the redirection messages may itself overwhelm the origin server.

2.2.2 Intermediate-Layer Solutions

There have been some intermediate-layer solution proposals for dealing with flash crowds, which utilize network resources to perform offload. Caching techniques help to alleviate server load during flash crowds by filtering out repeated requests from groups of clients which share a proxy cache.

In general, proxies on the Internet are divided into two types: forward proxies and reverse proxies. Forward proxies are placed near clients and thus far from the server end. Their typical functionality includes a firewall, and caching of static contents. They are usually shared by many clients and are reasonably powerful and stable. However, content providers do not have much control over them. Reverse proxies are placed near the back-end server farm, and act as agents of application providers. They serve requests on behalf of the back-end servers. Content providers can fully control their behavior. However, the scale of reverse proxies only goes as far as a content provider's network bandwidth allows [47].

Multi-Level Caching The solution using multi-level caching [4] argues that with proper replacement algorithms, a caching infrastructure designed to handle normal Web loads can be enough to handle flash crowds. It studies the effects of using different cache replacement algorithms, changing the placement of caches, using heterogeneous multi-level caching, and partitioning the ID space based on document size. The work concludes that using GDSF algorithm [5], the replacement policy in caches results in significant improvements to the client response times, and server and network loads.

Multi-Level Caching offers promising results for using caching to address flash crowds for small and static objects. The system needs a dedicated deployment of hierarchical caching placement, and complete control over the infrastructure of forward cache proxies. The system does not address the problem of 60% uncached objects, and thus it may not provide the desired level of protection to the origin server at the initial stage. In addition, it currently lacks an adaptive mechanism for handling flash crowds flexibly.

BackSlash Backslash [44] uses content-addressable P2P overlays based on distributed hash tables (DHTs) to build distributed Web server systems. It places copies of contents on mirror servers which are specified by content providers. DHTs provide bases for the self-organization of participants, for routing requests, and for load balancing.

BackSlash uses Web servers and proxies to offload the network traffic. However, in BackSlash, the contents on mirror servers must be pre-placed and well-organized in advance, which incurs operation complexity and restricted extensibility of the system.

CoralCDN CoralCDN [17] leverages the aggregate bandwidth of volunteers to absorb and dissipate most of the traffic for Web sites using the system. As we have seen in Chap. 1, CoralCDN exploits overlay routing techniques on top of a key/value indexing infrastructure: a P2P distributed sloppy hash table, or DSHT, which allows nodes to locate nearby cached copies of Web objects without querying more distant nodes and which prevents hot spots in the infrastructure, even under degenerate loads. We also know that to use CoralCDN, a content publisher or someone posting a link to a high-traffic portal simply appends ".nyud.net:8090" to the hostname in a URL.

Coral uses volunteers' additional capacities to absorb the overwhelming traffic. It combines a set of P2P-based reverse proxies to create cache objects on demand, and adopts DNS to redirect client requests transparently. CoralCDN is always waiting for the incoming requests, whose URL needs to be manually configured by appending ".nyud.net:8090" in advance. With a modified URL, CoralCDN is capable of object-oriented redirection, however, it sacrifices user unawareness of the system.

2.2.3 Client-Layer Solutions

Client-side solutions make clients help each other in sharing objects so as to distribute the load burden from a centralized server. An origin Web server can mediate client cooperation by redirecting a client to another client that has recently downloaded the objects, as in Squirrel [28], Pseudoserving [23] and CoopNet [32]. Clients can also form P2P overlay networks and use search mechanisms to locate resources. For example, PROOFS [45] employs randomization to build client side P2P overlay networks, and BitTorrent [9] breaks large files into small parts for efficient retrieval. In general, these solutions rely on the client-side cooperation. They must be deployed on users' desktop PCs, which are thus likely to prevent their widespread deployment.

CoopNet Cooperative networking [32] is a P2P caching solution that complements traditional client-server and client-Web proxy communication rather than replacing it. It has previously-registered clients who have already downloaded content, and they in turn serve the content to other clients. CoopNet uses HTTP-based redirection to route requests, and to select peers according to their nearby location.

In CoopNet, P2P communication kicks in during flash crowds to share the load, and gets out of the way when the C/S communication works fine. CoopNet uses a server-based redirection, which has the risk of a "single point of failure".

PROOFS PROOFS [39, 45] is comprised of two protocols. The first forms and maintains a network overlay. The second performs a series of randomized, scoped searches for objects atop the overlay formed by the first protocol. Nodes continually perform what is called a "shuffle operation". The shuffle is an exchange of a subset of neighbors between a pair of clients, and can be initiated by any client. Shuffling is used to produce an overlay that is "well-mixed", in that a client's neighbors are essentially drawn at random from the set of all clients that participate in the overlay. Once a random state is reached, scoped searches for objects can be performed atop the overlay. Objects are located by randomly visiting sets of neighbors until a node is reached that contains the object. Through combination of theoretical results and simulation, PROOFS claims to be robust for the overlay partitioning, for peer dynamic joining/leaving, and for limiting participation in the system.

PROOFS uses an unstructured first generation P2P system, and thus requires a lower preparation cost, and it offers good performance under the condition of flash crowds. A significant amount of attention has been paid to second generation P2P architectures such as CAN [36], CHORD [46], and Pastry [38], in which participants have a sense of direction as to where to forward requests. They provide benefit over their first generation counterparts in terms of the amounts of network bandwidth utilized and the time taken to locate those documents. However, to be able to handle documents whose popularity suddenly spikes without inundating those nodes responsible for serving these documents, the first generation architectures (which are simpler and more lightweight) are preferable.

2.2.4 Other Works

Grid technologies allow "coordinated resource sharing and problem solving in dynamic, multi-institutional organizations" [16], with a focus on large-scale computational problems and complex applications that involve many participants and different types of activities and interactions. Internet data centers host multiple Web applications on shared hardware resources. A. Chandra, et al. suggest reacting to changing application loads by reallocating resources to overloaded applications, borrowing these resources from other under-utilized applications if necessary [13].

As a last resort, a Web site can use admission control [14, 15, 49] to prevent itself from being overloaded, by rejecting a fraction of the client requests and only admitting preferred clients.

3 FCAN: Flash Crowds Alleviation Network

FCAN [7, 33] is an intermediate-layer solution, using a CDN-like wide-area overlay network of caching proxies which stores objects, and delivers them to clients, like the surrogate servers in a CDN. Considering the short duration and unpredictability of flash crowds, FCAN invokes the overlay only when a server is overwhelmed by a large amount of requests, and reorganizes the overlay when necessary. This dynamicity is the most prominent characteristic of FCAN compared to most of the above-mentioned related works. The only exception is DotSlash, however, it lacks an adaptive reorganization feature.

FCAN aims at complementing an existing Web server infrastructure to handle short-term load spikes effectively, but it is not intended to support a request load that is constantly higher than the planned capacity of a Web site. It targets small Web sites, although large Web sites can also gain some benefit from it.

3.1 Requirements

Below are the functional and nonfunctional requirements which we analyzed in order to make FCAN flexible, reliable, and cost-effective.

Object Delivery First and foremost is the timely delivery of content objects. FCAN should maintain high availability of the delivery service at all times. Moreover, accessibility to non-flash-crowd objects on the same target server should also be ensured.

Workload Control FCAN should monitor changes in the increasing load and control it so that the server does not become overwhelmed. At the same time, when flooded requests are offloaded to the temporary surrogates, FCAN should also have

a workload monitor on each surrogate to detect the leaving of flash crowds, and to control the redirected requests so as not to overload the surrogate.

Adaptive Transition FCAN should be sensitive to the load increase and transit its architecture in a flexible fashion in order to obtain optimum performance output. The duration time should be short to take the transition into effect. Both the detection and transition should be conducted automatically.

Request Redirection There should be a mechanism to direct the flooded requests by finding temporary surrogates. Moreover, the most appropriate surrogate should be selected. It would be ideal if the redirection being carried out is uniformly balanced.

Client Transparency FCAN will be more acceptable if clients could remain unchanged. It is better for the clients to remain completely unaware of the existence of FCAN.

Scalability Because Internet-based infrastructures have the potential to reach the entire world-wide Internet community, FCAN requires the capability to expand its infrastructure easily, with minimal effort and disruption.

3.2 Design Overview

In peaceful times, the conventional C/S architecture satisfies most of the client requests. A member server and member cache proxies, both of which comprise FCAN, do little more than what normal ones do. When a flash crowd comes, the member server detects the increase in traffic load. It triggers a subset of the member proxies to form an overlay, through which all requests are conducted. All subsequent client requests are routed to this overlay by DNS-based redirection. If the subset of proxies

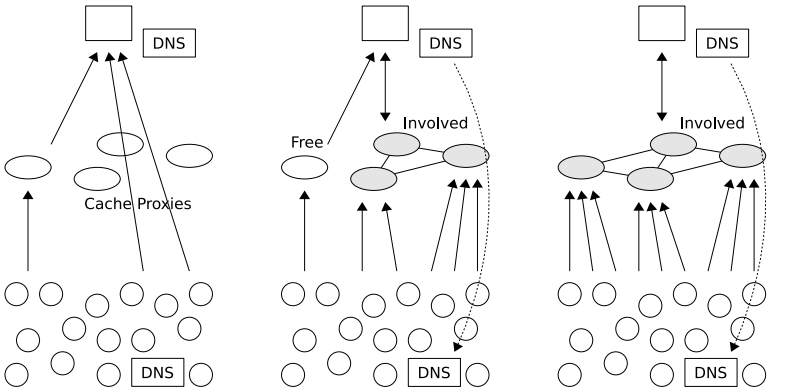


Fig. 3 FCAN overview

is not large enough to handle the amount of requests, new proxies are invited, and the overlay is enlarged. When the flash crowd declines, some proxies leave, so that the overlay shrinks and is eventually released. Figure 3 gives an overview of FCAN at three different states, namely, usual, initial, and enlarged state.

FCAN is not dedicated to a single member server. It is designed to be shared by several servers in need, even at the same time. Each member server uses its own overlay, small or large, and servers try mutually to prevent their overlays from overlapping as much as possible.

Each member proxy is primarily a regular forward cache proxy during its normal mode, however it acts as a surrogate, somewhat similar to a reverse cache proxy, serving requests from any user during the anti-flash-crowd mode. In reality, a member proxy serves for several servers, and it is possible that any one server suffers from flash crowds while the others do not. Therefore, each member proxy has the functionality of mixed-mode operations for the forward proxy mode and the surrogate (similar to reverse proxy) mode.

3.3 Flash Crowd Detection

As different resources such as network bandwidth, CPU and memory at a server may potentially become the bottleneck during a flash crowd, separate workload metrics should ideally be used for different resources. Each member server should do monitoring and overload/underload detection, and perform dynamic transition accordingly. The current design uses only the access arrival rate as the load metric, and uses a threshold-based scheme to trigger dynamic transition.

To detect the coming of a flash crowd, a server observes the volume of its load periodically. Once the server load exceeds the predefined threshold, T_{high}, the server treats it as the coming of a flash crowd.

During the flash crowd, each proxy involved in the overlay has a load monitor, which observes the number of client accesses, and the member server collects the load information from all the involved proxies periodically. When the load on the overlay of proxies decreases under a predefined threshold, T_{low} ($< T_{high}$), the member server treats it as the ending of the flash crowd.

3.4 Network Transition

When the member server detects the beginning of a flash crowd, it carries out the following procedure, in order to make some member cache proxies transit into the anti-flash-crowd mode in order to form a temporary overlay.

1. Selects a subset of proxies to form a CDN-like overlay of surrogates;

2. Triggers an update of DNS records to change the look-up entries of the Web site from the server's address to those of the proxies, so that subsequent requests are gradually redirected to the proxies along with DNS propagation;
3. Disseminates ("pushes") the flash-crowd object to the selected proxies, because more than 60% of the flash-crowd objects are uncached prior to the arrival of the flash crowd, as mentioned above;
4. Prepares to collect and evaluate statistics for the object from the involved proxies, so as to determine dynamic reorganization and release of the overlay;

Every member cache proxy carries out the following procedure upon request from the member server:

1. Changes its mode from a proxy to a surrogate (or, in the strict sense, a mixed mode of a forward proxy and a surrogate, as mentioned above);
2. Stores flash-crowd objects permanently, which should not expire until the flash crowd is over;
3. Begins monitoring the statistics of request rate and load, and reporting them to the server periodically,

The server selects the subset of proxies by probing them one by one first, because any proxy may already be involved in another flash crowd alleviation, or it may be overloaded due to some other reason. This prevents overlapping of more than one subsets for independent flash crowds. The subset can be small, even consisting of only one proxy, because FCAN has the feature of dynamic reorganization, as mentioned below. The current design expects network administrators to assign priorities to proxies for probing orders.

When the member server detects the leaving of the flash crowd, the involved proxies are dismissed one by one, in the reverse order of probing, with the following procedure:

1. The server updates the DNS records;
2. The server notifies the proxy to be dismissed;
3. The proxy changes its mode from a surrogate to a proxy.

The CDN-like overlay transits back to the normal C/S mode when all the proxies are dismissed. They are not all dismissed at once, since low load may be just temporary, and the system should therefore remain in the anti-flash-crowd mode.

3.5 Dynamic Reorganization

Every proxy has its local monitor, which observes the request rate and the overall load on itself. Proxies involved in the overlay, whether initial or additional, send feedback information to the server periodically, including the request rate for the flash-crowd object and the overall load on the proxy.

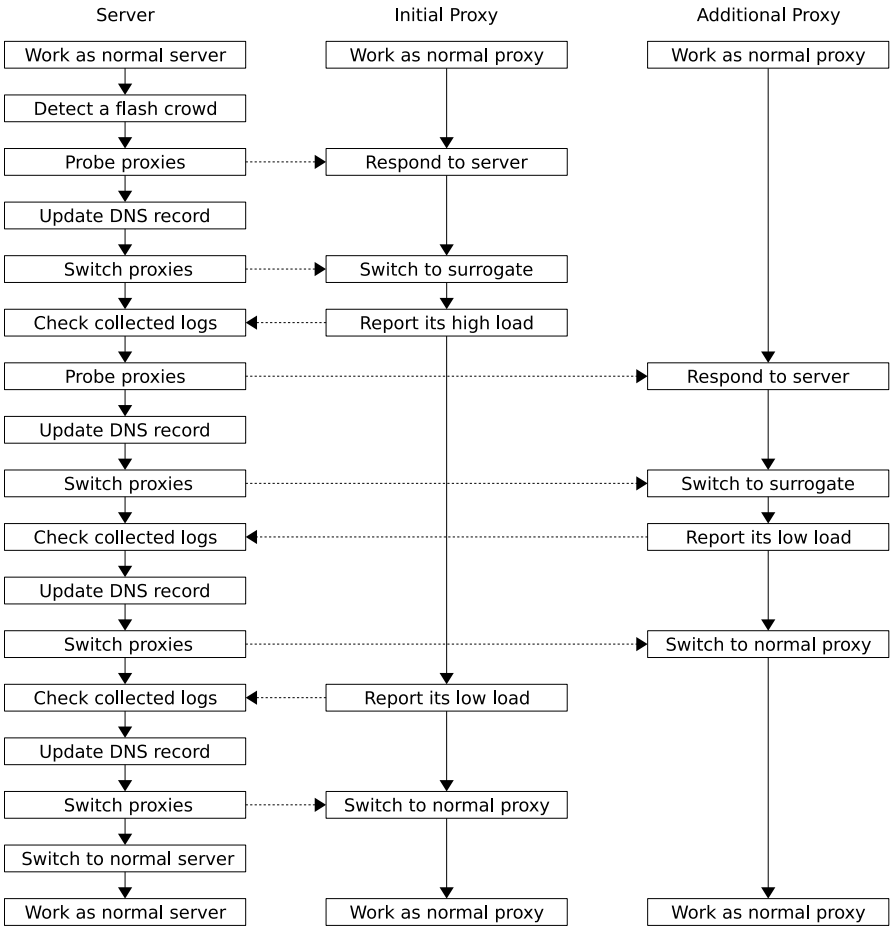


Fig. 4 Process flow overview in FCAN

When the request rate on the proxy is close to T_{high}, the proxy informs the server that the request rate is close to critical and increasing. When most of the proxies send the same information, the server starts inviting more proxies from the pool of other "free" member proxies which are not yet being involved in any overlay. The server probes the free proxies one by one to select a new proxy which can become utilized. Then the server and the new proxy carry out the same procedure as in 3.4.

When the load on any proxy is below T_{low} and no other proxies are suffering from a high load, the system dismisses them. The selection is done in the reverse order of invitation. Since DNS propagation may take a longer time, the change in proxy mode should be done later. If clients still reach the proxy after the DNS update, the proxy will act as a normal forward proxy, and retrieve the content from its local cache, or redirect the request to the member server or any temporary surrogate which is still involved in the overlay.

Figure 4 overviews the process flows of the server, and the initial and additional proxies, including network transition (presented in 3.4) and dynamic reorganization.

3.6 DNS-Based Redirection

To protect the server and network from overload, flooded requests must be redirected. In contrast to single site schemes where local load balancers work, this redirection is done within a wide-area environment, inside which the proxies may be geographically distributed. As mentioned above, we use DNS-based request redirection. DNS is an infrastructure for all the Internet applications including the Web, it is ubiquitous across the Internet, and it is transparent to clients.

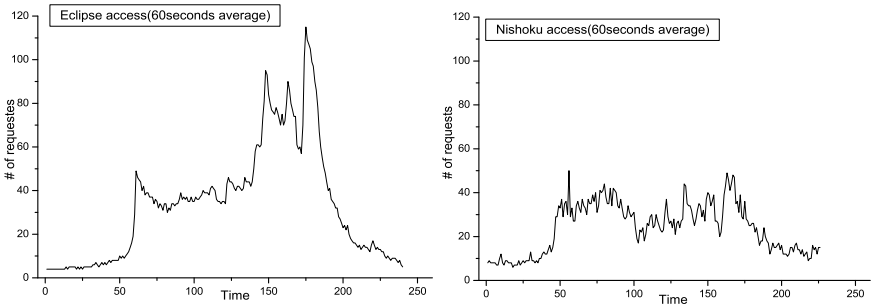
Authoritative DNS of the member server gives out the addresses of the involved member cache proxies instead of the address of the origin server when a client tries to resolve the server name through its local DNS server. The address given to the client may be any of the proxies under a certain selection policy, possibly in a simple round-robin manner or preferably in a load-balancing and proximity-based manner. Redirected requests for flash-crowd objects are conducted by the target proxy [48].

We use a specialized DNS server (or, in the strict sense, a DNS wrapper), called TENBIN [41, 42], on the server site which allows DNS look-up entries to be modified dynamically. TENBIN is one of our research products, and has already been used in practice, for example, in the "Ring Server" [37] and "LIVE! ECLIPSE" [26] projects. TENBIN also supports policy configuration for selecting an "appropriate" address. The policy could be based on a load-weighted algorithm, a proximity-based algorithm, a cache locality-based algorithm, or it could be conducted as simply as in a round-robin fashion.

Once being modified, the new addresses are propagated through the Internet to the client side DNS servers. One problem is that DNS caches may delay the propagation, with the result that the requests still continue to go to the origin server. This can be controlled by setting DNS records with a short expiration time, i.e. zero Time to Live(TTL). We have amassed much experience on DNS propagation both from experiments and from the practical use of TENBIN. It requires $10 \sim 15$ min. to complete worldwide propagation, but this is negligible compared to a typical flash crowd which may last for several hours or several days [12].

3.7 Simulation-Based Evaluations

For preliminary verification and evaluation of FCAN, we built a thread-based simulator of a virtual network with TCP/UDP and application layers. We have run experiments considering several scenarios of flash crowds, and below we present one with real access logs which were provided from the "LIVE! ECLIPSE" project [26]. On March 29th, 2006, from 9:00 to 11:30 GMT, the project delivered Web streaming,



(a) Access to www.live-eclipse.org (b) Access to www.nishoku.jp

Fig. 5 Accesses to two eclipse streaming sites

from two different server sites, for the solar eclipse that took place in Turkey, Libya, and Egypt. The two sites were:

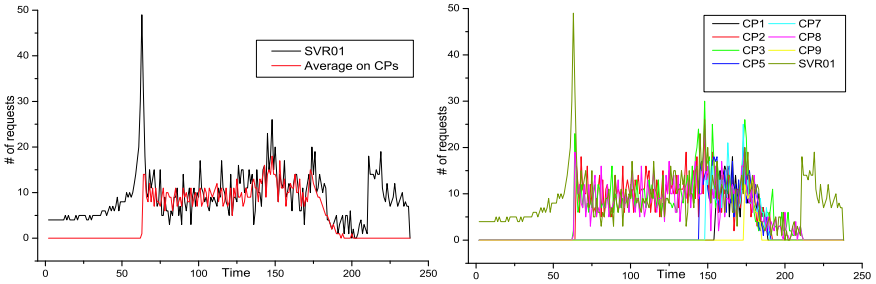
- http://www.live-eclipse.org
- http://www.nishoku.jp

While the former was accessed by clients from all over the world, the latter was accessed mostly by clients in Japan. There was a difference in access patterns for these two sites, since the expected access rate for the Live-Eclipse site was much higher than for the Nishoku site. Figure 5 shows the log data of the accesses for these sites for the period during which the eclipse was in process.

When fed to the simulator, these logs for the two sites were scaled down. The log of Live-Eclipse has been scaled down by 30, and the log for Nishoku by 10. Every simulation second corresponds to one minute of real time. Our experiment used two different member servers: one (SVR01) for Live-Eclipse, and the other (SVR02) for Nishoku. The experiment used ten member cache proxies for alleviation. The priorities (probing order) of the proxies for these member servers were set differently, and the initial subsets of proxies were also different between the member servers according to their priorities and the magnitude of the flash crowd.

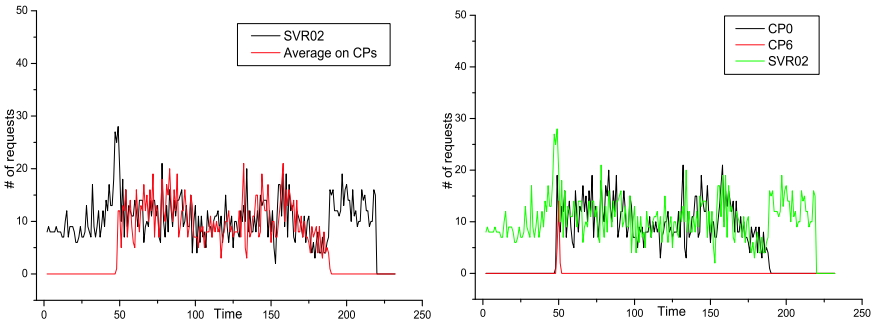
Figures 6 and 7 show the results of the simulation, where Fig. 6 shows the "Live Eclipse" overlay around SVR01, and Fig. 7 shows the "Nishoku" overlay around SVR02. The left graphs in Figs. 6 and 7 include average loads of the proxies, while the right graphs include individual loads.

In the "Live Eclipse" overlay, seven proxies, two initials and five additionals, were involved, as shown below:



(a) Server and average of proxies (b) Server and individual proxies

Fig. 6 Load alleviation in "Live Eclipse" overlay



(a) Server and average of proxies (b) Server and individual proxies

Fig. 7 Load alleviation in "Nishoku" overlay

SVR01	Joins at:	Leaves at:
CP8 (initial)	63	211
CP3 (initial)	63	211
CP2	65	191
CP5	145	190
CP7	149	189
CP1	155	188
CP9	174	184

For the first 60 sec., the server SVR01 handles the client requests by itself. The flash crowd to SVR01 starts at around the 60th second, and the server first invites two proxies to join in the alleviation process. These two and an additional one handle the load until the next rapid increase starting at around the 150th second. Then four more proxies are invited one by one. Using all of them, the average load on the system is kept below the threshold. After the 180th second, the amount of client requests starts decreasing, and the system dismisses the proxies one by one until the system is switched back to the C/S mode. The mode change occurs around the 200th second.

In the "Nishoku" overlay, only two proxies, one initial and one additional, are involved because of the relatively lower load, as presented below:

SVR02	Joins at:	Leaves at:
CP0 (initial)	48	189
CP6	49	51

The flash crowd to SVR02 starts at around the 50th second. At this moment, the highest peak of client requests is reached. CP0 is initially involved in the overlay, then immediately CP6 is invited, but only for 2 sec.

3.8 Concluding Remarks

The most prominent characteristics of FCAN are its dynamic and adaptive organization and reorganization features of the CDN-like overlay, which, as far as we know, cannot be found in any other related works. Here, we presented here that these unique features of FCAN are effective. The intermediate-layer solution using cache proxies in FCAN is, compared to the client-layer solutions, easier to manage and control. Moreover, compared to the server-layer solutions, it is more flexible and closer to clients.

The current design is the first version, and it still has some features which need to be improved. Threshold-based flash crowd detection should be more sophisticated, and this will be discussed later. Priority-based proxy grouping is now being replaced by an autonomous decentralized clustering scheme.

Another issue is the coarse granularity of redirection. A flash crowd is object-oriented, while DNS-based redirection is machine-oriented, since DNS deals only with machine names. It would be preferable to direct only the requests for flash-crowd objects to the proxy overlay and to pass other requests for the non-flash-crowd objects through, as usual. HTTP-based redirection and URL rewriting techniques offer fine-grained object-oriented redirection, however, they are not transparent to clients.

Quantitative and rigorous evaluations of FCAN are not included in the preliminary simulations so far. Real implementation on the Internet will be consisting of:

- A specialized Web server with a wrapper module for FCAN functions. Its core could be Apache for example, and the wrapper would intercept requests to the core server.
- A specialized cache proxy with a wrapper module for FCAN functions. Its core could be Squid for example, and the wrapper would intercept requests to the core proxy.
- An enhanced DNS server. TENBIN is a good candidate, which is actually already used in practice.

Table 1 Summary of design issues adopted by related systems

Design Issue		DD	DS	ML	BS	CO	CP	PF	FC
System Archtecture	Server-layer	√	√						
	Proxy-layer			√	√	√			√
	Client-layer						√	√	
Surrogate Servers	Dedicated Servers	√							
	Existing Servers		√		√				
	Existing Proxies			√	√	√			√
	Existing Clients			√			√	√	
Client Transparency	Client Unaware	√		√					√
	Browser Unchanged	√	√		√	√		√	√
Client Redirection	DNS-based	√	√		√	√			√
	URL Rewrite	√	√		√	√			
	HTTP-based						√		
Replica Placement	Mirror Replica	√	√						
	Caching on Demand	√	√	√		√		√	√
Object Locating	DHT-based P2P				√	√			
	Unstructured P2P						√	√	√
	Cooperative Caching			√					
Cache Miss Avoidance	Dynamic Delegation	√							
	Push Service								√
Adaptive Transition	Temporary Servers		√						
	Temporary Proxies								√
	Temporary Clients						√		

Note. DD: CDN with Dynamic Delegation, ML: Multi-Level Caching, BS: BackSlash, DS: Dot-Slash, CO: CoralCDN, CP: CoopNet, PF: PROOFS, FC: FCAN.

FCAN was originally designed for flash crowd protection, but in fact, it is not only limited to this. It adjusts server load under a predefined threshold facing against any unexpected traffic surges, and we can thus assume that some kinds of DDoS attacks could also be handled.

4 Visionary Thoughts for Practitioners

Table 1 summarizes some significant related research efforts (most of them are mentioned so far in this chapter) and compares them with FCAN. Our observations are presented in the following:

1. Over-provisioning based on peak demand or using CDNs to increase server locations in advance is costly and inefficient.
2. A client-side P2P overlay addresses flash crowds reasonably well, but not perfectly, since it loses client transparency and controllability.

3. In addition, some P2P systems have overheads, such as the flooding problem, which cannot be neglected while facing the ending of flash crowds.
4. Intermediate-layer solutions have advantages over other layer solutions. This is because the caching technique is promising in its ability to address flash crowds whose target objects are supposed to be small-sized and static.
5. However, most intermediate-layer solutions neglect the problem that more than 60% of objects are uncached at the beginning of a flash crowd, which results in the origin server being at risk for a surge of cache misses.
6. Forward proxies rather than servers are better employed as surrogate servers. Proxies are nearer to clients, and are thus more beneficial to client response time and network congestion.
7. To handle flash crowds flexibly and efficiently, an adaptive transition technique is necessary, which organizes the potential resources along the way, rather than occupying them all the time.

To sum up, each of the current research works have both merits and demerits. Through the comparison, we have come to conclude that there is still a lack of an efficient approach that can handle flash crowds in a flexible, reliable, and cost-effective manner, while remaining transparent to the end users.

5 Future Research Directions

While there are many research problems required to be addressed in the context of flash crowds alleviation, as future research directions, we focus on two main issues.

Early Detection of Flash Crowds We have already noticed the phenomenon that shortly before a flash crowd comes to a server, a number of requests sometimes floods to DNS servers to resolve the server's name. This must imply that if we had a technique for collecting the amount of requests from distributed DNS servers and for analyzing them, we could possibly predict the coming of a flash crowd, and thus give an advance warning to the target server.

Handling of Dynamic Objects It must be common to all the CDN systems to address dynamic object dissemination. Dynamic objects can be divided into two categories:

- Dynamically generated contents (mostly using script codes and a back-end database)
- Frequently updated contents (as often found in News sites)

The simplest way would be to replace a dynamic object with its trimmed static version under a heavily-loaded situation at the cost of its service quality [1].

It must be relatively easy to handle a dynamic object in the former category, if the back-end database is read-only. If not, or if a dynamic object falls in the latter category, we must provide a fast and reliable scheme for updating all the replicas in a consistent manner. This topic, update synchronization and coherence, has been

investigated extensively in the area of distributed databases, distributed caches, and distributed shared memories. Achievements out of these studies could be applied in this context.

Finally, some integration among server-layer, intermediate-layer and client-layer solutions could be interesting and promising.

6 Conclusion

Short-term Internet congestion, known as flash crowds, poses new challenges for designing scalable and efficient distributed server systems. This chapter analyzed the major characteristics of flash crowds, studied the related research works extensively, and pointed out the need for a dynamic network to handle short-term Internet congestion. Then we presented our original and unique idea of a dynamic CDN network which adaptively optimizes its own network architecture between C/S and CDN configurations to alleviate flash crowds. Our observations suggest that FCAN could be a good basis for performing early detection of flash crowds, and handling of dynamic objects. Therefore, we conclude that it could be a pathway to realize future innovations for handling flash crowds efficiently.

Acknowledgements This chapter is based on joint work with Prof. Toshihiko Shimokawa (Kyushu Sangyo University, Japan), Dr. Chenyu Pan (China), and Dr. Merdan Atajanov (Turkmenistan). We also thank Ms. Kate Miller for English proof reading.

References

1. Abdelzaher TF, Bhatti N (1999) Web Server QoS Management by Adaptive Content Delivery. In: Computer Networks, 31(11–16):1563–1577
2. Adler S (1999) The Slashdot Effect, an Analysis of Three Internet Publications. http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html
3. Akamai Technologies Inc. http://www.akamai.com
4. Ari I, Hong B, Miller EL, Brandt SA, Long DE (2003) Managing Flash Crowds on the Internet. In: Proc. 11th IEEE/ACM Int. Symp. on Modeling, Analysis, and Simulation of Comp. and Telecomm. Sys., 246–249
5. Arlitt M, Cherkasova L, Dilley J, Friedrich R, Jin T (1999) Evaluating Content Management Techniques for Web Proxy Caches. In: ACM SIGMETRICS Performance Evaluation Review, 27(4):3–11
6. Arlitt M, Jin T (2000) A Workload Characterization of the 1998 World Cup Web Site. In: IEEE Network, 14(3):30–37
7. Atajanov M, Shimokawa T, Yoshida N (2007) Autonomic Multi-Server Distribution in Flash Crowds Alleviation Network. In: Proc. IFIP 3rd Int. Symp. on Network Centric Ubiquitous Systems (LNCS 4809, Springer), 309–320
8. Barford P, Plonka D (2001) Characteristics of Network Traffic Flow Anomalies. In: Proc. ACM SIGCOMM Internet Measurement Workshop, 69–73
9. BitTorrent Website. http://www.bittorrent.com/
10. Breslau L, Cue P, Fan L, Phillips G, Shenker S (1999) Web Caching and Zipf-like Distributions: Evidence and Implications. In: Proc INFOCOM 1999, 126–134

11. CERT (1996) TCP SYN Flooding and IP Spoofing Attacks. Advisory CA-1996-21, http://www.cert.org/advisories/CA-1996-21.html
12. CERT (1999) Denial of Service Attacks. http://www.cert.org/tech_tips/denial_of_service.html
13. Chandra A, Shenoy P (2003) Effectiveness of Dynamic Resource Allocation for Handling Internet Flash Crowds. Tech. Report, TR03-37, Dept. of Computer Science, Univ. of Massachusetts Amherst
14. Chen X, Heidemann J (2002) Flash Crowd Mitigation via an Adaptive Admission Control Based on Application-Level Measurement. Tech. Report, ISI-TR-557, USC/ISI
15. Cherkasova L, Phaal P (2002) Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites. In: IEEE Trans. on Computers, 51(6):669–685
16. Foster I, Kesselman C, Tuecke S (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: Int. J. of High Performance Computing Applications, 15(3):200–222
17. Freedman MJ, Freudenthal E, Mazieres D (2004) Democratizing Content Publication with Coral. In: Proc. 1st USENIX/ACM Symp. on Networked Systems Design and Implementation
18. Houle KJ, Weaver GM, Long N, Thomas R (2001) Trends in Denial of Service Attack Technology. CERT Coordination Center White Paper, http://www.cert.org/archive/pdf/DoS trends.pdf
19. Iyengar AK, Squillante MS, Zhang L (1999) Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance. In: World Wide Web, 2(1–2):85–100
20. Joubert P, King R, Neves R, Russinovich M, Tracey J (2001) High-Performance Memory-Based Web Servers: Kernel and User-Space Performance. In: Proc. USENIX 2001, 175–188
21. Jung J, Krishnamurthy B, Rabinovich M (2002) Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In: Proc. 11th Int. World Wide Web Conf., 252–262
22. Kandula S, Katabi D, Jacob M, Berger A (2005) Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In: Proc. USENIX 2nd Symp. on Networked Systems Design and Implementation, 287–300
23. Kong K, Ghosal D (1999) Mitigating Server-Side Congestion in the Internet through Pseudoserving. In: IEEE/ACM Trans. on Networking, 7(4):530–544
24. LeFebvre W (2002) CNN.com: Facing a World Crisis. In: USENIX Annual Tech. Conf., http://tcsa.org/lisa2001/cnn.txt
25. LimeLight Networks. http://www.limelightnetworks.com/
26. LIVE! ECLIPSE. http://www.live-eclipse.org/index_e.html
27. Lorenz S (2000) Is Your Web Site Ready for the Flash Crowd? In: Sun Server Magazine 2000/11, http://www.westwindcos.com/pdf/sunserver_11900.pdf
28. Lyer S, Rowstron A, Druschel P (200) Squirrel: A Decentralized Peer-to-Peer Web Cache. In: Proc. 21th ACM Symp. on Principles of Distributed Comp., 213–222
29. Moore D (2001) The Spread of the Code-Red Worm (CRv2). http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml
30. Nah F (2004) A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? In: Behaviour and Information Technology, 23(3):153–163
31. Niven L (1973) Flash Crowd. In: The Flight of the Horse, Ballantine Books, 99–164
32. Padmanabhan VN, Sripanidkulchai K. (2002) The Case for Cooperative Networking. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems, 178–190
33. Pan C, Atajanov M, Hossain MB, Shimokawa T, Yoshida N (2006) FCAN: Flash Crowds Alleviation Network Using Adaptive P2P Overlay of Cache Proxies. In: IEICE Trans. on Communications, E89-B(4):1119–1126
34. Pan C (2006) Studies on Adaptive Network for Flash Crowds Alleviation. Ph. D. Thesis, Saitama University
35. Park K, Lee H (2001) On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In: Proc. ACM SIGCOMM 2001, 15–26
36. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A Scalable Content-Addressable Network. In: Proc. ACM SIGCOMM 2001, 161–172

37. The Ring Server Project. http://ring.aist.go.jp/index.html.en
38. Rowstron A, Druschel P (2001) Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In: Proc. ACM 18th Symp. on Operating Systems Principles, 188–201
39. Rubenstein D, Sahu S (2001) An Analysis of a Simple P2P Protocol for Flash Crowd Document Retrieval. Tech.Report, EE011109-1, Columbia Univ.
40. Saroiu S (2001) Bottleneck Bandwidths. http://www.cs.washington.edu/homes/tzoompy/sprobe/webb.htm
41. Shimokawa T, Yoshida N, Ushijima K (2000) Flexible Server Selection Using DNS. In: Proc. Int.Workshop on Internet 2000, in conjunction with IEEE-CS 20th Int. Conf. on Distributed Computing Systems, A76–A81
42. Shimokawa T, Yoshida N, Ushijima K (2006) Server Selection Mechanism with Pluggable Selection Policies. In: Electronics and Communications in Japan, III, 89(8):53–61
43. Sivasubramanian S, Szymaniak M, Pierre G, Steen M (2004) Replication for Web Hosting Systems. In: ACM Comp. Surveys, 36(3):291–334
44. Stading T, Maniatis P, Baker M (2002) Peer-to-peer Caching Schemes to Address Flash Crowds. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems, 203–213
45. Stavrou A, Rubenstein D, Sahu S (2004) A Lightweight, Robust P2P System to Handle Flash Crowds, In: IEEE J. on Selected Areas in Comm., 22(1):6–17
46. Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H (2001) Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM 2001, 149–160
47. Wang J (1999) A Survey of Web Caching Schemes for the Internet. In: ACM Comp. Comm. Review, 29(5):36–46
48. Wang L, Pai V, Peterson L (2002) The Effectiveness of Request Redirection on CDN Robustness. In: ACM Operating Systems Review, 36(SI):345–360
49. Welsh M, Culler D (2003) Adaptive Overload Control for Busy Internet Servers. In: Proc. USENIX Conf. on Internet Technologies and Systems
50. Zao W, Schulzrinne H (2004) DotSlash: A Self-Configuring and Scalable Rescue System for Handling Web Hotspots Effectively. In: Proc. Int. Workshop on Web Caching and Content Distribution, 1–18
51. Zona Research, Inc.(1999) The Economic Impacts of Unacceptable Web-Site Download Speeds. White Paper, http://www.webperf.net/info/wp_downloadspeed.pdf