Flash Crowd Mitigation via Adaptive Admission Control Based on Application-Level Observations

XUAN CHEN and JOHN HEIDEMANN
University of Southern California

We design an adaptive admission control mechanism, *network early warning system* (NEWS), to protect servers and networks from flash crowds and maintain high performance for end-users. NEWS detects flash crowds from performance degradation in responses and mitigates flash crowds by admitting incoming requests adaptively. We evaluate NEWS performance with both simulations and testbed experiments. We first investigate a network-limited scenarion in simulations. We find that NEWS detects flash crowds within 20 seconds. By discarding 32% of incoming requests, NEWS protects the target server and networks from overloading, reducing the response packet drop rate from 25% to 2%. For admitted requests, NEWS increases their response rate by two times. This performance is similar to the best static rate limiter deployed in the same scenario. We also investigate the impact of detection intervals on NEWS performance, showing it affects both detection delay and false alarm rate. We further consider a server memory-limited scenario in testbed experiments, confirming that NEWS is also effective in this case. We also examine the runtime cost of NEWS traffic monitoring in practice and find that it consumes little CPU time and relatively small memory. Finally, we show NEWS effectively protects bystander traffic from flash crowds.

Categories and Subject Descriptors: C.2.3 [**Computer-Communication Networks**]: Network Operations; C.4 [**Computer Systems Organization**]: Performance of Systems

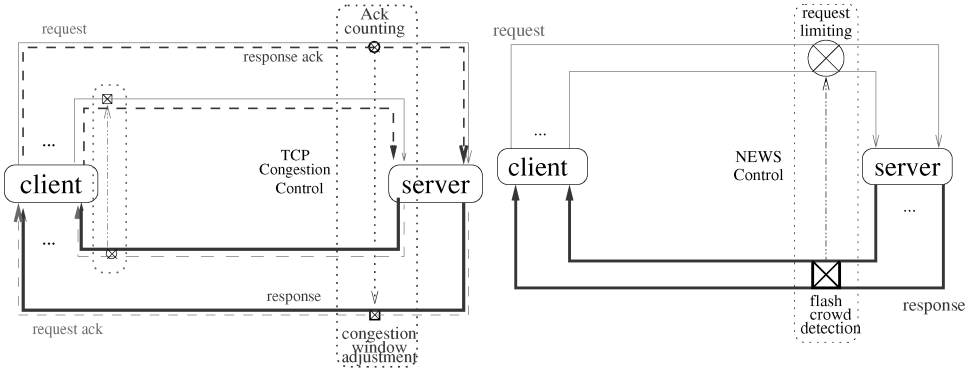
General Terms: System design, Performance and reliability

Additional Key Words and Phrases: Flash crowds, admission control, simulations, experimentation with testbeds

1. INTRODUCTION

Recent studies [Jung et al. 2002; Mahajan et al. 2001; Park and Lee 2001; Jamjoom and Shin 2003] show that the Internet is vulnerable to persistent overloading caused by flash crowds [Jung et al. 2002; Jamjoom and Shin 2003].

This material is based on work supported by DARPA via the Space and Naval Warfare Systems Center San Diego under Contract No. N66001-00-C-8066 ("SAMAN"), and by the CONSER project supported by the National Science Foundation.
Author's address: S. Chen, Information Sciences Institute, University of Southern California, Suite 1001, 4676 Admiralty Way, Marina del Rey, CA 90292; email: xuanc@catarina.usc.edu.



(a) TCP congestion control

(b) NEWS control between requests and responses

Fig. 1. Comparison of TCP congestion control and NEWS.

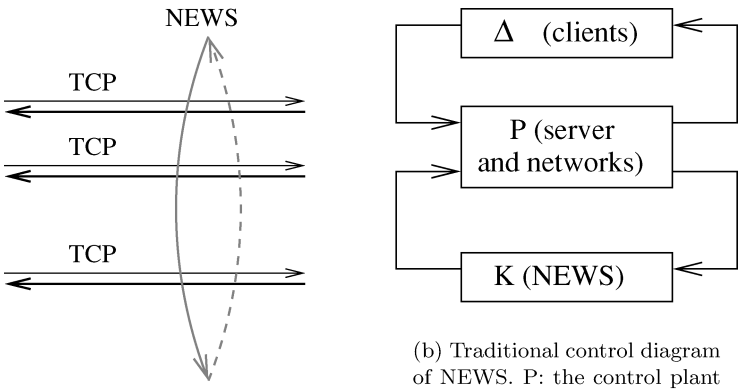
Flash crowds happen when many end-users simultaneously send requests to one Web site usually because of special events attracting interest of the mass population. These events could be pre-scheduled such as a webcast of a popular movie or unpredictable including natural disasters such as earthquakes, breaking news stories and links from popular Web sites (that is, the "slash-dot effect" [Adler 1999]).

Flash crowd traffic persistently overloads server or networks. When a flash crowd happens, the volume of requests to the *target Web server* increases dramatically. The magnitude may be tens or hundreds of times more than normal conditions. These requests may overload network connections [Mahajan et al. 2001; Jung et al. 2002] and the target server. In the meanwhile, response traffic could also congest networks (often in the first few links where traffic concentration is the largest). As a result, most users perceive unacceptably poor performance. In addition, flash crowds unintentionally deny services to other end-users who either share common network links with flash crowd traffic or retrieve unrelated information from the target server. We call the corresponding traffic *bystander traffic*.

1.1 Aggregate-Level Control Between Requests and Responses

The Internet applies TCP congestion control to cope with resource constraints [Floyd and Fall 1999; Jacobson 1988]. As shown in Figure 1(a), TCP adjusts its window size according to acknowledgments received and infers network congestion from packet loss. However, TCP is not able to relieve the persistent overloading during flash crowds because it only regulates per-connection behavior. Other congestion control algorithms that aggregate information per-host (for example, the congestion manager [Balakrishnan et al. 1999]) also fail to solve this problem because connections arrive from many hosts during flash crowds.

In this work, we propose *network early warning system* (NEWS), a router-based system, to impose aggregate-level control between requests and



(a) NEWS enforces cooperation among TCPs

(b) Traditional control diagram of NEWS. P: the control plant of NEWS, K: the control algorithm, Δ: the other part in control loop

Fig. 2. NEWS control diagrams.

responses. As shown in Figure 1(b), NEWS observes Web response performance (that is, flash crowd detection) and adjusts the incoming request rate to the target server accordingly.

Figure 2(a) shows the logical relationship between NEWS and TCP. The Internet has many individual TCP congestion control loops to enforce the right behavior at the connection level. However, they only operate on their own and do not cooperate with each other. It is the lack of cooperation that makes flash crowds possible. On the other hand, NEWS enforces cooperation among individual TCPs. As a result, we have a global control of TCP connections during flash crowds that will not allow excessive connections to overwhelm servers and networks. Figure 2(b) depicts the control logic of NEWS with a traditional control diagram.

1.2 Contributions

NEWS is a router-based adaptive admission control mechanism to impose aggregate-level control between requests and responses. It has two novel aspects.

First, NEWS does not consider per-flow service requirements for incoming requests like most admission control schemes [Breslau et al. 2000; Cetinkaya and Knightly 2000; Mundur et al. 1999; Rahin and Kara 1998]. Instead, it determines end-user perceived performance through measurement. Moreover, NEWS only measures performance for aggregates (details in Section 4.1), rather than for all existing flows like measurement-based admission control (MBAC) [Breslau et al. 2000; Cetinkaya and Knightly 2000].

NEWS is also different from schemes that monitor increases in requests directly [Blazek et al. 2001; Jung et al. 2002]: it detects flash crowds from performance degradation in Web *response*. As discussed in Section 3.2, observation in response traffic captures overloading both at the target server and in networks. On the other hand, an increase in request rate alone does not necessarily imply

low performance for end-users. Therefore, we believe that we should detect flash crowds by monitoring response performance.

Second, NEWS monitors changes in the performance of high-bandwidth responses because of their sensitivity to overloading conditions. Based on this observation, NEWS adjusts the admitted request rate automatically and adaptively. These two approaches make NEWS a self-tuning system, adaptive in both server- and network-limited scenarios (as we will show in Section 7.2, 8.2, and 8.3).

In the case of multiple servers connecting through the NEWS router, we propose the *hot-spot identification algorithm* (Section 4.4) to help NEWS regulate incoming requests intelligently, that is, automatically identifying one hot server even if many other servers are operational behind the NEWS router.

1.3 NEWS Performance Evaluation

We first evaluate NEWS performance through simulations in Section 7. We find that NEWS detects flash crowds within 20 seconds. By discarding about 32% of incoming requests, NEWS protects servers and networks from overloading: reducing the response packet drop rate from 25% to 2%. NEWS also increases the response rate for admitted requests by two times. This performance is similar to the best possible static rate limiter deployed in the same scenario.

We further implement NEWS on a Linux-based router (Section 6). With testbed experiments (Section 8), we validate NEWS performance in a network-limited scenario quantitatively in a more realistic experimental model than simulations.

We also configure a server-limited scenario where request processing exhausts the server's memory. We show that NEWS is an adaptive system that effectively prevents server and network overloading in both cases. More specifically, NEWS detects flash in crowds in around one detection interval (88 seconds with 64 seconds' detection interval). It automatically regulates incoming requests to a proper rate. As a result, NEWS protects target server and networks from overloading by discarding about 49% of excessive requests. NEWS shows a similar performance improvement for accepted end-users as in the simulation study.

We also find through both simulations and testbed experiments that NEWS effectively protects traffic to nearby, *bystander* servers. For example, it reduces median end-to-end latency for bystander Web traffic by about 10 times (Section 8.6).

We investigate the overhead of NEWS detection algorithm in Section 8.5.1. Our analysis shows that NEWS has similar algorithmic performance as a stateful fire-wall. We further examine NEWS' runtime overhead with testbed experiments, measuring router CPU and memory usage during a flash crowd. Our statistics (Section 8.5) show that NEWS consumes less than 5% of CPU time and 3–10MB memory. Overall, NEWS imposes small overhead on routers and is applicable to real networks. We also investigate system performance of a target server under flash crowds in detail.

We study the effect of detection intervals on NEWS performance in both simulations and testbed experiments. Our results show that NEWS triggers false alarms with small intervals like 15 or 30 seconds. On the other hand, larger detection intervals increase the detection delay of NEWS. We investigate this trade-off (between detection delay and false alarm rate) in Section 7.4 and 8.4.

2. RELATED WORK

In this section, we briefly describe mechanisms to accommodate flash crowds through resource provisioning. We also review other commonly used schemes to protect servers and networks from overloading such as admission control, congestion control, and overloading control.

2.1 Web Caching and Content Delivery Networks

Infrastructure vendors such as Akamai deploy Web caches and content delivery networks (CDN) to protect servers from overloading during flash crowds. These services need infrastructure support and are usually expensive. Further, recent studies [Arlitt and Jin 2000; Jung et al. 2002] show that the current Web caching scheme is not efficient because dynamic content such as pages for breaking news are not cached before flash crowds. Jung et al. [2002] proposed a new scheme—adaptive Web caching—for improvement. On the other hand, we propose a low-cost alternative by regulating incoming traffic adaptively.

Ideally, we should provide enough resources to prevent networks and servers from overloading. For example, we can replicate the target server to distribute its load. This approach needs infrastructure support. Further, in reality, there are circumstances where it is either difficult or impossible to estimate and provide enough required resources. So, we propose NEWS to regulate excessive requests based on currently available resources.

2.2 Admission Control

Admission control plays an important role in supporting applications with service requirements such as real-time constraint (for example, delay and jitter). Looking at public telephone networks [Gibbens et al. 1995] and integrated service networks [Clark et al. 1992], a call (or a new connection) explicitly describes its service requirement such as two channels for telephone conversation or 1Mbps bandwidth for Video-on-Demand service [Mundur et al. 1999]. Based on this service profile and currently available resources (circuits or network bandwidth), admission control makes a decision as to whether it should accept the incoming request or not.

Although appropriate for telephone and integrated service networks, it may be difficult for most Internet applications (e.g., Web and FTP) to accurately estimate and describe their service requirements. So, we believe that we need to determine application requirements dynamically through measurement. NEWS is such an example.

To avoid underutilization by estimating resource consumption with a statistical model, measurement-based admission control (MBAC) [Breslau et al. 2000; Cetinkaya and Knightly 2000; Jamin et al. 1995] determines the currently

available resources through traffic measurement. However, MBAC still needs the service profile of incoming requests. Further, MBAC aggregates the performance of all existing flows to evaluate current resource consumption, while NEWS only measures the transmission rate of high-bandwidth responses. Also, MBAC is more conservative and only accepts incoming requests upon sufficient resource; NEWS sends all requests through unless a flash crowd is detected.

In order to protect networks and servers from overloading, we could also apply a static rate limiter to regulate the incoming request rate below a certain threshold. It rejects excessive requests to ensure that networks and servers always work under capacity. Despite its simplicity, a static rate limiter lacks adaptivity to different environments. In order to work properly, network operators need to choose the rate limit carefully based on current configurations and their experience [Barford and Plonka 2001]. Usually, this choice is specific to a certain server or network connection and needs manual adjustment when the server's capacity or the connection's bandwidth changes. Contrary to this, we propose self-tuning traffic control algorithms, NEWS, which adapt to both network- and server-limited scenarios (as shown in Section 8.2 and 8.3).

Another similar work in this area is the network weather service (NWS) [Wolski et al. 1999]. It monitors and forecasts system performance such as link utilization and server load. Both NWS and NEWS apply change detection algorithms. Unlike NWS, NEWS does not rely on centralized data processing, and hence is more robust and adaptive in different scenarios. We also present the novel flash crowd detection algorithm of NEWS in Section 4.

2.3 Congestion Control at Different Levels

TCP and its variations apply end-to-end congestion control, which is fundamental to the stability of today's Internet. However, since it operates at flow-level, TCP is not sufficient to regulate aggregate behavior during flash crowds.

Congestion manager (CM) [Balakrishnan et al. 1999] is a per-host-based control algorithm, multiplexing concurrent flows among different applications of one end-host to ensure that they react to congestion cooperatively. Different from CM, NEWS does not regulate the behavior of individual hosts. As we will show in Section 3, per-host-based information does not help to solve the persistent overloading during flash crowds.

Researchers have also proposed to control Internet traffic at the aggregate level. For example, the aggregate-based congestion control (ACC) [Mahajan et al. 2001] regulates the rate of aggregates consuming most of the network bandwidth and promotes fair bandwidth allocation among different flows traversing the same link.

In terms of flash crowd mitigation, we can apply ACC to regulate response traffic in flash crowds which is likely to consume high bandwidth and cause large packet drops. However, ACC is not sufficient to mitigate flash crowds fundamentally because it does not have the complete control loop between requests and responses that NEWS applies. As we will show in Section 3, the aggregate-level control that NEWS imposes is important to protect server and networks from flash crowds.

2.4 Adaptive Queue Management Algorithms

Routers apply different queuing algorithms to support QoS for end-users and achieve fair bandwidth allocation for different flows. These algorithms differ in the flow states they keep. On the one hand, RED [Floyd and Jacobson 1993] do not keep per-flow states. On the other hand, Fair Queuing [Demers et al. 1989] maintains states for each flow and achieves fair bandwidth allocation for individual flows. CSFQ [Stoica et al. 1998] improves fair queuing by eliminating per-flow states from core routers. Instead, it keeps flow rate in packet headers.

There are some proposals between these two extremes which keep partial flow states [Lin and Morris 1997; Mahajan and Floyd 2001; Pan et al. 2000]. For example, FRED [Lin and Morris 1997] keeps states for flows that currently have packets in buffer. FRED determines the dropping probability for one flow according to the number of its packets being queued. RED with preferential dropping (RED-PD) [Mahajan and Floyd 2001] only keeps states for high-bandwidth flows and drops their packets preferentially.

Unlike these adaptive queuing management algorithms, the focus of this work is on flash crowd detection and mitigation. More specifically, NEWS captures abrupt changes in traffic observation (Section 4) and regulates aggregate behavior (e.g., rate-limit flash crowd traffic).

Persistent dropping (PD) [Jamjoom and Shin 2003] is proposed as an augment for adaptive queuing algorithms to mitigate flash crowd traffic. It applies a fine-grain connection-level control to persistently discard retransmitted packets. PD can also be integrated into our NEWS framework as an alternative reaction scheme. On the other hand, it does not address important issues such as flash crowd detection and bystander traffic protection as NEWS does.

2.5 Server Overloading Control

Recent studies [Cetinkaya and Knightly 2000; Chen and Mohapatra 2002; Lu et al. 2001; Welsh and Culler 2003] propose to apply overloading control and service differentiation on Web servers to improve Web performance under heavy loading and flash crowds. These schemes are complimentary to router-based algorithm like NEWS. Further, we believe that NEWS is also applicable to Web servers. NEWS is more flexible than server-based overloading control, effectively protecting bystander traffic from flash crowds (Section 7.5 and 8.6).

3. FLASH CROWDS AND EARLY WARNING

Flash crowd traffic shows different patterns than normal traffic [Jung et al. 2002]. In this section, we show some of its characteristics by examining two server HTTP logs. One was collected when a flash crowd happened to a private server (slash-dot effect [Adler 1999]), the other is from 1998's World Cup Web site (www.france98.com). To better describe the characteristics of flash crowd traffic, we first define some terminology.

As depicted in Figure 3, a Web connection from client C to server S contains one (with HTTP/1.1 [Gettys et al. 1999]) or a series (with HTTP/1.0 [Berners-Lee et al. 1996]) of request and response exchanges (that is, request and corresponding response flows). We define the *lifetime* of this Web connection (T_w) as

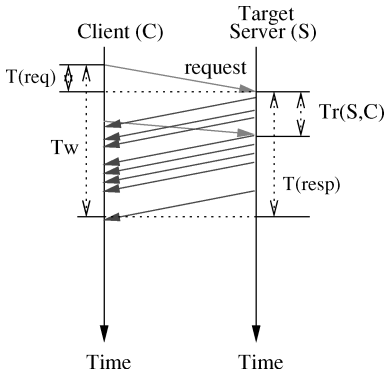


Fig. 3. Request and response transmission latency, Web connection lifetime, and request interval.

the time interval from a client sending the first request packet untill it receives all response packets.

We quantify the performance of Web request and response flows with the following two metrics. *Transmission latency*, $T(f)$, records the time interval (e.g., in seconds) from the first packet sent until the last packet is received. *Transmission rate*, $R(f)$, measures the flow data rate (in bytes per second, e.g.). Analytically, we have $R(f) = L/T(f)$, where L is the amount of data transferred by flow f (in bytes, for example). In this work, we measure $R(f)$ at the access router of the target server (as discussed in Section 3.3). Flows show various transmission latencies and rates due to the server load and congestion condition in the networks.

From the servers' point of view, we define *request interval* ($Tr(S, C)$) as the time between two adjacent requests that server S receives from client C. We denote $Tr(S)$ as the interval of request from all clients to server S. Alternatively, we define $R_p(S)$ as the *request rate* (in number of requests per second) observed by server S. $R_p(S)$ records the number of requests sent to S within one time unit.

3.1 Observations of Flash Crowd Traffic

We present two observations based on the statistics of two Web server log files. First, requests show very small interarrival time: the mean request interval is around 1 second in the slash-dot trace and less than 10ms in the World Cup trace. Sometimes, the target server even recorded the same arrival times for some requests due to its time granularity. So, the request rate shows a spike when a flash crowd happens.

Second, we find that most connections are short in the two log files we studied. That is, they retrieve small pages.[1] In both logs, most (90% for the slash-dot log and 50% for the World Cup log) requests are for pages less than 1K bytes. In the World Cup HTTP log, more than 90% of requests are for pages smaller than 10Kbytes. And, they carry about 50% of the network load. So, there is a

[1]The actual response size varies in different scenarios according to the content hosted on target servers.

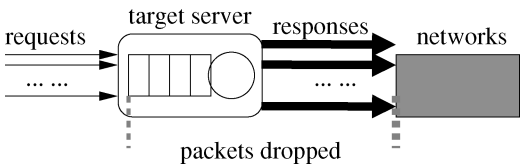


Fig. 4. Server and network overloading during a flash crowd.

concentration on small pages in the log files we studied. Based on these observations, we propose a simple two-layered flash crowd traffic model to test our design in the early stages.

The characteristics of flash crowd traffic impose challenges for detection. In this study, we focus on detecting flash crowds caused by unpredictable events. This is the most challenging case because of uncertainties in three aspects including happening time, user interest (i.e., the magnitude of the request rate increase), and response size. In this study, we propose an adaptive algorithm to detect flash crowds without knowledge of this information.

3.2 Reduced Web Performance During Flash Crowds

End-users may experience increased Web latency during flash crowds. As shown in Figure 4, several factors contribute to this increase. First, even though each request may contain just one small packet, too many of them can still cause congestion in networks. So, requests may be delayed or even dropped by networks. Second, the target server only accepts part of incoming requests, due to its CPU or memory constraint, and simply discards others. The target server is overloaded during flash crowds. It processes requests and generates responses slowly. Finally, responses contain a number of larger packets and inject more load than requests do. They are more likely to congest networks and increase transmission latency. When a response packet is dropped due to congestion, the target server needs to retransmit the lost packet even it is already overloaded.

Traditional approaches detect flash crowds from increases in request rate. However, as we will show, this increase does not necessarily correlate with overloading conditions during a flash crowd. Other factors such as system capacity and response size are also important.

In different circumstances, flash crowds may cause overloading at the target server or in networks. However, from an end-user point of view, they always perceive an increased Web latency or a decreased Web response rate. Therefore, we design NEWS to detect flash crowds from degradation in the Web response rate. As we will show in Section 8, this approach helps NEWS to adapt to both network- and server-limited scenarios automatically.

3.3 Overall Design of the Network Early Warning System

As shown in Figure 5, NEWS has three main components: the flash crowd detector, the request regulator, and the control logic. We design NEWS in a modular style so that we have the flexibility to apply new techniques without modifying its framework. For example, we could adopt Web caching techniques in the module of request regulator. We present the detailed design of flash crowd detection and regulation in Section 4 and 5.

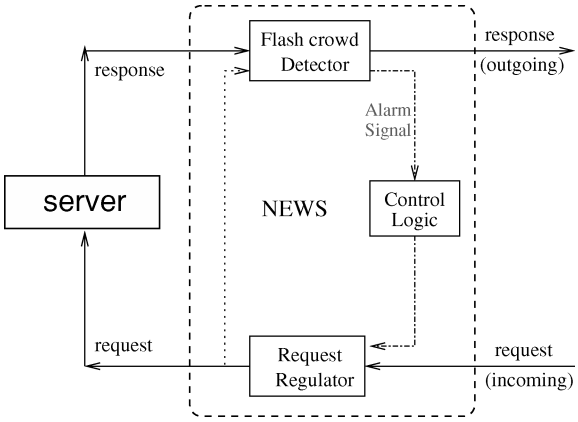


Fig. 5. Overall NEWS architecture (requests and responses physically traverse the same network connection).

NEWS imposes a global control loop over many individual TCPs (as shown in Figure 2(a). Unlike TCP's small time scale (i.e., RTT), this global control loop operates at a larger time scale such as 1 minute (discussed in more detail in Section 7.4 and 8.4). As a result, NEWS can apply more sophisticated techniques while only imposing small runtime CPU and memory overhead to routers (shown in Section 8.5). For example, we can apply complicated change detection algorithms or implement fine-grained request regulator.

We had three assumptions when designing NEWS. First, we should be able to deploy NEWS reasonably close to the target server. For example, we install NEWS on the access router of the target server or the server farm it operates.

Second, we assumed that requests and responses traverse the same access router. As depicted in Figure 5, when the access router is close to the target server, it is reasonable to treat incoming traffic as request and outgoing traffic as response. So, the access router does not need to decode application-level information for NEWS, for example, examining packet contents for HTTP header information. This assumption well holds for the case of the server farm. But sometimes the network behind the NEWS router might also provide Internet services for end-users. In this case, both incoming and outgoing traffic are mixed with Web requests and responses. So, the assumption above is questionable. Therefore, it is important to deploy NEWS near the target server.

In this work, we focus on a scenario that an enterprise network or a server farm is connected to the Internet with a single network link. A potentail research direction is to investigate the deployment of NEWS to a multihomed network where requests and responses of the same Web connection may traverse different routers.

4. DETECTING FLASH CROWDS

As shown in Figure 5, the flash crowd detector sets an alarm signal after it detects flash crowds. In this design, we consider the following three issues.

(1) *What to monitor and how?* As described previously, NEWS detects flash crowds by discovering a decrease in response rate. Since overloading either at the target server or in networks could cause low response rate, NEWS adapts to server- or network-limited scenarios [Eggert and Heidemann 1999] easily. We discuss our approach to monitor response performance in Section 4.1

(2) *How to detect changes?* The change detection algorithm [Basseville and Nikiforov 1993] is well studied in many fields such as signal processing and pattern recognition. Basically, it is the scheme that determines whether a change has occurred in the characteristics of a considered object. In this work, we tend to use a simple change detection algorithm to avoid computation complexity. On the other hand, we augment our algorithm with observations in network traffic to increase detection accuracy. We present our change detection algorithm in Section 4.2

(3) *When to set and reset the alarm signal?* To address this question, we need to consider two trade-offs. When setting the alarm signal, we intend to achieve reliable detection (i.e., low false alarm rate) at a cost of a relatively long detection delay. When resetting the alarm signal, we try to avoid fluctuations in output at the risk of penalizing more incoming requests. We have noticed in our earlier study that frequent variation in request regulation affects end-user performance and leads to resource underutilization.

4.1 High-Bandwidth Connections

Flash crowd traffic usually originates from hundreds or thousands of clients. Some clients may not have visited the target server before. We call them *cold* clients. Formally, a client C is cold with respect to server S if $Tr(C, S) > Tr_0$ (Tr_0 is a constant, e.g., $Tr_0 = 24$ hours). The existence of a cold client implies that we can not detect flash crowds by monitoring perceived performance for particular clients because they may not even attempt to access the target server before the flash crowd.

Also, as we have shown in Section 3.1, Web connections during flash crowds could be short (a few packets, e.g.). Therefore, we cannot monitor the performance of particular connections because they may disappear from our traffic observation.

Further, monitoring the mean rate of all responses observed is not helpful because flows react to congestion differently with fast connections noticing congestion quickly, while low-speed flows (such as to users connected through modems) showing very little change. Thus, congestion results in very little change in average response rate because of these inherently low-speed flows.

We propose a novel flash crowd detection algorithm by monitoring changes in the response performance of *fast* connections, that is, high-bandwidth connections (HBC). These connections are most sensitive to congestion. We call the response flows of HBCs high-bandwidth response flows (HBFs). These flows form an aggregate ϕ_{HBF}. We denote the number of flows in an aggregate as $|\phi|$. For example, $|\phi| = 10\% \times |\Phi|$, where Φ is a special aggregate containing all flows. We can formally define Φ as $\Phi = \phi(*, *)$ which implies that $\forall \phi \subseteq \Phi$.

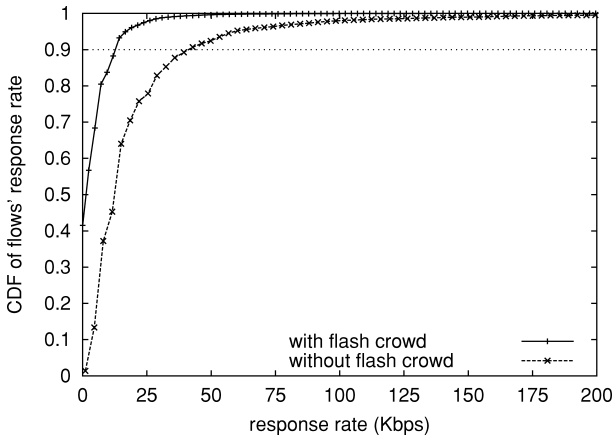


Fig. 6. CDF of flows' response rate in flash crowd and background traffic.

To quantify the performance of an aggregate ϕ, we define its *transmission rate* ($AR(\phi)$) as:

$$AR(\phi) = \frac{\sum_{f \in \phi} R(f)}{|\phi|}.$$

We investigate the sensitivity of HBF to network congestion through simulations (detailed simulation methodology is in Section 7.1). We measure the response rate ($R(f)$) of each individual flow before and during flash crowds and compare their cumulative distribution functions (CDFs) in Figure 6. If we choose 10% flows with the highest rate as HBFs, their transmission rate decreases about 78% during flash crowds.

On the other hand, if we average the response rate over all responses, the mean transmission rate only decreases by 52%. Even worse, the mean rate for the 10% slowest responses only reduced by about 40% during the flash crowd. Therefore, we confirm that HBFs are most sensitive to overloading conditions.

4.2 Change Detection Algorithm

We use a simple comparison-based scheme to detect changes in the response rate. Mathematically, the detector detects flash crowds if Condition 1 (listed in the following) holds. After it detects flash crowds, the NEWS detector watches the increase in the AR for HBFs which indicates performance recovery. The detector resets the alarm signal when Condition 3 holds.

$$AR < \overline{AR} \times (1 - \delta) \quad (1)$$

$$R_p > \overline{R_p} \times (1 + \delta) \quad (2)$$

$$AR > \overline{AR} \times (1 + \delta) \quad (3)$$

$$0 < \delta < 1$$

From the traffic control perspective, both Condition 1 and 3 choose the operation point for NEWS, that is, the long-term average of the transmission rate

for HBFs. The goal of NEWS is to regulate the interaction of the underlying system (including server, networks, and clients) so that its behavior (measured in AR for HBFs) is maintained within a reasonable range (represented by δ) of this operation point. In the above conditions, δ reflects the system's tolerance to changes. For example, with $\delta = 10\%$, the algorithm detects an increase when the current measurement is 110% larger than average.

We use Condition 2 to reduce potential false detections. Specifically, since clients may be cold and connections may be short (details in Section 3.2), there is the chance that only low-bandwidth hosts are active and responses only show a low rate. In that case, AR computed among these low-bandwidth flows will cause NEWS to trigger a false alarm.

To solve this potential problem, the flash crowd detector also checks the aggregate request rate R_p when the AR of HBFs decreases. We define *aggregate request rate* (R_p) as the rate of all requests passing through a router toward the target server. With this additional check, the detector triggers an alarm signal only when both AR decreases (Condition 1) and R_p increases (Condition 2). In this way, we are more confident that the decrease in AR is due to flash crowds rather than low-bandwidth connections.

We calculate these long-term averages ($\overline{AR}$ and $\overline{R_p}$) with a *high-low filter* (HLF). Very similar to the flip-flop filter [Kim and Noble 2001], HLF is essentially a combination of two *exponentially weighted moving average* (EWMA) filters with adjustable gains to fulfill different requirements. We present an EWMA filter mathematically as: $\overline{V(t)} = \alpha\overline{V(t-1)} + (1-\alpha)O(t)$. $O(t)$ is the current measurement (AR or R_p), $\overline{V(t)}$ is the long-term average calculated at time t ($\overline{AR}$ or $\overline{R_p}$), and α is the gain of the filter. A large α gives a stable output, while a small gain makes output sensitive to the current observation.

HLF uses a low gain ($\alpha = 0.125$) under common situations without an alarm signal for fast response to changes. When the alarm is set, we switch to a high gain ($\alpha = 0.875$) to keep output stable and avoid oscillations.

4.3 NEWS Flash Crowd Detection Algorithm

The flash crowd detector measures the transmission rate of response flows with the time-sliding window (TSW) algorithm [Clark and Fang 1998] to smooth the burntness of TCP traffic. We apply the configuration of the TSW rate estimator recommended in Fang et al. [2000]. The detector also measures the aggregate request rate R_p by counting the number of incoming requests within one time unit.

Every T seconds, the detector computes AR for HBFs. Since the number of flows observed at different times could vary dramatically, we choose the top p percent of responses with the highest rate as HBFs: $|\phi_{HBF}| = p \times |\Phi|$, where p is a tunable parameter. We choose $p = 10\%$ based on the observation in Figure 6. In our current implementation, we keep the transmission rate for all flows. Potentially, we can apply other schemes [Estan and Varghese 2002] to reduce this overhead.

The detector calculates the long-term average of the transmission rate for HBFs ($\overline{AR}$) and aggregate request rate ($\overline{R_p}$). Finally, the detector compares

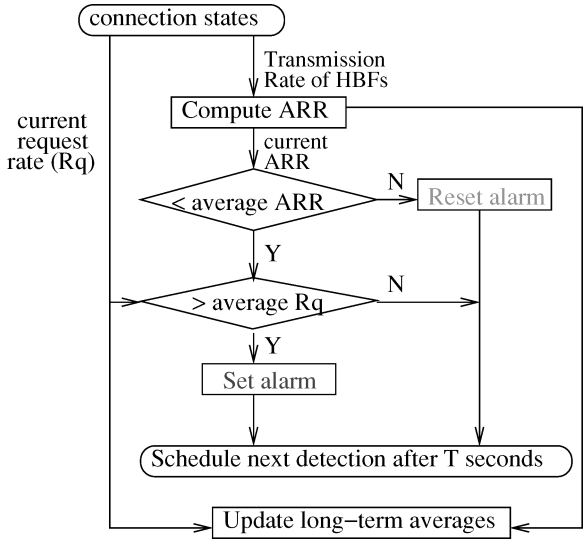


Fig. 7. Flow chart of the flash crowd detector.

AR with $\overline{AR}$ and R_p with $\overline{R_p}$. It sets or resets the alarm signal according to the conditions in Section 4.2. We show the complete procedure of flash crowd detection in Figure 7.

4.3.1 *Simple Analysis of the NEWS Detection Interval.* The detection interval T is a tunable parameter. As we will show in Section 7.4 and 8.4, NEWS with a smaller T detects traffic changes promptly but may trigger false alarms. On the other hand, a large T generates stable detection output but causes longer response time. In this section, we present a simple analysis to facilitate the choice of a proper detection interval. More specifically, we focus on the effect of the detection interval on false alarms.

According to Condition 1, NEWS detects the flash crowd when $R_c < (1 - \delta) \times R_0$, where R_c is the current measurement on the transission rate of HBFs. We also notice that the observed aggregate rate changes constantly due to underlying TCP window adjustment and network conditions. When this variation (more precisely, drop in response rate) is greater than the detector's tolerance δ, NEWS triggers a false alarm. Since there's no congestion before a flash crowd happens, we mainly focus on the effect of the TCP window adjustment which leads to dynamics in TCP flow rate, and hence the response rate observed. We are particularly interested in the slow start phase because it has a big impact on flow rate (double window size every RTT), and most Web responses are short [Guo and Matta 2001; Chen and Heidemann 2003].

Assume a response has a length of S Kbytes. If S is small, the lifetime of this response is $L = \log(S + 1) \times RTT$ seconds. Since most responses are less than 15Kbytes (according to Chen and Heidemann [2003]), we can estimate L as $4 \times RTT$. So we need to measure response rate in an interval of at least L seconds in order to correctly capture the flow rate of this response. An interval

less than L seconds may lead to underestimation of the flow rate and could cause NEWS to trigger false alarms. Therefore, we suggest choosing a NEWS detection interval of at least $4 \times RTT$ seconds.

4.4 Discovering Target Server

After NEWS detects a flash crowd, it starts to identify the *hot-spot*, that is, the target server of flash crowd traffic. In many cases, servers operate in a server farm and connect through one access router. If we deploy NEWS on that access router, it must distinguish between flash crowd traffic going to a hot-spot and low-rate, bystander traffic going to other servers. In this section, we propose a simple hash-based algorithm to discover the target server.

NEWS keeps a table of size N (hot-spot list). N is chosen based on the number of hot-spots required to identify the target server. Each entry in the hot-spot list records one server address observed in the following procedure. It maintains a counter to record the number of corresponding address being accessed (*hit-number*) within a certain time interval. Periodically, NEWS timeouts inactive entries in the list.

After NEWS detects a flash crowd, it randomly samples incoming packets and hashes their destination addresses into the hot-spot list with a function $h(x) = x \bmod N$. If the table entry has the same address or is not being used, NEWS increases the corresponding hit-number by 1. When the entry has been occupied by a different address, NEWS takes the next available entry in the hit-list. If the table is full, NEWS replaces the entry having the smallest hit-number and the longest inactive time with the new address observed. NEWS picks the address with the highest hit-number as the hot-spot.

Since we solely consider request destination addresses, HSI only classifies incoming traffic at the connection level. It cannot differentiate traffic to different ports on the same server (i.e., session-level classification). A potential future direction is to also consider destination port numbers, that is, to locate the target service at the target server.

5. MITIGATING FLASH CROWDS

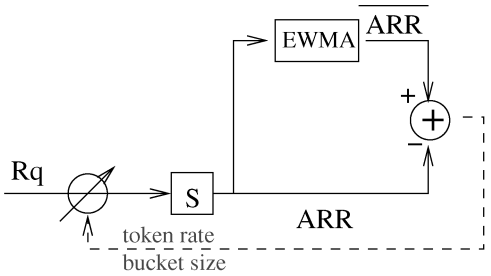
NEWS mitigates flash crowds by discarding excessive requests. In this section, we present a detailed design of request regulator and NEWS controller.

When NEWS detects a flash crowd, it differentiates requests to the target server from bystander traffic using the HSI algorithm. NEWS discards excessive requests during flash crowds with an adaptive rate limiter. As we will show in Section 7.2 and 8.2, dropped requests are retransmitted due to application or underlying TCP.

5.1 Regulating Requests

A request regulator should ensure that the admitted request rate converges to a reasonable value (denoted as R_{pc}). Ideally, when requests arrive at rate R_{pc}, they fully utilize the server and network resource. In the meanwhile, neither the target server nor the network is overloaded.

We propose a token-bucket based [Parekh and Gallagher 1993; Shenker and Wroclawski 1997] adaptive rate limiter to regulate requests. A token bucket



* S: server processes requests and generates responses

Fig. 8. NEWS control logic: scoreboard-based rate limit adjustment.

has two parameters: *bucket size* provides accommodation to bursty traffic and *token rate* limits long-term arrivals. In the long run, connections admitted by a token bucket converge to the token rate. Different from other simple token bucket algorithms, we design NEWS control logic to adaptively adjust the token rate according to the current observation on response performance and request rate. We present the detailed design of NEWS control logic in the next section.

In an our early design, we tried the approach of discarding excessive requests preferentially [Mahajan and Floyd 2001]. However, the resulting system is not stable: we observe oscillations in the admitted request rate, network load, and end-user performance. This is because probabilistic dropping only guarantees expected behavior but may not lead to smooth output [Mahajan et al. 2001]. Further, it is also difficult to determine dropping probability for requests based on measurement on response. This relationship may not be linear and is also affected by variance in response sizes. Therefore, we propose to approximate their relationship with NEWS control logic.

5.2 Controlling the Adaptive Rate Limiter

We depict the function of NEWS control logic in Figure 8. Given the alarm signal, it adjusts the rate limit of the request regulator so that it adapts to different scenarios automatically. NEWS control logic maintains two states: current and previous alarm signals. It adjusts the rate limit based on the rules described in the following.

When an alarm signal is set (transition from 0 to 1), NEWS control logic resets the rate limit to the current admitted request rate R_{p0} observed by the detector. Intuitively, requests arriving with rates higher than R_{p0} are likely to overload the server or networks and, therefore, cause a decrease in response performance. When an alarm signal changes back to 0 (transitions from 0 to 0 or from 1 to 0), NEWS control logic keeps the same rate limit.

If an alarm signal remains set (transition from 1 to 1), NEWS control logic adjusts the rate limit with a scoreboard (as shown in Figure 8). More specifically, it assigns scores to adjustments of increasing and decreasing the rate limit. It chooses the direction with the higher score. For example, if current scores for increasing and decreasing the rate limit are 5 and 3, respectively, NEWS control

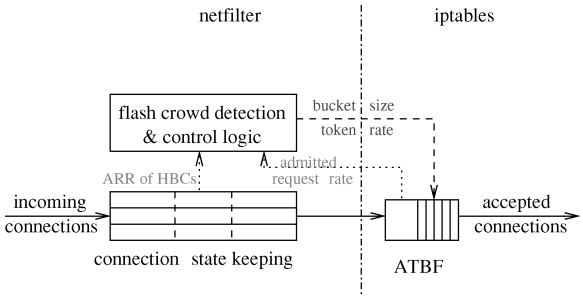


Fig. 9. NEWS implementation.

logic chooses to increase the rate limit. If an alarm resets (transition from 1 to 0) in the next period, a corresponding adjustment (that is, increasing rate limit) gets credit (now it is 6); otherwise it gets a penalty (reduced to 4). With this scheme, NEWS learns to make decisions automatically from history. It also makes NEWS adapt to different environments without human interference.

6. IMPLEMENTING NEWS

In order to investigate the applicability of NEWS to a real network environment, we prototype NEWS on a Linux-based router. We present NEWS implementation details in the following. We report our findings in testbed experiments in Section 8.

We implement NEWS under the framework of *iptables/netfilter* [Netfilter/iptables 2003]. Iptables/netfilter is the firewall subsystem for Linux kernel 2.4 and above. It provides various facilities such as stateful or stateless packet filtering, network address translation (NAT), and packet mangling. Netfilter and most functions of iptables (such as packet matching) are implemented in kernel. They process and manipulate packets according to different rules that users configure through iptables user interface.

As shown in Figure 9, we implement NEWS as three kernel modules: (1) connection state-keeping module, (2) flash-crowd detection and control logic, and (3) adaptive token bucket filter (ATBF). By dividing NEWS functions into different modules, we separate fast per-packet processing such as connection state keeping and packet matching and filtering from slow connection-based detection and control functions. We discuss the implementation of each module with details in the following.

6.1 Connection State Keeping

We build the connection state-keeping module based on the connection-tracking function in netfilter which keeps one record for each connection. Netfilter uses two specific terms to distinguish two directions of one connection, that is, *original* and *reply*. For example, a request in a Web connection is in original direction, and response is in reply direction. In our current implementation, NEWS keeps track of all connections. We measure memory consumption on the NEWS router and investigate options to reduce memory usage in Section 8.5.

We add a new state for each connection: response rate, that is, the data transmission rate (measured in bits per second) on reply direction. For Web traffic, this state records the actual data rate end-users perceive. We compute the connection response rate using the time-sliding window (TSW) algorithm [Clark and Fang 1998; Fang et al. 2000]. We intentionally avoid the first few control packets such as SYN and SYN-ACK to keep the measured response rate stable.

6.2 Flash Crowd Detection and Control Logic

Applying the algorithm described in Section 4, NEWS detects flash crowd periodically with an interval of T seconds. Based on an alarm signal, NEWS control logic regulates incoming requests adaptively (Section5).

The detection interval T is a tunable parameter. Since NEWS keeps traffic measurement for the last T seconds, NEWS with a smaller detection interval is more sensitive to traffic change and detects changes more promptly. However, the result is likely to oscillate, that is, have a high false alarm rate. Conversly, NEWS gives more stable output with a larger detection interval at the expense of a longer detection delay.

A smaller detection interval also consumes more CPU time. But, larger T needs more memory to keep connection states. We investigate the effect of different detection intervals in Sections 8.4 and 8.5. Based on our experience, we choose T as 64 seconds for our experiments.

6.3 Adaptive Token Bucket Filter

We use an adaptive token bucket filter (ATBF) to regulate incoming requests according to Section 5. We implement ATBF as an extension to iptables matching function. That is, any new connections that matche with ATBF are dropped. More specifically, each new connection adds some token into the bucket. It passes through ATBF when there are enough tokens. Otherwise, ATBF discards connections toward the target server by matching with the hot-spot list (Section 4.4). ATBF counts the number of accepted connections and reports it to the flash crowd detector as the current measurement of the request rate.

One observation in our experiments is that connections have very small interarrival times, for example, a few milliseconds. To accurately keep track of token number, we have to measure time at fine granularity, micro-second. Since we can not afford the computational overhead of floating point number division, we scale the number of tokens both in bucket and for acceptance of one connection by 1,000,000 times.

7. ALGORITHM EVALUATION THROUGH SIMULATIONS

We first evaluate the performance of NEWS algorithm through simulations using the *network simulator (ns-2.26)* [VINT 1997]. For fast algorithm design and evaluation, we prototype NEWS under the framework of the DiffServ model contributed by the Advanced IP Networks group at Nortel Networks [Pieda et al. 2000]. We further report our testbed experiments in the next section.

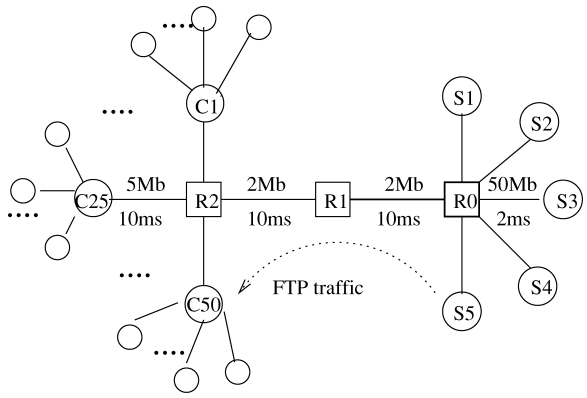


Fig. 10. The two-level dumbbell topology in simulations.

7.1 Methodology

Figure 10 shows the network topology in our simulations. Router R0 connects a server pool with 5 Web servers (S1–S5). S1 is the target server. Router R2 connects 50 access routers (C1–C50). Router R1 connects R0 and R2, representing the intermediate network connection between servers and clients.

Each access router (C1–C50) provides network connections for 20 clients. So, there are 1,000 clients in total. To reflect the variance of link properties in real networks, we determine link bandwidths and propagation delays for second-tier links (that is, between access router and end hosts) by RAMP [Lan and Heidemann 2002]. RAMP takes traffic measurement at USC/ISI and generates distributions of link properties such as bandwidth and delay. In this topology, the range of bandwidths and delays for second-tier links are 21K–10Mbps and 0.5ms–1.8 seconds, respectively. Although we cannot claim that this topology is representative, it does give us a scenario with intermediate queuing on the second-tier links and a mix of various link bandwidths and delays.

We study two types of traffic: Web traffic and bystander traffic (i.e., other traffic sharing common links with flash crowds, details in Section 7.5). We generate Web traffic (including background Web traffic and flash crowd traffic) based on real HTTP logs on the 1998 World Cup Web site.[2] There were 4 servers deployed for this Web site. Since they have shown similar patterns in request access [Arlitt and Jin 2000], we only consider those requests toward the server at Santa Clara, California. These HTTP logs recorded all requests sent between April 30, 1998, and July 26, 1998 [Labs 1992]. Each log entry keeps the following information: time when the server received a request, source and destination of a request, and size of the Web page requested. Our Web traffic model generates a Web request for one log entry.

Arlitt and Jin [2000] analyzed workload characteristics based on these HTTP logs. They found that there was a large increase (5–10 times) in request rates before each game. This increase caused a flash crowd. In our simulations, we

[2]We have also simulated flash crowd traffic with a simple two-level model in the early stage of this work.

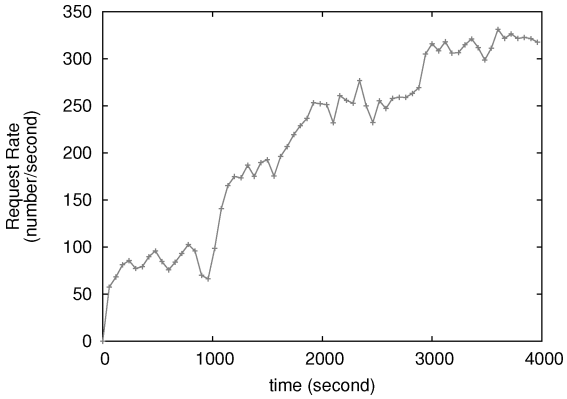


Fig. 11. Request rates to the World Cup'98 Web site (used in simulations).

consider the game between Brazil and Scotland on April 30. We show changes in the corresponding Web request rate in Figure 11. We observe normal traffic to the Web server before 1,000 seconds. Flash crowd happens at around 1,000 seconds with more than a 6 times' increase in request rates.

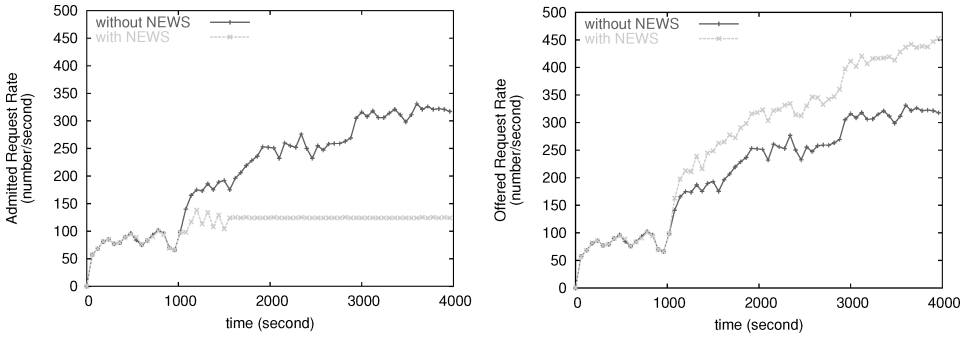
We deploy NEWS on router R0. NEWS monitors the Web response rate (from R0 to R1) and regulates incoming requests (from R1 to R0) when it detects a flash crowd. We initialize NEWS by setting the bucket size and token rate of adaptive rate limiter to large values. For example, we set the token rate as 1,000 connections per second. It is about three times larger than the maximum request rate observed in simulations (about 350 connections per second). So, NEWS accepts all incoming requests under normal conditions. In most simulations that follow, we configure the detection interval of NEWS as 60 seconds. We present our study on different detection intervals in Section 7.4.

We simulate scenarios with and without NEWS deployed. Each simulation runs for 4,000 seconds. We record offered and admitted request rates to the target server and the transmission rate of HBFs. In the following sections, we evaluate NEWS performance from both the target server and end-user perspectives. We also investigate the sensitivity of our simulation results by considering effects such as impatient end-users and server processing delays in Section 7.6.

7.2 Protecting Networks from Overloading

One goal in deploying NEWS is to protect the target server and networks from overloading. In this section, we simulate a network-limited scenario where flash crowd traffic overloads networks with a large amount of response traffic. Since NEWS detects flash crowds from observations of response traffic, we believe the following findings are also valid in server-limited scenarios. We verify this claim in Section 7.6. In order to consider the effect of server CPU and memory constraint on request processing, we further configure a server-limited scenario in our testbed experiments (Section 8.3).

We first measure the admitted request rate to the target server with and without NEWS deployed (shown in Figure 12(a)). We observe that NEWS



(a) NEWS Regulates the admitted request rate adaptively

(b) Increased offered request rate due to retransmission

Fig. 12. Request rates to the target Web server (observed in simulations).

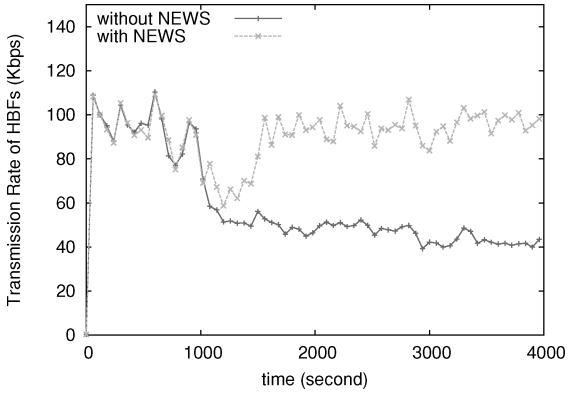


Fig. 13. NEWS maintains high transmission rates for HBFs during flash crowds (observed in simulations).

detects the flash crowd at about 1,020 seconds. That is, the detection latency is 20 seconds when the detection interval is set as 60 seconds.

After detecting the flash crowd, NEWS adjusts its rate limit automatically and regulates incoming requests to about 107 connections per second. As a result, the admitted request rate drops by 32%. This greatly reduces congestion in response traffic by reducing the packet drop rate from about 25% to 2%.

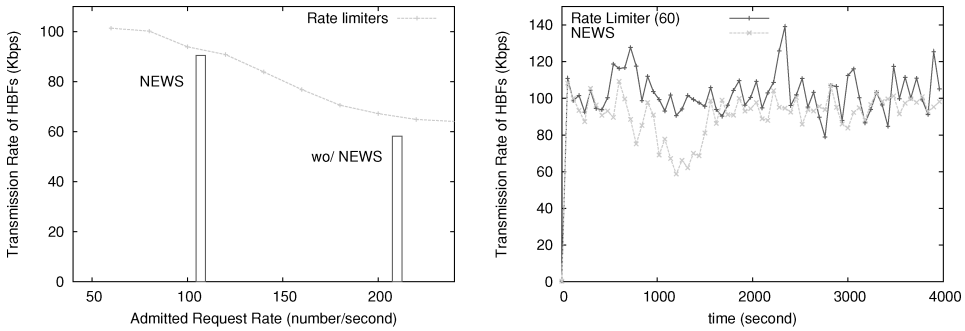
Due to the bandwidth limitation in networks, NEWS discards excessive requests. These requests may be retransmitted due to TCP, application, and client reaction behavior [Jamjoom and Shin 2003]. As a result, we observe an increase in the offered request rate as shown in Figure 12(b). Although NEWS can't serve all requests promptly, it does protect servers and networks from overloading. In the meanwhile, it gradually serves incoming requests.

7.3 Maintaining High Response Rate for Admitted Requests

Another important aspect of this evaluation is to study the effect of NEWS on end-user Web performance. As shown in Figure 13, HBFs suffer a 50%

Table I. Performance of NEWS and Different Static Rate Limiters in Simulations

Scenarios	Admitted Request Rate (number/s)	Request Rejection Percentage	Transmission Rate of HBFs (Kbps)	Response Loss Rate
Original traffic	209.7	0%	58.2	25%
NEWS	107.2	32.4%	90.5	2%
Static rate limiters	60	70.2%	101.36	0
	80	56.5%	100.2	0
	100	40.2%	93.95	1%
	120	32.8%	90.8	2%



(a) Comparison of NEWS and different rate limiters (b) Comparison of NEWS and the best rate limiter

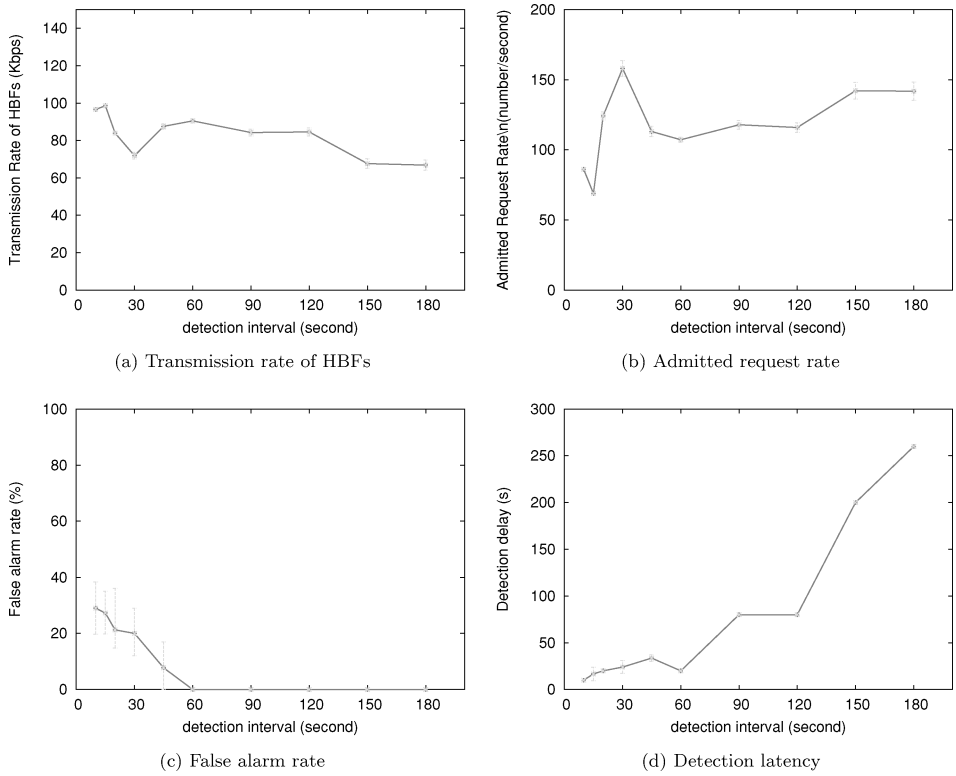
Fig. 14. Transmission rate of HBFs with NEWS and static rate limiters in simulations.

performance degradation during flash crowds. Their transmission rate drops from 96Kbps down to 58Kbps. By deploying NEWS, admitted end-users continuously receive high performance (90.5Kbps) during flash crowds. Therefore, we conclude that NEWS protects admitted requests from flash crowds.

NEWS requires sophisticated techniques to achieve this performance improvement. One could argue that a simple static rate limiter is also able to give comparable performance with careful configuration. Ideally, we want NEWS to perform similarly to the best possible rate limiter in the same scenario. We investigate this issue in our simulations.

We deploy a static rate limiter on router R0 to control incoming requests. As depicted in Table I and Figure 14(a), a static rate limiter shows different performance levels with different rate limits. In this particular scenario, we get the highest performance when we set the rate limit to 60 requests per second. We also anticipate higher end-user performance with an even stricter limit. However, these limits are so strict that more than two-thirds of incoming requests are discarded which may underutilize the underlying system.

We compare the performance of the static rate limiter with NEWS in Table I. We find that NEWS has similar performance to the best rate limiter: NEWS shows only about 11% less than the best static rate limiter in the transmission rate of HBFs. This result is encouraging because it verifies that the adaptive rate limiter in NEWS approaches the best request rate but without manual



(a) Transmission rate of HBFs

(b) Admitted request rate

(c) False alarm rate

(d) Detection latency

Fig. 15. The performance of NEWS under different detection intervals (observed in simulations).

adjustment. We believe that the overall performance difference between these two schemes reflects the cost for NEWS to adaptively discover the right rate limit after a flash crowd is detected (from 1,000 to 1,500 seconds, as shown in Figure 14(b)). On the other hand, NEWS only discards half of its incoming requests compared to the best rate limiter. We highlight the transmission rate of HBFs with NEWS and static rate limiters in Figure 14(a). Given similar performance, we believe it is an advantage to deploy NEWS because it alleviates the workload for manual operations by detecting and mitigating flash crowds adaptively.

7.4 Effect of Detection Interval

NEWS periodically checks changes in the transmission rate of HBFs with an interval of T. As we explain in Section 4, this parameter reflects the trade-off between fast detection and a low false alarm rate. In this section, we investigate NEWS performance with different detection intervals (see Figure 15).

From simulation results, we find that NEWS shows the best performance in terms of the admitted request rate and the transmission rate of HBFs when we set the detection interval as 60 seconds. We also notice that small time intervals (like 10, 15, and 20 seconds) give inconsistent performance for HBFs. This result indicates that it is hard to configure high-level controls like NEWS

at small timescales. Recall from our analysis in Section 4.3.1 that is due to the underestimation of TCP flow rate.

It is not unexpected that NEWS detects flash crowds quickly under very small intervals such as 15 seconds and 30 seconds. In fact, with these small time intervals, NEWS triggers a false alarm before flash crowds really happen. As a result, the adaptive rate limiter starts to regulate incoming requests. On the other hand, NEWS shows no false alarm with detection intervals larger than 90 seconds. But, the detection delay is also large. Based on these observations, we find that the detection interval of 60 seconds is feasible in the scenario we studied.

One possible way to reduce the NEWS false alarm rate is to separate the detection interval from the time period in which NEWS measures the performance for HBFs. We call this time period measurement window size. For example, NEWS measures the transmission rate for HBFs observed in the past 90 seconds but detects performance degradation with small intervals such as 30 seconds. We need to investigate its applicability to flash crowd detection in our future work.

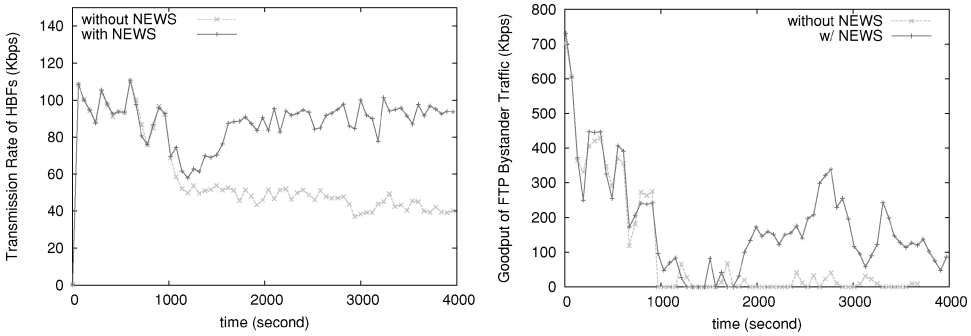
7.5 Bystander Traffic Protection

In real networks, there always exist other traffic (that is, bystanders) which we have not considered in our simulations so far, for example, Web or FTP traffic to other servers connected through NEWS router. We investigate the effect of flash crowds and NEWS on bystander traffic in this section. This study has two goals. First, we verify NEWS performance with the existence of bystander traffic. More specifically, we investigate both flash crowd detection and response performance for admitted requests. Second, we are also interested in NEWS protection of bystander traffic. Since NEWS is an adaptive system, we expect NEWS improves the performance for both admitted requests and bystanders. We now examine these two aspects.

We study NEWS protection for both long-lived (such as FTP) and interactive (for example, Web) bystander traffic. We report our findings in the first case that follows. In Section 8.6, we further investigate both cases with testbed experiments.

In our simulation, node S5 sends FTP traffic (long-lived) to C50 (refer to Figure 10). As shown in Figure 16(a), NEWS maintains high performance for admitted requests after detecting the flash crowd. The transmission rate for HBFs is 90.2Kbps before the flash crowd and reaches 86.4Kbps with NEWS deployed. Therefore, with the existence of bystander traffic, NEWS still shows consistent performance as in our previous study.

We further show the goodput of FTP bystander traffic without and with NEWS deployed in Figure 16(b). We find that the mean goodput is about 337Kbps before the flash crowd, then it drops to only about 10Kbps when flash crowd happens. With NEWS protection, the goodput of FTP traffic jumps back to about 120Kbps. So, NEWS improves goodput of FTP bystander traffic by more than 10 times. This improvement is because NEWS relieves congestion in networks by discarding excessive requests. The traffic regulation benefits both



(a) Maintaining high performance for admitted requests

(b) Improving the performance of FTP bystander traffic

Fig. 16. NEWS bystander traffic protection (in simulations).

Web traffic and bystanders. So, we conclude that NEWS is also able to protect bystander traffic from flash crowds.

7.6 Result Sensitivity to Other Simulation Parameters

The simulation results presented are from a much simpler scenario than reality. To investigate the sensitivity of our previous results, we relax two aspects of our simulation scenario by considering impatient end-users and server processing delay. While we can not claim to simulate the Internet [Floyd and Paxson 2001], this sensitivity study, together with our investigation on the effect of bystander traffic in Section 7.5, helps us to better understand the dynamics of NEWS. We also gain the confidence to deploy NEWS in real networks (we will report our findings in testbed experiments in the next section).

7.6.1 *Impatient End-Users.* In real the world, end-users may terminate their Web requests after a long waiting time. We investigate this effect with a simple model. Our model cancels Web requests that are not served after a certain time period. It also assumes that an end-user's waiting time has an exponential distribution, and end-users do not resume their requests after cancellation.

In our simulation, we choose the average waiting time as 60 seconds. After timeout, end-users may decide if they want to continue to wait or cancel their request with equal probability. We repeat simulations with 60-second detection intervals. Our results show that NEWS still archives similar performance with the existence of impatient end-users. We believe that this result is reasonable because excessive requests (and their responses) are delayed and even dropped during flash crowds. Impatient end-users are unlikely to have a large impact on this situation. On the other hand, Jamjoom and Shin [2003] have shown that retransmission due to persistent end-users (that is, very patient users) have a large effect on flash crowd traffic. This study also demonstrates that the existence of impatient users does not have much effect on the performance of NEWS.

7.6.2 *Server Processing Delays.* We focus on the network perspective in our simulation so far. In reality, Web servers impose processing delays to Web requests. To investigate this effect, we add a simple Web server model to our simulation. We assume that a Web server has a constant processing rate (for example, 1000KB/s in our study) and a buffer to hold incoming requests the server applies first come, first serve (FCFS) scheduling policy and always processes the first request in its buffer. The server drops incoming requests when its buffer is full.

We first investigate a scenario where the target server has infinite buffer space. Simulation results show that NEWS gives similar performance (admitted request rate and transmission rate of HBFs) as in our previous study. This result is not unexpected because our Web model only adds delays between requests and corresponding responses. It does not relieve the overloading condition in networks. The response traffic still congests network connections.

To get more realistic results, we configure the server with a limited buffer space of 1,000 requests. We find that incoming requests quickly fill the server buffer, and more than 50% of requests are dropped. This effect is very similar to the simple static request rate limiter described in Section 7.3. Therefore, neither server CPU nor network links are overloaded.

Based on these results, we conclude that our Web server model in simulation does not have a fundamental effect on the performance of NEWS. This is because our model focuses on adding server queuing and processing delays to incoming requests, limiting the server overloading condition where the CPU or memory is overwhelmed by incoming requests. To complete our study on the server overloading condition, we configure a server-limited scenario in our testbed experiments in Section 8.3.

8. SYSTEM EVALUATION THROUGH TESTBED EXPERIMENTS

In Section 7, we have evaluated the performance of NEWS in simulations and shown that it protects the target server and networks from overloading and maintains high performance for admitted requests. In this section, we evaluate NEWS implementation with testbed experiments. We quantitatively validate our previous simulation studies in a network-limited scenario. More importantly, testbed experiments allow us to investigate server performance and router overhead (such as CPU and memory usage) which are not modeled in network simulations.

We further evaluate the performance of NEWS in a server-limited scenario, confirming that NEWS is an adaptive system, capable of efficiently preventing overloading on the server or in networks. By examining its runtime overhead on the router, we also show that NEWS is a relatively light-weighted scheme. Finally, we evaluate the effectiveness of the hot-spot identification algorithm through experiments with bystander traffic.

8.1 Experiment Setup

Figure 17 shows our testbed environment. All machines have 100Mbps fast Ethernet network interfaces. The client pool has 7 machines with various

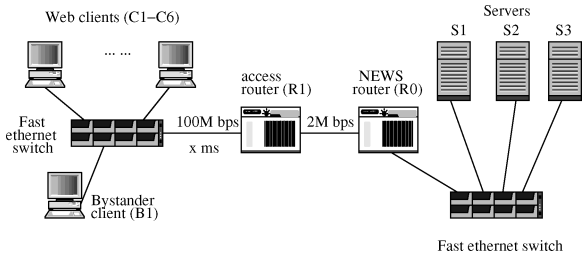


Fig. 17. Network topology in the testbed experiment.

configurations ranging from 677MHz Pentium III CPU with 128MB RAM to 2GHz Pentium 4 CPU with 512MB RAM. We use 6 machines (C1–C6) for flash crowd traffic generation and one (B1) for bystander traffic experiment.

Our server pool has 3 machines. Fast target server (S1) and bystander server (S3) have 2GHz Pentium 4 CPU and 512MB RAM. Slow target server (S2) has 180MHz Pentium Pro CPU and 128MB RAM. We connect client and server pools together through two routers R0 and R1. We use Linux Redhat 9 (kernel 2.4.20-8) on all machines, including two routers.

The NEWS router (R0) is a PC with 1.5GHz AMD Athlon 4 processor and 1GB RAM. We deploy NEWS on R0 by enabling iptables service with NEWS extension. In most experiments, we configure the NEWS detection interval as 64 seconds (see Section 7.4). We also investigate the effect of different detection intervals in Sections 8.4 and 8.5 .

We use NIST Net [NIST 1998] to introduce network dynamics into our experiments. NIST Net allows a Linux-based router to emulate a wide variety of network conditions such as link bandwidth, propagation delay, and packet loss. We configure link bandwidth between R0 and R1 as 2Mbps. We also set different propagation delays between R1 and client machines according to network measurement at USC/ISI [Lan and Heidemann 2002]. While we cannot claim that our testbed simulates the Internet [Floyd and Paxson 2001], this experimental study does help us to better understand the dynamics of NEWS in a relatively more real environment than simulations.

Our Web servers use TUX [Lever et al. 2000] to expedite request progressing. Running partially from within Linux kernel, TUX has demonstrated outstanding performance for serving static contents. We configure Web servers with large SYN buffer size and HTTP backlog to ensure that they are fed with enough workload.

We implement a traffic generator to playback the Web server log (discussed in Section 7.1) in our experiments. It sends Web requests to the server according to information in log entries such as time, server address, and request size. To simulate many concurrent connections, our traffic generation tool creates threads to accomplish request-response exchanges independently. Due to a memory constraint on one single client machine, we distribute traffic generation across six client machines (C1–C6).

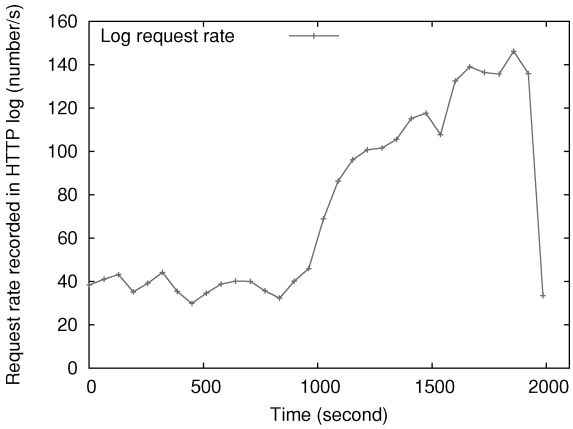
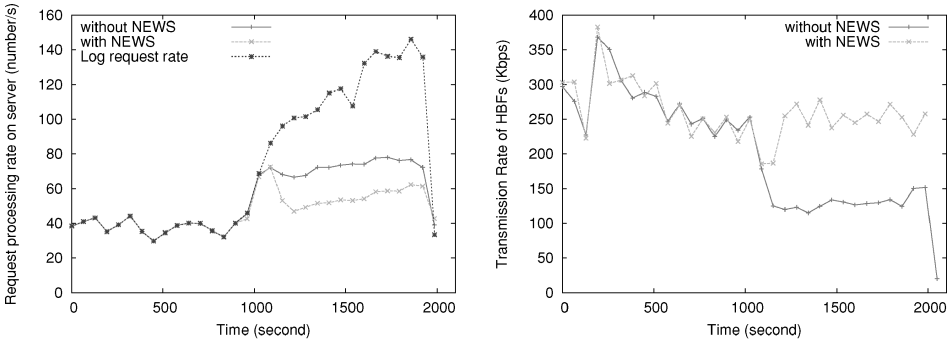


Fig. 18. Request rate to the World Cup'98 Web site (used in experiments).



(a) Request processing rate on target server

(b) Transmission rate of HBFs

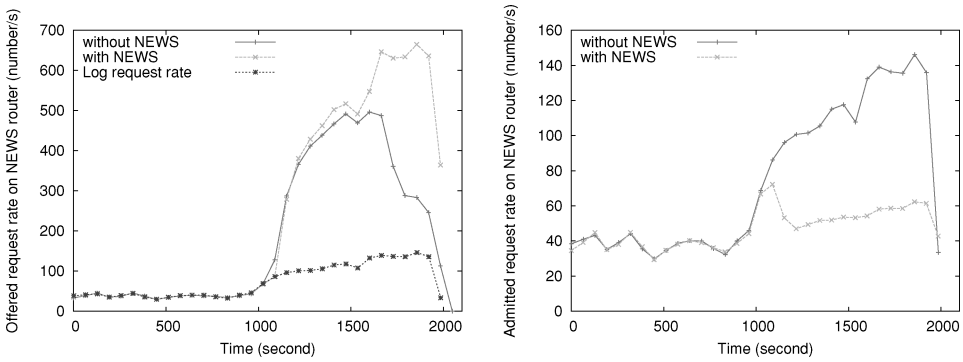
Fig. 19. Server and end-user performance during flash crowds (observed in experiments).

In our experiments, we use HTTP logs as described in Section 7.1 but choose a different game. More specifically, we playback 2,000 seconds of log from one semifinal between Brazil and Holland on July 7. As shown in Figure 18, the request rate increased by about 5 times in just several minutes after the game started at 1,000 seconds.

We monitor the Web request rate both at the NEWS router (R0) and the target Web server. To quantify end-user perceived performance, we record the transmission rate of HBFs on router R0. We also keep track of system and network statistics on each machine such as CPU and memory usage, network utilization, and packet drop rate. We are particularly interested in changes in these statistics when flash crowds happen.

8.2 Relieving Network Congestion

We have two different experiment configurations, namely network- and server-limited scenarios. We have studied NEWS performance in a network-limited scenario in simulations. In this section, we validate our simulation results quantitatively, confirming that NEWS effectively protects networks from



(a) Offered request rate (b) Admitted request rate

Fig. 20. Request rate observed on NEWS router in experiments.

overloading and maintains high performance for accepted requests. We further investigate system performance of the target server in detail.

In this experiment, we choose fast server S1 as the target. We observe that requests do not overload S1 during the flash crowd; in fact, their processing only consumes about 2–3% of CPU time and about 60% of memory on S1. However, the response traffic generated by S1 congests the network link between router R0 and R1. Our measurement shows 100% link utilization and about 60% packet drop rate.

As a result, S1 is kept busy with transmitting and retransmitting response traffic and is not able to catch up with all incoming requests. As shown in Figure 19(a), S1 can only process about 80 requests per second, 32% less than the average incoming request rate during flash crowds (117 requests per second). Congestion in the response network link and slow server processing (hence, large server backlog) greatly reduce end-user perceived performance: the transmission rate of HBFs drops by half as shown in Figure 19(b). Also, about 22% of connections timeout and fail due to packet loss. Failed connections resend SYN packets after timeout and thus increase the average offered request rate on the NEWS router from 110 to 318 requests per second (Figure 20(a)).

We enable NEWS on R0. With a 64-second detection interval, NEWS detects flash crowd at about 1,088 seconds, that is, 88 seconds after flash crowd starts. NEWS protects the target server and networks from overloading by regulating the admitted request rate down to 56 requests per second (Figure 20(b)). As less response traffic is generated, packet loss rate on link between R0 and R1 falls to less than 3%.

NEWS also maintains high performance for admitted requests during flash crowds, increasing the transmission rate of HBFs from 133Kbps to 261Kbps (Figure 19(b)). As shown in Figure 21, this performance is comparable to static token-bucket based request rate limiter with manually configured token rate. Therefore, our experiment confirms that NEWS protects the server and networks from flash crowds effectively, and maintains high performance for admitted requests.

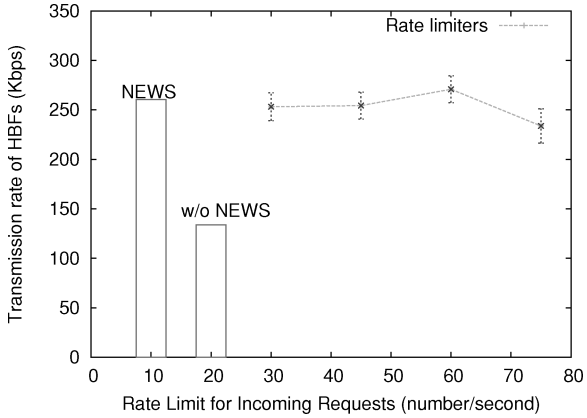


Fig. 21. Comparison of NEWS and static rate limiter in experiments.

As a cost for this performance, NEWS discards about 49% of excessive requests. As observed in Figure 20(a), the offered request rate to R0 increases by about 44% (from 318 to 459 requests per second) due to retransmissions as discussed in Section 7.2.

8.3 Protecting the Server From Flash Crowds

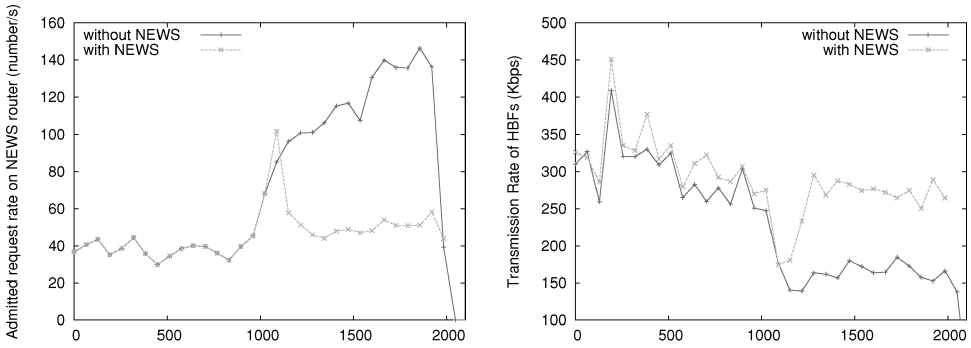
We further evaluate NEWS performance in a server-limited scenario. This evaluation was not possible in simulation because network-level simulators (like *ns*) do not include detailed models of server CPU, memory, and disk performance.

In this experiment, clients send Web requests to slow target server S2. Since Web requests in our experiments are only for static content, CPU usage on S2 is about 15%. At the same time, link utilization between R0 and R1 is only 60%. We do not observe any packet loss. However, we find that S2 uses up about 98% of its memory and starts swapping in order to serve the huge number of incoming requests.

Swapping slows down S2 and builds up server backlog. Eventually, about 20% of connections timeout due to long waiting time. As a result, end-users suffer from a low Web response rate even though there is no packet drop in the networks.

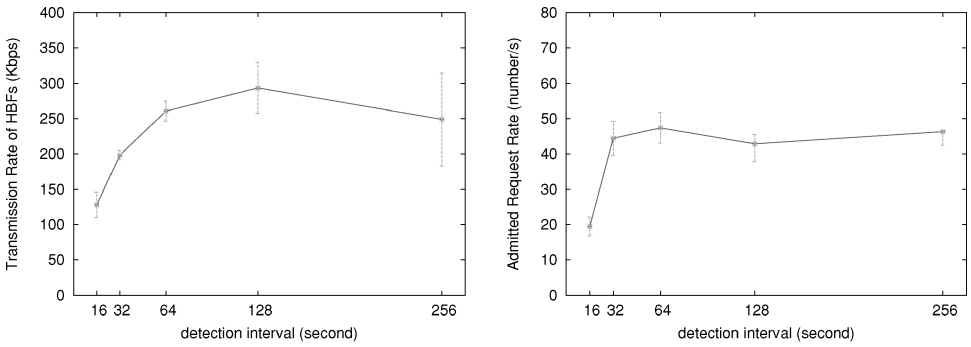
Since NEWS detects flash crowds by monitoring response performance, it adapts to the server-limited scenario automatically, detecting flash crowds withing about 80 seconds. By regulating incoming requests to 54 per second (as shown in Figure 22(a)), NEWS reduces server memory consumption to about 88%.

As the server stops swapping, it processes requests more promptly. Therefore, admitted requests receive much higher performance. As depicted in Figure 22(b), transmission rate of HBFs jumps from 158 Kbps to 260 Kbps. Based on these results, we conclude that NEWS is an adaptive scheme and can detect and prevent both network- and server-overloading during flash crowds.



(a) Admitted request rate (b) Transmission rate of HBFs

Fig. 22. NEWS performance in a server-limited scenario in experiments.



(a) Transmission rate of HBFs (b) Admitted request rate

Fig. 23. Different detection intervals affect NEWS performance (observed in experiments).

8.4 Effect of Different Detection Intervals

As we explained in Section 4, NEWS detects flash crowds by periodically (with an interval of T seconds) checking degradation in response performance. The detection interval T is an important parameter affecting the speed and accuracy of flash crowd detection and therefore the overall NEWS performance. With testbed experiments, we first validate our findings in the prior simulation study (Section 7.4). We further study the effect of different detection intervals on NEWS runtime overhead in the next section.

Figure 23 shows the transmission rate of HBFs and the admitted request rate under different Ts. NEWS shows its best performance with the detection interval at 64 or 128 seconds.

We find that the performance of NEWS falls with either large or very small detection intervals. For example, with a detection interval at 256 seconds, NEWS can't adjust the rate limit promptly as traffic changes. On the other hand, with very small intervals like 16 seconds, NEWS is so sensitive to traffic change that it triggers an alarm before flash crowd happens (at 760 seconds). Then, it sets the rate limit too low and about 65% of connections are discarded. These results confirm our findings in simulations.

With detection intervals larger than 16 seconds, NEWS always detects flash crowds slightly after one detection interval. Balancing detection delay and performance, we suggest configuring NEWS with a detection interval of 64 seconds.

In our experiments, NEWS does not trigger false alarms with detection intervals larger than 16 seconds. We believe this is because requests generated from one client machine are highly correlated. We plan to deploy NEWS to real networks to further study this issue.

8.5 NEWS Overhead on Routers

In this section, we first present an analysis of the algorithmic performance of the NEWS flash crowd detection algorithm. Then, we conduct a testbed experiment to further evaluate NEWS CPU and memory consumption on a Linux-based router.

8.5.1 *Algorithmic Performance of NEWS Flash Crowd Detection.* NEWS flash crowd detection consumes both CPU time and memory on routers. We analyze this overhead in the following. We further study NEWS runtime overhead with a testbed experiment in Section 8.5.

We first examine the overhead of NEWS flow state keeping. NEWS maintains stats for all active flows and updates flow response rates for every response packet. As shown in Section 6, NEWS uses the hash-based connection tracking function in Iptables/Netfilter [Netfilter/iptables 2003]. Therefore, the average overhead of updating flow rate is $O(1)$ ($O(N)$ in the worst case, N is the table size). On the other hand, NEWS requires the allocation of memory for its hash table. We investigate this consumption in our testbed experiment (Section 8.5).

Periodically (with an interval of T seconds), NEWS conducts a flash crowd detection procedure as described in Section 4.2. NEWS selects the 10% fastest flows and computes their transmission rate. This task can be accomplished efficiently with a randomized selection algorithm [Cormen et al. 2001] which has the average performance of $O(\log N)$ ($O(N)$ in the worst case, N is the number of flows kept). In our current implementation, NEWS first sorts flows based on their rates with quick-sort. Then, it picks the first 10% of entries in the sorted list. This imposes computational overhead of $O(N \log N)$.

Based on our analysis, NEWS imposes similar overhead to routers as a firewall keeping flow state. Therefore, we believe that NEWS is applicable to real networks to protect server and networks from flash crowds.

8.5.2 *NEWS CPU and Memory Overhead.* With testbed experiments, we are able to quantitatively investigate NEWS runtime overhead on routers, including CPU time and memory consumption. We report our findings here.

Table II shows the CPU usage of NEWS under different detection intervals and the offered request rate after flash crowds. The offered request rate before the flash crowd is about 40 requests per second. When flash crowd happens, the offered request rate varies from 530 to 259 requests per second with different NEWS detection intervals. In our experiments, we find that smaller detection intervals consume relatively larger CPU time because of more frequent flash

Table II. CPU Time Consumed by NEWS With Different Detection Intervals (Observed in Experiments)

Detection Intervals	Before Flash Crowds	After Flash Crowds	Offered Request Rate After Flash Crowds (request / s)
16 s	2–3.5%	5–6%	530
32 s	1.5–2%	3.5–5%	511
64 s	1–2%	2–4%	459
128 s and larger	about 1%	about 2%	less than 400

crowd detections. Overall, NEWS consumes less than 5% CPU time with a reasonable detection interval (for example, 64 seconds with an offered request rate of about 459 per second).

We also measure CPU usage when running NEWS on a slow machine (266MHz Pentium II CPU with 128MB RAM). NEWS consumes about 14% of CPU time on average after flash crowds happen. Therefore, we conclude that NEWS imposes relatively small computational overhead to routers. We further study NEWS memory consumption as follows.

NEWS needs memory to keep track of connection response rates. As shown in Section 6, we design this module based on connection-tracking support in netfilter which uses a hash table to keep states for each connection. So, NEWS memory consumption is determined by the number of new connections (initiated by requests). Through our experiments, we observe that memory consumption remains about 3M bytes before the flash crowd happens. In this stage, the incoming request rate is about 40 per second.

When a flash crowd happens, NEWS needs more memory as the request rate increases. Depending on the rate-limit that NEWS discovers, NEWS memory consumption varies from 10M to 12M bytes, corresponding to an incoming request rate of 60 to 100 requests per second. In general, NEWS memory consumption is not sensitive to detection intervals.

It is possible to reduce memory consumption for connection tracking by adjusting the value of connection timeout, that is, the measurement window size. The default configuration of netfilter is 24 hours. We are able to reduce memory consumption after flash crowds to about 6M bytes by timing out connections every 64 seconds. We can further reduce NEWS memory consumption by random sampling of incoming connections [Estan and Varghese 2002].

In real networks, routers usually have less memory than the PCs used in our experiments. To have a reference on the overhead of flow state keeping on commercial routers, we substitute R0 with a Cisco 3620 router which is widely used for mid-size networks. The router we use in this experiment has 100MHz R4700 CPU, 12MB processor, memory, and 8MB IO memory. We use NetFlow [NetFlow 2003] to keep states of individual unidirectional IP flows. With a default flow cache size of 256K bytes and flow timeout value of 15 seconds, our Cisco router is able to track about 4,000 active flows (or 2,000 bidirectional connections), each in a 70 byte record.

By repeating the previous experiments, we find that the average router CPU time is about 2–3% before flash crowd, and 7–8% after. The maximum CPU time reaches 10% during flash crowds. We do not observe any memory allocation

Table III. End-to-End Latency of Bystander Web Traffic in Different Experimental Scenarios

	Web Latency (seconds)	
Scenarios	Mean	Median
before flash crowd	0.22	0.18
during flash crowd	24.44	9.53
with NEWS	7.03	3.15
NEWS + HSI	0.42	0.32

failures for incoming flows. In another word, the default 256K byte flow cache is enough to keep states for all flows during the flash crowd in our experiments. Therefore, with the traffic condition in our experiments, per-flow state keeping only imposes small overhead even to real routers.

8.6 Bystander Traffic Protection

In our simulation study (Section 7.5), we have shown that NEWS protects bystander FTP bulk traffic from flash crowds. We partially validate this result with experiments. More specifically, we configure server S3 to constantly send 8K byte data chunks to client machine B1. Since NEWS measures aggregate request and response rates, it shows consistent performance in detecting and mitigating flash crowds.

We measure goodput received by B1 and find it drops dramatically from 327Kbps to 80Kbps as flash crowd traffic builds up. With NEWS deployed, B1 gets a consistent goodput of 318Kbps. Therefore, NEWS can protect the bystander bulk data transfer from flash crowds.

We further evaluate the hot-spot identification (HSI) algorithm through an experiment with bystander Web traffic. In this case, B1 sends Web requests to server S3 based on HTTP log under-light load (average request rates-less than 20 requests per second). Since most Web connections are short, we are interested in the end-to-end latency.

We summarize the experiment results in Table III. We observe that the flash crowd hurts bystander Web traffic greatly, increasing the median Web latency by more than 50 times. By discarding about 22% of connections, NEWS (with original algorithms) reduces the median Web latency for admitted bystander requests by more than 3 times. With the hot-spot identification technique, NEWS does not drop any bystander connections. As a result, NEWS further reduces the median Web latency for bystander Web traffic by about 10 times. Therefore, we conclude that the hot-spot identification algorithm is an effective technique to classify and protect bystander traffic from flash crowds.

9. FUTURE WORK

We have shown the effectiveness of NEWS in both network- and server-memory limited scenarios (Sections 7 and 8). As a potential direction for our future work, we can further evaluate the performance of NEWS in a scenario where server

CPU is overloaded with dynamic workload, for example, clients send database queries via CGI interface on the target server. This experiment will further test the adaptivity of the NEWS algorithm.

In Section 7.4 and 8.4, we demonstrated the impact of the detection interval on the performance of NEWS through simulations and testbed experiments. Modeling the overload protection of NEWS could be an interesting future direction for our research. With this analytic effort, we should be able to derive bounds to guide our choice of appropriate detection intervals for NEWS.

Currently, NEWS maintains states for each individual connection. A potential improvement of NEWS is to reduce this overhead and only keep track of high-bandwidth connections. One possible approach is to adopt schemes proposed by Estan and Varghese [2002].

DDoS attack [Savage et al. 2000; Shoeten et al. 2001; Park and Lee 2001] is another class of problem threatening network robustness. We can potentially extend the flash crowd detection techniques to detect DDoS attacks on a Web server. More specifically, we can design a system to monitor end-user perceived performance and infer ongoing DDoS attacks from performance degradation.

10. CONCLUSION

In this work, we propose NEWS to protect servers and networks from persistent overloading caused by flash crowds. NEWS detects flash crowds from performance degradation of Web responses and imposes aggregate-level control between requests and responses. We evaluate the performance of NEWS through both simulations and testbed experiments. Our results show that NEWS quickly detects flash crowds. By discarding excessive requests, NEWS protects both the target server and networks from overloading. NEWS also maintains high performance for end-users and protects bystander traffic from flash crowds. We also studied the effect of different detection intervals on the performance of NEWS.

Through this study, we demonstrate that an aggregate-level traffic control algorithm (like NEWS) can effectively protect infrastructure from persistent overloading during flash crowds. We also illustrate the importance of building an adaptive system. More specifically, NEWS infers persistent overloading from end-user perceived performance. It detects flash crowds effectively by measuring changes in the aggregated response rate of HBFs. As we have shown, this choice of traffic observation metrics makes the NEWS detection algorithm adaptive to both network- and server-limited scenarios. NEWS also discovers the right rate limit automatically.

We further studied the effect of detection intervals on the performance of NEWS, showing the trade-off between fast detection and a low false alarm rate. Since flash crowds are largely caused by human behavior, we focused on the accuracy of flash crowd detection at a cost of longer delays and used large detection intervals (for example, 1 minute). Further, our guideline also suggests using relatively long intervals (4–6 times of RTT) for TCP rate measurement.

ACKNOWLEDGMENTS

We would like to acknowledge Stephen Adler for providing the server trace of the slash-dot effect. We appreciate the fruitful discussion with members in the *SAMAN* and *CONSER* projects. We are graceful to Professor Edmond A. Jonckheere for his suggestions on presenting NEWS control diagrams and Professor Deborah Estrin, Dr. Ted Faber, and Dr. Sally Floyd for their feedback on the early work of NEWS.

REFERENCES

ADLER, S. 1999. The slashdot effect, an analysis of three Internet publications. Available at `http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html`.

ARLITT, M. AND JIN, T. 2000. A workload characterization study of the 1998 World Cup Web site. *IEEE Network, Special Issue on Web Performance 14*, 3 (May), 30–37.

BALAKRISHNAN, H., RAHUL, H., AND SESHAN, S. 1999. An integrated congestion management architecture for internet hosts. In *Proceedings of the ACM SIGCOMM*. Cambridge, MA, 175–187.

BARFORD, P. AND PLONKA, D. 2001. Characteristics of network traffic flow anomalies. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*. San Francisco, CA, 2–13.

BASSEVILLE, M. AND NIKIFOROV, I. V. 1993. *Detection of Abrupt Changes—Theory and Application*, 1st Ed. Prentice-Hall, Englewood Cliffs, NJ.

BERNERS-LEE, T., FIELDING, R., AND FRYSTYK, H. 1996. Hypertext transfer protocol—HTTP/1.0. RFC 1945. IETF. Available at `ftp://ftp.isi.edu/in-notes/rfc1945.txt`.

BLAZEK, R. B., KIM, H., ROZOVSKII, B., AND TARTAKOVSKY, A. 2001. A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods. In *Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance Workshop*. West Point, NY.

BRESLAU, L., KNIGHTLY, E. W., SHENKER, S., STOICA, I., AND ZHANG, H. 2000. Endpoint admission control: Architectural issues and performance. In *Proceedings of the ACM SIGCOMM*. Stockholm, Sweden, 57–69.

CETINKAYA, C. AND KNIGHTLY, E. 2000. Egress admission control. In *Proceedings of the IEEE Infocom*. Tel-Aviv, Israel, 1471–1480.

CHEN, H. AND MOHAPATRA, P. 2002. Session-based overload control in qos-aware Web servers. In *Proceedings of the IEEE Infocom*. New York, NY.

CHEN, X. AND HEIDEMANN, J. 2003. Preferential treatment for short flows to reduce Web latency. *Comput. Netw. 41*, 6 (April), 779–794.

CLARK, D. AND FANG, W. 1998. Explicit allocation of best effort packet delivery service. *ACM/IEEE Trans. Netw. 6*, 4 (Aug.), 362–373.

CLARK, D. D., SHENKER, S., AND ZHANG, L. 1992. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of the ACM SIGCOMM*. Baltimore, MD. 14–26. Available at `citeseer.nj.nec.com/clark92supporting.html`.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms*. 2nd Ed. The MIT Press, Cambridge, MA.

DEMERS, A., KESHAV, S., AND SHENKER, S. 1989. Analysis and simulation of a fair-queueing algorithm. In *Proceedings of the ACM SIGCOMM*. Austin, TX. 1–12.

EGGERT, L. AND HEIDEMANN, J. 1999. Application-level differentiated services for Web servers. *World-Wide Web J. 2*, 3 (Aug.), 133–142.

ESTAN, C. AND VARGHESE, G. 2002. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM*. Pittsburgh, PA. 323–336.

FANG, W., SEDDIGH, N., AND NANDY, B. 2000. A time-sliding window three colour marker (TSWTCM). RFC 2859. IETF.

FLOYD, S. AND FALL, K. 1999. Promoting the use of end-to-end congestion control in the Internet. *ACM/IEEE Trans. Netw. 7*, 4 (Aug.), 458–473.

FLOYD, S. AND JACOBSON, V. 1993. Random early detection gateways for congestion avoidance. *ACM/IEEE Trans. Netw. 1*, 4 (Aug.), 397–413.

FLOYD, S. AND PAXSON, V. 2001. Difficulties in simulating the Internet. *ACM/IEEE Trans. Netw. 9*, 4 (Aug.), 392–403.

GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. 1999. Hypertext transfer protocol—HTTP/1.1. RFC 2616. IETF. Available at `http://www.w3.org/Protocols/rfc2616/rfc2616.html`.

GIBBENS, R., KELLY, F. P., AND KEY, P. 1995. A decision-theoretic approach to call admission control in ATM networks. *IEEE J. Select. Areas Comm. 13*, 6 (Aug.), 30–37.

GUO, L. AND MATTA, I. 2001. The war between mice and elephants. In *Proceedings of the IEEE International Conference on Network Protocols*. Riverside, CA.

JACOBSON, V. 1988. Congestion avoidance and control. In *Proceedings of the ACM SIGCOMM*. Stanford, CA. 314–329.

JAMIN, S., DANZIG, P., SHENKER, S., AND ZHANG, L. 1995. Measurement-based admission control algorithms for controlled-load service. In *Proceedings of the ACM SIGCOMM*. Cambridge, MA. 2–13.

JAMJOOM, H. AND SHIN, K. G. 2003. Persistent dropping: An efficient control of traffic aggregates. In *Proceedings of the ACM SIGCOMM*. Karlsruhe, Germany, 287–298.

JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. 2002. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *Proceedings of the International World Wide Web Conference*. Hawaii, 252–262.

KIM, M. AND NOBLE, B. 2001. Mobile network estimation. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*. Rome, Italy, 298–309.

LABS, L.-B. 1992. The Internet Traffic Archive. Available at `http://ita.ee.lbl.gov/`.

LAN, K. AND HEIDEMANN, J. 2002. Rapid model parameterization from traffic measurement. *ACM Trans. Model. Comput. Simul. 12*, 3 (July), 201–229.

LEVER, C., ERIKSEN, M. A., AND MOLLOY, S. P. 2000. An analysis of the TUX Web server. Tech. rep., (Nov.) Center for Information Technology Integration, University of Michigan.

LIN, D. AND MORRIS, R. 1997. Dynamics of random early detection. In *Proceedings of the ACM SIGCOMM*. Cannes, France, 127–137.

LU, C., ABDELZAHER, T. F., STANKOVIC, J. A., AND SON, S. H. 2001. A feedback control approach for guaranteeing relative delays in Web servers. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*. Taipei, Taiwan.

MAHAJAN, R., BELLOVIN, S., FLOYD, S., IOANNIDIS, J., PAXSON, V., AND SHENKER, S. 2001. Controlling high bandwidth aggregates in the network. Tech. rep. International Computer Science Institute (ICSI), Berkeley, CA. (July).

MAHAJAN, R. AND FLOYD, S. 2001. Controlling high bandwidth flows at the congested router. In *Proceedings of the IEEE International Conference on Network Protocols*. Riverside, CA, 1–12.

MUNDUR, P., SIMON, R., AND SOOD, A. 1999. Integrated admission control in hierarchical video-on-demand systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*. Florence, Italy, 220–225.

NETFILTER/IPTABLES. 2003. Netfilter/iptables project. Available at `http://www.netfilter.org/`.

NETFLOW. 2003. NetFlow performance analysis. White paper. Cisco Systems, Inc. Available at `http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/ntfo_wp.htm`.

NIST. 1998. NIST net network emulator. Available at `http://is2.antd.nist.gov/itg/nistnet/`.

PAN, R., PRABHAKAR, B., AND PSOUNIS, K. 2000. CHOKE, a stateless active queue management scheme for approximating fair bandwidth allocation. In *Proceedings of the IEEE Infocom*. Tel-Aviv, Israel, 942–951.

PAREKH, A. AND GALLAGHER, R. G. 1993. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *ACM/IEEE Trans. Netw. 1*, 3 (June), 344–357.

PARK, K. AND LEE, H. 2001. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proceedings of the ACM SIGCOMM*. San Diego CA, 15–26.

PIEDA, P., ETHRIDGE, J., BAINES, M., AND SHALLWANI, F. 2000. A network simulator, differentiated services implementation. Available at `http://www7.nortel.com:8080/CTL/`.

RAHIN, M. A. AND KARA, M. 1998. Call admission control algorithms in ATM networks: A performance comparison and research directions. Research rep. (Feb.) University of Leeds.

SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. 2000. Practical network support for IP traceback. In *Proceedings of the ACM SIGCOMM*. Stockholm, Sweden, 295–306.

SHENKER, S. AND WROCLAWSKI, J. 1997. General characterization parameters for integrated service network elements. RFC 2215. IETF.

SHOETEN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., KENT, S. T., AND STRAYER, W. T. 2001. Hash-based IP traceback. In *Proceedings of the ACM SIGCOMM*. San Diego, CA. 3–14.

STOICA, I., SHENKER, S., AND ZHANG, H. 1998. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *Proceedings of the ACM SIGCOMM*. Vancouver, British Columbia, Canada, 118–130.

VINT. 1997. UCB/LBNL/VINT network simulator—ns (version 2). Available at `http://www.isi.edu/nsnam/ns/`.

WELSH, M. AND CULLER, D. 2003. Adaptive overload control for busy Internet servers. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*. Seattle, WA.

WOLSKI, R., SPRING, N. T., AND HAYES, J. 1999. The network weather service: A distributed resource performance forecasting service for metacomputing. *J. Fut. Gener. Comput. Syst. 15*, 6 (Oct.), 757–768.

Received March 2003; revised June 2004; accepted July 2004