Flash Crowd Mitigation via Adaptive Admission Control based on Application-level Observations

Xuan Chen and John Heidemann [*]

ISI-TR-557

Abstract

We propose the *network early warning system* (NEWS) to protect servers and networks from flash crowds, which usually happen when too many requests are sent to a web site simultaneously. NEWS is an self-tuning admission control mechanism, which imposes application-level congestion control (AppCC) between requests and responses. NEWS detects flash crowds from changes in web response rate. Based on the application-level observations, NEWS adjusts the admitted request rate automatically and adaptively. Simulation results show that NEWS detects flash crowds within 10 minutes (about 2–3 detection intervals). By delaying 56% of requests, NEWS is able to reduce the packet drop rate for responses from 17% to 1%. The aggregated response rate for admitted requests is twice as fast with NEWS as compared to without. This performance is similar to the best possible rate limiter.

1 Introduction

Recent studies [1, 2, 3] show that the Internet is vulnerable to persistent overloading caused by flash crowds [2] and the denial of service (DoS) attacks [3]. In this paper, we focus on imposing application-level control mechanism to mitigate flash crowds.

Flash crowds usually happen when many end-users simultaneously send requests to one web site because of a sudden new interest. Common examples of new interests are natural events such as earthquakes or breaking news stories, or links from popular web sites. In minutes or even seconds after the event, the volume of requests toward the *target web server* increases dramatically to tens or hundreds times more than normal. As shown in Figure 1, these requests overload the target server. The target server may reject some requests, and process those accepted ones slowly due to either resource limits (CPU or disk) at the server, or congestion on the response's network path (often in the first few links where the traffic concentration is the largest). As a result, most or all users perceive unacceptably poor performance. In addition, flash crowds may unintentionally deny service for other end-users who either share common links with flash crowd traffic or retrieve unrelated information from the target server.

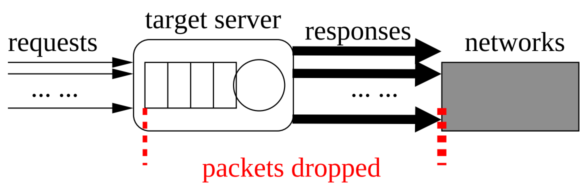


Figure 1: Request and Response Exchanges during Flash Crowds.

TCP congestion control is the basic mechanism the Internet uses to cope with resource constraints [4, 5]. However, TCP control is on a per-connection basis (Figure 2) and so is unable to deal with flash crowds where the problem is too many arriving connections. Congestion control mechanisms that aggregate information per-host (for example, the congestion manager [6]) also fail to address this problem because these connections arrive from many hosts.

We therefore argue that a high-level control

[*]Xuan Chen and John Heidemann are with University of Southern California, Information Sciences Institute. This material is based upon 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 NSF.

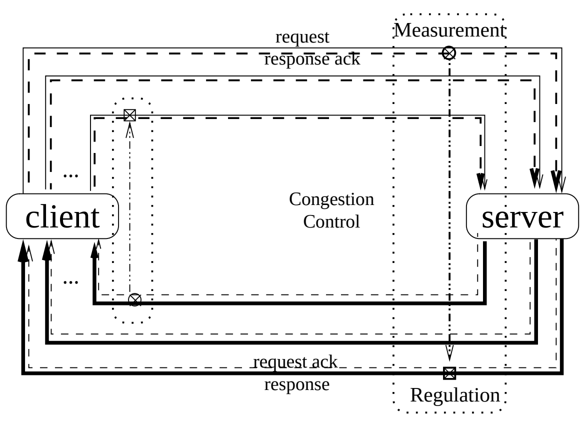


Figure 2: TCP Congestion Control.

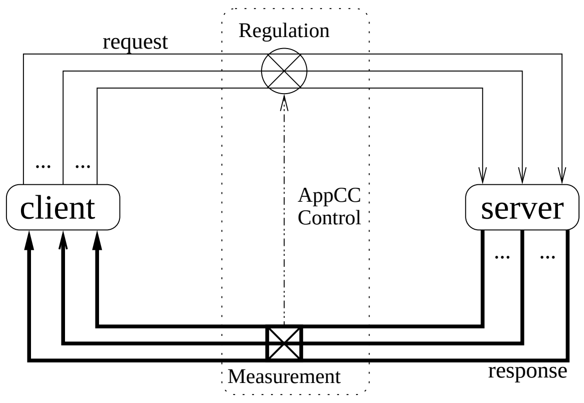


Figure 3: Application level Congestion Control.

mechanism between request and response is essential to mitigate flash crowds, that is, end-user applications or edge routers should observe web performance and adjust the rate of requests to the target server accordingly. For example, browsers should slow down the request rate upon the observation of increased web latency. Since this control scheme needs application level information such as web request rate, response latency or rate, we call it the Application-level Congestion Control (AppCC). We illustrate the control diagram of AppCC in Figure 3.

Similar to TCP end-to-end congestion control [5], a potential problem with AppCC is the mis-behaved end-users or applications: greedy end-users intentionally violet the principle of AppCC. So, we propose to apply incentives in routers to enforce AppCC. With routers' assistance, we also save the effort to modify end-user applications.

We propose *network early warning system* (NEWS) as a router-based adaptive admission control mechanism, which imposes AppCC between requests and responses. Different from other admission control schemes [7, 8, 9, 10], NEWS does not consider per-flow service requirement for incoming requests. Moreover, NEWS only measures the aggregated rate of high-bandwidth responses, while the measurement-based admission control (MBAC) aggregates the performance of all existing flows.

NEWS has the following two novel aspects: first, the approach of controlling request admission rate is based on measured performance of response (rather than previous approaches that admit requests based on either explicit service requests or measurements in requests). Second, NEWS observes changes in the rate of high-bandwidth responses and adjusts the request rate automatically and adaptively. These approaches make NEWS a self-tuning system, which adapts to different environment easily.

In the following sections, we present the detailed design of three main components in NEWS: flash crowd detector, request rate limiter, and controller. We evaluate the performance of NEWS through simulations and find that NEWS detects flash crowds within 10 minutes (about 2–3 detection intervals). By delaying 56% of requests, NEWS protects servers and networks from overloading: reduces the packet drop rate for responses from 17% to 1% For end-users, NEWS provides twice as fast the aggregated response rate for admitted requests. This performance is similar to that of the best possible rate limiter deployed in the same scenario.

2 Related Work

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

2.1 Web Caching and Content Delivery Networks

Infrastructure vendors such as Akamai [11] deploy web caches and content delivery networks (CDN) to accommodate the excessive web requests during flash crowds. Recent studies [2, 12] show that the current web caching scheme is not efficient because most web pages are not cached before flash crowds. Jung et. al.proposed a new scheme—the adaptive web caching [2] for improvement.

It is necessary to provide enough resources to prevent flash crowds from happening; options include increasing the capacity of servers and networks, duplicating the contents requested, and deploying web caches. However, there are circumstances that it is either difficult or impossible to estimate and provide "enough" required resources. Therefore, we believe that we need to impose control mechanism such as AppCC to mitigate flash crowds in these cases, so that servers and networks survive from overloading and some end-users may still perceive high performance.

2.2 Admission Control Schemes

Admission control schemes are important to support applications with service requirement such as real-time constraint. Look at the public telephone networks [13] and the integrated service networks [14], a new connection (or a call) explicitly describes the service required such as two channels for telephone conversation or 1Mbps bandwidth for Video-on-Demand service [8]. Based on this service profile and the current available resources (circuits or network bandwidth), admission control makes decision on whether it should accept the incoming request or not.

Although appropriate for telephone and integrated services networks, it is very difficult for many applications to accurately estimate their service requirements. So, this approach often under-utilizes the networks. NEWS avoids this problem by determining application requirements dynamically through measurement.

Except the general difference between NEWS and admission control schemes, Measurement-based admission control (MBAC) [15, 9, 10] looks very similar to NEWS: both schemes control the incoming traffic based on measurement of existing connections. However, MBAC aggregates the performance of all existing flows; while NEWS only measures the aggregated rate of high-bandwidth responses. Also, MBAC is more conservative and only accepts the incoming requests upon sufficient resource; while NEWS send all request through unless a flash crowd is detected.

In order to protect the target server, we could apply a rate limiter to ensure the incoming request always below a certain threshold. The rate limiter simply rejects excessive requests so that the target server always works under its capacity limit.

Despite its simplicity, rate limiter lacks the ability to adapt to different environment. In order for a rate limiter to work properly, network operators need to choose the rate limit carefully based on their experience [16]. Usually, this rate limit is specific to a certain server or network link, and needs manual adjustment when the server's capacity or the link's bandwidth changes. On the contrary, we design NEWS as a self-tuning system, which adapts to different environment easily and depends on no human interference such as pre-configuring rate limit.

Moreover, since NEWS detects flash crowds by examining response performance, it is capable to discover the overloaded condition either at servers or in networks equally efficiently. This is not a simple task for rate limiter, since web latency may be limited by networks or servers [17] and a threshold specific to the server may not work to relief congestion in networks.

Another similar work in this region is the network weather service (NWS) [18], which monitors and forecasts the performance such as link utilization and server load. Both NWS and NEWS share some common ideas in change detection algorithms. However, NWS applies a centralized data processing algorithm; while NEWS is a local algorithm.

2.3 Congestion Control Algorithms

TCP applies end-to-end congestion control at flow level for request and response. However, TCP is not sufficient to protect servers and networks from flash crowds because flash crowds are caused by too many concurrent connections. On the contrary, NEWS realizes a high-level control by imposing AppCC between request and response aggregates.

There are some other algorithms to control congestion caused by aggregates, such as the congestion manager (CM) [6] and the aggregate-based congestion control (ACC) [1]. CM is a per-host based congestion control scheme, which multiplexes concurrent flows among different applications of one end hosts to ensure that they react to congestion properly. Unfortunately, since most connections in flash crowds are short-lived, the information aggregated by CM at one host does not help to detect and mitigate flash crowds.

ACC is proposed to control persistent congestion caused by aggregates, which are composed of flows with TCP end-to-end congestion control. ACC captures aggregates which consume most of network bandwidth and regulates the rate of these aggregates with a virtual queue. ACC is a self-tunable control mechanism and can be applied to control response traffic in flash crowds, which is likely to consume high bandwidth and cause large packet drops.

However, CM and ACC can not mitigate flash crowds fundamentally. Similar to TCP congestion control, neither CM nor ACC has a complete control loop between requests and responses as AppCC (Figure 3), which we believe is essential to protect servers and networks from flash crowds. Therefore, these schemes have no control over the cause of flash crowds: too many concurrent requests.

3 Flash Crowds and Early Warning

To better describe the characteristics of flash crowd traffic [2] and the overall design of NEWS, we first define some basic terminologies. We depict some of them in Figure 4.

We refer a flow $f(s,d)$ as a series of pack-

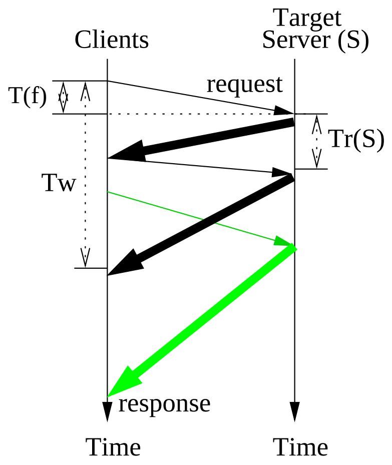


Figure 4: The Transmission Latency, the Life Time of Web Connection, and the Request Interval.

ets transferred from a source s to a destination d. We assign each flow an unique identification number; and use two metrics to quantify its performance: *transmission latency* $T(f)$ records the time interval (in second, for example) from s sending the first packet till d receiving the last packet, *transmission rate* $R(f)$ measures the bit rate (in bits per second, for example) of the flow. Flows show various transmission rates and latencies due to server load and congestion condition in networks.

Looking at web traffic, a web connection from client C to server S contains one (with HTTP/1.1 [19]) or a series (with HTTP/1.0 [20]) of request and response flows. We define the life time of a web connection (T_w) as the time interval from client sending the first request packet till it receiving all the response packets.

From servers' point of view, we define the *request interval* ($Tr(S,C)$) as the time between two adjacent requests from client C to server S. We denote $Tr(S)$ as the request interval from all clients to server S. Alternatively, we define $R_p(S)$ as the *request rate* (in number per second) observed by server S, which records number of requests sent to S within a time unit. For a router, it observes the packet rate of all requests pass through. which We define as the *aggregated request rate* R_p.

A set of flows with certain constraint forms an aggregate ϕ. The number of flows in an aggregate is $|\phi|$. As an example, aggregate $\phi_{\forall f(*,D)}$ contains all flows with the same destination address D. An aggregate may also choose any k flows randomly. K could be a constant such as 5 or a variable such as $k = 10\% \times |\Phi|$, where Φ is a special aggregate containing all flows[1]. To quantify the performance of an aggregate ϕ, we calculate its *aggregated transmission rate* (AR_ϕ) as:

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

3.1 Understanding Flash Crowd Traffic

Flash crowds is a term commonly used to describe sudden increase in the access to a web server that posts a breaking news after a sudden event or have links from a popular web site. Examples include the "slash-dot effect" [21]. When flash crowds happen, largely increased requests are sent to the target server almost simultaneously; the target server observes a spike in the request rate.

In this paper, we focus on the flash crowds caused by unpredictable events. This is the most challenging case since we don't have any clue when flash crowds will happen and how much the volume of requests will increase.

3.1.1 Increased Web Latency during Flash Crowds

End-users may experience increased web latency during flash crowds. As shown in Figure 1, several factors contribute to this increase. First, even each request just contains a tiny packet, too many of them may still cause congestion in networks. So, requests may be delayed or even dropped by networks. Second, the target server can only accept part of the incoming requests due to its capacity limit; and simply discards others. Because of its heavy load, the target server processes requests and generates responses slowly. These responses contain number of large packets and inject more load than requests. They are likely to congest networks and increase transmission latency or packet loss rate. When a response packet is lost, the target server needs to retransmit it despite it is already heavily loaded. Further investigation is needed to quantify the delay in these different phases within request and response exchanges.

Under different circumstances, flash crowds may cause overloading at the target server or in networks. However, from end-users' point of view, they always perceive an increased web latency or a decreased web response rate. By detecting flash crowds from application-level observations: web response rate, NEWS adapts to different circumstances automatically

3.1.2 Characteristics of Flash Crowd Traffic

Flash crowds may show different traffic pattern due to their causes [2]. In this study, we examine three server side traces which were collected during a slash-dot effect event [21]. Since systematically modeling flash crowds is beyond the scope of this work, we only investigate these traces roughly.

We have two observations. First, requests show very small inter-arrival time: the *request interval* is around 1–3 seconds. Second, quite some requests arrived at the target server in bulk, that is, their arrival time recorded by the target server are the same. We propose a simple flash crowd traffic model to reflect these characteristics in Section 6.1.

The characteristics of flash crowd traffic impose challenges to detection scheme. Flash crowd traffic is usually originated from hundreds and thousands clients. These clients may not have visited the target server before. We call these clients *cold-clients*. Formally, a client is "cold" with respect to server S if $Tr(S) > Tr_0$ (Tr_0 is a constant, for example $Tr_0 = 24hours$). The existence of cold-client implies little temporal correlation among requests during flash crowds. Therefore, we can not monitor the performance change of flows with particular source-destination pairs.

Further, flash crowd traffic usually contains many *short-lived* connections (assume not in a movie-downloading scenario). We define short-

[1]Therefore, we have $\Phi = \phi_{\forall f}$ and $\forall \phi \subseteq \Phi$.

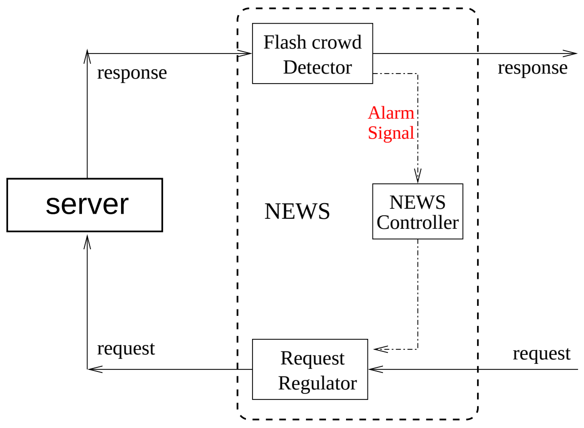


Figure 5: The Architecture of NEWS

lived connections as those that has $T_w < T_{w0}$ (T_{w0} is a constant, for example $T_{w0} = 1second$). We can also define short-lived flow in a similar way. As a result, it is meaningless to monitor the performance change of one particular connection.

3.2 Network Early Warning

As shown in Figure 5, NEWS has three main components: flash crowd detector, request regulator, and controller. This design is consistent with the diagram of AppCC (Figure 3). We present the detailed design of flash crowd detector in Section 4, and other two components in Section 5. We design NEWS in a modular style so that we have the flexibility to apply new techniques without modifying the structure of the system. For example, we could adopt web caching techniques in the module of request regulator.

NEWS is an self-tuning admission control scheme, which *detects flash crowd* from application-level observations. By regulating *requests* adaptively, NEWS protects networks and servers from flash crowds. NEWS also maintains high response rate for the admitted requests rather than leaves all end-users suffering the decreased performance.

Compared to TCP congestion control, NEWS is a control mechanism on large time scale. So, it can apply sophisticated techniques while still remain as a light-weighted scheme to the existing networks. For example, we can adopt complicated change detection algorithm or implement fine-grained request regulator.

In the design of NEWS, we do not make any assumption about the address and location of the target server. We just assume that we can always deploy NEWS reasonable close to the target server. For example, we install NEWS on the access link of the target server or its ISP.

We also assume that request and response traverse the same access link. However, this assumption does not hold in circumstances such as a multi-homed domain. In that case, we need to distribute NEWS to all access links of the ISP, and apply certain scheme to coordinate the detection and regulation at different points.

4 Detecting Flash Crowds

Flash crowd detector processes the traffic measurement with certain algorithm, and triggers alarm signal when it detects a flash crowd. When designing the detector for NEWS, we consider the following three issues:

1. *What to monitor and how?* We design the flash crowd detector to monitor the application level information: the transmission rate of web response (response rate, for short). The detector detects flash crowds by capturing decrease in response rate. Since heavy load either at the target server or in networks could cause low response rate, NEWS adapts to the server or networks limited scenarios easily.

2. *What change detection algorithm to use?* The change detection algorithm [22] is a well studied in many fields such as signal processing and pattern recognition. It is basically the scheme that determines whether a change has occurred in the characteristics of the considered object. When designing the flash crowd detector, we tend to use a simple change detection algorithm to avoid computation complexity; but augment it with network information. In future, we plan to try more sophisticated algorithm such as the Canonical Correlation Analysis (CCA) [23] for possible performance improvement such

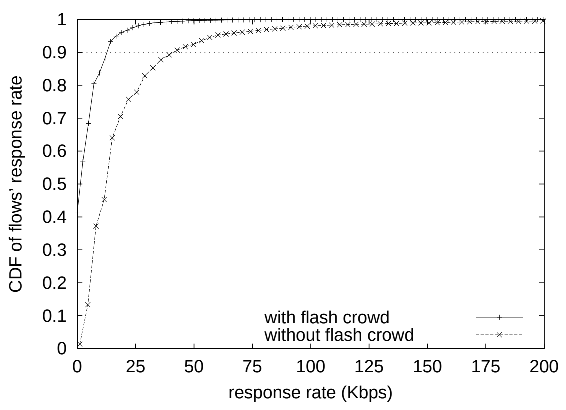


Figure 6: CDF of Flows' Response Rate in Flash Crowd and Background Traffic.

as reducing the detection latency.

3. *When to set and reset the alarm signal?* There are two trade offs to consider when answering this question. When setting the alarm signal, we intend to achieve fast response at the expense of possible false alarm. When resetting the alarm signal, we try to avoid fluctuation in output at the risk of penalizing more incoming requests.

We answer the first question in Section 4.1; and other two in Section 4.2. We present detailed algorithm in Section 4.3.

4.1 High-bandwidth Flows

Due to the existence of *cold clients* and *short-lived connections*, we can not detect flash crowds by monitoring the performance change of particular response flows (details in Section 3.1.2).

On the other hand, monitoring the mean rate of all responses is also not helpful. Flows react to congestion differently, with fast connections noticing the congestion very quickly, while low-speed flows (such as to users attached by modem) show very little change. Thus, if we average the rate of all responses, congestion results in very little change in average response rate because of many inherently low-speed flows.

Instead, we propose to detect flash crowds by monitoring changes in the aggregated performance of responses from *fast connections*, because those flows are most sensitive to congestion. We define these responses as high-bandwidth response flows (HBFs), which have rate greater than a constant R_0 (for example, $R_0 = 20Kbps$). The aggregate of HBFs are high-bandwidth aggregate ϕ_{HBF}; and those hosts sending or receiving HBFs are high-bandwidth hosts. Similarly, we can define low-bandwidth flows, aggregate, and hosts.

We verify this analysis through simulation (detailed simulation methodology in Section 6.1). We measure the response rate $R(f)$ of each individual flow with and without flash crowds, and compare their cumulated distribution functions (CDFs) in Figure 6. If we choose the 10% flows with the highest rate as HBFs, we find that their aggregated rate decreases about 70%.

We call the above observation HBF effect. Since most target servers (or their ISPs) are connected to the Internet in the similar way [24] as shown in Figure 8, we believe that we can observe HBF effect on the access link[2]of an ISP even in a general network topology. Further study is needed to verify this claim.

4.2 Change Detection Algorithm

The HBF effect implies that we can detect flash crowds by capturing the decrease in the aggregated rate of HBFs ($AR_{\phi_{HBF}}$). However, since clients may be cold and most flows are short lived (details in Section 3.1.2), there are chances that only low-bandwidth hosts are active and responses only show low rate. In that case, the $AR_{\phi_{HBF}}$ computed among these low-bandwidth flows is misleading.

To solve this potential problem, the detector also checks the aggregated request rate R_p when $AR_{\phi_{HBF}}$ decreases. More specifically, the detector triggers alarm signal only when both $AR_{\phi_{HBF}}$ decreases and R_p increases. In this way, we can make sure that the decrease in $AR_{\phi_{HBF}}$ is due to flash crowds rather than low-bandwidth hosts.

After it detects flash crowds, the detector watches increase in $AR_{\phi_{HBF}}$, which indicates the performance recovery in response. When that happens, the detector cleans the alarm signal.

[2]That is, the place we deploy NEWS.

We use a simple comparison based scheme to detect these changes. More specifically, the detector detects flash crowds if the two conditions 1 and 2 hold; and cleans the alarm signal when condition 3 holds.

$$AR_{\phi_{HBF}} < \overline{AR_{\phi_{HBF}}} \times (1-\delta) \tag{1}$$
$$R_p > \overline{R_p} \times (1+\delta) \tag{2}$$

$$AR_{\phi_{HBF}} > \overline{AR_{\phi_{HBF}}} \times (1+\delta) \tag{3}$$

$$0 < \delta < 1$$

δ reflects system's tolerance to the change. For example, with $\delta = 10\%$, the algorithm detects increase when the measurement is 110% larger and decrease when it is 90% smaller.

4.3 Algorithm Design

The flash crowd detector measures the transmission rate of response flows with time-sliding window (TSW) algorithm [25] to smooth the burntness of TCP traffic. The detector also measures request rate R_p.

Every T seconds, the detector computes $AR_{\phi_{HBF}}$. Since the number of flows observed at different time could vary dramatically, we choose the top p percent of responses with highest rate as HBFs: $|\phi_{HBF}| = p \times |\Phi|$. p is a tunable parameter, we choose $p = 10\%$ from Figure 6.

Then, the detector calculates the long-term average of aggregated rate for HBFs $\overline{AR_{\phi_{HBF}}}$ and the aggregated request rate $\overline{R_p}$. We propose to calculate these long-term averages with a High-Low Filter (HLF). HLF is essentially a combination of two Exponentially Weighted Moving Average (EWMA) filters with adjustable gains to fulfill different requirements.

We describe an EWMA filter as: $\overline{V(t)} = \alpha\overline{V(t-1)}+(1-\alpha)O(t)$, where $O(t)$ 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. Large α gives a stable output; while small gain makes the output sensitive to the current observation.

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

Finally, the detector compares AR with $\overline{AR}$ and R_p with $\overline{R_p}$. It sets and resets the alarm signal according to conditions in Section 4.2.

The detection interval T is a tunable parameter. A smaller T helps to detect changes promptly but the result may be unreliable; while a large T generates stable output but takes longer response time and causes more overhead in state keeping. Based on our experience in simulations, we set $T = 240 second$.

5 Mitigating Flash Crowds

In this section, we present the detailed design of the request regulator and NEWS controller. Given the alarm signal from flash crowd detector, the controller decides the proper reaction for the request regulator. NEWS applies an adaptive request rate limiter, which only functions after a flash crowd is detected.

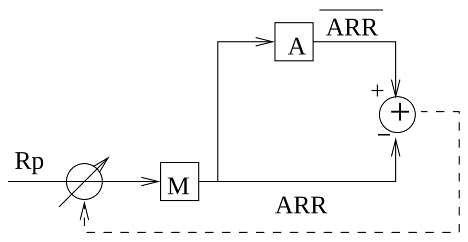
5.1 Regulating Requests

A request rate limiter should ensure that the admitted request rate converges to a reasonable value (R_{pc}). Ideally, with requests admitted at rate R_{pc}, neither target servers nor network links are overloaded.

We tired the approach of discarding excessive requests preferentially [26]. However, the result system is not stable: we observe osculations in system output. This is because that probabilistic dropping only guarantees expected behavior. Moreover, it is difficult to determine the dropping probability for requests based on the measurement on response. This relationship may not be a simple linear function.

To enforce the convergence in request rate, we switch to a token bucket based rate limiter. A token bucket has the following two parameters: *bucket size* provides accommodation to busrty traffic, and *token rate* limits the long-term packet arrival rate. In long run, the requests admitted by a token bucket converge to the token rate.

Above this token bucket based rate limiter, we



* M describes server processing request and generateing response.
* A is the algorithm to compute long−term average.

Figure 7: Controlling the Adaptive Rate Limiter.

		Current Alarm Signal	
		0	1
Previous Alarm Signal	0	No change	Reset rate limit
	1	No change	Adjust rate limit

Table 1: Diagram of State Transition in the Controller.

can further define more sophisticated policies according to different requirements such as distinguishing and protecting cross traffic from flash crowds, or maintain web performance for some particular end-users. However, as our goal is to investigate the design issues of a request regulator, the detailed policy definition is beyond the scope of this paper.

NEWS mitigates flash crowds so that it protects servers and networks from overloading and maintains a high performance for those admitted requests rather than leaving all end-users with the same poor performance. NEWS achieve these goals by delaying requests. As a result, the retransmissions of dropped requests inject more load to networks. Although we could tune NEWS to reduce the possible retransmissions, we still need to utilize some complementary schemes such as web caches to absorb the requests eventually.

5.2 Controlling the Adaptive Rate Limiter

We depict the function of the controller in Figure 7. Given the alarm signal from flash crowd detector, the controller adjusts the rate limit of the request regulator so that it adapts to different scenarios automatically. The NEWS controller maintains two states: the current and previous alarm signals. It adjusts the rate limit based on the state transition diagram in Table 1.

When the alarm signal is set (transition from 0 to 1), the controller resets the rate limit to the current admitted request rate R_{p0} observed by flash crowd detector. Intuitively, requests arriving with rate higher than R_{p0} are likely to increase the server and network load, and therefore cause decrease in response performance. When the alarm signal changes back to 0 (transitions from 0 to 0 or from 1 to 0), the controller keeps the same rate limit.

If the alarm remains set (transition from 1 to 1), the controller adjusts the rate limit with a score-board based scheme. Basically, the controller assigns scores to the adjustment of increasing and decreasing the rate limit and choose the direction with the higher core. For example, if the current scores for increasing and decreasing rate limit are 5 and 3 respectively, the controller chooses to increase the rate limit. If the alarm resets (transition from 1 to 0), the corresponding adjustment gets credit; otherwise it gets penalty. With this scheme, the controller learns to make decisions automatically from history, and it also adapts to different environment without human interference.

6 Algorithm Evaluation

We implement NEWS in *network simulator (ns-2.1b8)* [27] and evaluate its performance through simulations. For fast algorithm design and evaluation, we prototype NEWS under the framework of the DiffServ model contributed by the previous Advanced IP Networks group at Nortel Networks [28]. In the long run, we plan to migrate NEWS to a stand alone implementation.

6.1 Methodology

Figure 8 shows the simulated network topology. Router R0 connects a server pool with 5 web servers (S1–S5). S1 is the target server of flash crowd traffic. Router R1 connects two tiers of clients with different link bandwidths and propagation delays. We deploy NEWS to the access

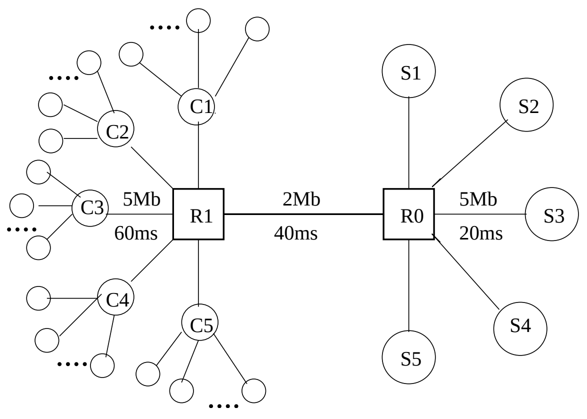


Figure 8: The Two-level Dumbbell Topology in Simulations.

link of the server pool: the link between R0 and R1. Initially, we set the bucket size of the adaptive rate limiter to some default value such as 50 packets. NEWS monitors responses (from R0 to R1), and regulates requests (from R1 to R0) if a flash crowds is detected.

We model flash crowd traffic in two layers: *background* models the normal web traffic pattern, and *flash crowd* captures the characteristics of flash crowd traffic. Background traffic exists throughout simulation, while flash crowd traffic only appears during a certain time period. We model both background and flash crowd layers based on the web traffic model in *ns* [29], where clients initiate a series of web sessions, each retrieving some web pages from randomly chosen servers; a web page contains several web objects, which is modeled by a heavy-tailed distribution to produce the self-similarity in web traffic [30]. Table 2 summarizes the attributes and corresponding distributions for background traffic. We choose the parameters according to a latest study on web traffic measurement [31].

The difference between flash crowd and background traffic is at session level, rather than at page level. So, we use the same page level attributes to model flash crowd traffic. We change session level attributes to reflect larger number of clients, shorter inter-session time, and smaller number of web pages within on session. We also model bulk arrivals (for example, the size of the bulk) for flash crowd traffic. We show the difference between flash crowd and background traffic model in Table 3.

The simulation runs for 20000 seconds. We inject flash crowd traffic at 1000 second. Without NEWS, the flash crowd ends at 8000 second. We conduct simulations with and without NEWS deployed. We record the offered and admitted request rate to the target server. We also measure the aggregated rate of HBFs, which reflects the end-users' perceived web performance. In the following sections, we evaluate the performance of NEWS from both target server's and end-users' perspectives.

6.2 NEWS Protects Servers and Networks from Overloading

One of the goals to deploy NEWS is to protect the target server and networks from overloading caused by flash crowds. In this work, we only simulate a network-limited scenario, where flash crowds overload networks with large amount of response traffic. Since NEWS detects flash crowds from application-level observations, we believe the following results are also valid in server-limited scenarios. Further simulation study is needed to verify this claim.

We measure the admitted request rate to the target server and compare the results without and with NEWS deployed in Figure 9(a). Looking at the early stage of flash crowds in Figure 9(b), NEWS detects flash crowd at 1595 second with the detection interval $T = 240 seconds$. That is, the detection latency is about 10 minutes, which is about 2 or 3 detection intervals.

After detecting the flash crowd, NEWS adjust its rate limit automatically and regulates the incoming requests to a proper rate. As a result, the admitted request rate drops; and the congestion in response traffic is released: reducing packet drop rate from 17% to 1%.

Due to the resource limitation at the target server or in networks, NEWS discards or delays excessive requests. In Figure 10, we show the offered request rate increases because of retransmissions as predicted in Section 5. Although NEWS can't serve all the requests, it does protects the server and networks and gain time for the deployment of other complimentary schemes such as web caches to absorb these excessive re-

Web Model Elements	**Element Attributes**	**Probabilistic Distributions**	**Parameters**
Web Session	Number of Web Sessions	Constant	value: 100
	Time Interval between Sessions (seconds)	Exponential	mean: 600
	Number of Web Pages per Session	Exponential	mean: 5
Web Page	Time Interval between Pages (seconds)	Exponential	mean: 30
	Number of Web Objects per Page	Exponential	mean: 5
Web Object	Time Interval between Web Objects (seconds)	Exponential	mean: 0.01
	Object Size (KBytes)	Pareto II	mean: 12, shape: 1.2

Table 2: The Attributes of Web Model and Corresponding Distributions of Background Traffic.

Traffic Type	**Session Number**	**Inter-session Time (seconds)**	**Number of Pages in one Session**	**Bulk Size**
background	100	600	5	NONE
flash crowd	1000	6	3	Exponential (mean: 3)

Table 3: Different Session-level attributes for Background and Flash crowd Traffic.

quests.

6.3 NEWS Maintains High Response Rate for Admitted Requests

We design NEWS to maintain high performance for admitted requests. As shown in Figure 11, HBFs suffer largely decreased performance during flash crowds: the aggregated rate of HBFs is only 50% of that before flash crowds. By deploying NEWS, the end-user perceived performance of those admitted requests remains consistently high even when flash crowds happen. This result verifies that NEWS protects the admitted requests from flash crowds rather than leaves equally low performance for all end-users.

NEWS applies sophisticated techniques for the above performance improvement, which rate limiter could also achieve with careful configuration. Ideally, we want NEWS to perform similarly as the best possible rate limiter. We investigate this issue below.

6.4 NEWS Achieves Similar Performance as the Best Rate Limiter

We first deploy request rate limiters to the link between R0 and R1. By adjusting the rate limit, we get different aggregated rate for HBFs as shown in Figure 12(a). In this particular network environment, we find that rate limiter gives best performance when the rate limit is set to 4 requests per second.

We compare NEWS with the best rate limiter in Figure 12(b) and summarize their performance in Table 4. We find that NEWS shows similar performance as the best rate limiter: NEWS shows only around 15% less in the aggregated rate of HBFs. It is also encouraging to find that the adaptive rate limiter in NEWS discovers the best request rate. We highlight the aggregated rate of HBFs in Figure 13.

Since NEWS needs to adjust its rate limit to the right value, it does not maintain a high aggregated rate for HBFs in the early stage. We believe this process of hunting the right rate limit reduces the overall performance of NEWS. On the other hand, NEWS rejects 16% less requests than the best rate limiter in average, which avoids packet retransmissions.

Given similar performance, we believe it is an advantage to deploy NEWS since it is self-tuning and adapts to different environments easily. By deploying NEWS, we save the effort for manual configurations.

Scenarios	**Admitted Request Rate (number/s)**	**Request Rejection Rate**	**Aggregated Rate of HBFs (Kbps)**	**Response Loss Rate**
Original traffic	4	0%	48.9	17%
NEWS	3.6	56%	102	1%
Rate limiter	3.6	67%	117.8	0.1%

Table 4: Comparison of NEWS with Rate Limiter.

7 Conclusion

In this paper, we propose NEWS to protect servers and networks from persistent overloading caused by flash crowds. We design NEWS as a self-tuning admission control scheme, which imposes AppCC between requests and responses. NEWS controls the incoming requests based on the measurement in response performance.

NEWS is a self-tuning system and adapts to different environments easily. We present the design details of its three main components. We show that NEWS detects flash crowds effectively by measuring changes in the aggregated rate of HBFs. With the adaptive rate limiter and the controller, NEWS mitigates flash crowds by regulating incoming requests.

We evaluate the performance of NEWS through simulation. Simulation results show that NEWS detects flash crowds within 10 minutes (about 2–3 detection intervals). By dropping excessive requests, NEWS reduces the packet loss rate in response from 17% to 1%. Also, the aggregated response rate for admitted requests is twice as fast with NEWS as compared to without NEWS. This performance is as good as the best possible rate limiter, while NEWS does not need specific configuration.

Acknowledgments

We would like to acknowledge Stephen Adler for providing server trace of slash-dot effect. We appreciate the fruitful discussion with members in *SAMAN* and *CONSER* projects. We are graceful to Dr. Chuanxiong Guo and many people who have given us insightful suggestions for this work.

References

[1] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. Technical report, ICSI, 2001.

[2] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *Proceedings of the International World Wide Web Conference*, pages 252–262. IEEE, May 2002.

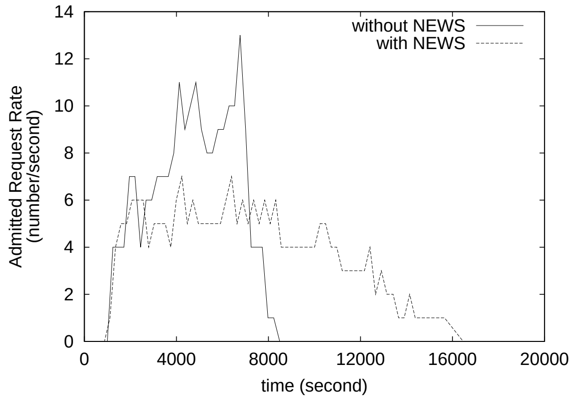
[3] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of the ACM SIGCOMM*, pages 15–26, San Diego CA, USA, August 2001. ACM.

[4] Van Jacobson. Congestion avoidance and control. In *Proceedings of the SIGCOMM '88*, pages 314–329, Stanford, California, August 1988. ACM.

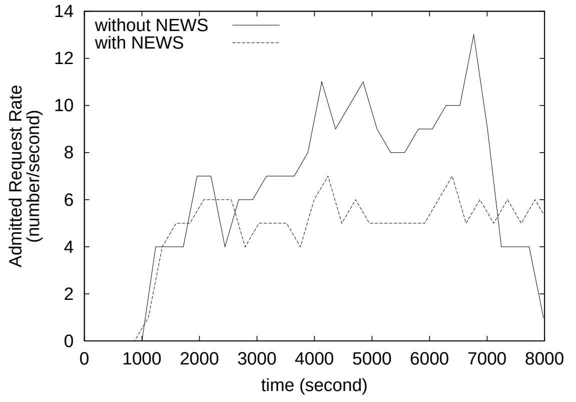
[5] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *ACM/IEEE Transactions on Networking*, 7(4):458–473, August 1999.

[6] Hari Balakrishnan, Hariharan Rahul, and Srinivasan Seshan. An integrated congestion management architecture for internet hosts. In *Proceedings of the ACM SIGCOMM*, pages 175–187, Cambridge, MA, USA, 1999. ACM.

[7] Mohammad A. Rahin and Mourad Kara. Call admission control algorithms in atm networks: A performance comparison and



(a) NEWS Regulates the Admitted Request Rate Adaptively.



(b) Early Period of the Above Figure.

Figure 9: Changes in Admitted Request Rate with NEWS.

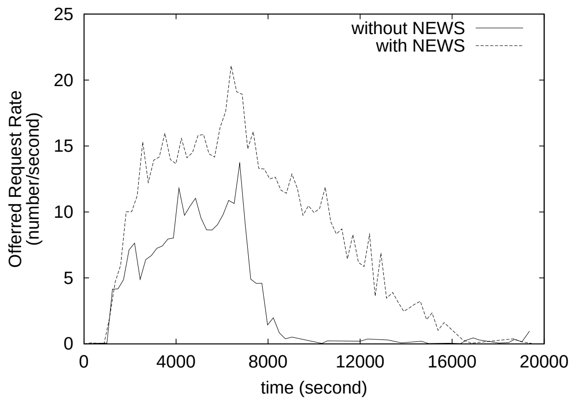


Figure 10: Increased Offered Request Rate due to Retransmission.

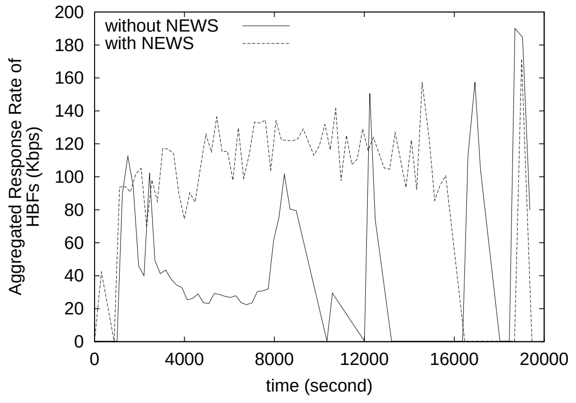


Figure 11: NEWS Maintains High Aggregated Rate for HBFs during Flash Crowds.

research directions. Research report, University of Leeds, February 1998.

[8] P. Mundur, R. Simon, and A. Sood. Integrated admission control in hierarchical video-on-demand systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 220–225, Florence, Italy, June 1999. IEEE.

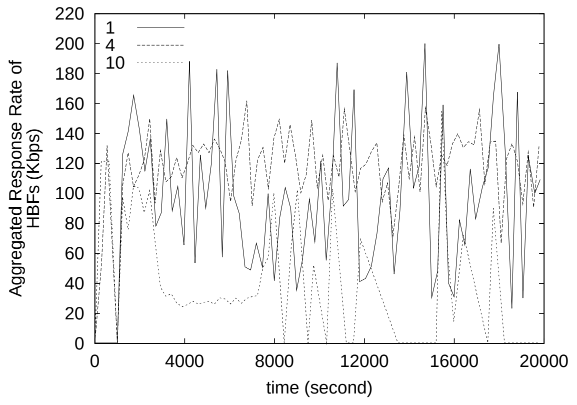
[9] C. Cetinkaya and E. Knightly. Egress admission control. In *Proceedings of the IEEE Infocom*, Tel-Aviv, Israel, March 2000. IEEE.

[10] Lee Breslau, Edward W. Knightly, Scott Shenker, Ion Stoica, and Hui Zhang. Endpoint admission control: Architectural issues and performance. In *Proceedings of the ACM SIGCOMM*, pages 57–69, Stockholm, Sweden, August 2000.

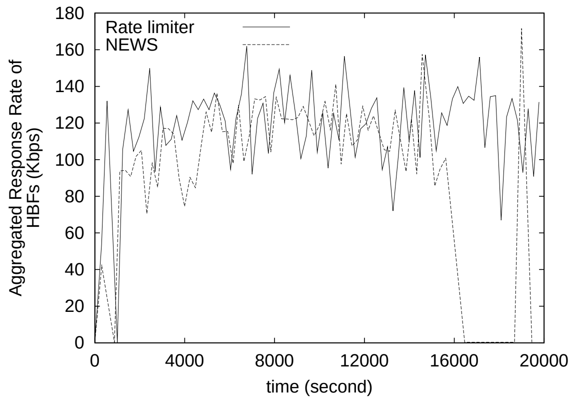
[11] Akamai Technologies Inc. http://www.akamai.com.

[12] Martin Arlitt and Tai Jin. A workload characterization study of the 1998 world cup web site. *IEEE Network, Special Issue on Web Performance*, 14(3):30–37, May 2000.

[13] R.J. Gibbens, F. P. Kelly, and P.B. Key. A decision-theoretic approach to call admission control in atm networks. *IEEE Jour-*



(a) The Aggregated Rate of HBFs with Different Rate Limiters.



(b) Compare the Best Rate Limiter with NEWS.

Figure 12: Aggregated Rate of HBFs with Rate Limiter and NEWS.

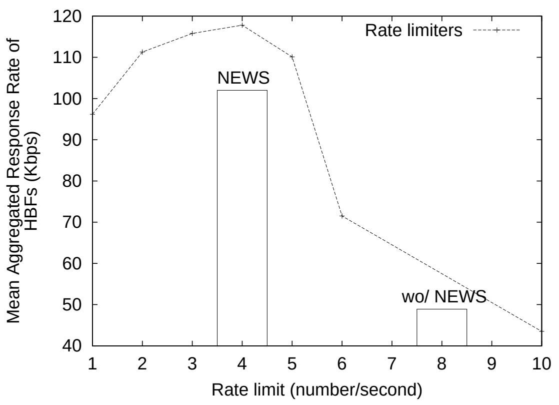


Figure 13: The Aggregated Rate of HBFs with Different Schemes.

nal on Selected Areas in Communications, 13(6):30–37, August 1995.

[14] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of the ACM SIGCOMM*, pages 14–26, Baltimore, MD, USA, August 1992.

[15] Jamin. S., Danzig. P.B., Shenker. S, and Zhang. L. Measurement-based admission control algorithms for controlled-load service. In *Proceedings of the ACM SIGCOMM*, pages 2–13, Cambridge, MA, October 1995. ACM.

[16] P. Barford and D. Plonka. Characteristics of network traffic flow anomalies. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pages 2–13, San Francisco, CA, November 2001. ACM.

[17] Lars Eggert and John Heidemann. Application-level differentiated services for web servers. *World-Wide Web Journal*, 2(3):133–142, August 1999.

[18] Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. Technical report, September 1998.

[19] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol—HTTP/1.1. RFC 2616, Internet Request For Comments, June 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[20] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol—HTTP/1.0. RFC 1945, Internet Request For Comments, May 1996. ftp://ftp.isi.edu/in-notes/rfc1945.txt.

[21] Stephen Adler. The slashdot effect, an analysis of three internet publications. February 1999.

[22] Michele Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall, Englewood Cliffs, NJ, first edition edition, April 1993.

[23] E. Jonckheere, K. Shah, and S. Bohacek. Dynamic modeling of Internet traffic for intrusion detection. In *Proceedings of the IEEE American Control Conference*, Alaska, USA, May 2002. IEEE.

[24] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the ACM SIGCOMM*, pages 251–256, Cambridge, MA, USA, August 1999. ACM.

[25] D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *ACM/IEEE Transactions on Networking*, 6(4):362–373, August 1998.

[26] R. Mahajan and S. Floyd. Controlling high bandwidth flows at the congested router. In *Proceedings of the IEEE International Conference on Network Protocols*, pages 1–12, Riverside, CA, USA, November 2001. IEEE.

[27] VINT group. UCB/LBNL/VINT network simulator—ns (version 2). http://www.isi.edu/nsnam/ns/.

[28] Peter Pieda, Jeremy Ethridge, Mandeep Baines, and Farhan Shallwani. A network simulator, differentiated services implementation. http://www7.nortel.com:8080/CTL/, 2000.

[29] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proceedings of the ACM SIGCOMM*, pages 301–313, Cambridge, MA, USA, August 1999. ACM.

[30] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. In *Proceedings of the ACM SIGMETRICS*, pages 160–169, Philadelphia, Pennsylvania, May 1996. ACM.

[31] F.D. Smith, F. Hernandez Campos, K. Jeffay, and D. Ott. What TCP/IP protocol headers can tell us about the web. In *Proceedings of the ACM SIGMETRICS*, pages 245–256, Cambridge, MA, June 2001.