Protecting Bursty Applications Against Traffic Aggressiveness

Anat Bremler-Barr * Nir Halachmi† Hanoch Levy ‡

April 16, 2007

Abstract

Aggressive use of networks, in particular the Internet, either by malicious or innocent users, threatens the service availability and quality of polite applications. Common queueing mechanisms which supposedly solve the problem, are shown in this work to be ineffective for bursty applications, including Web applications. This can be exploited by malicious users to conduct a new kind of denial of service attacks.

We propose a new traffic control mechanism called *Aggressiveness Protective Queuing (APQ)* which is based on attributing importance weights to the users and which solves this problem by dynamically decreasing the weight of the aggressive users. The actual weight used for a flow is a dynamically varying parameter reflecting the past bandwidth usage of the flow. We show that under heavy load (deterministic model), *APQ* significantly restricts the amount of traffic an aggressive user can send and bounds it , at most, to *twice* the amount of traffic sent by a polite (regular) user. Simulation results demonstrate the effectiveness of APQ under a stochastic environment.

Keywords: Security, Quality of Service, Denial of Service, Queuing

1 Introduction

The Internet is characterized by a variety of applications varying in their traffic characteristics and Quality of Service (QoS) demands. One of the major objectives of network control is to provide an application certain performance as to grant it proper quality. This must be done in a tough environment where many other unpredicted applications compete for the network resources.

*School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. bremler@idc.ac.il
†School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. halachmi.nir@idc.ac.il
‡School of Computer Science, Tel-Aviv University, Israel. hanoch@cs.tau.ac.il

In recent years the welfare of the Internet has been threatened by malicious attacks of Distributed Denial of Service (DDoS). The DDoS attacker consumes the resources of the victim, a server or a network, causing a degradation in performance or even total failure of the victim. In a common type of DDoS attack, the attacker sends a huge amount of traffic, consuming the CPU or the bandwidth of the victim. One of the common remedies suggested in the literature [16] and practically used [2] is to use traffic control mechanisms to bound the amount of traffic that can reach the victim from a source address [1].

Weighted Fair Queuing is one of the common control mechanisms that have been thoroughly studied and widely used in network devices such as network routers and Web servers. WFQ aims at granting an application an "equal share" of the network resource (link, server); in a nutshell - a WFQ mechanism provides that the instantaneous fraction of the resource is fairly (typically equally) divided among all applications competing for that resource at that instance.

Applying WFQ to the sources that send traffic to a victim which is under DDoS attack, will slow down the traffic rate of an attacker to be as low as that of a regular user. However, as shown in this work, in an environment of bursty applications this is not enough. In a bursty environment, the malicious users can conduct a new type of DDoS attack which can overcome the WFQ mechanism.

Bursty applications make up significant part of the Internet traffic. These are applications whose traffic demands vary heavily over time. Perhaps the most important of these is the Web application whose burstiness is inherently implied from the behavior of a Web session. Such a session is characterized by transmission of file (web document) from the Web server to the client, followed by a random think time (browsing of the document by the user), followed by another file transfer (as a response to a new request by the client) and another think time, and so on. Typical think times can range between a second to tens of seconds. Thus the application is characterized by short traffic demands separated by longer "silent" periods.

Attackers can use this traffic characteristic to conduct a new type of attack by running "aggressive applications". These are applications that continuously pose traffic demands over long periods of time. That is, the application does not act according to the lineament of bursty applications, and in addition to sending traffic during the burst it sends traffic also in the "silent" periods.

As we show in this paper, in such environments the pure WFQ mechanism cannot protect

[1] Using traffic control mechanism is applicable to attacks that use non spoof packets, and where the number of attackers is moderate in comparison to the number of legitimate users. Other type of attacks are out of the scope of this paper.

the typical non-aggressive (which we denote *polite*) applications from the aggressive applications, and thus the polite applications are doomed to receive poor QoS. This is due to the fact that the network capacity is planned by taking into account the fact that bursty applications are not continuously active and by utilizing statistical multiplexing among them[2]. An aggressive application, in contrast, consumes in the long run more resources than a polite application, for which the system was designed, and hence harms the allocated bandwidth of the polite application in the system. WFQ cannot remedy this problem since it aims at equally dividing the network resources, at every epoch, among all applications who have data to transmit at that epoch. An aggressive application that behaves at *every isolated epoch* like an innocent one will not be distinguished by WFQ; however, since in the long run it requests much more than a polite application, it can damage the performance of the system and of the polite applications. In section 5 we provide a simple analysis demonstrating the magnitude of this problem.

So far, we have demonstrated the problem arising due to "malicious" applications, such as a DDoS attacker. Nonetheless, the same problem may also show up with "innocent applications" such as 1) Pre-fetching devices operating on a client's Web browser and making pre-fetch requests of files the user might need in the future, and 2) Users who conduct a massive download of files (songs, videos) from a network whose capacity was not planned for such applications. Thus, if the network is resource bounded, aggressive users can adversely affect the performance of polite users. [3]

Recognizing this fact, our objective is to devise a new traffic control mechanism that is be flexible enough to adjust to the time-varying demands of the polite bursty applications while protecting them from the aggressive applications. To this end, we propose a new mechanism called *Aggressiveness Protective Queueing* (APQ), which is based on the principles of the WFQ scheme and which uses *dynamic weights* for its operation. APQ uses a simple mechanism for tracking the resource usage by the flows and uses this accounting to properly and dynamically control the weights of WFQ. The actual weight used for a flow is the product of its original weight (that would be granted to it by the administrator of WFQ) multiplied by a dynamically varying variable reflecting the past bandwidth usage of the application. As such, APQ forms a simple algorithm that can take an existing WFQ algorithm together with its weights and make it robust against aggressive users. APQ uses two types of parameters, one that can be related to the average traffic load of a flow and one that relates to the burstiness. APQ is a flexible

[2]That is, in most cases the number of applications concurrently competing for the resource is not too large.

[3]Some readers might feel that they have the "right", performance-wise, to download files as much as they wish. While we do not get into the argument of whether this "right" is proper or not, our point is that this cannot come on account of the more polite users.

mechanism that with a proper configuration can handle authorized applications that require heavy volume (either bursty or continuous) and light volume concurrently while protecting them against *unauthorized users* that place continuous heavy loads.

The scheme devised is, nonetheless, a non-idling scheme that continues to process requests at full rate as long as there are requests in the system. Thus, when polite users do not have requests to process, the aggressive users can make use of the full system processing power. Having devised the APQ algorithm, we turn to analyze it. We show that an aggressive user cannot affect polite users in any significant way. This results from the fact that once the aggressive user becomes aggressive, its weight reduces drastically compared to that of the polite user. We analyze the algorithm under the assumption that the ratio between the maximal load that can be placed by an aggressive user and the load of an innocent user is given by j. We show that when the network is overloaded, at any interval of time Δt the amount of traffic sent by an aggressive user is bound by twice the traffic sent by a polite user (compared to a ratio of j, when APQ is not used) and bound by one time the traffic sent by a polite user in the case that this is a continuous naive aggressive user. Furthermore, we show that even a sophisticated aggressive user, that is aware of all the details of the APQ algorithm and uses them to its benefits, is bound to send $\sqrt[3]{3 \cdot (j-1)} - 1)$ times the traffic of a polite user.

The proper place for implementing APQ in the network is either in Edge Routers, where the network administrator can control it's configuration, or at Web servers. APQ is designed not to change the pure version WFQ, but to add a wrapper around it. APQ can be used to improve fairness in the network as well as a DDoS protection mechanism.

The rest of the paper is organized as follows. The next section provides the relevant related work. In Section 3 we demonstrate the problem discussing WFQ with static weights and describe why it cannot properly handle bursty traffic in the presence of aggressive users; a queueing system that models this problem is presented and is used to quantify the problem. In Section 4 we introduce the use of dynamic weights in WFQ to address this problem. In Section 5 we provide analysis results and in section 6 we provide numerical results obtained via simulation to demonstrate the effectiveness of APQ in controlling bursty traffic. In Section 7 we discuss practical considerations in implementing APQ.

2 Related Work

Fairness is a desirable characteristic of scheduling mechanisms. It provides protection against traffic hogs by ensuring that they cannot overrun others, since the system allocates to them the same rate as to the other polite flows in the system. General Processor Sharing (GPS)

is a theoretical mechanism that is believed by many to provide "ultimate" fairness of packet scheduling. Its fairness is achieved by allocating the resources at *every (infinitesimal) epoch* equally (or weighted equally) among the backlogged flows. WFQ is a practical *approximated* implementation of GPS [18].

However, in bursty environments even GPS cannot guarantee protecting the polite flows from the aggressive ones. The reason is that GPS ensures fairness only in an (infinitesimal) momentary fashion as a function of the momentary status of the flows. However, in bursty traffic applications, where the polite flows must be compared to the aggressive ones over longer time scales,it will fail and will not be able to provide longer time scale fairness. That is, over a long period of time the aggressive users can send much more traffic than the regular polite bursty users. We argue in this paper that in order to be fair in bursty environment one needs to define fairness differently than that of GPS. Fairness in bursty environment should be defined by bounding the total amount of traffic the user can send over a longer period.

As far as we are aware of, we are the first to point the shortcomings of using the common fairness definition of GPS in the bursty environment and to propose an alternative one.

Mechanisms for per-flow treatment at the router can be classified as based on either scheduling or preferential dropping. Scheduling approaches, like WFQ, place flows in different scheduling queues, and determine packets of which queue should be transmitted. In contrast, preferential dropping mechanisms, like RED[15] or RIO [7], decide which packets to drop.

The papers [8, 7] deal with the general subject of "large flows" vs. "small flows", only in the context of the dropping mechanisms and address an issue entirely different from the one we address. The issue addressed by [7] is that under flow control mechanisms like TCP, the nature of the flow control tends to negatively discriminate the "small flows" (in comparison to the "large flows"). This is because short connections do not gain detailed knowledge of the network state, and therefore, due to the conservative nature of the TCP congestion control algorithm ("slow start") they are doomed to be less competitive. Specifically, due to the "slow start" mechanism, the congestion window of short connections is relatively small most of the time, and thus packet losses they incur always require a timeout which translates to disastrous effect on the short connection performance. The desirable goal is to achieve the regular fairness in an environment of "large flows" and "small flows". To overcome these difficulties of TCP, [7] and [8] propose different queue management policies to be used in the router, that once interact with the TCP congestion mechanism will still achieve fairness (in the GPS notions). Their main idea is to compensate for this flow-control discrimination by giving preferential treatment to short flows, and to drop packets or provide congestion notification to large flows

and not to short ones.

As such, they do not deal at all with the fairness of GPS in bursty environments. Further, they are tailored only to deal with the slow start problem of the flow control mechanism and thus are not designed to general traffic (e.g., for traffic with no flow control, like UDP, they may end up with positively discriminating the short flows). Dealing with any type of traffic is, of course, crucial in designing a mechanism for malicious environments, where the attacker can send any type of traffic (UDP) or not obey to TCP rules.

[9] deals with problem of scalability in scheduling mechanism, where it is difficult to maintain state information for a very large number of flows. This is achieved by applying preferential dropping mechanisms specifically on high-bandwidth flow and by maintaining state only to these flows. The technique. named RED-PD, explicitly leverages the skewed bandwidth distribution, where only small number of flows is responsible for the most of the bytes sent in the internet. The paper does not deal with the problem of GPS fairness in bursty environment over long time scale.

In our paper we concentrate on WFQ and its bursty traffic problem, since this is de-facto the common mechanism implemented in today's network elements for achieving fairness. One of the advantages of our proposed solution APQ, is that it can be implemented as a wrap around WFQ. One of the disadvantages of WFQ, which our solution, inherits, is the scalability issue, since the mechanism keeps states per each flow.

We note that preferential dropping mechanism (e.g. RED-PD [9]) may serve as alternative mechanisms to WFQ. To this end, a subject for future research is whether and how one can use the idea of preferential treatment[7] in packets dropping in order to bound the total amount of traffic an aggressive user can submit over long time scale, regardless of the traffic type and in scalable manner. Note also that the scalability of RED-PD[9], is based on a normal internet skewed bandwidth distribution, where only small number of flows is responsible for most of the bytes sent in the internet. This might no longer be true in malicious aggressive environments. Moreover, the technique is narrower than WFQ in its functionality, since RED-PD per-se lacks the capabilities of assigning different weights to flows.

There are many creative ways to conduct DDoS attacks. Similar to other works [16, 17, 13] we relate to DDoS as a resource management problem. As such, Denial of Service attacks can be mitigated using different resource management solutions. Unfortunately, as has occurred many times, an attacker can find "holes" in those mechanisms and still create an attack. In [11] it was shown that another traffic control mechanism, the admission control, can be used to produce a new type of attack, reduction of quality. To the best of our knowledge this work

is the first to expose the vulnerability of WFQ as a DDoS solution and its ineffectiveness to protect against aggressive attackers in a bursty application environment.

"Burstiness" issues have been addressed in the WFQ literature but in a completely different context from how we have dealt with them in this paper. Formally it dealt with the "burstiness" (or smoothness) of the *output process* of WFQ, and it was shown that the original WFQ can significantly deviate from the GPS schedule and thus generate non-smooth (or "bursty") output. The WF^2Q scheduling proposed in [10] overcomes this difficulty and the deviation of its output from GPS is much smaller.

Our solution to the fairness problem of weighted fair queuing with bursty flows, is to adaptively change the weights. Adaptive weighted or Dynamic weighted fair queuing has previously been proposed in the literature. Adaptive (Dynamic) WFQ, changes the weights dynamically in a response to events occurring in the network.

The work [16] proposes an adaptive WFQ mechanism as part of an architecture to defend against DDoS attacks. The basic motivation is to decrease a flow's weight as a *penalty mechanism* designed to decrease the attackers resources and minimize their effect on the network. That work does not deal with the problem of WFQ with bursty traffic specifically, and the "penalty" weight is determined as a function of the *trust* the system has in the source (thus weight can reduce if the flow does not pass certain legitimacy tests). In contrast, our work focuses on shedding light on the problems pure-WFQ encounters with bursty traffic, and how to solve them. We further offer a specific function for WFQ weight control, in order to minimize the affects of the problem. In [14] and [12] adaptive WFQ was proposed in order to cope with the problem that in some small temporal timescales, the static WFQ weights may be inaccurate and need to be adjusted. Those studies change the flow's weight to adjust to the flow's current rate, that is when the flow's offered traffic is larger than earlier estimated, they increase the flow's weight, and thus might have adverse affects in light of aggressive users; in contrast, we adjust the weight to be inversely proportional to the user's offered traffic to prevent aggressive users from dominating the link.

A short abstract version of this paper was presented at a conference [6].

3 WFQ: Unfair Service to Bursty Flows in the Presence of Aggressive Users

As stated above, Web traffic is characterized by a bursty behavior. This burstiness results from the nature of the user's activity, in which the user requests a Web page (including all its sub-documents, such as GIFs and other pictures and elements) stops for a while to read the

document and then requests another Web page (with its associated pieces), and so on. Thus, the load inflicted on the system by a typical user is bursty: Relatively high demands for a short period, followed by very low (or zero) demand for a longer period. A typical scenario of a user is to request the page and all its associated documents (e.g. 15 documents all together) within a few seconds (e.g., 3 seconds) and then to be quiet for tens of seconds (e.g., 30 seconds) before making the next page request.

This type of behavior inflicts two types of request rates: 1) Peak Request Rate, which is the request rate when the user makes the requests (15 documents per 3 seconds in the above example), and which can be translated to the more useful measure of *Peak Packet Rate - PPR* (which can also be measured in bytes) and 2) Average Request Rate, which is the average number of requests over time (15 documents per 33 seconds in the above example), translating to the *Average Packet Rate – APR*. Note that APR is a "long term rate" while PPR is a "temporal rate". Note that this user behavior allows admission control and network design to be based on the fact that typically the requests of the different users spread over time and thus the number of users that can be accommodated in the system is much larger than the capacity divided by the PPR value.

The presence of users that do not comply with this behavior can significantly degrade the network performance, and increase packet loss and delay. We will refer to these users as *aggressive users* (while referring to the regular users as *polite users*). Their motivation can be either to exploit the network in a selfish manner (such as using a WEB pre-fetcher) or even to maliciously harm the network (DDoS attackers). An aggressive user takes advantage of the idle time in order to send data, transmitting at PPR all the time. WFQ cannot handle polite users in a fair way in the presence of aggressive users. The reason is that a polite user will receive its share of the bandwidth only when it is not idle (which is a small fraction of the time), while the aggressive user "requests" and receives its share of the bandwidth "all of the time".

To appreciate the potential magnitude of the problem one should note that the average request rate of a polite user is given by APR while that of an aggressive user is given by PPR. Thus, the load placed on the system by an aggressive user is equivalent roughly to that placed by APR/PPR polite users. In the above example, this ratio equals 10, and it can easily reach higher values. With the current scheduling of WFQ, the aggressive user's data is transmitted 10 times more than the data of a regular user over a long time period.

Our main point is that WFQ is fair at every epoch, and divides the network resources equally at each such epoch. But WFQ is not fair over a long time period in the case of a mixed

environment of aggressive flows and bursty-traffic flows. From our perspective, fairness means that the total bandwidth a user can send over a long time period should be bound similarly for all users (i.e., the data transmitted by the queueing model should have the same average packet rate.).

3.1 Can A Rate-Limiter (in front of WFQ) Resolve the Problem?

A natural approach to deal with the problem could be to use the "standard approach" of placing rate-limiter units (Leaky-bucket type), each dedicated to a user, in front of the WFQ scheduler. The rate-limiter then prevents a user's flow from entering the WFQ system at a too high rate. While this can provide full protection to polite users it suffers from a major drawback: it limits each user to a certain rate and thus the system will not operate well at epochs of low load, or epochs where the number of active users is relatively low. At such an epoch, one would like to use WFQ at its full processing rate in order to provide better service to the currently active users, as well as to empty their buffers in order to improve the system performance in future epochs. However, the rate-limiter mechanism limits the rate entering WFQ and thus we may reach a situation where some of the user buffers are loaded (at the rate limiter) while the processor does not operate at full rate, and thus can be quite inefficient.

A more severe drawback of this mechanism is that it will significantly hurt the performance of a polite application regardless of whether there are aggressive users in the system or not. This will occur when the inter-arrival times between the requests of a polite user vary, or if the amount of data (web page size) requested in each request vary; both of these are in fact inherent properties of the Web. Under this situation, the rate limiter is likely to prevent the polite application from processing requests that are relatively closely adjacent to each other (since it limits the peak rate of the application) even when the system is relatively empty. Thus, many of the statistical-multiplexing advantages of the system cannot come into effect.

4 Aggressiveness Protective Queuing (APQ)

Our solution is a new mechanism, Aggressiveness Protective Queuing (APQ), which is a variation on WFQ. APQ satisfies the following requirements:

1. It supports bursty users with a Peak Packet[4] Rate (PPR), and an Average Packet Rate (APR) that is significantly smaller than the peak (since polite users are inactive for long

[4]All rates and traffic volumes in this paper are measured in packets. Transformation of packets to bits may be reasonable for certain applications and can be done easily.

periods).

2. It negatively discriminates users that are using more than the average rate (APR) for a long period of time.

3. The limitation imposed by the system on the users is a function of the system load, i.e., if overall, the system is under-loaded, then it imposes weaker constraints on the users (any user). That is, while satisfying requirements 1 and 2 above, the mechanism does not constrain users unless it is necessary for keeping the system from getting over-loaded.

APQ satisfies the above requirements using a dynamic weight function that reduces the weight assigned to aggressive users. For every user, the mechanism counts the amount of traffic that the user has generated in the near history and uses this amount to affect the weight given to it. For example, if the counted amount for some user is large, then the function reduces the weight assigned to this user. The counters are calculated over a sliding window, where the window size covers the short history relevant to the dynamic weight function. Specifically, the window size Δ (measured in time units) is set to roughly the expected inter-burst time, namely about ABS/APR, where ABS is the average burst size (measured in packets) and APR is the average packet rate.

In order to describe the function, we define the following parameters. Let $R(t)$ be the rate (measured in packets) of the user offered traffic at time t. Let $D_{in}(x,y) = \int_x^y R(t)dt$ be the total traffic offered by the user to the queueing system in the interval $[x,y]$. Let $SM(t) = D_{in}(min(0, t-\Delta), t)$ be the amount of offered traffic during the last sliding window.

Let w_o be the original fixed weight assigned to a user and let $w(t)$ be the dynamic weight assigned to the user time t.

In this paper we explore the following dynamic weight function of APQ:

$$w(t) = \begin{cases} w_o & SM(t) \leq ABS \\ w_o \cdot (\frac{ABS}{SM(t)})^\alpha & SM(t) > ABS, \end{cases}$$

where α is a penalty factor which is configurable in the system. We require that $\alpha \geq 1$. The greater α is, the smaller the weight that is assigned to an aggressive user.

The function adaptively changes the weight assigned to the user in a way that for a user that offers to the queueing mechanism more than ABS traffic in a period of duration Δ, the weight is reduced proportionally to the deviation from ABS. The system configuration of APQ requires the configuration of two parameters per application type in the system: the burst size ABS and the window size Δ. Legitimate applications that require high constant bit

rate, will be configured with a suitable high ABS. These are in addition to the user's original weights w_o required by WFQ, and which may or may not be identical for all users.

5 Analysis of APQ

We analyze the effect of APQ against two types of aggressive users: **Naive Aggressive** - A user that is not aware of APQ and its dynamic weight modification. Its strategy is to *continually* transmit at peak rate (PPR) (as opposed to a polite user who goes idle between bursts).[5]

Sophisticated Aggressive - A user that is aware of the APQ algorithm, namely the actual dynamic weight function. Its aim is to maximize the bandwidth APQ will grant it and its strategy is to offer traffic in a sophisticated way as to achieve this objective.

5.1 Queueing Model

In order to get some evaluation of APQ, we consider a simplistic scenario that can shed light on the relative performance of these policies.[6]

The scenario is as follows:

- Each aggressive user transmits at constant peak rate R.
- Each polite user transmits at peak rate R for a duration T_{on} and then stays idle for a duration T_{off}; both T_{on} and T_{off} are constants. Thus, we have $ABS = R \cdot T_{on}$.
- The number of polite users in the system who are *concurrently active* (that is, they are at their T_{on} period) plus the number of aggressive users are fixed N, and K of them are aggressive users.[7] This implies that the number of concurrently active users is roughly f times smaller than the overall number of polite users in the system, where $f = \frac{T_{on}+T_{off}}{T_{on}}$.
- For each user the original weight is 1 i.e., $w_o = 1$.

[5]The assumption is based on the fact that aggressive users do not wish to be detected. Moreover WFQ constant weight deals with an anomaly in PRR, since in time of congestion it divides equally the network resources among all the users with the same constant weight.

[6]It is very hard to analyze WFQ (and furthermore APQ) in a general setting. The transmission rate of a flow (user) depends on the overall system condition, i.e, the number of active users and their assigned weights.

[7]This is, off course, not a practical situation, since the number of active users constantly changes due to the stochastic nature of the system. However it will serve as a basis to approximately evaluate the relative performance of the methods

- Queuing space is limited and thus packets that are not transmitted within a period of Δ from their arrival time are dropped.

- We choose the window size of APQ, Δ, to be equal to the periodicity of a user, i.e. $T_{on} + T_{off}$.

Let B be the capacity of (or bandwidth allocated to) the output link of the queuing system. In the analysis we are interested in cases where the network is in congestion. That is, we assume that $B \leq R(N-K)$. Note that when the network is not in congestion, APQ will let all users, including the aggressive users, enjoy the free bandwidth.

Let f be the burst factor $= \frac{T_{on}+T_{off}}{T_{on}} = \frac{\Delta}{T_{on}}$ (since we choose the window size to be equal to Δ). This factor plays a major role in the analysis results. Let $D_{user}^{schedule}$ be the data transmitted by the queuing schedule (WFQ or APQ) for the particular type of user (*polite*, *naive-aggressive* or *sophisticated-aggressive*) during an interval of size Δ.

5.2 Vulnerability Factor

To evaluate the quality of a scheduling algorithm, we introduce the *Vulnerability Factor*, $V_{user}^{schedule}$ of a scheduling policy with respect to the "aggressiveness" of the aggressive users. The factor indicates the number of denied-service polite-users per aggressive user. That is, it is the number of *polite* users that do not have room in the system (and thus will have to be denied access at admission control) due to the presence of a single aggressive user. Specifically, we will evaluate it via $V_{user}^{schedule} = D_{user}^{schedule}/D_{polite}^{schedule}$ where *user* is either *naive-aggressive* or *sophisticated-aggressive*. Note the behavior of this factor: the higher the vulnerability factor the less protection the scheduler provides to polite users.

5.3 Polite Users

- Offered rate: $R(t)$ periodically equals R for a duration of T_{on} and 0 for a duration of T_{off}.

- Total offered traffic : $ABS = R \cdot T_{on}$

- Transmitted traffic under WFQ: Using this model, the transmitted rate of a polite user is $R = \frac{B}{N}$ under WFQ (the link bandwidth divided by the sum of weights of the flows in the system). Hence, we have $ABS = \frac{B}{N} \cdot T_{on}$ and

$$D_{polite}^{WFQ} = \frac{B}{N} \cdot T_{on}. \tag{1}$$

- Transmitted traffic under APQ: Under APQ the transferred rate of each user is between $\frac{B}{N-K}$ to $\frac{B}{N}$ transmitted rate (due to the reduction in the effect of the aggressive users). Hence

$$\frac{B}{N} \cdot T_{on} \leq D_{polite}^{APQ} \leq \frac{B}{N-K} \cdot T_{on} \tag{2}$$

5.4 Naive Aggressive Users

- Offered rate: R
- Total offered traffic: $f \cdot ABS$ (recall $f = \frac{\Delta}{T_{on}}$).
- Transmitted traffic under WFQ:

$$D_{naive}^{WFQ} = \frac{B}{N} \cdot \Delta \tag{3}$$

- **Corrolary 1** *The vulnerability factor of WFQ for a naive aggressive user is given by:*

$$V_{naive}^{WFQ} = D_{naive}^{WFQ} / D_{polite}^{WFQ} = \Delta / T_{on} = f \tag{4}$$

- **Transmitted traffic under APQ:** We turn to analyze the upper bounds of a user's transmitted traffic. To this end, we assume that the aggressive user did not offer any traffic in the previous window (In the next bullet item we will analyze the maximum transmitted traffic under the assumption that the aggressive user was also active in the previous window).

$$\begin{aligned} D_{naive}^{APQ} &\geq \int_0^{T_{on}} \frac{B}{N} dt + \int_{T_{on}}^{\Delta} (\frac{ABS}{R \cdot t})^{\alpha} \cdot \frac{B}{N} dt \\ &= \int_0^{T_{on}} \frac{B}{N} dt + \int_{T_{on}}^{\Delta} (\frac{R \cdot T_{on}}{R \cdot t})^{\alpha} \cdot \frac{B}{N} dt \\ &= \frac{B}{N} \cdot (T_{on} + T_{on}^{\alpha} \cdot \int_{T_{on}}^{\Delta} \frac{1}{t^{\alpha}} dt), \end{aligned} \tag{5}$$

where the inequality results from the same arguments leading to Equation 2. Similarly,

$$D_{naive}^{APQ} \leq \frac{B}{N-K} \cdot (T_{on} + T_{on}^{\alpha} \cdot \int_{T_{on}}^{\Delta} \frac{1}{t^{a}} dt). \tag{6}$$

For penalty factor $\alpha = 1$:

$$D_{naive}^{APQ} \leq \frac{B}{N-K} \cdot T_{on}(1 + \log f)$$

Similarly,

$$D_{naive}^{APQ} \geq \frac{B}{N} \cdot T_{on}(1 + \log f). \tag{7}$$

For penalty factor $\alpha = 2$:

$$D_{naive}^{APQ} \leq \frac{B}{N-K} \cdot (T_{on} + T_{on}^2 \cdot \int_{T_{on}}^{\Delta} \frac{1}{t^2} dt) =$$
$$\frac{B}{N-K} \cdot T_{on}(1 + T_{on} \cdot (-\frac{1}{\Delta} + \frac{1}{T_{on}})) =$$
$$\frac{B}{N-K} \cdot T_{on}(1 + \frac{f-1}{f}) \tag{8}$$

Similarly,

$$D_{naive}^{APQ} \geq \frac{B}{N} \cdot T_{on}(1 + \frac{f-1}{f}) \tag{9}$$

Corrolary 2 *The vulnerability factor of APQ with $\alpha = 1$ and $\alpha = 2$ for a naive aggressive user are bound by*

$$V_{naive-aggressive}^{APQ(\alpha=1)} \leq (1 + \log f) \cdot \frac{N}{N-K} \tag{10}$$

$$V_{naive-aggressive}^{APQ(\alpha=2)} \leq 2 \cdot \frac{N}{N-K}. \tag{11}$$

- **Continuous naive aggressive users:** We analyze the transmitted traffic of an aggressive user that was also active in the previous window size. The assigned weight of this user is fixed since in every time unit $SM(t) = R \cdot \Delta$ and hence $w(t)$ is fixed to

$$w(t) = (\frac{ABS}{R \cdot \Delta})^{\alpha} = (\frac{1}{f})^{\alpha}, \tag{12}$$

and hence

$$D_{cont-naive}^{APQ} \leq \int_0^{\Delta} (\frac{1}{f})^{\alpha} \cdot \frac{B}{N-K} dt =$$
$$(\frac{1}{f})^{\alpha} \cdot \frac{B}{N-K} \cdot \Delta = (\frac{1}{f})^{(\alpha-1)} \frac{B}{N-K} \cdot T_{on}. \tag{13}$$

Similarly,

$$D_{cont-naive}^{APQ} \geq (\frac{1}{f})^{(\alpha-1)} \frac{B}{N} \cdot T_{on}. \tag{14}$$

For Penalty factor $\alpha = 1$ we get:

$$\frac{B}{N} \cdot T_{on} \leq D_{cont-naive}^{APQ} \leq \frac{B}{N-K} \cdot T_{on}.$$

For Penalty factor $\alpha = 2$ we get:

$$(\frac{1}{f}) \cdot \frac{B}{N} \cdot T_{on} \leq D_{cont-naive}^{APQ} \leq (\frac{1}{f}) \cdot \frac{B}{N-K} \cdot T_{on}.$$

Thus, we have:

Corrolary 3 *The vulnerability factor of APQ with $\alpha = 1$ and $\alpha = 2$ for a continuous naive aggressive user obeys*

$$V_{cont-naive}^{APQ(\alpha=1)} \leq 1 \cdot \frac{N}{N-K}, \tag{15}$$

$$V_{cont-naive}^{APQ(\alpha=2)} \leq \frac{1}{f} \cdot \frac{N}{N-K}. \tag{16}$$

5.5 Sophisticated Aggressive Users

In this subsection we analyze the amount of traffic that a sophisticated aggressive user can possibly transmit under *APQ*. This user is assumed to know the function used by APQ and optimizes its offered traffic in order to maximize the traffic APQ will transmit for him. An approach that intuitively sounds optimal for the sophisticated aggressive user is to offer the same amount of traffic as the mechanism allows him to transmit. This is due to the fact that the dynamic weight function is in accordance with the offered traffic and hence any traffic that the user offers but does not transmit, only decreases his weight. However, in order to choose the above strategy, the sophisticated user must also be aware of the load in the network in order to offer traffic which matches the capacity the system has allocated to him. Naturally, this is knowledge the sophisticated aggressive user does not posses since it requires him to have details from the system in run time.

For the sake of analysis, we will assume that the network is in a condition where the number of concurrent users N' is fixed and it obeys $\frac{B}{N'} = R$ (where R is the peak rate allowed per user, PPR). Further, since we seek an upper bound on the amount of traffic the sophisticated user can transmit, we will assume that this user is knowledgeable of the value R.

To carry out the analysis, we prove in Lemma 5.1 that the traffic the sophisticated aggressive user can transmit, using any strategy of traffic offering, is bound from above by $(\sqrt{2 \cdot (f-1)} + 2) \cdot ABS$ (recall that $ABS = (B/N)T_{on}$) traffic units under APQ with $\alpha = 1$ and by $(\sqrt[3]{3 \cdot (f-1)} + 2) \cdot ABS$ traffic units under APQ with $\alpha = 2$. Second, in Lemma 5.2 we show that indeed using the strategy where the user offered rate is $R(t) = w(t) \cdot R$ allows the user to transmit as much as $(\sqrt{2 \cdot (f-1)} - 1) \cdot ABS$ under APQ with $\alpha = 1$ and by $(\sqrt[3]{3 \cdot (f-1)} - 1) \cdot ABS$ under APQ with $\alpha = 2$. These results imply that the above strategy is indeed optimal (up to a constant difference of $3ABS$) and achieves transmitted traffic volume to the order of $\Theta(\sqrt{f} \cdot ABS)$, which is to the order of $\Theta(\sqrt{f})$ times larger than the amount of traffic transmitted for a polite user.

Lemma 5.1 *Under APQ a sophisticated aggressive user cannot transmit in a period of duration Δ more than $(m+2)\cdot ABS$ traffic where m is derived from the equation $\sum_{i=1}^{m} i^{\alpha} \leq f-1$. In case that $\alpha = 1$ this translates to*

$$D_{out}^{APQ} \leq (\sqrt{2\cdot(f-1)}+2)\cdot ABS. \tag{17}$$

In case that $\alpha = 2$ this translates to

$$D_{out}^{APQ} \leq (\sqrt[3]{3\cdot(f-1)}+2)\cdot ABS. \tag{18}$$

Sketch of the proof: We will consider an arbitrary time interval $[t_s, t_e]$ of size Δ, and show that regardless of the offered traffic rate of the user, it cannot transmit more than $\sqrt{2\cdot(f-1)}+1\cdot ABS$ over that interval. The key idea of the proof is to break the traffic transferred in $[t_s, t_e]$ to "chunks" of size ABS (see Figure 1), and to calculate the minimal amount of time it takes to transmit the ith chunk, $i \geq 1$. We will show that transmitting the ith chunk takes at least a duration of $(i-1)^{\alpha}\cdot T_{on}$ (by using the fact that the weight is assigned according to the offered traffic which is at least the amount of transmitted traffic). Using this we will derive a boundry on the amount of traffic that can be transmitted in the interval of size Δ.

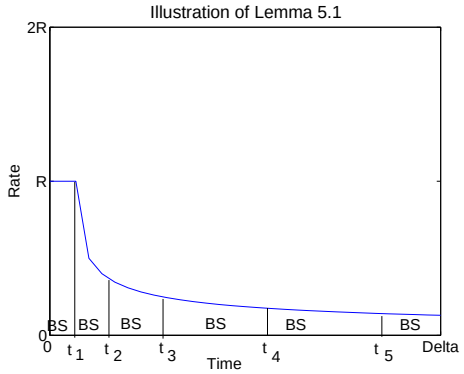


Figure 1: An illustrating figure for Proof of Lemma 5.1

Starting to count time and traffic at t_s, let $t_0 = t_s$ and let t_i, $i \geq 1$ denote the epoch at which exactly $i \cdot ABS$ traffic units are transmitted. Let w_i, $i \geq 1$, be the weight assigned to the user at t_i. Let $m \geq 0$ be the largest i such that $t_i \leq t_e$ and such that the user transmits $i \cdot ABS$ in (t_s, t_i)(In Figure 1 we illustrate these quantities).

First, let's examine t_1: Clearly the amount of traffic transferred in (t_s, t_1) is exactly ABS. Next, consider t_2: The amount of traffic transferred in (t_s, t_2) is $2 \cdot ABS$. From the definition of the weight function and since $t_1 - t_s < \Delta$ we get that $w_1 \leq \frac{1}{1^{\alpha}}$. Similarly, for an arbitrary value of i, $i \leq m$ we get that $w_i \leq \frac{1}{i^{\alpha}}$.

For the same reason, for every $1 \leq i \leq m-1$ the weight at epoch t, $t_i \leq t \leq t_{i+1}$, is bound by $\frac{1}{i^{a}}$. The amount of traffic that is transmitted between t_i and t_{i+i} is ABS. Hence $(t_{i+1} - t_i) \cdot \frac{1}{i^{a}} \cdot R \geq ABS$. Similarly, and since less than ABS traffic is transferred in (t_m, t_e) we have $(t_e - t_m) \cdot \frac{1}{m^{a}} \cdot R \geq ABS$.

Hence, $t_{i+1} - t_i \geq i^\alpha \cdot T_{on}$ (for $1 \leq i \leq m-1$) and $t_e - t_m \geq m^\alpha \cdot T_{on}$. Also, we must have $t - 1 - t_e \geq T_{on}$ (since ABS traffic is transferred at this interval). Now, since $\Delta = t_e - t_s = (t_e - t_m) + (t_1 - t_s) + \sum_{i=1}^{m-1}(t_{i+1} - t_i)$ we get $\sum_{i=1}^{m} i^a \cdot T_{on} + T_{on} \leq \Delta$ which leads to $\sum_{i=1}^{m} i^\alpha \leq f - 1$.

For $\alpha = 1$, we get that $\sum_{i=1}^{m} i^1 \leq f - 1$. That is, $\sum_{i=1}^{m} i = \frac{m\cdot(m+1)}{2} \leq f - 1$ and thus $m \leq \sqrt{2 \cdot (f-1)}$. A user can transmit in (t_s, t_e) at most $(m+2) \cdot ABS$ traffic units: one in the interval (t_s, t_1) plus m in the interval $(t_1 t_m)$ plus one in the interval (t_i, t_e) . Hence we receive that $D_{out}^{APQ} \leq ABS \cdot (\sqrt{2 \cdot (f-1)} + 2)$.

For $\alpha = 2$ we get that $\sum_{i=1}^{m} i^2 \leq f - 1$. That is, $\sum_{i=1}^{m} i^2 = \frac{m\cdot(m+1)\cdot(2m+1)}{6} \leq f - 1$ and thus $m \leq \sqrt[3]{3 \cdot (f-1)}$. Since a user can transmit in (t_s, t_e) at most $(m+2) \cdot ABS$ traffic units, we get that $D_{out}^{APQ} \leq ABS \cdot (\sqrt[3]{3 \cdot (f-1)} + 2)$ ∎

Corrolary 4 *The vulnerability factor of APQ with $\alpha = 1$ and $\alpha = 2$ for a sophisticated aggressive user are bound by*

$$V_{sophis-aggressive}^{APQ(\alpha=1)} \leq \sqrt{2 \cdot (f-1)} + 2, \tag{19}$$

$$V_{sophis-aggressive}^{APQ(\alpha=2)} \leq \sqrt[3]{3 \cdot (f-1)} + 2. \tag{20}$$

Lemma 5.2 *There is a strategy where during an interval of length Δ the sophisticated aggressive user can transmit under APQ with α at least $(m+1) \cdot ABS$ traffic where m is derived from the equation: $\sum_{i=1}^{m}(i+1)^\alpha \geq f - 1$. In case that $\alpha = 1$ this translates to*

$$D_{out}^{APQ} \geq (\sqrt{2 \cdot (f-1)} - 1) \cdot ABS \tag{21}$$

and in case that $\alpha = 2$ this translates to

$$D_{out}^{APQ} \geq (\sqrt[3]{3 \cdot (f-1)} - 1) \cdot ABS. \tag{22}$$

Proof: Consider the following strategy of the sophisticated aggressive user: The sophisticated user starts with a history of zero traffic sent, and sends an amount of traffic exactly identical to what the algorithm allocates to him according to the weight (i.e., $R(t) = w(t) \cdot R$). Note that the user starts with weight equal to 1 ($w(0) = 1$) since he did not send any traffic in the pervious window. We will show that using this strategy the sophisticated user can transmit $\sqrt{2 \cdot (f-1) - 2} \cdot ABS$ traffic.

We use the formulation and the notation used in Lemma 5.1. Consider i, $1 \leq i \leq m-1$: Similarly to Lemma 5.1 we have that for t obeying $t_i \leq t \leq t_{i+1}$, $w(t) \geq \frac{1}{(i+1)^\alpha}$. Thus, we get that $ABS \geq (t_{i+1} - t_i)\frac{1}{(i+1)^\alpha} \cdot R$ and thus for $1 \leq i \leq m-1$, $t_{i+1} - t_i \leq (i+1)^\alpha \cdot T_{on}$. Similarly, we can see that $ABS \geq (t_e - t_m)\frac{1}{(m+1)^\alpha}$ and thus $t_e - t_m \leq (m+1)^\alpha \cdot T_{on}$. Also $t_1 - t_s = T_{on}$, since the user transmits at rate R at this segment.

User Type	WFQ	APQ ($\alpha = 1$)	APQ ($\alpha = 2$)
Naive aggressive user	f	$1 + \log f$	2
Continuous naive aggressive user	f	1	$\frac{1}{f}$
Sophisticated aggressive user	f	$\sqrt{2 \cdot (f-1)} + 2$	$\sqrt[3]{3 \cdot (f-1)} + 2$

Table 1: The Vulnerability Factor for APQ and WFQ .

Since $(t_1 - t_s) + (t_e - t_m) + \sum_{i=1}^{m-1}(t_{i+1} - t_i) = t_e - ts = \Delta$ we get that $\sum_{i=1}^{m}(i+1)^{\alpha} \cdot T_{on} + T_{on} \geq \Delta$ which leads to $\sum_{i=1}^{m}(i+1)^{\alpha} \geq f - 1$.

For $\alpha = 1$ the above sum yields $\sum_{i=1}^{m}(i+1) = \frac{(m+2)\cdot m}{2} \geq f - 1$, hence $m \geq \sqrt{2 \cdot (f-1)} - 2$. The minimum amount of traffic that the user transmits in (t_s, t_e) is $(m+1) \cdot ABS$: One "chunk" of ABS in the interval (t_s, t_1) and m "chunks" in the interval $[t_1, t_m]$. Hence, $D_{out}^{APQ} \geq ABS \cdot (\sqrt{2 \cdot (f-1)} - 1)$.

For $\alpha = 2$ the above sum yields $\sum_{i=1}^{m}(i+1)^2 = \frac{(m+1)\cdot(m+2)\cdot(2m+3)}{6} - 1 \geq f - 1$, hence $m \geq \sqrt[3]{3 \cdot (f-1)} - 2$. Since, the minimum amount of traffic that the user transmits in (t_s, t_e) is $(m+1) \cdot ABS$ we get that $D_{out}^{APQ} \geq ABS \cdot (\sqrt[3]{3 \cdot (f-1)} - 1)$.

Corrolary 5 *Under APQ with $\alpha = 1$ or 2, a sophisticated user can transfer $\Theta(\sqrt[\alpha]{\alpha \cdot f} \cdot ABS)$ traffic units in an interval of length Δ. $\Theta(\sqrt[\alpha]{\alpha \cdot f} \cdot ABS)$ is also the maximum amount of traffic the user can transmit.*

The following corollary is based on a strategy where the user alternates between idling for one period of length Δ and then following the strategy described in Lemma 5.2 for the next period of length Δ:

Corrolary 6 *Consider intervals of duration $K\Delta$, $K > 1$: Under APQ with $\alpha = 1$ or 2 a sophisticated user can transfer at least $\frac{K}{2}(\sqrt[\alpha]{\alpha \cdot (f-1)} - 1 \cdot ABS)$ traffic units over an interval of length $K\Delta$ units.*

5.6 Summary of Analysis Results

In Table 1, we present a summary of the results, and show the vulnerability factor under the various scheduling pollices and for the various users. The assumptions are the same as those we use in the analysis of the sophisticated aggressive user, i.e., the amount of data that is allocated to a user with weight 1 is exactly R. In the example, which aims to portray Web applications, $T_{on} = 2$ and $\Delta = 30$ and hence $f = 15$. If $\alpha = 1$ and the user is naive aggressive, then he can send $\log f + 1$ more than the polite user, which in our example leads to less than

five and up to one for the continuous naive user. If $\alpha = 2$ and the user is a naive aggressive, then he can send twice as much as the polite user, and up to $\frac{1}{15}$ of the continuous naive user. We feel that the interesting and practically relevant case to evaluate the system performance is the case of the naive user and, especially, the continuous naive user.

6 Stochastic Environment: Simulation Results

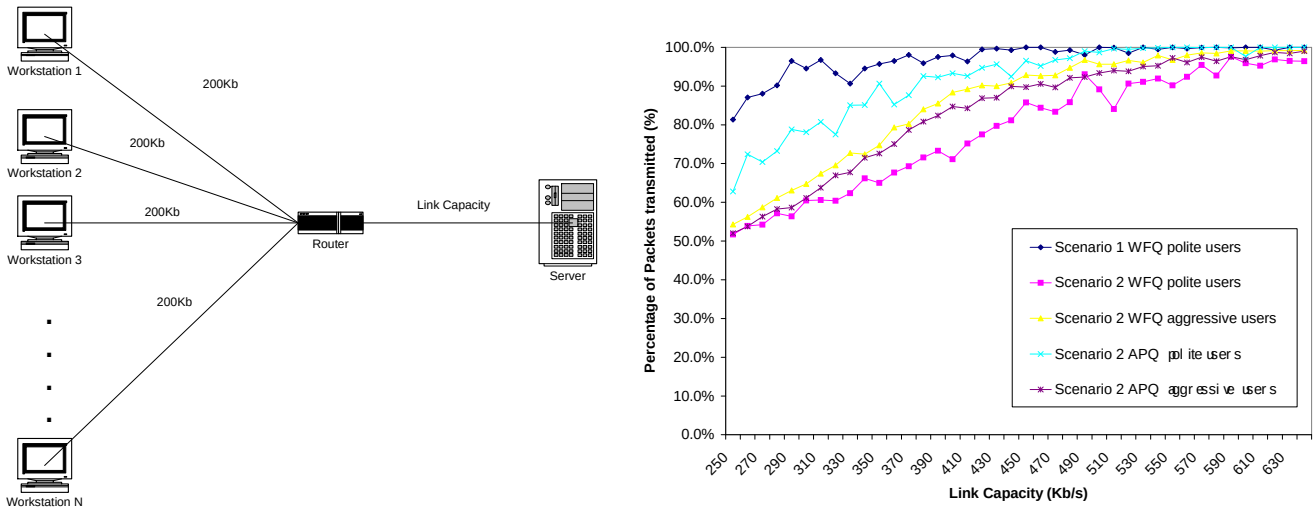
In this section we evaluate the behavior and efficiency of APQ by simulation results in a stochastic environment. We conduct the simulation using the NS2 network simulator [4] and use the WFQ implementation contributed by Paolo Losi [5] as a base line for developing the algorithm (APQ). Our simulation implementation wraps the WFQ code without modifying it, making it easy to implement on existing systems, as well as not jeopardizing the correctness of the WFQ implementation. The implementation collects the statistics every second of simulation time for each user, sums up the offered traffic of the user during a sliding window, and updates the weights (if required) accordingly.

The system simulated is depicted in Figure 2 (a) where N is the number of users in the network, and $link-capacity$ is the capacity of the Router - Server link. Each user uses a 200Kb link to the Router, and each user transmits in 200Kb rate ($R(t) = 200$Kb). We measure the percentage of packets transmitted by the system (namely the ratio between the offered traffic rate and the transmitted traffic rate), for each user. We consider two types of users: polite user and aggressive user. A polite user transmits bursty traffic in which $T_{on} = 2$ seconds (and the offered packet rate during this period is 200Kb/sec), and T_{off} is a random variable with uniform distribution lying between 10 to 60 seconds.[8] Aggressive users transmit at a constant rate of $R = 200Kb/sec$. The router maintains a private buffer for each of the users to store the user's packets. Packets that arrived to a full buffer are dropped. The duration of each simulation run was 300 seconds (simulation time). Each run was repeated several times, and the results were averaged.

We present two experiments. In the first experiment we examine the percentage of packets transmitted per user as a function of the link capacity (varying between 250 Kb/sec and 650 Kb/sec). The examination is done both for WFQ and APQ. Under this experiment we consider two scenarios. In *Scenario 1* there are $N = 12$ users in the system, all of them are polite users. In *Scenario 2* there are $N = 12$ users, where two of them are aggressive users and the rest are polite users.

The results of this experiment are depicted in Figure 2 (b). The results show that on

[8]Note that the ratio of PPR/APR is between 5 to 30 which correspond to our log analysis in Section 7



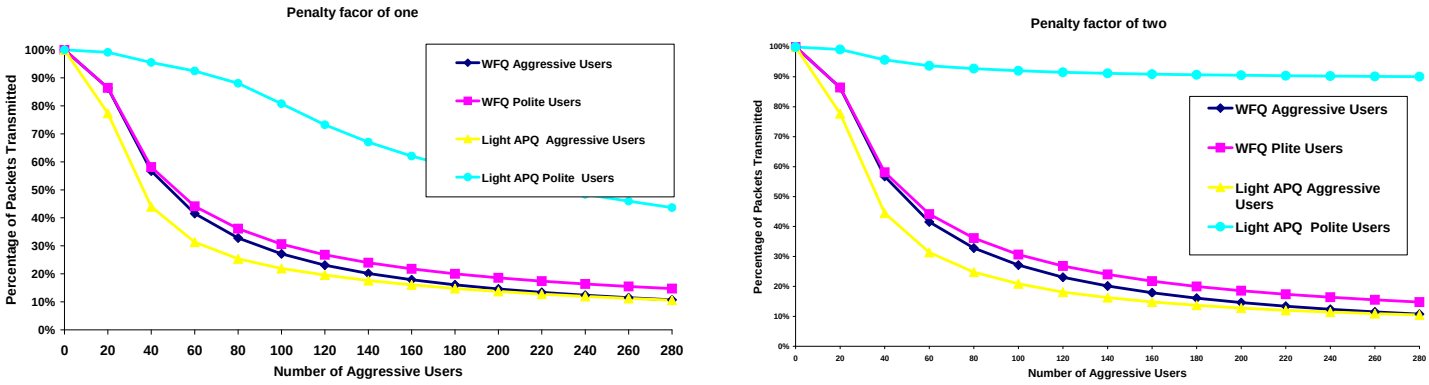
(a) (b)

Figure 2: (a) Test set up (b) Experiment 1 - Percentage of transmitted packets as function of Link Capacity

Scenario 1 the percentage of transmitted packets is very high in all capacity values. In Scenario 2, under WFQ, the percentage of transmitted packets of the polite users drops to 60% at capacity of 360Kb/sec. The link capacity needs to be roughly doubled in order to grant the polite users the same performance level measured in a fraction of packets transmitted as of Scenario 1. When APQ with penalty factor equalling one is used in Scenario 2 (that is, at the presence of aggressive users), the polite users experience the same, or even better, performance, than they experience in Scenario 1 (that is, when no aggressive users are present). In contrast, the aggressive users experience worse performance (lower percentage of transmitted packets) than they experience in Scenario 2 using WFQ.

In the second experiment (Figure 3), we examine the number of aggressive users a given network can handle without negatively affecting the polite users. We further examine how a large number of aggressive users affect the network and how well APQ handles them. The setup of the second experiment is the same as that of the first one, with a difference in the total number of users (polite and aggressive) which is fixed at $N = 300$ and the link-capacity set to $9000Kb/sec$.

In order to get intuition about the effect of our algorithm we did the setup with a lighter version of APQ. In the light version of APQ the algorithm decreases the weight of a user, only once, after the first window. The performance of the lighter version of APQ is a lower bound of the performance of APQ.



(a) (b)

Figure 3: Fraction of transmitted packets as a function of Number of Aggressive users using Light APQ with (a) penalty factor of one (b) penalty factor of two

The figure 3 (a) shows that under WFQ with 100 aggressive users, only 30% of the traffic of polite users is transmitted. While with light APQ with penalty factor of one, with 100 aggressive users, 80% of the traffic of polite users is transmitted. The figure 3 (b) shows that with penalty factor of two, 90% of the traffic of polite users is transmitted by the system, even if the majority of users (above 90%) are aggressive. This result corresponds to our analytic results (in Section 5). The burst factor, f, in our example is between 5 to 30. Without WFQ the aggressive user would transmit in each window 5 to 30 times the traffic of a regular user. With the penalty factor of two, the aggressive user can send only two times the traffic of a regular user in the first window, and only $\frac{1}{5}$ to $\frac{1}{30}$ of traffic of a regular user in the consecutive windows after the first window. In this case the aggressive users create congestion only in the first window, and thereafter the additional load is marginal even when compared to regular users.

7 Practical Considerations

In this section we demonstrate how the APQ mechanism is applied to a real life scenario of a typical Web server and estimate its impact. We take as a case study a web server of our University and analyze a 24 hours log that recorded the requests received by the server[9]. The bursty nature of http is clearly shown in Figure 4, where we plot a scatter of requests of 500 users over a period of 5 minutes (300 sec).

[9]The number of requests can yield a good estimate of the request bandwidth if one wants to apply the APQ mechanism on bandwidth

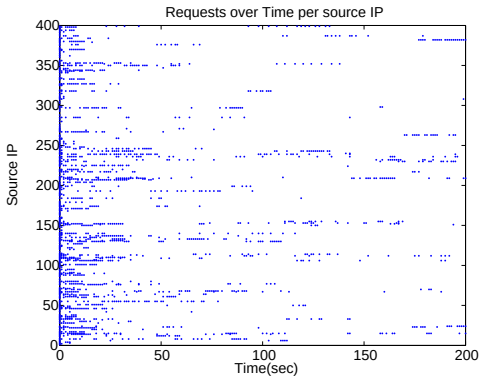


Figure 4: Scatter of the request of 500 users over period of 5 minutes

APQ requires configuring three parameters: The penalty factor (α), the window size (Δ) and the Burst size (ABS). We choose $\alpha = 1$, i.e., a moderate penalty factor. The window size, Δ, should be chosen based on the characteristics of the application. In case of http a value between 30 sec to 60 sec seems reasonable[10]. We have chosen $\Delta = 30$ for this analysis. The most important parameter is ABS, whose value is sensitive to the application type. In order to estimate ABS, we calculate the total number of requests per window for all of the source addresses, and we chose ABS to be the 95th percentile of these evaluations, which in our case is equal to 136 requests per window of 30 seconds.

In order to understand the APQ impact, we here give some sense of the aggressive user's capabilities with and without APQ. Recall that the load placed on the system by an aggressive user is equivalent roughly to that placed by $PRR/ARR \sim PPR/APR$ polite users[11]. As a reasonable assumption, we assume that a user's momentary rate (rate over a 1 second interval) can reach or exceed the maximum momentary rate observed in the data. The maximal momentary rate equals 140 requests per second, after cleaning some outliers. Hence, roughly, in the case of WFQ, an aggressive user can send $140 * 30$ requests during a window and in the case of APQ this value is bound by ABS equaling 136 (in the case of continuous aggressive user). Hence a simple aggressive user can inflict about thirty-fold higher on the system load than that of a normal user.

In some cases we may want to have a precise ABS that is sensitive to the specific use of the application, i.e., in the specific characteristics of the server (long or short burst size). In this case "peace-time learning" is required to evaluate the ABS. "Peace time learning", is a process

[10] A further analysis of how to choose Δ is out of the scope of this paper

[11] Peak Request Rate divided by Average Request Rate; see Section 3

whose objective is to learn the characteristics of regular legitimate traffic, and is a common technique used in traffic anomaly detection devices as well as in denial of service mitigation devices [1, 3].

8 Concluding Remarks

Our study revealed that fairness in todays networks can be very fragile. While WFQ provides fairness to users in an environment where all flows send data at roughly a constant data rate, it has a fairness problem when dealing with bursty users, whose average rate differs significantly from their peak rate. The fairness problem shows up when aggressive users attempt to exploit the network by continuously placing demands on the network. In such cases the polite, non-aggressive users, may receive very poor QoS since their network fair share is not delivered. We propose the APQ mechanism to solve this problem by taking into account the amount of traffic sent by the flow in recent history (recent window), and dynamically modifying the WFQ weight as a function of this amount of traffic.

References

[1] Cisco ddos protection solution. www.cisco.com/application/pdf/en/us/guest/netsol/ns615/c654/cdccont_0900aecd80322dae.pdf.

[2] Fair queuing combats ddos. *NANOG mailiing list 2005.* http://www.cctec.com/maillists/nanog/historical/0002/msg00298.html.

[3] Mazu networks. http://www.mazunetworks.com/solutions/.

[4] NS2 network simulator. http://www.isi.edu/nsnam/ns/.

[5] WFQ implementation code donated by paolo losi. http://www.cclinf.polito.it/ s77950/.

[6] Hanoch Levy, Bremler-Barr and Nir Halachmi. Aggressiveness protective fair queueing for bursty applications (short abstract). In *IWQoS*, 2006.

[7] L. Guo and I. Matta. The War Between Mice and Elephants IN *IEEE ICNP*, 2001.

[8] L. Le, J. Aikat, K. Jeffay, and F. Smith. Differential Congestion Notification: Taming the Elephants In *IEEE ICNP*, 2004.

[9] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth flows at the congested router In *IEEE ICNP*, 2001.

[10] H. Zhang C. Bennett. WF2Q: worst-case fair weighted fair queueing. *Proceedings of IEEE INFOCOM*, pages 120–128, 1996.

[11] Mina Guirguis, Azer Bestavros, Ibrahim Matta, and Yuting Zhang. Reduction of Quality (RoQ) attacks on internet end-systems. In *Proceedings of IEEE INFOCOM*, 2005.

[12] D. Makrakis J. Gallardo. Dynamic predictive weighted fair queueing for differentiated services. *IEEE International Conference on Communications*, pages 2380–2384, 2001.

[13] R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott. Controlling high bandwidth aggregates in the network, 2001.

[14] J. Shin, J. Kim, D. Lee, and C. Kuo. Adaptive packet forwarding for relative differentiated service and categorized packet video. *IEEE International Conference on Communications'*, pages 763–767, 2001.

[15] Sally Floyd, Van Jacobson. Random Early Detection Gateways for Congestion Avoidance (1993). *IEEE/ACM Transactions on Networking* vol 1(4) pages, 397-413,1993 .

[16] R. Thomas, B. Mark, T. Johnson, and J. Croall. Netbouncer: Client legitimacy-based high-preformance ddosfiltering. *DISCEX*, page 14, 2003.

[17] David K. Y. Yau, John C. S. Lui, Feng Liang, and Yeung Yam. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. on Networking*, pages 29–42, 2005.

[18] A. Parekh, R. Gallager. A generalized processor sharing approach to flow control - the single node case. *IEEE/ACM Trans. on Networking*, pages 344-357, 1(3), 1993