Designing Traffic Profiles for Bursty Internet Traffic

Xiaowei Yang
MIT Laboratory for Computer Science
Cambridge, MA 02139
yxw@lcs.mit.edu

Abstract

This paper proposes a new class of traffic profiles that is better suited for metering bursty Internet traffic streams than the traditional token bucket profile. A good traffic profile should satisfy two criteria: first, it should consider packets from a conforming traffic stream as in-profile with high probability to ensure a strong QoS guarantee; second, it should limit the network resources consumed by a non-conforming traffic stream to no more than that consumed by a conforming stream. We model a bursty Internet traffic stream as an ON/OFF stream, where both the ON-period and the OFF-period have a heavy-tailed distribution. Our study shows that the heavy-tailed distribution leads to an excessive randomness in the long-term session rate distribution. Therefore, it is inherently difficult for any profile that limits the long-term average session rate to give a strong QoS guarantee for the conforming traffic streams. Our simulation demonstrates that a token bucket profile that couples the average rate control and the burst size control has a weak QoS guarantee. Based on this result, we propose a new class of traffic profiles that decouples the long term average rate control from the burst size control. Compared to a token bucket profile, this profile improves the level of QoS for a conforming traffic stream, yet limits the "effective bandwidth" consumed by a non-conforming traffic stream.

1. Introduction

This paper discusses the design of traffic profiles for bursty Internet traffic. We propose a new class of traffic profiles that performs better than the traditional token bucket profile. In the Diffserv [6] architecture, a traffic profile defines the rules for a meter to decide whether a packet from a traffic stream is in-profile or out-of-profile. The packet may be subject to different conditioning actions. When there is a Service Level Agreement (SLA) present, the essential goal of a profile is not to smooth bursty traffic. Instead, it is to check whether a traffic stream conforms to the SLA. A good traffic profile needs to satisfy two criteria. First, from the perspective of providing a QoS guarantee, it is desirable to have a high probability guarantee that packets from a conforming stream will be identified as in-profile. A conforming stream is a traffic stream whose statistics are allowed by a traffic profile. Second, from the perspective of policing against misbehavior, a traffic profile should limit the "effective bandwidth" [14] consumed by the non-conforming stream to be no more than that consumed by a conforming stream. Thus the misbehavior of the non-conforming stream will not degrade the level of service for a conforming stream. An example of a non-conforming stream is one that has a much higher expected transmission rate than that allowed by a traffic profile.

It is desirable to have a range of traffic profiles for different classes of traffic, and for differentiation within a class. For example, traffic streams generated by constant bit rate video applications have quite different statistics than those generated by web browsing sessions, thus require different profiles to meet their QoS requirements. It is worth noting that the type of traffic we consider in this paper is most often generated by document transfers, such as web browsing. Traditionally, the QoS requirement for such traffic (a.k.a. elastic traffic) is less obvious than that for the real-time traffic. In this paper, the QoS requirement we consider is *transparency* [20]. If a burst is sent at the specified peak rate, we say the transfer is *transparent*. If a burst is not sent at the specified peak rate, a user will notice delay, spend time waiting, and become dissatisfied. Thus, we consider *transparency* as a valid QoS requirement for such interactive applications. We admit that the ultimate QoS measurement is user satisfaction, which can not be completely determined by the measurement of network performance. For an interactive traffic stream, a user may value some bursts more than others. Since we do not have a precise method to quantify user satisfaction, as an approximation, we assume that the higher the percentage of bursts that are sent transparently , and the more the bytes that are sent transparently, the higher the user's satisfaction is. There have been related studies on traffic profiling algorithms, such as different window policing mechanisms [18], multiple token banks [5], and the *fractal leaky bucket* [13] algorithm.

In this paper, we model a bursty Internet traffic stream as an ON/OFF stream, where both the ON-periodand the OFF-periodhave heavy-tailed distributions. This model is a simplified derivation from recent results of traffic measurement and modeling [17, 10, 15]. According to Barford and Crovella's work [3], users' Web browsing behaviors follow the ON/OFF pattern, where ON-periodhas a heavy-tailed distribution attributed to the heavy-tailed file size distribution.

Our simulation and analysis show that there is no "natural" average session rate for a single traffic stream. Thus, it is unlikely for any profile that limits the long-term average session rate to give a strong QoS guarantee. Our simulation also demonstrates that a token bucket profile that couples the average rate control and the burst size control has a weak QoS guarantee. Based on this result, we propose a new class of traffic profiles that decouples the long term average rate control from the burst size control. Compared to a token bucket profile, this profile improves the level of QoS for a conforming stream, yet effectively limits the "effective bandwidth" consumed by a non-conforming traffic stream.

2. Traffic Model, Simulation Method and Notations

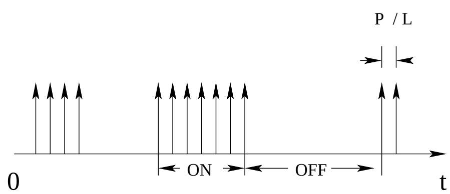

Figure 1: The ON/OFF Traffic Model.

In this section, we describe the traffic model, the simulation method and some notation. In our simulation, a bursty traffic stream from a single source is modeled as an ON/OFF traffic stream. The ON-periodmodels a single flow, such as the transfer of a single web page, and the OFF-periodmodels users' thinking time. ON-periods and OFF-periods are strictly alternating. The stream models traffic generated during a session, i.e., a busy period of user activities, such as web browsing for an hour. For bursty Internet traffic, both ON-periods and OFF-periods are modeled by Pareto distributions. In an ON-period, packet arrivals are spaced out by P/L, where P is the packet size and L is the peak rate. Figure 1 depicts the traffic model. In our simulation, the traffic stream is passed through a meter that decides whether a packet is in-profile or out-of-profile according to a traffic profile. As we are interested in the *transparency* QoS requirement, out-of-profile traffic is shaped, instead of dropped or marked. Thus, the transfer time of a burst can be used to measure performance.

Traditional traffic models assume that the burst size is exponentially distributed. To better understand the difference between these two models, we simulate both the Exponential distribution and the Pareto distribution in some experiments, and compare the results. If a Pareto distribution has mean μ and location parameter m, the corresponding exponential distribution is defined as $P\{X < x\} = 1 - e^{\frac{x-m}{\mu-m}}, x > m$.

In this paper, a token bucket profile with a token rate r bytes/s and a token depth b bytes is denoted by the pair (r, b). We always assume our simulation starts at time 0. And $A(t)$ denotes the cumulative arrivals of a traffic stream up to time t. Simulation results shown are obtained using the parameters in Table 1 unless otherwise specified. These parameters for the Pareto distribution are chosen based on recent research results on traffic measurement and modeling [3, 15].

3. Average Session Rate of a Bursty Source

For a heavy-tailed ON-perioddistribution, there is no "natural" burst size [17], to which the token bucket size can be set. In this section, we look at the distribution of the average session rate of a traffic stream at different time scales via simulation. A session is simulated by a single run with some length T. Each distribution curve is sampled from 10^6 runs. Figure 2 and Figure 3 show the cumulative distribution function of the normalized session rate of a Pareto distribution and an Exponential distribution over multiple time scales. A normalized session rate is computed by dividing the average rate in a simulated session by the expected rate of the ON/OFF stream. The tail is shortened quickly for the Exponential stream, which shows that the throughput of an exponential stream averaging over an hour duration has a small variance and there exists a "natural" session rate to which the token rate can be set. In contrast, the tail shape for the Pareto distribution does not change much for simulations ranging from ten seconds to two hours, which suggests that even averaging over a duration as long as two hours, the session rate of a heavy-tailed ON/OFF stream has a large variance. Hence it has non-negligible probabilities of reaching quite large or quite small values. The observed maximum throughput is only limited by the peak rate.

Intuitively, the simulation results can be explained by the Central Limit Theorem (CLT) and the corresponding limit theorem for random variables with infinite variance [12, 9]. Let X_i be the size of burst i, t_{ON} be the mean burst transfer time, and t_{OFF} be the mean off time. The mean arrival time for one ON/OFF cycle will be $t_{period} = t_{ON} + t_{OFF}$. In a session of length T, the average number of bursts arrived can be approximated as $N = T/t_{period}$. Therefore, the average session rate r_T is roughly:

$$r_T = \frac{\sum_{i=1} N X_i}{T} \tag{1}$$

Let μ_{r_T} denote the mean of r_T. For large N, when X_i is an exponential distribution, $(r_T - \mu_{r_T})N^{-1/2}$ converges to a Normal distribution, whose tail probability decreases exponentially fast. When X_i is a Pareto distribution, $(r_T - \mu_{r_T})N^{-1+1/\alpha}$ converges to a levy-stable distribution, which has a power-law tail with the same shape parameter α as that of X_i. Our analysis roughly explained the results in Figure 2 and Figure 3. Notice that r_T is actually limited by the peak rate L and therefore it cannot be simply expressed as the sum of N random variables as in Equation 1. The analysis is not accurate. But, since r_T has finite range, we do not require many samples in our simulation to derive the distribution. Otherwise, as pointed out by Crovella and Lipsky [9], it requires 10^{22} samples to get two digits accuracy for estimating the population mean of a Pareto variable with a shape parameter $\alpha = 1.1$ and infinite range.

This slow-convergence property poses a problem for traffic profiling. A traffic profile has two functions [4]. First, it is a SLA between a user and a network operator. It helps both the user and the network to decide whether a packet is in-profile or out-profile. Second, it provides the network operator with relevant information for resource management. However, a bursty traffic stream generated by a particular application is a random process. We hope we can predicate its behavior from previous observations. For an Exponential ON-OFF traffic stream, as we have seen, it is highly possible to predicate the stream's long term behavior. However, for a heavy-tailed ON-OFF stream, even we assume the application always generates traffic conforming to a known distribution, the behavior of an individual session is still beyond predictability. For any profile that limits the average session rate to r, in a duration t, the number of arrived tokens $A(t)$ is limited by rt. Both our simulation and analysis conclude

Table 1: **Simulation Parameters**

Period	*Shape* (α)	*Location* (β)	*Distribution*	*Mean*
ON	1.05	1024 Bytes	$P\{X < x\} = 1 - (\frac{1024}{x})^{1.05}, x > 1024$	14026 Bytes
OFF	1.4	1 Second	$P\{X < x\} = 1 - (\frac{1}{x})^{1.4}, x > 1$	3.4939 Seconds
Packet Length		1024 Bytes		
Peak Rate		1.5Mbps		

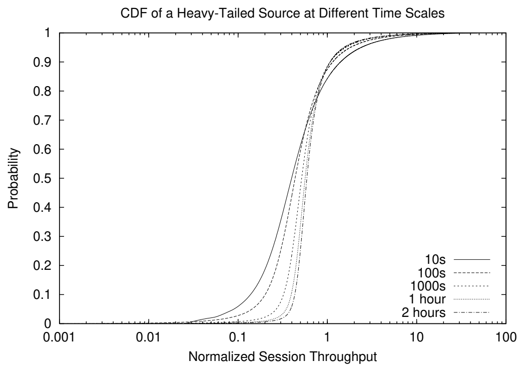

Figure 2: The convergence of a Pareto ON/OFF Traffic Stream. The different curves are for session lengths of 10s, 100s, 1000s, 1 hour and 2 hours.

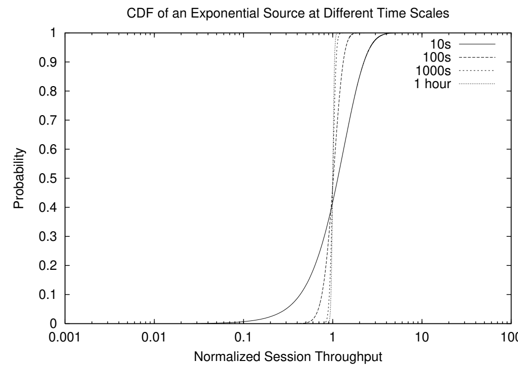

Figure 3: The convergence of an Exponential ON/OFF Traffic Stream. The different curves are for session lengths of 10s, 100s, 1000s, 1 hour and 2 hours.

that for any time scale t, there is a non-negligible probability that an application may generate near-peak traffic for the entire duration t, which indicates that unless $A(t)$ is close to Lt (recall that L is the peak rate), with non-negligible probability, an application will generate out-of-profile traffic, and the QoS may not be guaranteed. For example, even if a user's browsing behavior can be perfectly modeled by a heavy tailed ON/OFF process, within a browsing session, he is likely to download a large file with non-trivial probability. Thus if he has negotiated a profile limiting his average session rate to r, with r equaling to the expected session rate, according to Figure 2, for a one-hour session, with probability one out of ten, his traffic demand will exceed the limit allowed by the profile, i.e., rt, and thus be marked as out-of-profile. As the probability of getting a large value decreases polynomially, increasing the token rate r from the expected rate only reduces the probability of marking his traffic out-of-profile polynomially.

4. Performance of Token Bucket Profiles

In this section, we look at the performance of a token bucket profile. A token bucket profile (r, b) limits the maximum burst size of a traffic stream to b, as well as the average rate of a session with length t to $r + b/t$, thus enhances the predicability of the output traffic stream. It has no long term memory. When the bucket is full, extra tokens are discarded, and the good behavior of a stream is forgotten. The decision made on a packet depends only on the short term history of a traffic stream, i.e., whether the previous burst has consumed all the tokens, but not on the long term history.

4.1 Performance of a Token Bucket Profile

We simulate the case where an ON/OFF traffic stream is policed against a token bucket profile. When the transfer of a burst is finished, an OFFtime follows before a new burst arrives. The maximum bucket size is set at ten times the mean burst size. At the beginning of each simulation, the bucket is full. For each run, we record the time it takes for each burst to be transmitted. The time is normalized using the peak-rate transfer time of each burst. We then draw a distribution of the normalized transfer time, both in the number of bursts and in the number of bytes. Each distribution is sampled from 10,000 runs. Each simulation runs for the duration which it takes to transmit the amount of traffic that would be sent in a one-hour session with no traffic profile present. Figure 4 and Figure 5 show the performance of a token bucket profile for shaping a Pareto ON/OFF source and an Exponential ON/OFF source. The token rate varies from the expected session rate to twice the expected session rate.

As seen from the figures, when the token rate is set to the expected session rate, for a Pareto ON/OFF stream, more than 98% of bursts can be sent without distortion. However, those bursts only consist of 45% of the total bytes sent. For an Exponential ON/OFF stream, more than 90% of files and 80% of bytes can be sent without distortion. When the token rate is increased to twice the average session rate, the performance for the Exponential stream is improved much more significantly than that for the Pareto stream. This result can be predicted from the session rate distribution, as a quite high session rate may happen for a Pareto ON/OFF stream with non-negligible probability. The session rate may be very close to the peak rate. Hence increasing the token rate does not increase the performance significantly for the Pareto ON/OFF stream.

4.2 How to Improve QoS Guarantees

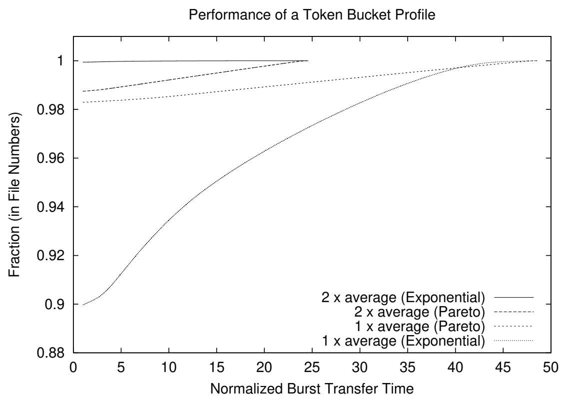

Figure 4: Fraction counted by the number of bursts for an ON/OFF traffic stream. When the token rate doubles, most bursts of the Exponential stream are sent at the peak rate.

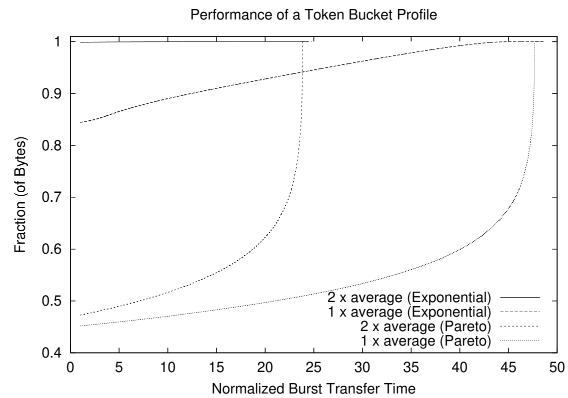

Figure 5: Fraction counted by bytes. When the token rate doubles, most bytes of the Exponential stream are sent at the peak rate. For the heavy-tailed traffic stream, even more than 98% of the bursts are sent at the peak rate, they constitute merely about 45% of the bytes.

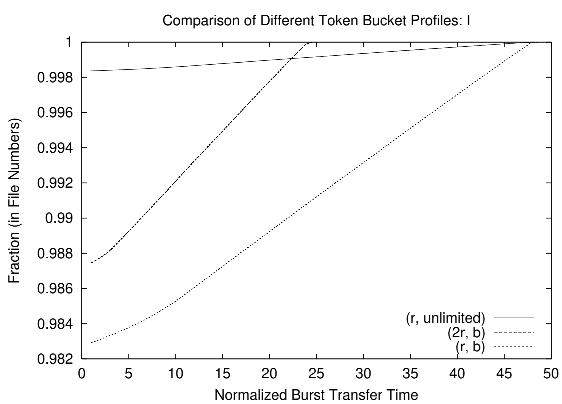

Figure 6: Fraction counted by the number of bursts.

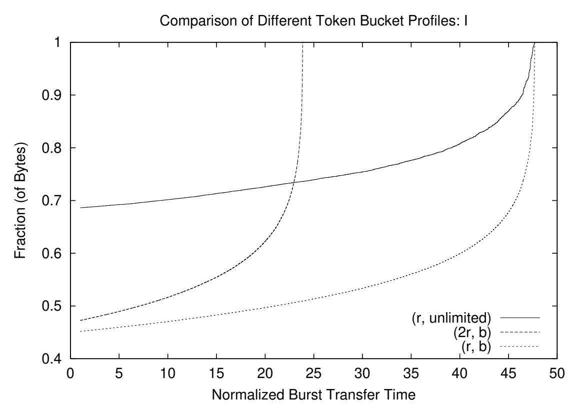

Figure 7: Fraction counted by bytes.

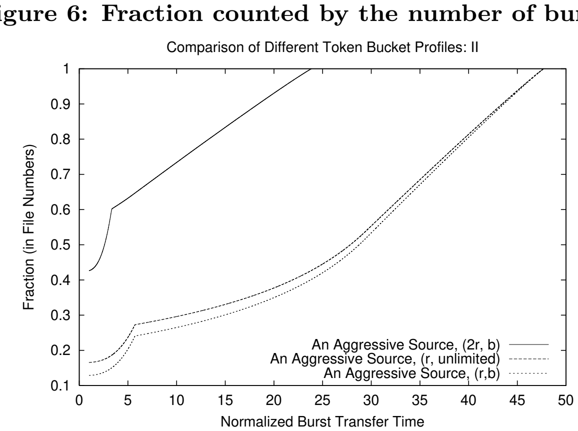

Figure 8: Fraction counted by the number of bursts.

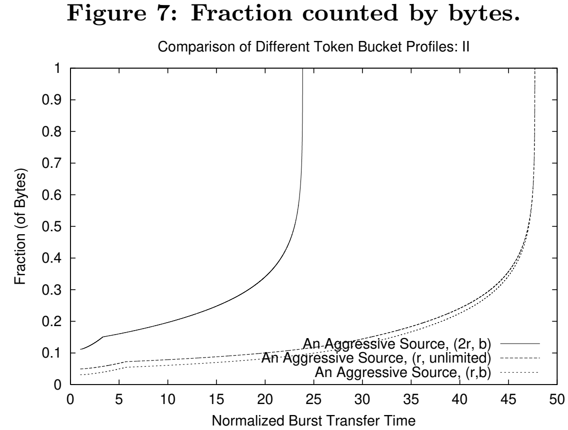

Figure 9: Fraction counted by bytes.

A token bucket profile limits both the burst size and the average session rate. We can either increase the limit on the average session rate or increase the limit on the maximum burst size to improve QoS guarantees. To formulate ideas on how to improve the performance of a token bucket profile for a heavy-tailed ON/OFF stream, we compare the effectiveness of increasing the bucket size and increasing the token rate from three aspects. First, we compare how effective they are in improving the QoS guarantees for a single conforming traffic stream. By "conforming", we mean that the traffic stream is generated from the pre-negotiated distribution. Second, we compare how effective they are in limiting the QoS guarantees for a single aggressive stream. By "aggressive", we mean that the traffic stream consumes more bandwidth than pre-negotiated. Thirdly, we compare how effective they are in controlling aggregated aggressive streams. An effective profile should limit the peak bandwidth of the multiplexed aggressive streams to be no more than the peak bandwidth of the multiplexed conforming streams without traffic profiles. Thus, in the worse case, when the majority of the users are misbehaving, a network provider can still provide QoS for the in-profile traffic, given that he has provisioned his network according to the peak bandwidth of the multiplexed conforming streams.

To conduct the first comparison, we run simulations with a token bucket profile that allows unbounded bucket size. And the initial number of tokens in the bucket is set at ten times the average burst size. The result from this profile is compared against the normal token bucket profile (r, b) and $(2r, b)$, respectively. Figure 6 and Figure 7 show the performance comparison when the stream's expected rate is r. The token bucket profile $(2r, b)$ shortens the token-rate burst transfer time by half, but it does not significantly increase the number of bursts or bytes that are sent at the peak rate. In contrast, the profile $(r, unlimited)$ reduces the non-peak rate transfered files from 1.7% to less than 0.2% and the non-peak rate transfered bytes from 55% to 33%.

For the second comparison, we set the minimum burst size of an aggressive stream at ten times that of a conforming stream. Such an aggressive stream is policed by profiles (r, b), $(2r, b)$ and $(r, unlimited)$, where r is the expected rate of a conforming stream and b is ten times the mean burst size of a conforming stream. The results are shown in Figure 8 and Figure 9. As can be seen, the profile $(2r, b)$ is not as effective as the profile $(r, unlimited)$. The profile $(2r, b)$ allows more bursts and bytes to be sent without distortion or with less distortion and can not limit the long term session rate to be r. Thus, it is not as effective as the profile $(r, unlimited)$ at discriminating conforming and aggressive streams.

For the third comparison, we compare the normalized peak data rate of aggregated aggressive traffic streams when shaped by the three profiles (r, b), $(2r, b)$ and $(r, unlimited)$. A normalized peak data rate is computed as P/nr, where P is the measured average peak rate of aggregated streams, r is the expected data rate of a single conforming traffic stream, and n is the number of streams. The aggregated peak rate P_r is a random variable, with an upper bound $n \times L$, where L is the peak data rate of a single traffic stream. In our simulation, all streams are homogeneous. For each n, we ran $10000/n$ simulations. Each of them simulates an 8-hour trace. We record the peak arrival rate seen in $5ms$ intervals in the trace of each run. The length of the interval is chosen because the spacing between packets from a single source is about $5ms$ in our simulation. We then compute the average peak rate P and the standard deviation of P over $10000/n$ runs. As we do not know the distribution of P_r and each simulation is run for a long time, we use P as a representative value such that with a high probability, the peak rate of the aggregated traffic streams P_r is less than P.

Figure 10 shows the simulation results, together with the normalized data rate for conforming traffic streams when there is no profile and no shaping. As shown in Figure 10, the normalized data rate degenerates as the number of multiplexed sources increases, indicating a statistical multiplexing gain for bursty traffic. The normalized peak data rates of the aggregated aggressive streams shaped by the profile $(r, unlimited)$ and (r, b) are quite close to that of the aggregated conforming streams without shaping. For the profile $(2r, b)$, the aggregated normalized peak data rate exceeds that of conforming streams, which means that the profile is not effective in limiting the effective bandwidth of aggressive streams. A network will become congested when the peak of the aggregated traffic exceeds the bandwidth provisioned according to the conforming sources without profiles, and thus can not guarantee the QoS for the in-profile traffic.

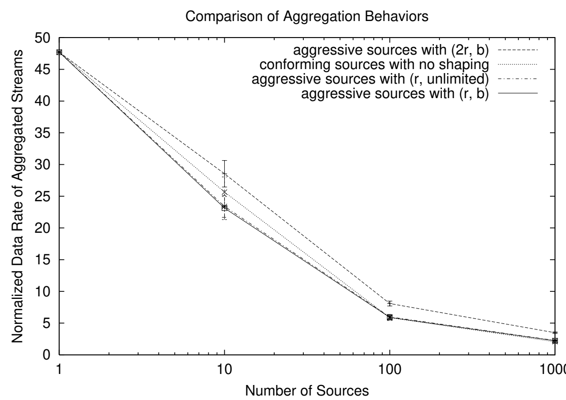

Figure 10: The normalized data rate of aggregated traffic when applying no profile to conforming traffic streams, profile (r, b), $(2r, b)$ and $(r, unlimited)$ to aggressive traffic streams.

5. A New Traffic Profile: the 2Bucket Profile

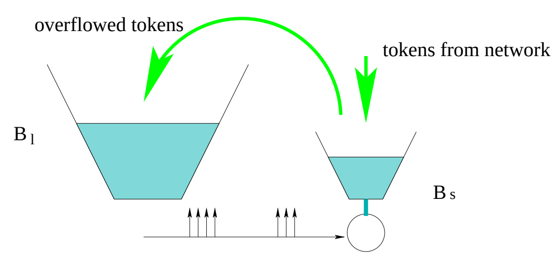

Figure 13: The 2Bucket Profile

From the above experiments, we conclude that to effectively improve QoS without compromising discrimination against misbehavior, it is better to increase the token bucket size than to increase the token rate. However, the $(r, unlimited)$ profile offers no control over burst size. A malicious user may keep silent for almost the entire course of a session and then burst out all his accumulated tokens. This observation leads to our design of a new class of traffic profile: the 2Bucket profile. A visualization of the profile is shown in Figure 13.

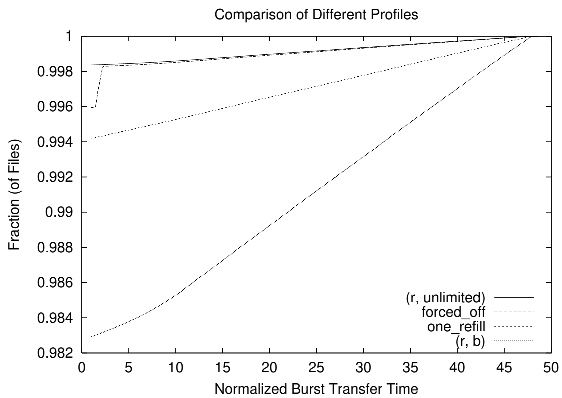

Figure 11: Fraction counted by the number of bursts.

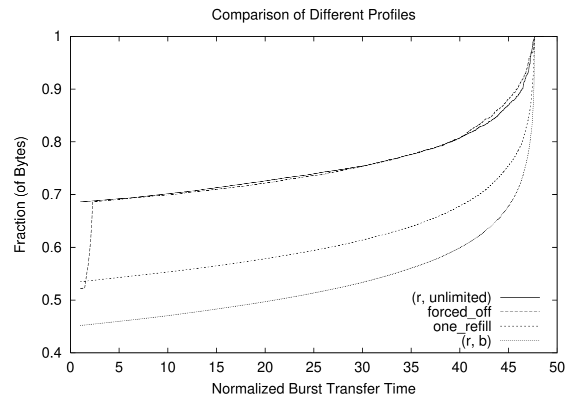

Figure 12: Fraction counted by bytes.

One bucket B_s that has a maximum capacity b is used to control the burst size. The other bucket B_l, which has an unlimited capacity, is used to record the long term history. Tokens arriving from the network are stored into B_s first. If B_s is overflowed, tokens are dumped into B_l. When a packet arrives, if B_s has enough tokens, the packet is transmitted immediately. If B_s is empty and B_l has extra tokens, a refill process may be triggered. This process is triggered according to a refill policy. The choice of refill policies is a key design parameter of the 2Bucket profile. In a refill, tokens are taken from B_l to fill B_s. After B_s is filled with enough tokens, the packet may be sent. If both B_s and B_l are empty, or B_s is empty and a new refill process is not allowed, the packet has to wait for tokens to come from the network. A refill policy decides when and how tokens are refilled from B_l into B_s. We use the terms B_s and B_l to refer to both the available tokens in the two buckets and the buckets themselves, as the meanings will be clear from context. Recall that $A(t)$ is the cumulative arrivals from a stream in a time period t. This two-bucket profile ensures that as long as $A(t) < rt + b$, $B_l + B_s > 0$.

The behavior of the 2Bucket profile is quite different when different refill policies are deployed. A refill policy determines the burstiness allowed by the profile. We will describe two possible policies and analyze their performance. It is also possible to design other refill policies, depending on the type of controls we need. The first policy is called "*forced_off*". For this policy, a refill process fills B_s with $\min(b, B_l)$ tokens at the peak rate. Two consecutive refills must be separated by at least the time interval *forced_off*. This policy will break a long burst into several small bursts, as long as the long term behavior of a traffic stream is good, i.e., $B_l + B_s$ is not empty. If the long burst is a document transfer, this break-down will increase the transfer time. Depending on the values of b and *forced_off*, the transfer time can still be much less than the token-rate transfer time. If the burst is actually a video stream (for example, a user asks for a cheap bursty-traffic profile but uses it to watch on-line video), this break-down will give intolerable performance. Thus, it can discourage the misuse of a profile.

We call the second policy *one_refill*. For this policy, refill is more strict. Only one refill of $\min(b, B_l)$ tokens is allowed in any ONperiod. It limits the maximum burst size to be at most $2 \times b$. If a stream's long term behavior is good, each ON-periodis guaranteed a refill. In contrast, for a token-bucket profile, if the previous burst exhausts all available tokens in the bucket, the next burst has to wait for more tokens.

Performance of this new profile with two policies is shown in Figure 11 and Figure 12, along with the corresponding token-bucket profiles (r, b) and $(r, unlimited)$. In our simulation, for the *forced_off* policy, the parameter *forced_off* is set to the location parameter of OFF-perioddistribution, *min_off*. For the *one_refill* policy, if we have not seen a packet from a source for a time period *min_off*, we conclude one burst has ended and the stream is eligible for a new refill. The profile $(r, unlimited)$ is actually a special case of the 2Bucket profile. When the parameter *forced_off* is set to zero, the 2Bucket profile behaves exactly the same as $(r, unlimited)$.

Notice that there is a sharp knee in the curve for *forced_off*, and the performance of *forced_off* and $(r, unlimited)$ are almost the same after the knee. When examining the raw transfer time recorded in the trace file, we found that the bursts after the knee were sent when B_l ran out of tokens. Therefore, the rest of the bursts can only be sent at the token rate, despite the refill policies. When B_l runs out of tokens, the long term behavior of a stream reaches the limitation imposed by the token rate. Clearly neither *forced_off* nor $(r, unlimited)$ can improve the performance in this case. The steep region of the curve consists of bursts that are transmitted when B_l has extra tokens. In the $(r, unlimited)$ case, bursts are sent at the peak rate immediately. In the case of *forced_off*, a *forced_off* token-rate period is forced between two bursts. The transfer time is slightly lengthened under this policy. Compared to the token bucket profile, both *forced_off* and *one_refill* improve performance.

6. Other Possible Improvements

6.1 Fundamental Issue

The fundamental issue of designing a traffic profile for bursty traffic is to find a good descriptor with a few parameters to capture the statistical behaviors of a traffic stream. Our proposed 2Bucket profile keeps track of the long term behavior of a traffic stream and relaxes the constraint on the maximum burst size if a stream has a good history. As the

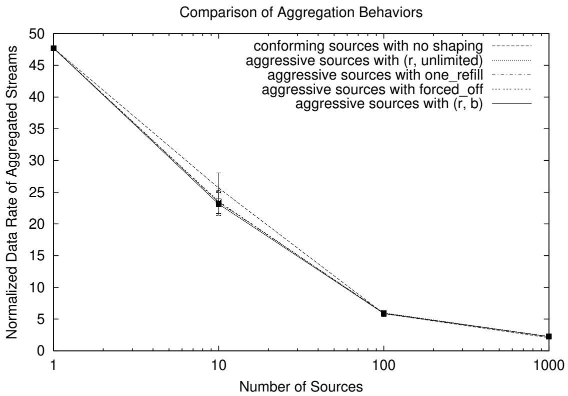

Figure 14: The normalized data rate of aggregated traffic with profile $(r, unlimited)$, *one_refill*, *forced_off* and (r, b) applied to aggressive traffic streams, and the normalized peak data rate for conforming traffic streams when there is no profile and no shaping. The performance of each profile is similar.

burst size distribution for a heavy-tailed $ON-OFF$ source has an infinite variance, this relaxation effectively improve the performance. However, this profile still limits a session's long term average rate, which is highly variable in the case of bursty traffic. This constraint can not be deterministically relaxed. Otherwise, a profile cannot limit misbehaviors of aggressive streams, as shown in Figure 10. Here we discuss two possible future improvements.

6.2 A Probabilistic Profile

The profiles we have discussed so far use a deterministic algorithm to decide whether a packet is in-profile or out-of-profile. Thus, the same measure will be applied to all sessions presumedly drawn from the same distribution. Another possible approach is to probabilistically decide whether a packet is in-profile or out-of-profile based on a stream's long term history. When a user negotiates a profile with a network, the user describes the statistical characteristics of his traffic stream to the network. For example, a Pareto ON-OFF stream may let the network know the parameters in Table 1. Suppose at time point t after the profile is set up, a stream has sent a bytes. When a new packet with length l arrives, based on the parameters a user declares, the network can compute the probability $p = P\{A(t) > a + l\}$. If $a + l > rt$, both B_s and B_l will be empty. Then with probability monotonic in p, the network sends the packet at the peak rate. Otherwise, the packet has to wait for tokens.

This probabilistic profile imposes no hard limit on how much a stream can send. Instead, it probabilistically relaxes the limitation. A user that sends more will have a lower probability to send future packets at peak rate after it exceeds the average rate limit. In the current profiles, this probability is zero. Thus, this probabilistic profile may give better QoS guarantees than deterministic profiles.

However, if the out-of-profile traffic will be offered Best-Efforts service instead of being shaped, the need for this relaxation will be alleviated. Thus, how well the proposed probabilistic profile performs and whether it is worth the efforts need further study.

6.3 Renegotiate New Profiles

In our study, we assume a profile is set up for a session, which usually lasts for an hourly time scale [grammar]. The profile remains unchanged during the session. However, if we assume a stream can dynamically renegotiate new profiles, the performance can be further improved. For a heavy-tailed ON-OFF stream, the size of a burst increases with the bytes seen from the burst. Thus, if the transfer time of a burst exceeds some threshold, it is likely the burst will be very large. Based on this information, a user can explicitly request a new contract for transmitting the long document. This profile may cost less and the network may give low priority to its traffic so as to automatically achieve shortest job first scheduling and improve the overall response time. This profile may also cost more and the network will allow the entire burst to be sent at the peak rate.

7. Related Work

Rathgeb [18] compared different policing mechanisms proposed for controlling individual cell streams in the ATM networks using analytical results. The mechanisms he compared include the Leaky Bucket Mechanism(LB), the Jumping Window Mechanism(JW), the Triggered Jumping Window Mechanism (TJW), the Exponentially Weighted Moving Average Mechanism (EWMA), and the Moving Window Mechanism(MW). His comparison also used an ON-OFF model, where the ON-period had a geometry distribution and the OFF-period had a negative exponential distribution. The comparison that took into account aspects including the violation probability of a conforming statistical source and the worse-case admitted traffic showed the LB and EWMA were most promising mechanism. The author concluded to limit the violation probability for a well-behaving stochastic source, a large bucket size or a large window size is required. In our study, we chose a different source model, and argued that for any algorithm that limits the average rate of a source, it is impossible to limit the violation probability to some negligible value.

Courcoubetis and Kelly et al [8] proposed a charging scheme based on the effective bandwidth of a traffic stream. The charging schemes consider both the a prior knowledge of a traffic class and the posterior usage of a particular traffic stream. A total charge consists of a charge according to the usage time and a charge according to the total volume sent. A charging scheme is used to regulate misbehavior, as the more a user sends, the more he will be charged.

Berger [4] described an algorithm that defines a generalized form of traffic profiles. The token-bucket profile is its specialized form. In the paper, Berger showed the conditions on the algorithm to produce certain network properties, such as penalizing the throughput of bursty traffic. The algorithm has one single parameter. It is interesting to see whether our proposed profile, which has two state parameters, can be generalized using a similar approach.

Fonseca et al [13] proposed a novel traffic profiling algorithm called *fractal leaky bucket*[1] (FLB). The design of FLB is based on the observation that the cumulative arrival process $A(t)$ of a self-similarity process has an extra term kt^{α}, $0 < \alpha < 1$. Though $\lim_{t\to\inf} A'(t)/t = \mu$, where μ is

[1]There is a subtle different between a leaky bucket profile and a token bucket profile. For the purpose of our analysis, we can ignore the difference.

the expected rate of the arrival process, at any finite time, $\mu t < A(t)$. Thus, for a leaky bucket profile, if the token rate is set to μ, the arrived tokens will be less than the arrived bytes. This will cause packets (or cells) from a conforming process to be marked unconforming. The proposed FLB algorithm compares the total arrived packets within a time interval τ with those computed by the envelop process $A'(t)$ of the cumulative arrival process $A(t)$. If the arrived packets exceed those allowed by $A'(t)$ extra packets are considered out-of-profile, and τ is increased. On the other hand, if the arrived packets are less than those allowed by $A'(t)$, τ is decreased. However, there are at least three problems associated with the FLB algorithm. First, it is hard to compute the parameters for the envelop process. Second, decisions are made at discrete time intervals τ, which increases the delay jitter. Third, it is quite likely the envelop process allows a source to send at the peak rate for a fairly long time. As shown in Section 3, an envelop process of the ON-OFF Pareto process will be one that only limits the peak rate of the source.

Clark et al [7] designed the time-sliding window (TSW) algorithm to keep the average rate of a TCP flow within a target rate. The algorithm has two parameters: the sliding window length *window* and the target rate R_t. The algorithm also keeps a state variable *last* that records the arrival time of the last packet and a state variable *est* that estimates the average rate of a TCP flow. Upon the arrival of a packet with length L at time *now*, the algorithm updates *est* as $(L + est * window)/(now - last + window)$. Due to the saw-tooth behavior of a TCP flow, the sending rate of a conforming TCP may oscillate between $0.66R_t$ and $1.33R_t$. If *window* is large and $est > R_t$, the algorithm can decide a packet out-of-profile with probability $(est - R_t)/R_t$. If *window* is on the order of the round trip time, and $est > 1.33R_t$, the algorithm can decide a packet out-of-profile. Lin [16] et al proposed the *enhanced* TSW (ETSW) algorithm, which keeps track of both a short term estimation and a long term estimation of a TCP's sending rate. ETSW dynamically maintains a state variable *watermark* between [1.0, 1.33]. The short term estimation of the sending rate is compared against $watermark * R_t$ to decide whether a packet is out-of-profile or in-profile. Elysee [11] modified the TSW algorithm into the Average Rate Control Usage Profile (ARCUP) algorithm as a profile for bursty Internet traffic. Unlike TSW, in ARCUP, the rate estimation is done on a per file basis, and the policing action is to cut down the peak transfer rate of a file, instead of marking a packet as *in* or *out*. Elysee also compared the performance of the ARCUP algorithm and the leaky bucket algorithm as a traffic profile for four benchmark data sets [1]. In the early stage of our project, we also compared ARCUP and TSW algorithms. Our results did not show the two algorithms are superior to the leaky bucket algorithm. When the window size is set small, the algorithms are not able to filter out the instantaneous rate fluctuation. When the window size is large, the estimation error is propagated to the entire *window* duration, which makes the algorithm sensitive to the initial setting of the estimation rate.

Berger et al [5] developed a similar traffic profile using multiple token banks. In this traffic profile, each flow is assigned a token bank. And there is one system token bank that absorbs the overflowed tokens from individual token banks. The tokens in that token bank can be shared among different flows. This traffic profile provides a scheme to give a bonus to a traffic stream if the system is under-loaded.

Reibman et al [19] compares the suitability of two traffic profiles: the token bucket profile and the sliding window profile for teleconference calls in the context of ATM network. The parameters are chosen after the video sequence is known. Due to the burstiness of the traffic, they found that for the video sequences to be conforming to the traffic profiles, a large bucket size or a large window size is necessary, even when the negotiated average rate is twice the true average rate.

Ashmawi [2] et al evaluated the impact of different settings of the token bucket profile on the user perceived quality of video streams over both local and wide area networks. They investigates the Expedited Forwarding service of Diff-Serv, in which the bucket depth is limited to at most two Maximum Transmission Units (MTUs). However, in their experiments, they observed that even constant rate encoded video stream exhibited bursty behaviors. Consequently, they found a small increase in the bucket depth could lead to a significant increase in the quality of the video stream. In a vague sense, this finding coincides with ours. That is, increasing the bucket size is effective in providing QoS and limiting misbehaviors.

8. Conclusions and Future Work

In this paper, we describe a new class of traffic profiles — the 2Bucket profile. When metering bursty traffic, this profile achieves better performance than the traditional token bucket profile by decoupling the long term average rate control from the burst size control. Not only is the profile able to remember the long term behavior of a traffic stream, it also enables the network to control the burst size. We also present two possible refill policies that can be applied to the 2Bucket profile to demonstrate how a network can use the profile to enforce different QoS policies.

In the future, we intend to build a prototype system and test the effectiveness of the 2Bucket profile through user experiments and trace-driven simulation. In particular, we are interested in testing how different policies and parameter settings affect users' QoS assessments.

The performance of usage profiles is obtained via simulation. It will be helpful if we can derive some analytical models for analyzing the performance of different profiles.

9. Acknowledgment

I am grateful to my advisor David Clark for his guidance and support. I thank Arthur Berger and John Wroclawski for helpful discussions and comments on early drafts of the paper. I also thank the anonymous reviewers for their perceptive comments.

10. References

[1] BU Web Traces. Internet Trace Archive: `http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html`.

[2] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson. On the Impact of Policing and Rate Guarantees in Diff-Serv Networks: A Video Streaming Application Perspective. In *Proceedings of ACM SIGCOMM2001*, San Diego, California, USA, August 2001.

[3] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of ACM SIGMETRICS'97*, pages 151–160, June 1998.

[4] A. W. Berger. Desirable Properties of Traffic Descriptors for ATM Connections in a Broadband ISDN. In J. Labetoulle and J. Roberts, editors, *The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks*, volume 1a, pages 233–242. Elsevier Science B.V., 1994.

[5] A. W. Berger and W. Whitt. A Multi-Class Input-Regulation Throttle. In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, Hawaii, December 1990.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, December 1998. RFC 2475.

[7] D. D. Clark and W. Fang. Explicit Allocation of Best-Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking*, 6(4), August 1998.

[8] C. Courcoubetis, F. Kelly, and R. Weber. Measurement-based usage charges in communication networks, 1997.

[9] M. Crovella and L. Lipsky. Long-lasting transient conditions in simulations with heavy-tailed workloads. In *Winter Simulation Conference*, pages 1005–1012, 1997.

[10] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[11] P. A. Elysee. Usage Profiles : Allocation of Network Capacity to Internet Users. Master's thesis, M.I.T., 2001.

[12] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume II, chapter VI. John Wiley & Sons, 1971.

[13] N. L. S. Fonseca, G. S. Mayor, and C. A. Neto. On the Equivalent Bandwidth of Self-Similar Sources. *ACM Transactions on Modeling and Computer Simulation*, 10(2):104–124, 2000.

[14] F. Kelly. Notes on Effective Bandwidths. In F. Kelly, S. Zachary, and I. Ziedins, editors, *Stochastic Networks*, pages 141–168. Oxford Science Publications, 1996.

[15] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking*, 2(1):354–367, February 1994.

[16] W. Lin, R. Zheng, and J. C. Hou. How to Make Assured Services More Assured. In *the 7th International Conference on Network Protocols (ICNP'99)*, Toronto, Canada, October 1999.

[17] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[18] E. Rathgeb. Modeling and Performance Comparison of Policing Mechanisms for ATM Networks. *IEEE Journal on Selected Areas in Communications*, 9(3):325–334, April 1991.

[19] A. R. Reibman and A. W. Berger. Traffic Descriptors for VBR Video Teleconferencing Over ATM Networks. *IEEE/ACM Transactions on Networking*, 3(3), June 1995.

[20] J. Roberts. Engineering for Quality of Service. In K. Park and W. Willinger, editors, *Self-Similar Network Traffic and Performance Evaluation*, pages 401–420. John Wiley & Sons, Inc., 2000.