SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 3, ISSUE 5
ISSN: 2054 -7420

Efficient On-Line Traffic Policing for Confidence Level based Traffic Model

Lie Qian
Dept. of Chemistry, Computer & Physics Science, Southeastern Oklahoma State University, Durant, OK, USA
lqian@se.edu

ABSTRACT

On-line traffic, such as conversational call, live video, serves a large group of applications in the internet now days. An important feature of on-line traffic is that they are not pre-recorded and no exact information about each session's traffic is known before the traffic happens. S-BIND (Confidence-level-based Statistical Bounding Interval-length Dependent) traffic model was proposed to characterize such traffic for QoS admission (GammaH-BIND) and policing purpose. A state-dependent token bucket based statistical regulator was proposed to police the traffic using S-BIND parameters. However, the proposed regulator can output traffic within the expected S-BIND parameters if the input traffic is random or is just trying to transmit large amount of traffic without exploiting the regulator's token bucket design. In this paper, the author shows that if the source of the traffic understands the bucket's behavior, it can tune the traffic and cause significant violations in the regulator's output traffic. A new design of state-dependent token bucket for the regulator is proposed in this paper to remove such potential problem and an optimization algorithm is given to improve the regulator's efficiency by removing redundant token buckets in the regulator.

Keywords: Traffic Model, QoS, Policing

1 Introduction

On-line traffic, such as conversational call, live video, serves a large group of applications in the internet now days. Most of such on-line traffic are delay sensitive and have Quality of Service (QoS) requirements. QoS refers to the capability of a network to provide better service to selected network traffic or flows [1]. QoS requirements can be described by parameters such as delay, packet loss rate, jitter, and etc.

To fulfil QoS requirements, network cannot allow unlimited amount of traffic enter the network without admission control. Admission control algorithms are used to decide if a new incoming traffic flow will affect the QoS of the existing traffic in the network while it trying to receive its expected QoS. Proposed admission control algorithms can be classified into two categories: measurement-based and traffic descriptor-based admission control [1]. Measurement-based admission control algorithms [3-6] measure the traffic condition in the network and estimate the QoS in the network with the new arriving traffic flows. The admission decisions are made based on such estimation. Such measuring requires significant changes in the network's architecture. Traffic descriptor-based admission controls [7-12] allow the

DOI: 10.14738/tnc.35.1446
Publication Date: 20th September 2015
URL: http://dx.doi.org/10.14738/tnc.35.1446

network to make admission control decisions based on existing and new coming traffic's characterizations (parameters), which are used to calculate the QoS in the network that can be received by the existing and the new arriving traffic flows. This kind of admission control relieves network from constant traffic measuring in the network, which could be a significant burden to the network, and requires less modification in the network architecture. Furthermore, with the introduction of DiffServ QoS architecture [13], where the per-flow information can be maintained in a centralized QoS controller (bandwidth broker) [14, 15], traffic descriptor-based admission control process can be decoupled from routers to solve scalability problems in routers.

The effectiveness of the traffic descriptor-based admission control depends heavily on the accuracy of the traffic models that are used to characterize the existing and incoming new traffic. Among the proposed traffic models [1, 16-24], Confidence-level-based Statistical Bounding Interval-length Dependent (S-BIND) [1] model doesn't require pre-existing traffic traces to get the parameters and thus could be applied to On-Line traffic. GammaH-BIND admission control algorithm was proposed in [1] to perform admission control based on S-BIND traffic model parameters. After being admitted, network traffic described by S-BIND parameters need to be regulated at the entrance of the network to ensure that the traffic going through the network does follow its declared S-BIND parameters. In [2], the authors propose a state-dependent token bucket based statistical regulator to police the S-BIND network traffic. The regulator is composed of a series of state-dependent token buckets. Each token bucket is established based on the parameters of on linear section on the S-BIND's constraint function. However, the regulator proposed in [2] could output traffic violating the S-BIND's parameters under certain circumstances when the source understands the regulator's design and tunes its traffic to get the worst case from the token buckets in the regulator. Also setting up a token bucket for each linear section could be redundant when some buckets could be stricter than the others and make them unnecessary in the regulator.

This paper analyses the real constraint function for the state-dependent token buckets proposed in [2] and their worst cases which could generate output traffic violating the S-BIND parameters. Based on the analysis, a new design of the state-dependent token buckets is first proposed to output the traffic within S-BIND limit even in the worst case. Secondly an optimization algorithm will be proposed to remove the redundant token buckets in the regulator to improve the regulator's efficiency. The empirical analysis presented in this paper show that the new token buckets always output traffic within the S-BIND parameter's limit and the optimization algorithm could detect redundant token buckets in the regulator.

The remainder of this paper is organized as follows. The related works in traffic models and traffic regulating are discussed in Section 2. Section 3 presents the worst case analysis of the token buckets, the new token bucket design and the optimization algorithm. The empirical analysis results are demonstrated in Section 5. The conclusions are given in Section 6.

2 Related works

In this section, a brief review of the network traffic models and policing is given. First let's define the network model used in this paper composed of a centralized traffic controller in a network domain. The traffic controller uses traffic admission processes to admit new traffic going through the domain based on the new traffic's descriptors (parameters) while ensuring all existing traffic's QoS in the domain. Ingress edge routers are the routers through which the network traffic enter the domain. After the new traffic is

admitted, regulators need to be deployed at the ingress edge routers to ensure that the traffic behave as described in the admission control process

2.1 Confidence-level-based Statistical BIND (S-BIND) traffic model

Among traffic descriptors, binding traffic models bound the traffic volume. Traffic constraint function, denoted as b(t), is the essential part for binding traffic models. A network traffic flow is said to be bounded by b(t) if during any interval of length of u, the amount of traffic transmitted by this flow is less or equal to b(u). Other than (σ,ρ) pair [21, 22], ($X_{min},X_{ave},I,S_{max}$) model proposed in [7], and Bounding Interval-dependent (BIND) based models are proposed [16, 23, 24], D-BIND [16] is a traffic model specifying the maximum arrival rates for several time intervals known as multiple rate interval pairs. D-BIND works well for deterministic bounding and off-line traffic, but is not suitable for on-line traffic. For on-line traffic other than video, bounding the transmission in each interval with the worst case rate estimated from the traffic behavior properties may compromise the effectiveness of D-BIND. In [1], authors developed a new Confidence-level-based Statistical BIND (S-BIND) traffic model. The S-BIND model defines p time interval rate pairs as:

$$\{(R_k, I_k) \mid k = 1,..., p\} \tag{1}$$

where, $I_1<I_2<...<I_{k-1}<I_k$. I_k denotes the time interval length, and R_k is the rate, at which the flow is allowed to send in any period of I_k statistically. In S-BIND, random variable S_k is defined as:

$$S_k(a) = prob\left(\frac{A_j[t, t+I_k]}{I_k} \le a\right), \forall t \ge 0 \tag{2}$$

Here, $A_j[t_1,t_2]$ denotes the amount of arrivals in traffic flow j within time interval $[t_1,t_2]$. S_k reflects the distribution of flow's transmission rate over time interval I_k and has density function $s_k(a)$. For each time interval I_k, R_k is defined in S-BIND by using random variable S_k's density function $s_k(a)$ as following:

$$\int_0^{R_k} s_k(t)dt = \varepsilon \tag{3}$$

When ε=1, R_k will be the same as in D-BIND. The smaller ε we set, the smaller R_k we will get, and the higher network utilization can be expected in admission control because more traffic will be admitted. For different kinds of on-line traffic such as conversation, videoconference, high motion video, low motion video, game data and etc., S-BIND parameters can be pre-defined with different confidence levels through experiments or statistical data analysis.

Along with GammaH-BIND admission algorithm developed in [1], S-BIND can achieve maximum valid network utilization for both low bursty and high bursty traffic.

2.2 Token Bucket Based Statistical Regulator

After a network traffic flow being admitted by the domain's controller based on the flow's descriptors, the traffic flow needs to be monitored during its life time to ensure that it doesn't violate its claimed traffic descriptor by sending too much data into the domain. Regulators at the ingress router are responsible for this kind of monitoring. The traffic flow violating its claimed descriptor could be disconnected or its excessive data packets could be dropped at the ingress routers.

Most known policing mechanisms, such as Leaky Bucket, Token Bucket, the Triggered Jumping Window [25], Dual Leaky Bucket [26], BLLB [27], FBLLB [28] cannot handle statistical traffic models, such as S-BIND, where the traffic rates are allowed to exceed the parameters statistically. The authors in [2] developed a Token Bucket Based Statistical Regulator. Multiple state-dependent token buckets with dynamic token generation rates are cascaded together to regulate S-BIND traffic. One bucket is deployed for each linear segment in the constraint function *b(t)* derived form S-BIND traffic model. Each bucket handles packets statistically. A state diagram is given in **Figure 1** to show the bucket's behavior.

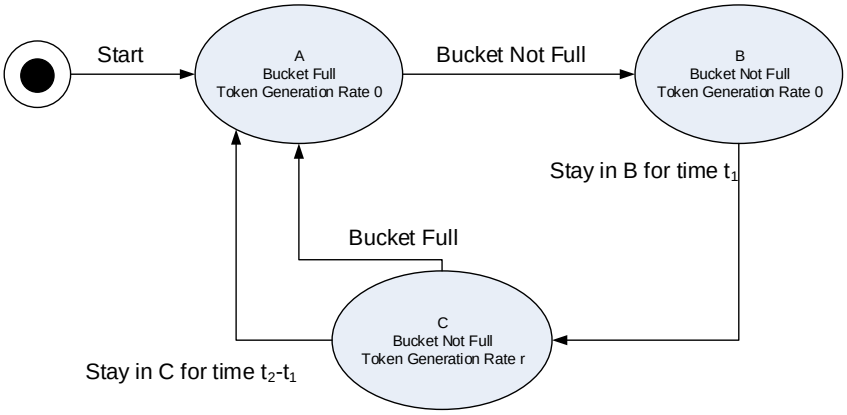

Figure 1: Token Bucket's Behavior

For each linear segment in the constraint function $b(t)$, it has two end points, one at (t_1,b_1), and one at (t_2,b_2), where $t_1<t_2$, $b_1\leq b_2$. A state-dependent token bucket corresponding to this segment has bucket depth b_1, and the token generation rates $r=(b_2-b_1)/(t_2-t_1)$. In the example shown in **Figure 2**, the segment for Bucket1 has parameters: $t_1=a$, $t_2=b$, $b_1=c$, $b_2=d$. The segment for Bucket2 has parameters: $t_1=b$, $t_2=f$, $b_1=d$, $b_2=e$.

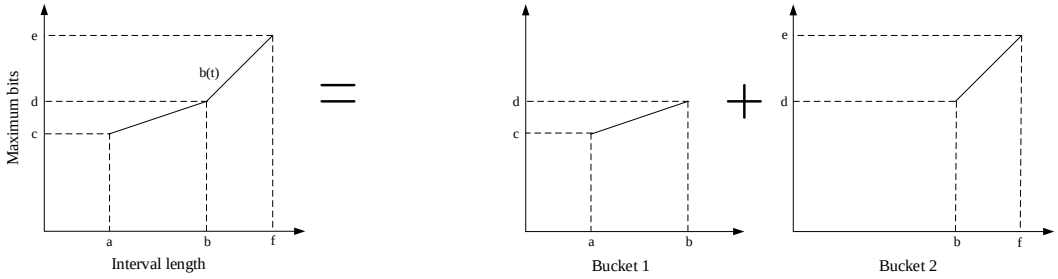

Figure 2: Cascade of Token Buckets for Non-Concave Constraint Function

Different from traditional deterministic token buckets which drop incoming packets when there is no enough token left in the bucket, regulator proposed in [2] is allowed to forward packets even when the token is not enough statistically based on the confidence level specified in S-BIND traffic model. When a packet p with size s needs to be transmitted while the token left in bucket is $t<s$, the transmission probability of the packet is calculated as (4).

$$P_{transmit}=\begin{cases}\dfrac{(1-\varepsilon)-\dfrac{t_{neg}}{t_{neg}+t_{pos}}}{(1-\varepsilon)} & if\ \dfrac{t_{neg}}{t_{neg}+t_{pos}}<(1-\varepsilon)\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

In (4), the transmitting probability mainly depends on the variables t_{neg} and t_{pos}, while ε is a confidence level parameter from S-BIND traffic model, which should be a constant for the life time of the bucket. t_{neg} is counted as the time when the token bucket has negative amount of token and t_{pos} is calculated as the total time minus t_{neg}.

3 Efficient Token Bucket Based Statistical Regulator Design

In this section, the author first derives the complete constrain function for each token bucket in the statistical regulator proposed in [2], then proposes a new bucket design and an optimization algorithm to remove the redundant token buckets to reduce the ingress routers' policing burden.

3.1 Constrain Function for Each Token Bucket

The regulator proposed in [2] is composed of a series of token buckets, one for each linear section on the constrain function derived from S-BIND traffic model. Let's say we have a token bucket B following the linear section between points (t_s, b_s) and (t_e, b_e) on the constrain function diagram as shown in Figure 3. All traffic of this flow need pass bucket B to be admitted into the domain (of course, also need pass all other token buckets in the same regulator). Here a question we are facing is what the output traffic will like under different time intervals.

Formally, let $A_j[t_1,t_2]$ denote the total number of bits sent by flow j between time t_1 and t_2. Flow j is bounded by constraint function $b(t)$ if $A_j[t,t+u] \le b(u)$, for all t, $u>0$. Based on the token bucket's behavior state diagram in Figure 2, let's discuss the maximum volume of traffic could be forwarded in bucket B within any time interval $t_1 \le \Delta t$, where $\Delta t = t_e - t_s$. We consider the following 3 scenarios:

Scenario 1: t_1 starts in state A. Immediately before t_1 starts, the token bucket shifts its state from *C* to *A* and have a full bucket of token (bucket size is b_s). Therefore during t_1, b_s amount of traffic could be forwarded at least and the token bucket shift to state *B* immediately after t_1 starts. However, state *B* requires the token bucket stay in state B for time length t_s with no new token generated. Let's say, in the worst case, t_1 is greater than t_s and before the end of t_1 the state shifts to *C*, in which tokens are generated at the rate of $r = \Delta b/\Delta t$, where $\Delta b = b_e - b_s$. Because t_1 is less than Δt, which is the time the bucket need to stay in state *C* to achieve maximum token generated, the bucket will remain in state *C* at the end of t_1. Therefore, at the end of t_1, in the worst case, the token bucket is still in state C and the total amount of traffic delivered during t_1 is $b_s + \Delta b = b_e$.

Scenario 2: t_1 starts in state *B*. In order to get the worst scenario, we assume that t_1 starts at the very end of state *B* (so that more token could be generated very soon after switching to state *C*) and the token volume left in the bucket is $b_s - \varepsilon$, where ε is a very small non zero value (could be the size of the smallest packet in the network). After t_1 starts, in the worst case, the token bucket immediately switches to state *C* and generates token in the rate of $r = \Delta b/\Delta t$ for Δt time, i.e. generate Δb traffic and at the end of t_1 the token bucket becomes full and switches to state *A*. The consumed token volume during this period is $b_s - \varepsilon + \Delta b + b_s = b_e + b_s - \varepsilon$ which has upper bound of $b_e + b_s$.

Scenario 3: If t_1 starts during state *C*, the token generated before the token bucket is full is less than Δb and the bucket could become full when the state switch to *A*. The token generated is $\Delta b + b_s = b_e$.

Among the 3 scenarios above, it is obvious that scenario 2 is the worst case in which the token generated is bounded by

$$b(t_1) = \begin{cases} b_s + \frac{\Delta b}{\Delta t} t_1 & if\ t_1 < \Delta t \\ b_e + b_s & if\ t_1 = \Delta t \end{cases} \quad (5)$$

In scenario 2, after t_1 ends in the worst case, the bucket will stay in state *B* with no token generated (i.e. no traffic can be forwarded) for at least t_s amount of time, which gives us the bound for time interval $\Delta t+t_s=t_e$ as second part in (6) below.

$$b(t) = \begin{cases} b_s + \frac{\Delta b}{\Delta t} t & if\ t < \Delta t \\ b_e + b_s & if\ t \geq \Delta t\ and\ t \leq t_e \end{cases} \quad (6)$$

Therefore for t≤te, we can have the constraint function for a linear segment B depicted in the Figure 3 below.

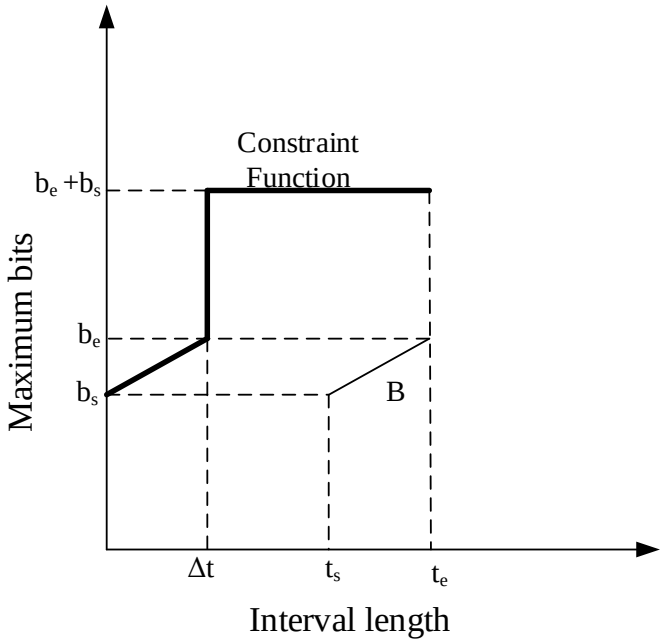

Figure 3: Constraint Function for One Token Bucket in [2] t≤t_e

For the time interval *t* that is greater than t_e, its first t_e amount of time will send in the worst case b_e+b_s amount of traffic as shown in Figure 3. After that, the bucket will come back to the beginning of scenario 2 and get ready to shift to state *C*. At this moment, the tokens in the bucket has been counted in $b(t_e)= b_e+b_s$ already. Therefore only the newly generated tokens in state *C* will be counted and added to the constraint function. The token generated (i.e. traffic allowed to be forwarded) in the period of the second period of te is only Δb+bs=be, in which Δb is the amount of tokens generated in state C with rate r=Δb/Δt and bs is the amount of tokens generated when the bucket switches to state A. Put all rounds together, we can get the general constraint function for the linear section between (ts, bs) and (te, be) as below in (7) and depicted in Figure 4.

$$b(t) = \begin{cases} b_s + \left\lfloor \frac{t}{t_e} \right\rfloor b_e + \frac{\Delta b}{\Delta t} t & if\ t\%t_e < \Delta t \\ b_s + \left(\left\lfloor \frac{t}{t_e} \right\rfloor + 1 \right) b_e & if\ t\%t_e \geq \Delta t \end{cases} \quad (7)$$

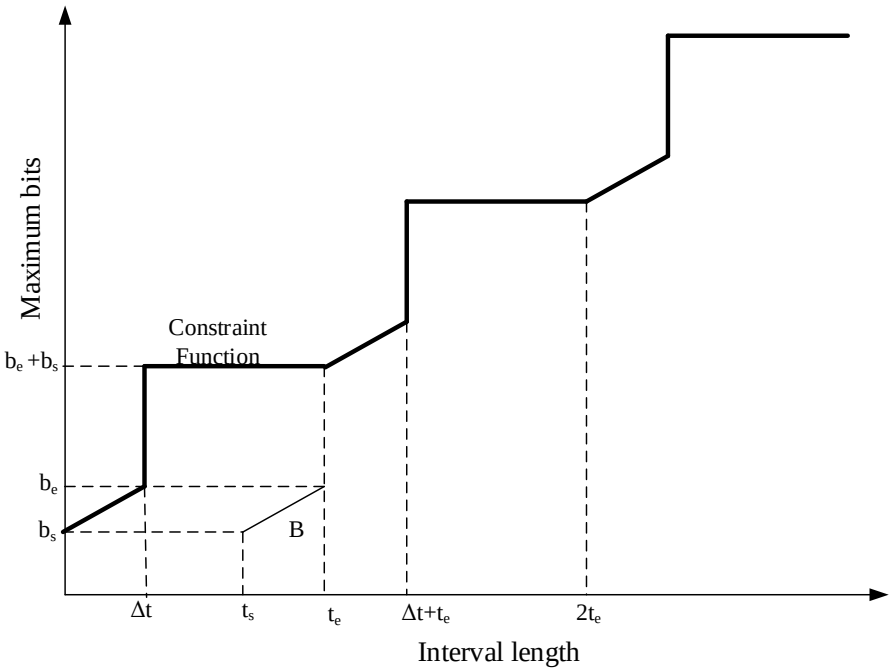

Figure 4: Constraint Function for One Token Bucket in [2]

3.2 Token Bucket Based Statistical Regulator Optimization Algorithm

In this subsection, a new state dependent token bucket is proposed to have tighter constraint function compared to [2], which is analyzed in subsection 3.1. This newly designed token bucket's token generation behavior could be summarized in Figure 5, where Δt is the difference between t_e and t_s of the corresponding linear segment between (t_s, b_s) and (t_e, b_e) in the S-BIND model. The token bucket for this linear segment has bucket size of b_s. The initial state of the token bucket is state *A1* with full bucket of tokens and 0 token generation rate. The first difference here compared to the design in [2] is that after state *B*, the bucket will go back to state *A* then to *B* for one more round before it goes to state *C*. Therefore, 2 more states are added to separate different visits to states *A* and *B*. Now state *A* is divided into states *A1* and *A2*; *B* is divided into *B1* and *B2*. The second difference is that the bucket switches to state *B1* or *B2* from *A1* or *A2* respectively only after the bucket is empty. By doing this, the starting of state *B1* or *B2* are postponed compared to [2] which can reduce the traffic crowded at the transition time between *B2* and *C* under the worst case as described in scenario 2 in subsection 3.1. The third change is in the transition from state *C* to *A1*. If the transition is triggered by the volume of tokens in the bucket in *C* reaching b_s, no token needs to be added. If the transition is triggered by the expiration of Δt, the volume of tokens need to be added during the transition, noted as b_{jump}, following the formula (8).

$$b_{jump} = \begin{cases} b_s - \Delta b & if\ \Delta t < t_s \\ b_s - t_s r & if\ \Delta t \geq t_s \end{cases} \quad (8)$$

where Δb is the difference between b_s and b_e, $r=\Delta b/\Delta t$. With this new design, we can see that after the transition from *C* to *A1*, the bucket is not necessarily full. The reduction in the b_{jump} from b_s in [2] helps to give a tighter constraint function.

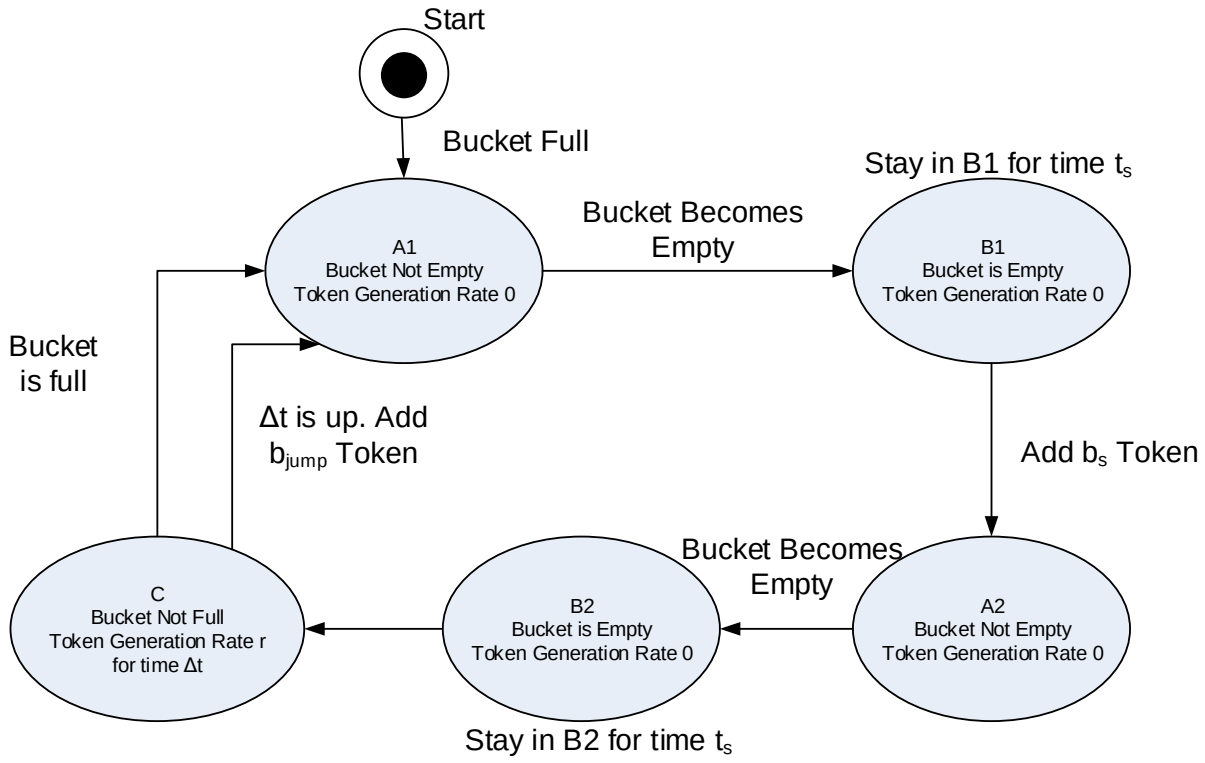

Figure 5: New Token Bucket's Behavior

Based on the new design, the maximum amount of consumed token cannot exceed b_s for any time interval less or equal to t_s. Let's discuss the constraint function for the time interval *t* less or equal to t_s under following 3 scenarios.

Scenario 1: We have a full bucket of token (b_s in total) in state *A2* and time interval *t* starts. Under this scenario, because the bucket will generate no token for consumption for t_s amount of time after the bucket is empty, the time interval *t* cannot reach state *C* for any new token to be generated. The amount of traffic that can be forwarded is limited by the volume of token available in state *A2*, which is bounded by the bucket size b_s.

Scenario 2: We have an empty bucket at the beginning of state *C* and the time interval *t* starts. If Δt is greater than t_s, time interval *t* will end before the switching to *A1* happen and the generated tokens are limited within Δb. If Δt is less than t_s, second case of b_{jump} in (8) could happen and the overall generated tokens are $\Delta b+b_{jump}=\Delta b+b_s-t_sr<b_s$. That's because $t_sr> \Delta tr=\Delta b$. Therefore the volume of token could be generated within the time interval *t* is still limited by the greater one between b_s and Δb.

Scenario 3: This scenario is trivial, if the time interval *t* starts in state *A1*, the bucket has to wait at least t_s amount of time in *B1* for the next token to be added.

Now let's discuss the tokens generated in the next Δt amount of time after the end of the first period of t_s just discussed.

Following scenario 1, the next Δt amount of time in state *C*, Δb tokens could be generated in the worst case under rate *r*, which bounds the tokens in time interval $t_e=t_s+\Delta t$ within $b_s+\Delta b=b_e$. At the end of t_e, b_{jump} could happen in the worst case and the scenario go back to the beginning of scenario 1. Before that the total tokens generated in the period of t_e is b_e+b_{jump}.

Following scenario 2, the next Δt amount of time is spent in state *C* and *A1*. In the worst case the whole period of t_e could span the state *C* and *A1* altogether (include the first t_s amount of time discussed in scenario 2). State *C* generates $\Delta b+b_{jump}$. At the end of *A1*, b_s is added. In total, $\Delta b+b_{jump}+b_s=b_e+b_{jump}$. After that, the scenario go back to the beginning of scenario 3.

Based on the discussion above, we can depict the constraint function for linear segment *B* as below in Figure 6. Compare to Figure 4, this newly designed regulator's constraint function is significantly tighter.

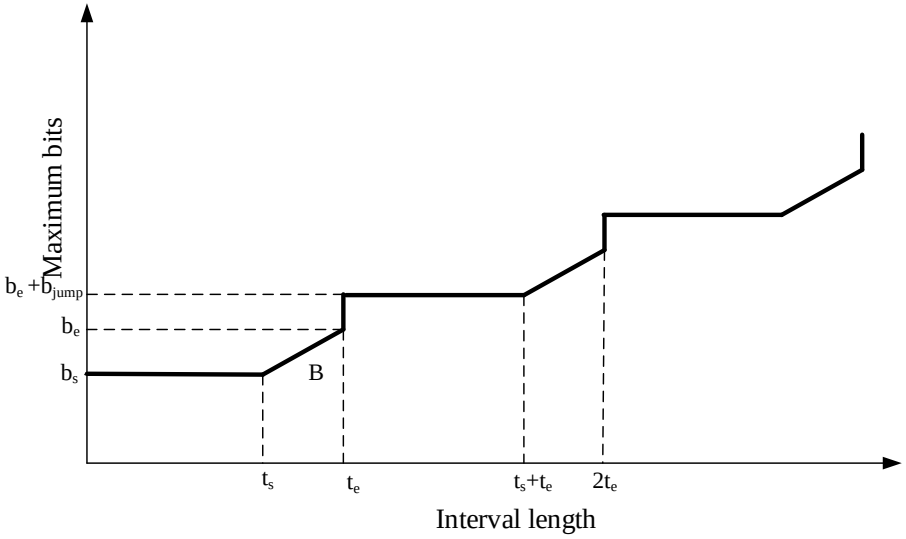

Figure 6: New Token Bucket Constraint Function

Based on the constraint function depicted in Figure 6, we can get the general constraint function for the linear section between (t_s, b_s) and (t_e, b_e) as below.

$$b(t) = \begin{cases} b_s + \left\lfloor \frac{t}{t_e} \right\rfloor (\Delta b + b_{jump}) & if\ t\%t_e < t_s \\ b_s + \left\lfloor \frac{t}{t_e} \right\rfloor (\Delta b + b_{jump}) + (t\%t_e - t_s)r & if\ t\%t_e \geq t_s \end{cases} \quad (9)$$

3.3 Remove Redundant Buckets

After having the complete constraint function (9) for a given token bucket in this new design, we can take a look at the token buckets needed in the regulator. Based on the original design in [2], one token bucket is setup for each linear segment on the constraint function of the S-BIND descriptor. In reality, because of convex segments in *b(t)*, it is possible that a bucket, say *B*, has its constraint function below another bucket *C*'s constraint function at all time. In another word, bucket *B*'s output is always smaller than the amount of traffic allowed by bucket *C* at any time interval. In this case, there is no need for bucket *C* and we can reduce the traffic policing burden in the regulator by removing *C*'s bucket to be more efficient in resource consuming.

Here is a new method used to decide when a token bucket is redundant. For a token bucket *B`* for segment from point *(t`1, b`1)* to *(t`2, b`2)*, if there exists one bucket B for the segment between *(t1, b1)* and *(t2, b2)*, we can remove B` in our regulator if and only if:

a) B`'s linear rate is higher than B, i.e. $\frac{b`_2-b`_1}{t`_2-t`_1} > \frac{b_2-b_1}{t_2-t_1}$ AND

b) B`'s segment is somewhere after B's in the overall b(t) derived from multiple rate interval pairs, i.e. $t`_1 > t_2$ AND

c) B`'s constraint function $b_{B`}(t)$ is above *B*'s function anywhere between $t`_1$ and $t`_2$, i.e. $\forall t, where\ t`_1 \le t \le t`_2, b`_1+(t-t`_1)\frac{b`_2-b`_1}{t`_2-t`_1} \ge b_B(t)$ is true, where $b_B(t)$ follows (9).

For example in Figure 7, after we have a bucket for *B*, segment *C* doesn't require an extra bucket in policing because it is totally above $b_B(t)$. However, if we have a segment *D* instead of *C*, then we still need a bucket for *D* to properly regulate the traffic. By using the method above, after admitting one flow, the network domain can calculate and setup only the necessary buckets to regulate the incoming traffic efficiently.

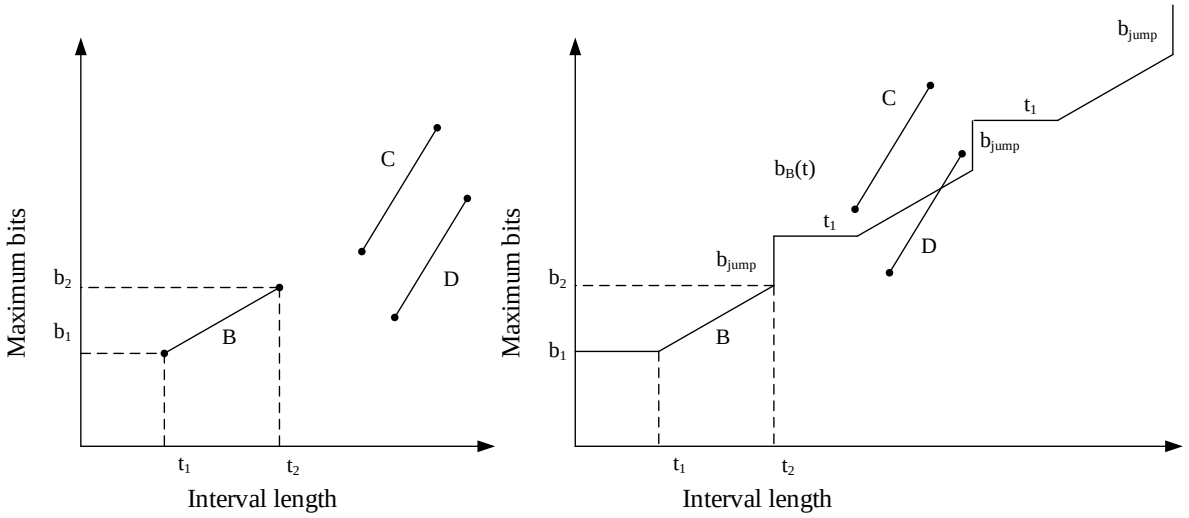

Figure 7: Redundant Token Bucket in Regulator

4 Empirical Analysis

In this section, simulation results are presented for empirical analysis. In [2], the input malicious traffic is modeled as ON-OFF Markov process. The traffic volume is large (when the malicious traffic keep staying in ON state for most of the time), but the pattern of the traffic cannot get the worst out from the buckets in the regulator. In this paper, simulation is built based on a specially designed traffic pattern, which could put the buckets into their worst case scenario and we can see that the output traffic from the regulator in [2] violates the S-BIND traffic descriptor.

4.1 Violating Input Traffic Pattern

The S-BIND traffic descriptors used to set up the regulator are derived from ON-OFF Markov process modeling low-bursty traffic (1/α=1.587s, 1/β=1.004s, R=64Kbps), which simulates speech audio based on ITU-T specification. S-BIND intervals are 500ms, 750ms, 1000ms, 2500ms, 4000ms, and 5000ms. Confidence levels are 90%, 80%, 70% and 60%. The descriptors are listed in the table below, which are used to setup the regulator's token buckets in simulations.

Table 1: S-BIND Parameters Used to Setup Regulator

	500ms	750ms	1000ms	2500ms	4000ms	5000ms
90%	32Kb	48 Kb	64 Kb	132.86 Kb	186.88 Kb	223.49 Kb
80%	32 Kb	45.63 Kb	54.66 Kb	107.78 Kb	154.62 Kb	187.14 Kb
70%	24.96 Kb	33.22 Kb	41.80 Kb	87.62 Kb	133.31 Kb	163.90 Kb
60%	14.14 Kb	22.14 Kb	28.54 Kb	72.32 Kb	110.85 Kb	139.65 Kb

Among all these linear segments, due to the length of the paper, the results of the sections between 500ms and 750ms, 1000ms and 2500ms at 70% confidence level are presented in this paper for analysis. The buckets established for these two sections has the following parameters (b_s=24.96Kb, b_e=33.22Kb, t_s=500ms, t_e=750ms), (b_s=41.8Kb, b_e=87.62Kb, t_s=1000ms, t_e=2500ms).

To get the worst case out from the buckets, 2 kinds of input traffic are setup as following. The first kind of input traffic (Traffic Pattern I) is desinged to get the worst case of the buckets designed in [2]. In Traffic Pattern I, the source will send a very small amount of traffic at very beginning (state *A* for old buckets and state *A1*, *A2* for newly designed buckets), then the source will wait for t_s amount of time without sending anything. After that, the source will send large amount of traffic trying to exhuast any token in the bucket for Δt amount of time. Then the souce will again send a very small amount of traffic to restart the loop. The second kind of input traffic (Traffic Pattern II) is designed to get the worst case of the newly designed buckets in this paper. Because the 0 token generation rate in the new buckets will last for t_s amount of time when the bucket is empty, the small amount of traffic in Traffic Pattern I will have no effect in the state transition under our newly designed bucket. Therefore, Traffic Pattern II tries to send unlimited amount of traffic all the time and tries to exaust the tokens in the buckets all the time.

4.2 Experiment Comparison

Traffic Pattern I is tested on both old and our newly designed buckets. Traffic Pattern II is tested on our newly designed bucket. Results are presented in Table 2.

Table 2: Results Comparing New Buckets with Old Buckets in [2]

	Bucket (500ms-750ms)		Bucket (1000ms-2500ms)	
Bucket Interval (t_s, t_e)	500ms	750ms	1000ms	2500ms
Bucket Volume (b_s, b_e)	24.96Kb	33.22Kb	41.80Kb	87.62Kb
70% Traffic I->Old Bucket*	**33.1Kb**	33.1Kb	**49.5Kb**	87.5Kb
70% Traffic I->New Bucket**	24.9Kb	24.9Kb	26.5Kb	57Kb
70% Traffic II->New Bucket***	24.9Kb	29Kb	40.3Kb	77.1Kb

We can see in Table 2 the old buckets cannot handle Traffic Pattern I correctly, the output traffic's S-BIND parameters (row with *) violate the expected bucket's b_s value. On the other hand, the newly designed buckets' outputs are all bounded by the expected parameters (rows with ** and ***).

The newly proposed bucket optimization algorithm described in Section 3.3 is tested on the S-BIND parameters in Table 1. The result shows that at 70% confidence level, the bucket between 4000ms and 5000ms can totally be removed from the regulator, because its constraint function is completely above the constraint function of the bucket between 1000ms and 2500ms as we derived in section 3.3. Therefore

the bucket between 400ms and 5000ms will block no traffic at all if it takes input from the bucket between 1000ms and 2500ms. By removing one bucket from the 5 buckets in the regulator, about 20% computation cost can be saved from the regulator.

5 Conclusion

In this paper, a new token bucket design is proposed for traffic using S-BIND traffic descriptor. In the new design, the bucket can regulate the traffic within the S-BIND parameter even when the source tunes the input traffic to hit the worst case of the regulator. Other than the bucket design, a new bucket optimization algorithm is proposed to reduce the number of buckets needed in a regulator by checking the constraint functions of all the buckets to remove the redundancy.

REFERENCE

[1] Lie Qian, Anard Krishnamurthy, Yuke Wang, Yiyan Tang, P. Dauchy, and Albert Conte, A New Traffic Model and Statistical Admission Control Algorithm for Providing QoS Guarantees to On-Line Traffic. Proceedings of IEEE Global Telecommunications Conference, 2004, GLOBECOM, vol. 3, pp. 1401-1405.

[2] Lie Qian, Yuke Wang, and Hong Shen, Token Bucket Based Statistical Regulator for S-BIND Modeled On-Line Traffic. Proceedings of IEEE International Conference on Communications, 2005, ICC, vol. 1, pp. 125-129.

[3] J. Qiu and E. W. Knightly, Measurement-based admission control with aggregate traffic envelopes, IEEE/ACM Transactions on Networking, vol. 9, no. 2, pp. 199-210, April 2001.

[4] B. Statovci-Halimi, Adaptive admission control for supporting class-based QoS, 2010 6th EURO-NF Conference on Next Generation Internet (NGI), pp. 1-8, 2010.

[5] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, Endpoint admission control: architectural issues and performance, Proc. Of ACM SIGCOMM'00, pp. 57-69, September 2000.

[6] V. Elek, G. Karlsson, and R. Ronngre, Admission control based on end-to-end measurements, Proc. Of IEEE INFOCOM, March 2000.

[7] D. Ferrari and D. C. Verma, A scheme for real-time channel establishment in wide-area networks, IEEE Journal on Selected Areas in Communications, vol. 8, Issue 3, pp. 368-379, April 1990.

[8] E. W. Knightly, H-BIND: a new approach to providing statistical performance guarantees to VBR traffic, Proc. Of IEEE INFOCOM '96, pp. 1091--1099, March 1996.

[9] E. W. Knightly and N. B. Shroff, Admission control for statistical QoS: theory and practice, IEEE Network, vol. 13, Issue 2, pp. 20--29, 1999.

[10] George Kesidis, Jean Walrand and Cheng-Shang Chang, Effective bandwidths for multiclass Markov fluids and other ATM sources, IEEE/ACM Transactions on Networking, vol. 1, pp. 424-428, 1993.

[11] H. A. Harhira, and S. Pierre, A Mathematical Model for the Admission Control Problem in MPLS Networks with End-to-End delay guarantees, Proc. Of 16th International Conference on Computer Communications and Networks, pp. 1193-1197, 2007.

[12] S. Alwakeel, and A. Prasetijo, Probability admission control in class-based Video-on-Demand system, 2011 International Conference on Multimedia Computing and Systems (ICMCS), pp. 1-6, 2011.

[13] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, An architecture for differential services, IETF, RFC 2475, December 1998.

[14] Z.-L. Zhang, Z. Duan, L. Gao, and Y. Hou, Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services, ACM SIGCOMM Computer Communication Review, vol. 30, Issue 4, 2000.

[15] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, A two tier resource management model for the Internet, Proc. Of GLOBECOM '99, vol. 3 , pp. 1779 -1791, 1999.

[16] E. W. Knightly and H. Zhang, D-BIND: an accurate traffic model for providing QoS guarantees to VBR traffic, IEEE ACM Transactions on Networking, vol. 5, Issue 2, pp. 219-231, April 1997.

[17] H. P. Stern, S. A. Mahmoud, and K. K. Wong, A comprehensive model for voice activity in conversational speech-development and application to performance analysis of new-generation wireless communications system, Wireless Networks, vol. 2, no. 4, pp. 359-367, December 1996.

[18] F. Beritelli, A. Lombardo, S. Palazzo, and G. Schembra, Performance analysis of an ATM multiplexer loaded with VBR traffic generated by multimode speech coders, IEEE Journal on Selected Areas in Communications, vol. 17, no. 1, pp. 63-81, January 1999.

[19] P. R. Jelenkovic, A. A. Lazar, and N. Semret, The effect of multiple time scales and subexponentiality in MPEG video streams on queueing behavior, IEEE Journal on Selected Areas in Communications, vol. 15, no. 6, pp. 1052-1071, August 1997.

[20] A. Lombardo, G. Morabito, and G. Schembra, An accurate and treatable markov model of MPEG-video traffic, IEEE INFOCOM, pp. 217-224, March 1998.

[21] R. Cruz, A calculus for network delay, part I: Network elements in isolation, IEEE Transactions on Information Theory, vol. 37, Issue 1, pp. 114-121, January 1991.

[22] S. Chong and S. Li, Probabilistic burstiness-curve-based connection control for real-time multimedia services in ATM networks, IEEE Journal on Selected Areas in Communications, vol. 15, no. 6, pp. 1072-1086, August 1997.

[23] J. Kurose, On computing per-session performance bounds in high-speed multi-hop computer networks, ACM Sigmetrics'92, vol. 20, Issue 1, 1992.

[24] H. Zhang and E. Knightly, Providing end-to-end statistical performance guarantees with interval dependent stochastic models, ACM Sigmetrics'94, vol. 22, Issue 1, 1994.

[25] E. P. Rathgeb, Modeling and performance comparison of policing mechanisms for ATM networks, IEEE Journal on Selected Areas in Communications, vol. 9, Issue 3, pp. 325-334, April 1991.

[26] J. Turner, New directions in communications (or which way to the information age), IEEE Communication Magazine, vol. 24, Issue 10, pp. 8-15, October 1986.

[27] M. Salamah and H. Lababidi, BLLB: a novel traffic policing mechanism for ATM networks, 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 411-415, August 2000.

[28] M. Salamah and H. Lababidi, FBLLB: a fuzzy-based traffic policing mechanism for ATM networks, ACS/IEEE International Conference on Computer Systems and Applications, pp. 31-35, June 2001.

