Leaky Buckets: Sizing and Admission Control *

V.G. Kulkarni
and
N. Gautam
Department of Operations Research
University of North Carolina
Chapel Hill, N.C. 27599-3180
vg.kulkarni@unc.edu
gautam@or.unc.edu

October 9, 1996

Abstract

In this paper we propose a new unified approach to address simultaneously the issues of traffic policing (using leaky buckets), admission control and network dimensioning. First we compute the effective bandwidth of the output of the leaky bucket (both buffered and unbuffered). We find a surprising result that effective bandwidth of the output of a buffered leaky bucket is independent of the token pool size. Furthermore, the output effective bandwidth exhibits discontinuous behavior as the token pool size approaches infinity. We use these results in an optimization model to find the “optimal” leaky bucket parameters. We explain how this optimization program can be used to do network dimensioning if the input traffic characteristics are known, or to do connection admission control if the network parameters are fixed.

---

*This work was partially supported by NSF Grant No. NCR-9406823, and by the Center for Advanced Computing and Communication at Duke University.


1 Introduction

The emerging high speed networks are expected to carry a broad range of traffic (video, audio and data). In the networks using asynchronous transfer mode (ATM), each traffic-source is described by its stochastic characteristics, and is assured a quality of service (QoS), as measured by cell-loss probability, delay, delay-jitter, etc. The network dimensioning, traffic policing and connection admission control is done with the aim of guaranteeing a specified quality of service (QoS) to all the users.

The concept of effective bandwidth (also known as equivalent capacity) and its use in admission control for the statistical multiplexing of bursty sources is now well-documented and accepted (See [17], [4], [11], [6], [12] and [9]). The effective bandwidth is a number associated with a traffic-source such that if the sum of the effective bandwidths of all the sources multiplexed onto a buffer is less than the output rate of that buffer, then the QoS is satisfied for each source.

The above methodology assumes that the source traffic will conform to its stated statistical properties. However, in practice we need a policing mechanism to make sure that any non-conforming source does not affect the network adversely. Due to the high data transmission speed of the network, one needs to use rate based open loop traffic policing mechanisms, rather than the "more optimal" feedback based mechanisms. (See [10], [1], [18] and [25]). One common method of policing the user traffic is called the Leaky Bucket. (See [18], [2], [19], [26], [27] and [28].)

A leaky bucket is essentially a credit management mechanism that controls the traffic entering the network as follows: Tokens enter a token pool of size $M$ at rate $\gamma$. (Tokens generated while the token pool is full are lost.) If there is a token in the token pool, an incoming packet immediately takes one and enters the network. If the token pool is empty then we have two alternative implementations:

- **Buffered Leaky Bucket**: there is an (infinite capacity) buffer where packets wait for tokens to arrive.

- **Unbuffered Leaky Bucket**: there is no buffer for the packets and any packet that does not find a token enters the network carrying a "violation" tag. Later such violation traffic can be dropped if congestion develops.
The admitted packets enter the network and may be dropped if they encounter full buffers along the way. This cell loss probability will depend upon the link speeds and the buffer sizes in the network, as well as on the route followed by the packet.

One of the important questions is how to set the parameters of the leaky bucket, viz., $M$ and $\gamma$, and the network parameters, viz., buffer sizes and link speeds. The leaky bucket sizing has been addressed in the literature by many researchers, see [3], [26], [16], etc. Bandwidth allocation in general has also received considerable attention, see [7], [13], [14], [15], etc.

It is clear from the above literature that the problems of network design, bandwidth allocation and call admission control are treated separately so far. Here we present a combined approach that treats these problems simultaneously, thus continuing the approach taken in [15] by incorporating the more recent results on effective bandwidths.

Thus the main problem we deal with in this paper concerns the optimal choice of the leaky bucket parameters $\gamma$ and $M$ for each source, the network link speeds, and the buffer sizes. The actual formulation is given in the Section 4. The rest of the paper is organized as follows: In the next section we recapitulate some of the recent results on effective bandwidths that are relevant to our work. In Section 3 we apply these results to study the effective bandwidth of the output from a leaky bucket. We study the buffered as well as the unbuffered case. These results are then used in an optimization model that is presented in Section 4. An efficient algorithm to solve the optimization problem is then described in Section 5. This section also presents numerical results illustrating many different uses of the optimization problem.

2 Preliminary Results

In this section we recapitulate some of the more recent results on effective bandwidths from [4], [5], [8], [9], [20] and [21].

2.1 Asymptotic Log Moment Generating Function of a Traffic Source

Consider a single buffer fluid model driven by a random environment process $\{Z(t), t \geq 0\}$. When the environment is in state $Z(t)$, the fluid enters the buffer at rate $r(Z(t))$. Let $X(t)$ be the amount
of fluid in the buffer at time $t$. The buffer has infinite capacity and is serviced by a channel of constant output rate $c$.

Let $A(t)$ be the total amount of fluid input from the source to the buffer in time $t$. Thus

$$A(t) = \int_0^t r(Z(u))du.$$ 

The asymptotic log moment generating function (ALMGF), $h(v)$, is defined to be

$$h(v) = \lim_{t \to \infty} \frac{1}{t} \log E\{\exp(vA(t))\}.$$  

(1)

In practice we want to bound the overflow probability of the finite buffer of size $B$ by $\epsilon$. This can be achieved if the infinite buffer process satisfies

$$\lim_{t \to \infty} P(X(t) > B) < \epsilon.$$  

(2)

In the asymptotic region

$$B \to \infty \text{ and } \epsilon \to 0, \text{ such that } -\log(\epsilon)/B \to \delta > 0,$$  

(3)

the inequality (2) is satisfied if

$$h(\delta)/\delta < \epsilon.$$  

(4)

The quantity $h(\delta)/\delta$ is called the effective bandwidth of the input source.

It is not easy to calculate the ALMGF using equation (1). However, when the environment process can be modeled as Markov Regenerative Process (MRGP) or regenerative process, we can compute their effective bandwidths in the following manner as shown in [21]. We restate these results in the next two subsections.

2.2 MRGP Sources.

Let $\{Z(t), t \geq 0\}$ be an MRGP with an embedded Markov renewal sequence $\{(Y_n, S_n), n \geq 0\}$. Assume that $\{Y_n, n \geq 0\}$ is an irreducible Discrete Time Markov Chain (DTMC) with a finite state-space $\{1, 2, \ldots, m\}$. Let

$$F_1 = \int_0^{S_1} r(Z(t))dt$$

be the total fluid generated by the source during $[0, S_1]$. Define

$$\phi_{ij}(u, v) = E\{e^{-uS_1+vF_1}; Y_1 = j \mid Y_0 = i\},$$  

(5)
for \(i, j = 1, 2, \ldots, m\) and \(-\infty < u, v < \infty\). Let

\[ \Phi(u, v) = [\phi_{ij}(u, v)] \]

be an \(m \times m\) matrix. Let \(\epsilon(u, v)\) be the largest real positive eigenvalue (the Perron-Frobenius eigenvalue) of \(\Phi(u, v)\). Define

\[ \epsilon^*(v) = \sup_{\{u : \epsilon(u, v) < \infty\}} \{\epsilon(u, v)\} \]

and

\[ u^*(v) = \inf\{u > 0 : \epsilon(u, v) < \infty\}. \]

Then for a given \(v\),

(a) if \(\epsilon^*(v) \geq 1\), \(h(v)\) is a unique solution to \(\epsilon(h(v), v) = 1\),

(b) if \(\epsilon^*(v) < 1\), \(h(v) = u^*(v)\).

### 2.3 Regenerative Source

Let \(\{Z(t), t \geq 0\}\) be a regenerative process and \(\{S_n, n \geq 0\}\), with \(S_0 = 0\), be a sequence of regeneration epochs. Then

\[ \Phi(u, v) = \hat{\phi}(u, v) = E\{e^{-uS_1+vF_1}\}, \]

and hence

\[ \epsilon(u, v) = \hat{\phi}(u, v). \]

The quantities \(\epsilon^*(v), u^*(v)\) and \(h(v)\) can be obtained from Section 2.2 on MRGP sources.

For example, consider an on-off source with general on and off times. When the source is on, traffic is generated at rate \(r\), otherwise it does not generate any traffic. The successive on and off times (generically denoted by \(U\) and \(D\), respectively) are independent. This on-off source can be modeled as a regenerative process with \(S_1 = U + D\) and \(F_1 = rU\). Hence,

\[ \hat{\phi}(u, v) = E\{e^{-u(U+D)+uvU}\} = \hat{U}(u - rv)\hat{D}(u), \]

where \(\hat{U}\) and \(\hat{D}\) are the Laplace Stieltjes Transforms (LSTs) of the on and off time distributions respectively. Hence one can find the ALMGF \((h(v))\) for general on-off sources by using the results of Section 2.2.

We shall use the results of this subsection to study the ALMGF of the output of the leaky bucket in the next section.
3 Output of a Single Leaky Bucket

Consider a single leaky bucket as shown in Figure 1. Input to the data buffer is from an exponential on-off source with on time parameter $\alpha$ and off time parameter $\beta$. When the source is on it generates traffic at rate $\tau$ and at rate 0 when off. Let $X(t)$ be the amount of traffic in the data buffer at time $t$.

There is a token pool of size $M$ into which tokens are generated continuously at a fixed rate $\gamma$. (The new tokens are discarded if the token pool is full.) Let $Y(t)$ be the number of tokens in the token pool at time $t$ ($Y(t) \leq M$).

In this subsection we characterize the output from the leaky bucket and calculate its ALMGF (and hence the effective bandwidth). This is useful since the output of the leaky bucket acts as an input to a network node. Define a process $W(t)$ as

$$W(t) = X(t) + M - Y(t).$$

We use the $W(t)$ process for two different scenarios:

- Buffered Leaky bucket (data buffer of infinite capacity).
- Unbuffered Leaky bucket (no data buffer).

Note that for the unbuffered leaky bucket $X(t) = 0$, for all $t \geq 0$, and hence $W(t)$ would be just $M - Y(t)$.
### 3.1 Buffered Leaky Bucket

A typical sample path of $W(t)$ is shown in Figure 2. Following [10] we note that $W(t)$ process behaves like the buffer content process of a single infinite buffer fluid model with an exponential on-off source with parameters $\alpha$ and $\beta$. Traffic is generated at rate $r$ when the source is on, and at rate 0 when it is off. The output rate is a constant $\gamma$. $W(t)$ increases at rate $r - \gamma$ when the source is on (for an exponential amount of time with parameter $\alpha$). When the source is off, (for an exponential amount of time with parameter $\beta$) $W(t)$ either decreases at rate $\gamma$ if $W(t) > 0$ or stays constant at 0 if $W(t) = 0$. Let $Z(t)$ be the state of the source (0 if off and 1 if on) at time $t$.

The output rate from the leaky bucket is $R(t)$ at time $t$ and is given by

$$R(t) = \begin{cases} \gamma & \text{if } W(t) \geq M \\ r & \text{if } W(t) < M \text{ and } Z(t) = 1 \\ 0 & \text{if } W(t) < M \text{ and } Z(t) = 0. \end{cases}$$

Define the following:

$$U = \inf \{ t > 0 : W(t) = 0 | W(0) = 0, Z(0) = 1 \}$$

See Figure 2 for an illustration of $U$. Let $V$ be the total amount of traffic output from the leaky bucket in time $U$. It is clear that $W(t) > 0$ and token pool is non-full during $(0, U)$. Hence the tokens enter the token pool at rate $\gamma$ during $(0, U)$. Since the token pool is full at 0 and $U$, it is clear that the total number of tokens removed from the pool over $(0, U)$ must be the same as the total number of tokens that entered the pool over $(0, U)$. Hence we get

$$V = \gamma U.$$
Theorem 1 Let $D(t)$ be the total output from the leaky bucket over $[0,t]$. The ALMGF of the output of the leaky bucket

$$h_D(v) = \lim_{t \to \infty} \frac{1}{t} \log E\{\exp(vD(t))\}$$

is given in terms of the ALMGF of the input, $h_A(v)$, as

$$h_D(v) = \begin{cases} h_A(v) & \text{if } 0 \leq v \leq v^* \\ h_A(v^*) - \gamma v^* + \gamma v & \text{if } v > v^*, \end{cases} \quad (9)$$

where

$$v^* = \frac{\beta}{r} \left( \frac{\gamma \alpha}{\beta(r - \gamma)} - 1 \right) + \frac{\alpha}{r} \left( 1 - \sqrt{\frac{\beta(r - \gamma)}{\gamma \alpha}} \right)$$

and

$$h_A(v) = \frac{1}{2} \left( rv - \alpha - \beta + \sqrt{(rv - \alpha - \beta)^2 + 4\beta rv} \right).$$

Proof: The output from a buffered leaky bucket is modulated by the bivariate process \(((W(t), Z(t)), t \geq 0)\) according to Eq. (6). Now, suppose $W(0) = 0$ and $Z(0) = 1$. Then, \(((W(t), Z(t)), t \geq 0)\) is a regenerative process that regenerates whenever it reaches the state $(0, 1)$. The length of the regenerative cycle is seen to be $S_1 = U + D$, where $U$ is as in Eq. (7) and $D$ is an $\exp(\beta)$ random variable. Now, from Eq. (8), the total output during $U$ is $\gamma U$, while the total output during $D$ is $0$. Hence the total output during the first regenerative cycle is $\gamma U$. Thus the output from the leaky bucket has the same characteristics as that of an on-off source discussed in Section 2.3, with $\gamma$ as the peak rate, $U$ as on-time and $D$ as off-time. Let $\tilde{U}$ and $\tilde{D}$ be the LSTs of $U$ and $D$ respectively. Using the results from [23] and [24] we get

$$\tilde{D}(w) = \frac{\beta}{\beta + w},$$

and

$$\tilde{U}(w) = \begin{cases} \frac{w + \beta + \gamma s_0(w)}{\beta} & \text{if } b^2 + 4w(w + \alpha + \beta)\gamma(r - \gamma) \geq 0 \\ \infty & \text{otherwise} \end{cases} \quad (10)$$

where

$$s_0(w) = -\frac{b - \sqrt{b^2 + 4w(w + \alpha + \beta)\gamma(r - \gamma)}}{2\gamma(r - \gamma)}$$

and

$$b = (r - 2\gamma)w + (r - \gamma)\beta - \gamma \alpha.$$
Thus the ALMGF of the buffered leaky bucket is identical to that of the output from a queue (with output rate $\gamma$), and is independent of $M$. This means that $M$ does not play any role as far as reducing the effective bandwidth, but acts strictly as a policing device that prevents arbitrarily large peak-rate bursts from entering the network.

We would like to comment upon a curious discontinuous behavior at this point. Although the ALMGF of the output is related to that of the input as stated in Theorem 1 for all $M < \infty$, we have $h_D(v) = h_A(v)$ if $M = \infty$. This is because the $\{(W(t), Z(t)), t \geq 0\}$ process is transient if $M = \infty$, thus making the above analysis inapplicable. However, in that case, the leaky bucket is transparent and $D(t) = A(t)$ for all $t \geq 0$, thus making the two ALMGFs identical. In practice using $M = \infty$ is never a good idea, and hence this discontinuity will not bother us in the later analysis.

### 3.2 Unbuffered Leaky Bucket

![Figure 3.](image)

In the unbuffered leaky bucket case, a packet that arrives at the leaky bucket is sent into the network with a “violation” tag if no tokens are available at the time of its arrival. We will concentrate on the untagged packets as the tagged ones would be dropped in the event of a congestion. The sample path of $W(t)$ is shown in Figure 3. Since there is no data buffer, $W(t) = M - Y(t)$ and $W(t)$ ranges from 0 to $M$. Note that $W(t)$ process is identical to a buffer content process of a fluid queue with on-off input and constant output with rate $\gamma$ and a finite buffer of size $M$.

We follow the steps of Section 3.1. The output rate from the leaky bucket is $R(t)$ at time $t$ and
is given by
\[
R(t) = \begin{cases} 
\gamma & \text{if } W(t) = M \\
r & \text{if } W(t) < M \text{ and } Z(t) = 1 \\
0 & \text{if } W(t) < M \text{ and } Z(t) = 0.
\end{cases}
\] (11)

Let \( U \) be as in Eq. (7). Then Eq. (8) remains valid in the unbuffered case. Hence the ALMGF of the output process is similar to that in Theorem 1 except that we need to use modified expressions for the LSTs \( \tilde{U} \) and \( \tilde{D} \). In the unbuffered case, the LST of the distribution of \( U \) depends on \( M \) and \( \gamma \), and can be computed using techniques of [23]. The final result is as follows:
\[
\tilde{U}(w) = E\{e^{-wU}\} = \frac{(\beta + w + \gamma s_1)e^{(w-s_1)}M(w + \beta + \gamma s_0 + \alpha \gamma s_0) - (\beta + w + \gamma s_0)(w + \beta + \gamma s_1 + \alpha \gamma s_1)}{\beta e^{(w-s_1)}M(w^2 + w \beta + w \gamma s_0 + \alpha w + \alpha \gamma s_0) + (-w^2 - w \beta - w \gamma s_1 - \alpha w - \alpha \gamma s_1)},
\]
where
\[
s_0 = (-\hat{b} - \sqrt{\hat{b}^2 + 4w(w + \alpha + \beta)\gamma (r - \gamma)})/(2\gamma (r - \gamma)),
\]
\[
s_1 = (-\hat{b} + \sqrt{\hat{b}^2 + 4w(w + \alpha + \beta)\gamma (r - \gamma)})/(2\gamma (r - \gamma)),
\]
and
\[
\hat{b} = (r - 2\gamma)w + (r - \gamma)\beta - \gamma \alpha.
\]
\( \tilde{D}(w) \) is as in the buffered case. Therefore from Section 2.3 we have that, for a given \( v > 0 \), the ALMGF of the output of the unbuffered leaky bucket, \( h_D(v) \), is computed as follows:

(a) if \( c^*(v) \geq 1 \), \( h_D(v) \) is a unique solution to \( \tilde{U}(h_D(v) - \gamma v)\tilde{D}(h_D(v)) = 1 \),

(b) if \( c^*(v) < 1 \), \( h_D(v) = u^*(v) \).

Though closed form expressions for \( h_D(v) \) are complicated, they are straightforward to obtain numerically.

When \( M = 0 \), it can be shown that the \( h_D(v) \) function reduces to the ALMGF of an on-off source with \( \exp(\alpha) \) on-times, \( \exp(\beta) \) off-times and peak rate \( \gamma \). This is as expected. However, we get the same discontinuous behavior as \( M \to \infty \) as in the buffered case, and it arises for the same reason.
We assume that there are $K$ (a fixed positive integer) sources of traffic. The $i^{th}$ ($i = 1, 2, \ldots, K$) source is an exponential on-off one with on-time parameter $\alpha_i$, off-time parameter $\beta_i$, and peak rate $r_i$. The mean input rate is $m_i = \frac{r_i\beta_i}{\beta_i + \alpha_i}$. We assume that the $i^{th}$ input source is policed by a leaky bucket with parameters $\gamma_i$ and $M_i$. We shall consider both the buffered and unbuffered leaky bucket cases.

The output from the $K$ leaky buckets is multiplexed onto a single buffer of size $B$ and constant output rate $c$. In the unbuffered case all the packets (tagged or untagged) enter the buffer. However, when the buffer gets full the tagged packets are dropped from the buffer before any of the untagged ones are affected.

### 4.1 The Optimization Problem

We now conceptually formulate the problem of selecting the parameters $M_i, \gamma_i$, ($1 \leq i \leq K$). We consider the following issues arising from the contract between the sources and the network:

- In case of the buffered leaky bucket, the contract specifies that as long as the $i^{th}$ source adheres to its agreed upon characteristics, the fraction of the traffic that faces a delay of more than a fixed amount $d_i^*$ is bounded above by $\zeta_i$. 

```plaintext
Figure 4.
```
• In the case of the unbuffered leaky bucket, the contract specifies that as long as the $i^{th}$ source adheres to its agreed upon characteristics, the fraction of the traffic that gets tagged as violation traffic is bounded above by $\zeta_i$.

• In the case of the buffered leaky bucket, the fraction of the traffic from the $i^{th}$ source that is discarded by the network due to buffer overflows is bounded above by $\epsilon_i$. (This is called the cell loss probability constraint.)

• In the case of the unbuffered leaky bucket, the fraction of the non-violation traffic from source $i$ that is discarded by the network due to buffer overflow is bounded above by $\epsilon_i$.

Now consider the following optimization problem:

\[ \text{P: Minimize } \sum_{i=1}^{K} \gamma_i \]

Subject to:

1. Waiting time or tagging constraint at the leaky bucket,

2. $m_i < \gamma_i \leq r_i$, for $1 \leq i \leq K$.

3. $\sum_{i=1}^{K} M_i \leq M^*$.

The objective function and constraint (3) need some explanation. The parameter $\gamma_i$ of the $i^{th}$ leaky bucket serves the following function: No matter how badly the source behaves, the data rate in the arbitrarily long bursts that it can send into the network is bounded above by $\gamma_i$. Thus if all sources were to simultaneously misbehave, the network will get traffic at maximum sustainable rate $\sum_{i=1}^{K} \gamma_i$. Hence it make sense to ensure that this worst case situation is kept the best possible. Similarly, the parameter $M_i$ can be thought of the largest instantaneous burst that the leaky bucket will allow from the $i^{th}$ source. Thus if $M^*$ is the largest burst that the network can handle, (for example we may set $M^* = B$ or $M^* = B/2$.) then it makes sense to add constraint number (3).

It will be seen that this turns out to be a very crucial constraint.

Constraint (1) is self explanatory, arising out of the contract stipulations. Constraint $m_i < \gamma_i$ is needed for stability, and $\gamma_i \leq r_i$ is needed to keep the the leaky bucket operation non-trivial. The quantitative expression for constraint (1), to be derived later, is valid only in the range $m_i < \gamma_i \leq r_i$. 

11
Note that the packet loss constraint in the network is not explicitly included in the above optimization problem. We show next how this constraint can be handled.

4.2 Uses of the Model

We illustrate two uses of the solution of the optimization problem $P$.

- **Determining the channel capacity**

Suppose we want to design a network to handle a given set of sources. We first choose $B$ to satisfy budget constraint or the maximum delay constraint. Then we solve $P$ for these given sources and this buffer size $B$, and obtain the optimal values of $\gamma_i$ and $M_i$ for $1 \leq i \leq K$.

Now, let $D_i(t)$ be the total output from the $i^{th}$ leaky bucket over time $[0, t)$. (For an unbuffered leaky bucket, it is defined to be the total non-violation traffic admitted by the leaky bucket over $[0, t)$. ) Using the optimal parameters $\gamma_i$ and $M_i$, use the results of Section 3.1 and 3.2 to obtain the ALMGF $h_i(t)$ of $D_i(t)$. Consider the case when all $\epsilon_i = \epsilon$. (This means all sources require the same Quality of Service.) Then it is known that the QoS criterion is satisfied if

$$\sum_{i=1}^{K} \frac{h_i(\hat{\delta})}{\hat{\delta}} < c$$

where

$$\hat{\delta} = -\frac{\log \epsilon}{\log B}.$$ 

(See [4], [20].) This result is valid in the asymptotic region

$$B \to \infty, \epsilon \to 0 \quad \text{so that} \quad -\log(\epsilon)/B \to \hat{\delta} \in (0, \infty).$$

Then the minimum capacity needed to satisfy the packet loss constraints is given by

$$c^* = \sum_{i=1}^{K} \frac{h_i(\hat{\delta})}{\hat{\delta}}.$$ 

- **Call admission control**

Suppose the capacity $c$ and the buffer size $B$ are given. Suppose we have $K$ sources requesting service. We first solve $P$ and compute the optimal parameters $\gamma_i$ and $M_i$. Using these we compute $h_i(\hat{\delta})$ as in the previous paragraph. Let

$$c^* = \sum_{i=1}^{K} \frac{h_i(\hat{\delta})}{\hat{\delta}}.$$
If \( c^* < c \) then we can admit all the sources, otherwise some will have to be denied access. Furthermore, if we have already admitted \( K \) sources, and a new \( K + 1 \)st source arrives, we resolve \( P \) and compute the new optimal parameters \( M_i \) and \( \gamma_i \) \((1 \leq i \leq K + 1)\), and compute
\[
c^* = \sum_{i=1}^{K+1} \frac{b_i(\delta)}{\delta}.
\]
If \( c^* < c \) we can admit the new source and reset the leaky bucket parameters of all the existing sources. Note that this dynamic change should be transparent to the users, since their service contract will continue to be honored.

4.3 Constraint (1)

We now state the explicit expressions for constraint (1) below for the buffered and unbuffered case.

- **Buffered Leaky Buckets**

Consider the buffered leaky bucket system where the data buffer has infinite capacity. We assume that \( d_i^* = 0 \), for all \( i \). Then delay constraint at the input buffer can be stated using the steady state analysis of the \( \{(W(t), Z(t)), t \geq 0\} \) process as (see [16])
\[
e^{-\theta_i M_i} \leq \zeta_i, \text{ for } i = 1, 2, \ldots, K,
\]
where
\[
\theta_i = \frac{r_i(\gamma_i - m_i)\alpha_i}{(r_i - \gamma_i)(r_i - m_i)\gamma_i},
\]
and \( \zeta_i \) is the expected fraction of traffic from source \( i \) that experience a non-zero delay.

- **Unbuffered Leaky Buckets**

Consider the unbuffered leaky bucket system where there are no input buffers or data buffers and packets enter the network with “violation” tag if there are no tokens available to them. The tagging constraint can be obtained by suitably modifying the delay constraint at the input buffer as stated by [16] as follows:
\[
\frac{r_i - \gamma_i}{m_i} e^{-\theta_i M_i} \leq \zeta_i, \text{ for } i = 1, 2, \ldots, K,
\]
where
\[
\theta_i = \frac{r_i(\gamma_i - m_i)\alpha_i}{(r_i - \gamma_i)(r_i - m_i)\gamma_i},
\]
and \( \zeta_i \) is the upper bound on the expected fraction of tagged traffic for source \( i \).
5 Algorithms and Numerical Results

5.1 Buffered Leaky Buckets

In this section we study the optimization problem $P$ for the buffered leaky bucket in more detail and derive an efficient numerical algorithm to solve it. First we restate $P$ as:

$$PB: \quad \min \left\{ \sum_{i=1}^{K} \gamma_i \right\},$$

subject to the constraints,

$$e^{-\theta_i M_i} \leq \zeta_i, \text{ for } i = 1, 2, \ldots, K, \quad (12)$$

$$M_1 + M_2 + \cdots + M_K \leq B, \quad (13)$$

$$\frac{r_i \beta_i}{\beta_i + a_i} < \gamma_i \leq r_i, \text{ for } i = 1, 2, \ldots, K. \quad (14)$$

Before we go ahead and employ non-linear programming techniques to solve PB, we modify the problem to simplify the analysis. The constraint

$$\frac{r_i \beta_i}{\beta_i + a_i} < \gamma_i$$

would be automatically satisfied as $\gamma_i = m_i$ will imply that $M_i = \infty$ which is not possible. Hence we can drop the above constraint. Also note that the larger the $M_i$, the smaller the $\gamma_i$ and hence the constraint

$$M_1 + M_2 + \cdots + M_K \leq B$$

will be binding. Therefore the optimality problem can be restated as follows:

$$PB1: \quad \min \left\{ \sum_{i=1}^{K} \gamma_i \right\},$$

subject to the constraints,

$$e^{-\theta_i M_i} \leq \zeta_i, \text{ for } i = 1, 2, \ldots, K, \quad (15)$$

$$M_1 + M_2 + \cdots + M_K = B, \quad (16)$$

$$\gamma_i \leq r_i, \text{ for } i = 1, 2, \ldots, K. \quad (17)$$

Define the Lagrangian

$$L(\gamma_1, \ldots, \gamma_K, M_1, \ldots, M_K) = \sum_{i=1}^{K} \gamma_i + \sum_{i=1}^{K} \lambda_i (e^{-\theta_i M_i} - \zeta_i) + \mu (\sum_{i=1}^{K} M_i - B) + \sum_{i=1}^{K} \nu_i (\gamma_i - r_i),$$

$$14$$
where $\lambda_i$, $\mu$, and $\nu_i$'s are associated Lagrange multipliers. Using the standard Lagrangian approach, it is easy to reduce the first order necessary conditions for optimality to

$$\sum_{i=1}^{K} \frac{-\log(\zeta_i)/\theta_i}{\mu_i} = B, \quad \text{and}$$

either

$$\frac{m_i(r_i - m_i)}{r_i \alpha_i (\gamma_i - m_i)^2} + \frac{r_i - m_i}{r_i \alpha_i} = \frac{-1}{\mu \log(\zeta_i)}, \quad \nu_i = 0 \quad \text{and} \quad \gamma_i < r_i,$$

or

$$\lambda_i = 0, \quad M_i = 0, \quad \gamma_i = r_i \quad \text{and} \quad \mu_i \nu_i < 0.$$

Using the above conditions, we derive the following algorithm to solve PB1. First note that as $\gamma_i$ varies from $m_i$ to $r_i$ in Eq. (19), $1/\mu$ varies from 0 to $-\alpha_i/\log(\zeta_i)$. Thus for a fixed $\mu \in (0, \max_i \{-\alpha_i/\log(\zeta_i)\})$ one can solve Eq. (19) to get

$$\gamma_i = \gamma_i = m_i + \sqrt{x},$$

where

$$x = \frac{m_i(r_i - m_i)^2}{r_i \alpha_i}, \quad \text{and}$$

$$y = -\frac{r_i - m_i}{r_i \alpha_i} - \frac{1}{\mu \log(\zeta_i)}.$$

Let

$$\gamma_i(\mu) = \min \{\gamma_i, r_i\}.$$ 

It can be seen that $\gamma_i(\mu)$ is a monotone function of $\mu$. This fact is used in the following algorithm

1. Using binary search over $\mu \in (0, \max_i \{-\alpha_i/\log(\zeta_i)\})$ solve

$$\sum_{i : \gamma_i(\mu) < r_i} -\log(\zeta_i)/\theta_i = B,$$

where $\theta_i$ is computed by using $\gamma_i = \gamma_i(\mu)$. Let the final value of $\mu$ be $\mu^*$. 

2. Set $\gamma_i^* = \gamma_i(\mu^*)$, and $M_i^* = -\frac{\log(\zeta_i)}{\theta_i^*}$, where $\theta_i^*$ is computed by using $\gamma_i = \gamma_i(\mu^*)$.

It is easy to show that the optimal solution corresponds to the objective function being minimized. We illustrate the algorithm by means of two numerical examples.

**Example 1:** Consider the case of $K$ iid sources with common parameters $\alpha$, $\beta$ and $r$. It is easy to observe that $M_i^* = M^*$ and $\gamma_i^* = \gamma^*$ for all $i$, as given below:
\[ M^* = \frac{B}{K}, \]
\[ \gamma^* = -\frac{a_2}{a_1} + \frac{\sqrt{a_2^2 - 4a_1 a_3}}{2a_1}, \]

where \( a_1 = r - m, a_2 = \frac{\alpha B}{K \zeta} - r(r - m) \) and \( a_3 = maB \gamma / K \log(\zeta) \).

Therefore using \( \gamma^* \) and \( M^* \), the ALMGF \( h(\delta) = h_i(\delta, \gamma^*, M^*) \) can be obtained from Theorem 1. The QoS criteria for cell-loss probability is satisfied if

\[ K h(\delta)/\delta < c. \] (21)

Thus in the design problem we should choose \( c \) to satisfy the above; while in the call admission problem we should keep admitting calls as long as the number of calls \( K \) in the system satisfies (21).

As a numerical example, consider the following input parameters: \( \alpha = 1, \beta = 0.4, r = 1.2, \epsilon = 10^{-7}, \zeta = 0.003, \) and \( K = 27 \). Then the optimal leaky bucket parameters are \( \gamma^* = 1.1711 \) and \( M^* = 0.1695 \), which yields \( h(\delta)/\delta = 0.7360 \). Thus Eq. (21) is satisfied if \( c > 27 \times 0.7360 = 19.87 \).

Notice that the optimal leaky bucket parameters, and hence the value of \( h(\delta)/\delta \) will change with \( K \), and hence the problem PB1 will need to be solved repeatedly for different values of \( K \).

**Example 2:** Suppose there are two types of sources. There are \( k_1 \) iid type 1 sources with \( \alpha_1 = 2.4, \beta_1 = 0.4, r_1 = 2.0 \) and \( \zeta_1 = 10^{-5} \), and \( k_2 \) iid type 2 sources with \( \alpha_2 = 1, \beta_2 = 0.4, r_2 = 1.2 \) and \( \zeta_2 = 0.003 \). The network buffer has capacity \( B = 10, \epsilon = 10^{-7} \) for both types of sources. It is clear that the optimal leaky bucket parameters will depend only on the type of the source. For a given \((k_1, k_2)\), we can obtain the optimal \( \gamma_1^*, \gamma_2^*, M_1^* \) and \( M_2^* \) by solving PB1. The effective bandwidth of the output of a type \( i \) source will be \( h_i(\delta)/\delta \). The QoS criterion is then

\[ k_1 \frac{h_1(\delta)}{\delta} + k_2 \frac{h_2(\delta)}{\delta} < c. \]

If this is satisfied we say that the pair \((k_1, k_2)\) is feasible. Table 1 gives the values of \( \gamma_1^*, \gamma_2^*, M_1^*, M_2^*, h_1(\delta)/\delta, \) and \( h_2(\delta)/\delta \) for the pairs \( \{ (k_1, k_2) : 1 \leq k_1 \leq 6, 1 \leq k_2 \leq 6 \} \). They are obtained using the algorithm mentioned in the previous subsection. This table can be used for both the design problem as well as admission control problem as follows.
<table>
<thead>
<tr>
<th>$k_2$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.861</td>
<td>0.7917</td>
<td>1.173</td>
<td>1.038</td>
<td>1.3629</td>
<td>1.1903</td>
</tr>
<tr>
<td></td>
<td>7.012</td>
<td>2.9883</td>
<td>4.498</td>
<td>1.003</td>
<td>3.3145</td>
<td>0.0566</td>
</tr>
<tr>
<td></td>
<td>0.681</td>
<td>0.6698</td>
<td>0.796</td>
<td>0.736</td>
<td>0.8356</td>
<td>0.7360</td>
</tr>
<tr>
<td>2</td>
<td>0.988</td>
<td>0.8922</td>
<td>1.215</td>
<td>1.072</td>
<td>1.3650</td>
<td>1.1920</td>
</tr>
<tr>
<td></td>
<td>5.851</td>
<td>2.0746</td>
<td>4.219</td>
<td>0.781</td>
<td>3.3022</td>
<td>0.0467</td>
</tr>
<tr>
<td></td>
<td>0.735</td>
<td>0.7100</td>
<td>0.807</td>
<td>0.736</td>
<td>0.8359</td>
<td>0.7360</td>
</tr>
<tr>
<td>3</td>
<td>1.068</td>
<td>0.9547</td>
<td>1.243</td>
<td>1.094</td>
<td>1.3665</td>
<td>1.1932</td>
</tr>
<tr>
<td></td>
<td>5.235</td>
<td>1.5882</td>
<td>4.041</td>
<td>0.639</td>
<td>3.2935</td>
<td>0.0398</td>
</tr>
<tr>
<td></td>
<td>0.763</td>
<td>0.7269</td>
<td>0.813</td>
<td>0.736</td>
<td>0.8361</td>
<td>0.7360</td>
</tr>
<tr>
<td>4</td>
<td>1.120</td>
<td>0.9966</td>
<td>1.263</td>
<td>1.110</td>
<td>1.3676</td>
<td>1.1940</td>
</tr>
<tr>
<td></td>
<td>4.855</td>
<td>1.2864</td>
<td>3.918</td>
<td>0.541</td>
<td>3.2871</td>
<td>0.0347</td>
</tr>
<tr>
<td></td>
<td>0.781</td>
<td>0.7337</td>
<td>0.818</td>
<td>0.736</td>
<td>0.8363</td>
<td>0.7360</td>
</tr>
<tr>
<td>5</td>
<td>1.156</td>
<td>1.0265</td>
<td>1.277</td>
<td>1.122</td>
<td>1.3684</td>
<td>1.1947</td>
</tr>
<tr>
<td></td>
<td>4.596</td>
<td>1.0808</td>
<td>3.828</td>
<td>0.469</td>
<td>3.2822</td>
<td>0.0307</td>
</tr>
<tr>
<td></td>
<td>0.792</td>
<td>0.7359</td>
<td>0.821</td>
<td>0.736</td>
<td>0.8364</td>
<td>0.7360</td>
</tr>
<tr>
<td>6</td>
<td>1.186</td>
<td>1.0488</td>
<td>1.288</td>
<td>1.131</td>
<td>1.3691</td>
<td>1.1953</td>
</tr>
<tr>
<td></td>
<td>4.409</td>
<td>0.9319</td>
<td>3.759</td>
<td>0.414</td>
<td>3.2783</td>
<td>0.0275</td>
</tr>
<tr>
<td></td>
<td>0.799</td>
<td>0.7360</td>
<td>0.823</td>
<td>0.736</td>
<td>0.8365</td>
<td>0.7360</td>
</tr>
</tbody>
</table>

Legend:

<table>
<thead>
<tr>
<th>$\gamma_1$</th>
<th>$\gamma_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>$M_2$</td>
</tr>
<tr>
<td>$b_1(k)$</td>
<td>$b_2(k)$</td>
</tr>
</tbody>
</table>

Table 1.

For example, suppose we want to be able to handle 4 sources of type 1 and 5 of type 2. Then for the pair $(4, 5)$ we see that the sum of the output effective bandwidths is $4 \times 0.8478 + 5 \times 0.7360 = 7.0712$. Hence we must choose $c > 7.0712$ in order to handle this traffic. On the other hand suppose $c = 6.2$ is given. Then the pair $(3, 4)$ is feasible if we use the optimal parameters from Table 1, however the pair $(3, 5)$ is infeasible. Thus the call admission can be done using a table like this. As a final example, all feasible pairs $(k_1, k_2)$ are shown in the region $R$ (including the boundary) of Figure 5 for the case of $c = 18.1$. 

17
5.2 Unbuffered Leaky Buckets

For the unbuffered leaky buckets, the optimization problem is very similar except for the tagging probability constraint. The Lagrangian method can be adopted here too to optimally solve for $M^*_i$ and $\gamma^*_i$, $(i = 1, 2, \ldots, K)$. Then the ALMGF $h_i(k, \gamma^*_i, M^*_i)$ can be obtained using the procedure in Section 3.2. Hence the design problem and the call admission problem can be solved in a manner similar to the buffered leaky bucket case.

References


