韓國經營科學會誌
第22卷 第2號
1997年 6月

Leaky Bucket 시스템에서 트래픽제어에 관한 대기행렬모형

황철희* · 이호우** · 윤승현** · 안부용** · 박노익**

A Queueing Model for Traffic Control in Leaky Bucket System

C.H. Hwang · H.W. Lee · S.H. Yoon · B.Y. Ahn · N.I. Park

Abstract

We build a queueing model for buffered leaky bucket system. First, we set up the system equations and then calculate the steady-state probabilities at an arbitrary time epoch by recursive method.

We derive the mean waiting time and the mean number of cells in the input buffer, and evaluate the performance of the buffered leaky bucket system to find the optimal queue capacity and token generation rate that meet the quality of service(QoS).

1. 서 론

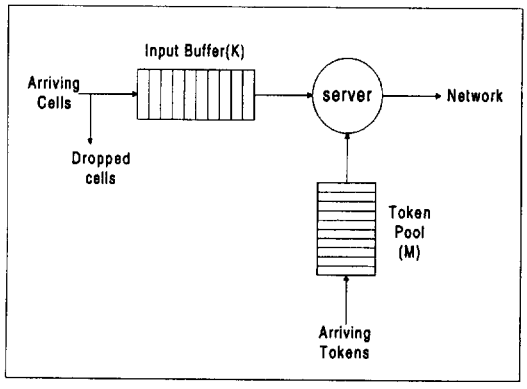
요즘 관심이 집중되고 있는 광대역 정보통신망(Broadband Integrated Service Digital Network: B-ISDN)은 영상 서비스, 고속데이타 서비스등 다양한 광대역 정보통신 서비스를 통합적으로 제공할 수 있도록 표준화시킨 통신망이다. B-ISDN의 요구 사항을 만족시켜 주기 위하여 교환(switching)과 다중화(multiplexing)를 위해 제안된 기술 중에서 패킷교환방식을 기초로 한 비동기식 전송방식(Asynchronous Transfer Mode : ATM)이 적합한 것으로 알려지고 있다.

ATM망에서 대부분의 트래픽 입력원(traffic source)은 버스티(bursty)하다. 버스티한 입력원은 아주 짧은 시간 동안에 많은 셀을 망에 보낼 수도 있고, 긴 시간 동안에 어떠한 셀도 망에 보내지 않을 수도 있다. ATM망은 다수의 트래픽

* 삼성데이터시스템
** 성균관대학교 산업공학과

입력원에서 나오는 셀에 대한 서비스를 실행하고 있다. 대역폭을 할당할 때 각 입력원의 최고 셀 발생률에 맞추어서 대역폭을 할당하는 것은 비효율적이기 때문에, 통계적 다중화를 통해 효율적인 대역폭을 할당한다. 그러나 만약 다수의 트래픽 입력원에서 동시에 많은 셀이 망에 유입되게 된다면 망에는 폭주가 일어나게 되고, 제 기능을 발휘하지 못할 수도 있다. 따라서 폭주를 제어해야할 필요성이 있다.

기존의 망을 위해 개발된 폭주제어 방식은 폭주가 일어난 이후 이에 대처하는 반응제어의 범주에서 개발되었다. 이 반응제어는 ATM망의 빠른 링크 속도와 버스티한 트래픽특성 때문에 효과적이지 못하다. 예방제어의 방법에는 호 수락제어(Call Admission Control : CAC)와 사용 파라미터제어(Usage Parameter Control : UPC)가 있다. 사용 파라미터 제어방법에는 Leaky Bucket 방법과 Window-Based Technique이 있다. 이 중에서 Leaky Bucket 방법이 사용 파라미터제어를 위한 가장 효율적인 방법으로 알려지고 있다[8].



〈그림 1〉 입력버퍼가 있는 Leaky Bucket 시스템

본 논문에서는 입력버퍼가 있는 Leaky Bucket 방법을 대기행렬 모형으로 모형화하고 시스템의 안정상태 확률을 순환방법으로 구한 후 평균 대기시간과 평균 대기고객수, 셀 손실확률 등을 구하여 Leaky Bucket 방법의 특성을 분석한다. 여기서 도착과정은 포아송과정(Poisson process)을 따르고 토큰의 생성시간은 동일한 일반분포를 따른다고 가정한다. 입력버퍼의 크기는 K로 토큰풀의 크기는 M으로 고정시킨다.

본 논문에서는 또한 도착률, 토큰생성률, 입력버퍼의 크기, 토큰풀의 크기에 대한 관계를 알아보고, 망이 요구하는 서비스 요구사항(Quality of Service; QoS)을 만족시키기 위한 최적의 입력버퍼 크기와 토큰풀의 크기, 토큰생성률의 관계 및 이들이 시스템의 성능에 미치는 영향을 분석하였다.

Leaky Bucket 방식은 1986년에 Turner[13]에 의해 처음으로 제안된 후 다양한 형태로 발전되어 왔다. Leaky Bucket의 기본적인 아이디어는 셀이 망에 들어 가기 위해서는 반드시 토큰풀(token pool)로부터 하나의 토큰을 가지고 Leaky Bucket을 빠져나가야 한다는 것이다. 〈그림 1〉에서 보는 바와 같이 Leaky Bucket 방식은 간단하게 대기행렬모형으로 표현할 수 있다. 셀이 도착하였을 때 토큰풀이 비어 있으면 도착한 셀은 입력버퍼에 쌓아 두고 토큰풀 내에 토큰이 존재하면 셀은 토큰풀에서 토큰 하나를 빼내어 Leaky Bucket을 통과하여 망으로 들어가게 된다. 토큰은 토큰 발생기에 의해 만들어져 토큰풀에 쌓이게 되고 토큰이 생성되었을 때 토큰풀이 차 있으면 만들어지는 토큰은 손실된다. 토큰이 생성될 때 입력버퍼에 기다리고 있는 셀이 있다면 입력버퍼의 첫 번째 셀은 즉시 Leaky Bucket을 빠져 나간다. 입력버퍼 크기는 유한하고 버퍼가 차 있을 때 도착하는 셀은 폐기된다.

토큰풀의 크기는 입력원이 얼마나 버스티한 트래픽인지에 따라 결정되는 변수이다. 토큰풀

크기를 크게 정하면 지연시간을 단축시키는 반면 버스트성(burstiness)이 커지고 망에서의 셀 손실확률은 증가한다. 입력원이 버스트성이 큰 트래픽을 보내는 경우, 한꺼번에 많은 셀이 짧은 시간 동안 망에 보내지게 된다. 따라서 입력 트래픽의 버스트성이 클수록 토큰풀의 크기는 커져야 한다[1],[15].

B-ISDN에서 트래픽 제어의 기본 구조를 이해하기 위해 호 설정의 절차는 Bae and Suda[3]에 잘 나타나 있다.

입력버퍼를 무한대로 두는 Leaky Bucket 모형에서는 도착한 셀은 무한 입력버퍼에 들어가고 셀 손실은 전혀 없다. 이에 대한 연구로서 Anantharam et al.[1]은 Leaky Bucket을 통과한 이후의 셀들의 버스트성을 검사하여, Leaky Bucket을 통과하면 전보다 버스트성이 감소한다는 것과 버스트성이 토큰풀 크기에 따라 증가한다는 것을 보였다. Liu et al.[10]은 입력버퍼가 있는 경우에도 같은 결과가 나타난다는 것을 보였다.

Cidon et al.[7]이 셀 지연시간과 셀 손실확률간의 절충을 위하여 입력버퍼를 제안한 이래로 입력버퍼가 있는 경우에 대한 많은 연구가 이루어졌다. Sidi et al[12]은 도착과정을 포아송과정으로 가정하고 셀 지연시간, 셀 이탈 간격분포, 셀 손실 확률 등을 구하였다. Wu et al.[16]는 도착과정을 베르누이과정으로 가정하고 Leaky Bucket 방법의 성능 분석을 하였다. Berger[5]는 도착과정이 MMPP(Markov Modulated Poisson Process)인 경우에 입력버퍼가 있는 Leaky Bucket 방법에 대해 Matrix-Geometric방법을 이용하여 분석하였다. 또한 Bala et al.[4]은 최대 셀률로 셀을 보내는 기간이 길수록 평균 셀률의 제어가 어려우며 토큰 버퍼의 크기를 조절하는 것보다 토큰 발생기의 토큰 생성속도를 조절하는 것이 더 효과적임을 보였다. Butto et al.[6]은 유체근사법(fluid approximation)을 이용하여 근사 성능 분석을 하였다. Rathgeb [11]은 Leaky Bucket방식을 $G/D/1/N$ 지연-손실시스템으로 모형화하고 셀 손실확률이 입력 트래픽의 특성에 크게 의존함을 보였다.

이전의 논문들에서는 성능 분석을 위한 방법으로 Imbedded Markov Chain기법이나 유체근사법 등을 이용하여 시스템을 분석하여 왔다. 이러한 방법으로는 정확한 안정상태 확률을 구하거나 패키지화 시키기 힘들다는 단점이 있다. 순환방법을 이용하면 이러한 단점을 극복할 수 있는데 이에 대한 연구로서 Van Hoorn[14]는 $M^X/G/1$ 대기행렬 시스템에 대하여 도착시점과 임의 시점에서의 시스템 크기에 대한 안정상태 확률분포를 구하는 순환방법을 제시하였다. Baba[2]는 $M^X/G/1$ 유한 대기행렬 시스템에 대하여 임의 시점에서의 시스템 크기에 대한 안정상태 확률분포를 구하는 순환방법을 제시하였다. Gupta와 Srini Vasa Rao[9]는 단일서어버 상태종속 도착율을 갖는 대기행렬 시스템 ($\lambda(n)/G/1/K$)에서 임의 시점에서의 시스템내 고객수에 대한 안정상태 확률을 구하기 위한 순환방법을 제시하였다.

2. 시스템 방정식

본 논문의 연구 대상은 사용 파라미터제어 방법의 한 형태인 Leaky Bucket 시스템에 관한 것이다. 입력버퍼가 있는 Leaky Bucket 시스템에 대한 설명은 다음과 같다(<그림 1> 참조).

(1) 셀이 Leaky Bucket 시스템을 빠져나가기

위해서는 한 개의 토큰을 필요로 한다.

(2) 토큰을 가지고 Leaky Bucket 시스템을 빠져나가는 데 걸리는 시간은 무시할 수 있다. 따라서 대기시간(queue waiting time)과 체제시간(system sojourn time)이 같으며 토큰풀이 비어 있지 않은 상황에서 입력버퍼에는 셀이 있을 수 없다. 또한 입력버퍼에 기다리고 있는 셀이 존재하면 토큰풀은 반드시 비어 있는 상태이다.

(3) 입력버퍼의 크기는 K이다. 이것은 대기행렬 내의 평균고객수와 셀 손실확률을 제어하기 위한 것이다.

(4) 토큰풀의 크기는 M이다. 이것은 Leaky Bucket 시스템을 빠져나가는 셀들의 버스트성을 제어하기 위한 것이다. M이 크면 Leaky Bucket 시스템을 빠져나가는 셀들의 버스트성은 커진다.

(5) 대기행렬의 운용 규칙은 다음과 같다.
- 입력버퍼가 차면 (즉 대기고객수가 K이면) 도착하는 셀은 손실된다.
- 토큰풀이 차면 (즉 대기토큰수가 M이면) 생성되는 토큰은 버린다.

Leaky Bucket 시스템에 대한 가정은 다음과 같다.

(i) Leaky Bucket 시스템에 들어오는 고객(셀)은 도착률 λ의 포아송과정을 따른다.

(ii) 토큰의 생성시간은 서로 독립이고, 동일한 일반분포(general distribution)를 따른다.

(iii) 동시에 셀이 도착하고 토큰이 생성되는 경우는 없다.

본 장에서는 Leaky Bucket 시스템에 대하여 잔여서비스시간을 부가변수로 하여 임의 시점에서의 시스템내 고객수에 대한 안정상태 확률분포를 구한다. 또한 시스템 성능척도인 손실확률, 평균대기시간을 구한다.

다음과 같은 기호와 확률들을 정의하자.

λ : 고객(셀)의 도착률
K : 입력버퍼 크기
M : 토큰풀 크기
$s(x)$: 토큰 생성 시간의 확률밀도함수(pdf)
$S^*(\theta)$: $s(x)$의 Laplace 변환(LT)
$P_{cell\ loss}$: 셀 손실확률
$P_{token\ loss}$: 토큰 손실확률
$R(t)$: 시점 t에서의 잔여 토큰생성시간
$N_1(t)$: 시점 t에 입력버퍼내에 있는 고객(셀)수
$N_2(t)$: 시점 t에 토큰풀내에 있는 토큰수

$$P_{i,j}(x,t) = Pr[N_1(t)=i, N_2(t)=j, x \le R(t) \le x+\Delta t] \quad (i=0,\ldots,K,\ j=0,\ldots,M)$$

$$P_{i,j}(x) = \lim_{t\to\infty} P_{i,j}(x,t)$$

$$P_{i,j}(t) = Pr[N_1(t)=i, N_2(t)=j] \quad (i=0,\ldots,K,\ j=0,\ldots,M)$$

위에서 정의된 확률들을 이용하면 다음과 같은 안정상태(steady state)에서의 시스템방정식을 구할 수 있다.

$$-\frac{d}{dx}P_{0,0}(x) = -\lambda P_{0,0}(x) + \lambda P_{0,1}(x) + P_{1,0}(0)s(x) \qquad (2.1)$$

$$-\frac{d}{dx}P_{0,j}(x)=-\lambda P_{0,j}(x)+\lambda P_{0,j+1}(x)+P_{0,j-1}(0)s(x) \quad (j=1,\ldots,M-1) \tag{2.2}$$

$$-\frac{d}{dx}P_{0,M}(x)=-\lambda P_{0,M}(x)+P_{0,M-1}(0)s(x)+P_{0,M}(0)s(x) \tag{2.3}$$

$$-\frac{d}{dx}P_{i,0}(x)=-\lambda P_{i,0}(x)+\lambda P_{i-1,0}(x)+P_{i+1,0}(0)s(x) \quad (i=1,\ldots,K-1) \tag{2.4}$$

$$-\frac{d}{dx}P_{K,0}(x)=\lambda P_{K-1,0}(x) \tag{2.5}$$

다음과 같은 Laplace 변환을 정의하자.

$$P^*_{i,j}(\theta)=\int_0^\infty e^{-\theta x}P_{i,j}(x)dx \quad (0\le i\le K,\ 0\le j\le M)$$

식 (2.1), (2.2), (2.3), (2.4), (2.5)에 Laplace 변환을 적용하면 다음과 같은 차등변환방정식을 구할 수 있다.

$$(\lambda-\theta)P^*_{0,0}(\theta)=\lambda P^*_{0,1}(\theta)+P_{1,0}(0)S^*(\theta)-P_{0,0}(0) \tag{2.6}$$

$$(\lambda-\theta)P^*_{0,j}(\theta)=\lambda P^*_{0,j+1}(\theta)+P_{0,j-1}(0)S^*(\theta)-P_{0,j}(0) \quad (j=1,\ldots,M-1) \tag{2.7}$$

$$(\lambda-\theta)P^*_{0,M}(\theta)=P_{0,M-1}(0)S^*(\theta)+P_{0,M}(0)S^*(\theta)-P_{0,M}(0) \tag{2.8}$$

$$(\lambda-\theta)P^*_{i,0}(\theta)=\lambda P^*_{i-1,0}(\theta)+P_{i+1,0}(0)S^*(\theta)-P_{i,0}(0) \quad (i=1,\ldots,K-1) \tag{2.9}$$

$$0=\lambda P^*_{K-1,0}(\theta)+\theta P^*_{K,0}(\theta)-P_{K,0}(0) \tag{2.10}$$

2.1. 입력버퍼내의 고객(셀)수에 대한 안정상태 확률

본 절에서는 임의시점에서 입력버퍼내에 있는 고객(셀)수에 대한 안정상태확률을 구하려고 한다.

다음과 같은 확률을 정의하자.

$$P_{i,j}=P^*_{i,j}(0)=\int_0^\infty P_{i,j}(x)dx \quad (i=0,\ldots,K,\ j=0,\ldots,M) \tag{2.11}$$

이 확률은 임의시점에서 입력버퍼내에 i개의 셀이, 토큰풀내에 j개의 토큰이 있을 결합확률(joint probability)이다.

식 (2.8)에 $\theta=0$, $\theta=\lambda$를 각각 대입하면 다음과 같은 두 식을 얻는다.

$$P_{0,M-1}(0)=\lambda P^*_{0,M}(0) \tag{2.12}$$

$$P_{0,M}(0)=\frac{P_{0,M-1}(0)S^*(\lambda)}{1-S^*(\lambda)} \tag{2.13}$$

식 (2.7)에 $\theta=\lambda$를 대입하면 다음과 같은 식을 얻는다.

$$P_{0,j-1}(0)=\frac{P_{0,j}(0)-\lambda P^*_{0,j+1}(\lambda)}{S^*(\lambda)} \quad (j=1,\ldots,M-1) \tag{2.14}$$

Laplace 변환의 θ 에 대한 h 차 미분을 다음과 같이 정의하자.

$$\frac{d^h P^*_{i,j}(\theta)}{d\theta^h} = P^{*(h)}_{i,j}(\theta)$$
$$(i=0,\ldots,K,\ j=0,\ldots,M)$$

식 (2.14)에 포함된 $P^*_{0,j+1}(\lambda)$를 구하기 위하여 먼저 식 (2.8)과 식 (2.7)에 의해서 다음과 같은 식을 구한다. 여기서 $P^{*(0)}_{0,j}(\lambda)$는 $P^*_{0,j}(\lambda)$ 이다.

$$P_{0,M}{}^{*(h)}(\lambda) = -\frac{1}{h+1}[P_{0,M-1}(0) + P_{0,M}(0)]S^{*(h+1)}(\lambda) \quad (2.15)$$
$$(h=0,\ldots,K+M-1)$$

$$P_{0,j}{}^{*(h)}(\lambda) = -\frac{1}{h+1}[\lambda P^{*(h+1)}_{0,j+1}(\lambda) + P_{0,j-1}(0)S^{*(h+1)}(\lambda)] \quad (2.16)$$
$$(j=1,\ldots,M-1\ ,\ h=0,\ldots,K+j-1)$$

식 (2.15)와 (2.16)을 순환적으로 이용하면 $P_{0,j}(0)$ $(0\le j\le M-2)$를 $P^*_{0,M}(0)$의 식으로 얻을 수 있다.

식 (2.14)로부터 다음과 같은 식을 구한다.

$$P_{0,0}(0) = \frac{P_{0,1}(0) - \lambda P^*_{0,2}(\lambda)}{S^*(\lambda)} \quad (2.17)$$

식 (2.16)과 (2.17)으로부터 다음과 같은 식을 구한다.

$$P^{*(h)}_{0,1}(\lambda) = -\frac{1}{h+1}[\lambda P^{*(h+1)}_{0,2}(\lambda) + P_{0,0}(0)S^{*(h+1)}(\lambda)] \quad (2.18)$$
$$(h=0,\ldots,K)$$

식 (2.6)에서 $\theta=\lambda$를 대입한 식과 (2.17), (2.18)로부터 다음과 같은 식을 얻는다.

$$P_{1,0}(0) = \frac{P_{0,0}(0) - \lambda P^*_{0,1}(\lambda)}{S^*(\lambda)} \quad (2.19)$$

식 (2.9)에서 $\theta=\lambda$라 하면 다음을 얻는다.

$$P_{i+1,0}(0) = \frac{P_{i,0}(0) - \lambda P^*_{i-1,0}(\lambda)}{S^*(\lambda)} \quad (2.20)$$
$$(i=1,\ldots,k-1)$$

식 (2.20)에 포함된 $P^*_{i-1,0}(\lambda)$를 구하기 위하여 우선 식 (2.6)과 식 (2.9)에 의해서 다음과 같은 식을 구한다. 여기서 $P^{*(0)}_{i,0}(\lambda)$는 $P^*_{i,0}(\lambda)$ 이다.

$$P^{*(h)}_{0,0}(\lambda) = -\frac{1}{h+1}[\lambda P^{*(h+1)}_{0,1}(\lambda) + P_{1,0}(0)S^{*(h+1)}(\lambda)] \quad (2.21)$$
$$(h=0,\ldots,K-1)$$

$$P^{*(h)}_{i,0}(\lambda) = -\frac{1}{h+1}[\lambda P^{*(h+1)}_{i-1,0}(\lambda) + P_{i+1,0}(0)S^{*(h+1)}(\lambda)] \quad (2.22)$$
$$(i=1,\ldots,K-1,\ h=0,\ldots,K-i-1)$$

식 (2.21)과 (2.22)를 순환적으로 이용하면 $P_{i+1,0}(0)$ $(0\le i\le K-2)$를 $P^*_{0,M}(0)$의 식으로 얻을 수 있다.

식 (2.7)에서 $\theta=0$이라 하면 다음을 얻는다.

$$P^*_{0,j}(0) = \frac{\lambda P^*_{0,j+1}(0) + P_{0,j-1}(0) - P_{0,j}(0)}{\lambda}$$
$$(j=1,\ldots,M-1) \quad (2.23)$$

식 (2.6)에서 $\theta=0$이라 하면 다음을 얻는다.

$$P^*_{0,0}(0)=\frac{\lambda P^*_{0,1}(0)+P_{1,0}(0)-P_{0,0}(0)}{\lambda} \quad (2.24)$$

식 (2.8)에서 $\theta=0$이라 하면 다음을 얻는다.

$$P^*_{i,0}(0)=\frac{\lambda P^*_{i-1,0}(0)+P_{i+1,0}(0)-P_{i,0}(0)}{\lambda}$$
$$(i=1,\ldots,K-1) \quad (2.25)$$

식 (2.12), (2.22), (2.23), (2.24)로부터 정리하면 다음과 같은 식을 구한다.

$$P^*_{0,j}(0)=\frac{P_{0,j-1}(0)}{\lambda} \quad (j=1,\ldots,M-1) \quad (2.26)$$

$$P^*_{i,0}(0)=\frac{P_{i+1,0}(0)}{\lambda} \quad (i=0,\ldots,K-1) \quad (2.27)$$

$P^*_{K,0}(0)$을 구하기 위하여 식 (2.10)으로부터 θ 에 대하여 1차 미분을 하고 $\theta=0$이라 하면 다음과 같은 식을 얻는다.

$$P^*_{K,0}(0)=-\lambda P^{*(1)}_{K-1,0}(0) \quad (2.28)$$

$P^{*(1)}_{K-1,0}(0)$을 구하기 위하여 식 (2.9), (2.6), (2.7), (2.8) 각각에 대해 θ 에 관하여 1차미분을 하고 $\theta=0$이라 하면 다음과 같은 식을 구한다.

$$P^{*(1)}_{i,0}(0)=[\lambda P^*_{i-1,0}(0)+P_{i+1,0}(0)S^{*(1)}(0)]/\lambda$$
$$+[\lambda P^*_{i-1,0}(0)+P_{i+1,0}(0)-P_{i,0}(0)]/\lambda^2 \quad (i=1,\ldots,K-1) \quad (2.29)$$

$$P^{*(1)}_{0,0}(0)=[\lambda P^{*(1)}_{0,1}(0)+P_{1,0}(0)S^{*(1)}(0)]/\lambda$$
$$+[\lambda P^*_{0,1}(0)+P_{1,0}(0)-P_{0,0}(0)]/\lambda^2 \quad (2.30)$$

$$P^{*(1)}_{0,j}(0)=[\lambda P^{*(1)}_{0,j+1}(0)+P_{0,j-1}(0)S^{*(1)}(0)]/\lambda$$
$$+[\lambda P^*_{0,j+1}(0)+P_{0,j-1}(0)-P_{0,j}(0)]/\lambda^2 \quad (j=1,\ldots,M-1) \quad (2.31)$$

$$P^{*(1)}_{0,M}(0)=[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(1)}(0)/\lambda+P_{0,M-1}(0)/\lambda^2 \quad (2.32)$$

따라서 식 (2.29), (2.30), (2.31)와 (2.32)를 사용하여 $P^*_{K,0}(0)$을 구할 수 있다.

그러므로 $P^*_{i,j}(0)$ $(i=0,\ldots,K, j=0,\ldots,M)$는 다음과 같은 정규화조건을 사용하여 $P^*_{0,M}(0)$의 식으로 구할 수 있다.

$$\sum_{i=0}^{K}P^*_{i,0}(0)+\sum_{j=1}^{M-1}P^*_{0,j}(0)+P^*_{0,M}(0)=1 \quad (2.33)$$

$P^*_{i,j}(0)$ $(i=0,\ldots.K, j=0,\ldots,M)$을 $P^*_{0,M}(0)$의 식으로 표현하면 다음과 같이 쓸 수 있다. 여기서 $D_{i,j}(i=0,\ldots,K, j=0,\ldots,M)$은 $P^*_{0,M}(0)$의 계수인 양의 상수이다.

$$P^*_{i,0}(0)=D_{i,0}\cdot P^*_{0,M}(0) \quad (i=0,\ldots,K) \quad (2.34)$$

$$P^*_{0,j}(0)=D_{0,j}\cdot P^*_{0,M}(0) \quad (j=0,\ldots,M) \quad (2.35)$$

따라서 식 (2.34)와 (2.35)에 의해서 식 (2.33)은 다음과 같이 쓸 수 있다.

$$\sum_{i=0}^{K}D_{i,0}\cdot P^*_{0,M}(0)+\sum_{j=1}^{M-1}D_{0,j}\cdot P^*_{0,M}(0)+P^*_{0,M}(0)$$
$$=P^*_{0,M}(0)\left[1+\sum_{i=0}^{K}D_{i,0}+\sum_{j=1}^{M-1}D_{0,j}\right]=1 \quad (2.36)$$

식 (2.36)으로 부터 $P^*_{0,M}(0)$을 다음과 같이 구할 수 있다.

$$P^*_{0,M}(0)=\frac{1}{1+\sum_{i=1}^{K}D_{i,0}+\sum_{j=0}^{M-1}D_{0,j}} \tag{2.37}$$

여기서 $P^*_{0,M}(0)$은 식 (2.11)에 의해서 $P_{0,M}$과 같다.

식 (2.34), (2.35), (2.37)로부터 임의시점에서의 시스템내 고객수에 대한 안정상태확률은 다음과 같이 구한다.

$$P_{i,j}=\begin{cases}\dfrac{1}{1+\sum_{i=1}^{K}D_{i,0}+\sum_{j=0}^{M-1}D_{0,j}} & ,\ (i=0\ ,\ j=M)\\ \dfrac{D_{i,0}}{1+\sum_{i=1}^{K}D_{i,0}+\sum_{j=0}^{M-1}D_{0,j}} & ,\ (i=1,\ldots,\ k,\ j=0)\\ \dfrac{D_{0,j}}{1+\sum_{i=1}^{K}D_{i,0}+\sum_{j=0}^{M-1}D_{0,j}} & ,\ (i=0\ ,\ j=0,\ldots,M-1)\\ 0 & ,\ o/w\end{cases} \tag{2.38}$$

위에서 구한 입력버퍼내의 셀수에 대한 안정상태확률을 구하기 위한 체계화된 알고리즘은 다음과 같다.

(단계 1 : 초기값 계산)

식 (2.12)로부터 $P_{0,M-1}(0)$ 계산

식 (2.13)으로부터 $P_{0,M}(0)$ 계산

(단계 2 : $P_{0,j}(0)$ $(j=0,\ldots,M-2)$ 계산)

식(2.15)로부터 $P^{*(h)}_{0,M}(\lambda)$ $(h=0,\ldots,K+M-1)$ 계산

식 (2.16)으로부터 $P^{*(h)}_{0,j}(\lambda)$ $(j=1,\ldots,M-1\ ,\ h=0,\ldots,K+j-1)$ 계산

식 (2.14)로부터 $P_{0,j}(0)$ $(j=0,\ldots,M-2)$ 계산

(단계 3 : $P_{i,0}(0)$ $(i=1,\ldots,K)$ 계산)

식 (2.19)로부터 $P_{1,0}(0)$ 계산

식 (2.21)로부터 $P^{*(h)}_{0,0}(\lambda)$ $(h=0,\ldots,K-1)$ 계산

식 (2.22)로부터 $P^{*(h)}_{i,0}(\lambda)$ $(i=1,\ldots,K-1\ ,\ h=0,\ldots,K-i-1)$ 계산

식 (2.20)으로부터 $P_{i,0}(0)$ $(i=2,\ldots,K)$ 계산

(단계 4 : $P^*_{0,j}(0)$ $(j=1,\ldots,M-1)$, $P^*_{i,0}(0)$ $(i=0,\ldots,K-1)$ 계산)

식 (2.26)으로부터 $P^*_{0,j}(0)$ $(j=1,\ldots,M-1)$ 계산

식 (2.27)로부터 $P^*_{i,0}(0)$ $(i=0,\ldots,K-1)$ 계산

(단계 5 : $P^*_{K,0}(0)$ 계산)

식 (2.32)로부터 $P^{*(1)}_{0,M}(0)$ 계산

식 (2.31)로부터 $P^{*(1)}_{0,j}(0)$ $(j=1,\ldots,M-1)$ 계산

식 (2.30)으로부터 $P^{*(1)}_{0,0}(0)$ 계산

식 (2.29)로부터 $P^{*(1)}_{i,0}(0)$ $(i=1,\ldots,K-1)$ 계산

식 (2.28)로부터 $P^*_{K,0}(0)$ 계산

(단계 6 : 안정상태확률 계산)

$$SUM=1+\sum_{i=0}^{K}\sum_{j=0}^{M-1}P^*_{i,j}(0)$$

$$P_{0,M}=\frac{1}{SUM}$$

$$P_{i,0}=\frac{P^*_{i,0}(0)}{SUM} \quad (i=1,\ldots,K)$$

$$P_{0,j}=\frac{P^*_{0,j}(0)}{SUM} \quad (j=0,\ldots,M-1)$$

위의 알고리즘을 이용하여 수치예를 보이기로 한다.

수치예

토큰생성시간이 확정적분포를 따르고 시스템의 모수가 다음과 같이 주어졌을 때 임의시점에서의 입력버퍼내의 고객(셀)수에 대한 안정상태확률을 구하는 예를 보인다.

고객의 도착률 (λ) : 1.0
토큰 생성시간 (D) : 1.0
입력버퍼 크기 (K) : 3
토큰풀 크기 (M) : 3

평균 서비스 시간이 D인 확정적 서비스 시간의 라플라스변환은 $e^{-D\theta}$이다. 따라서 평균 토큰 생성

시간이 1이므로 $S^*(\theta)=e^{-\theta}$ 이다. 위에서 구한 알고리즘을 사용하면 다음과 같다.

(단계 1 : 초기값 계산)

$P_{0,2}(0)=P^*_{0,3}(0)=1$

$P_{0,3}(0)=P_{0,2}(0)S^*(\lambda)/(1-S^*(\lambda))=0.581976707$

(단계 2 : $P_{0,j}(0)$ $(j=0,\ldots,M-2)$ 계산)

$P^*_{0,3}(\lambda)=-[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(1)}(\lambda)=0.581976707$

$P^{*(1)}_{0,3}(\lambda)=-1/2[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(2)}(\lambda)=0.290988353$

$P^{*(2)}_{0,3}(\lambda)=-1/3[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(3)}(\lambda)=0.193992236$

$P^{*(3)}_{0,3}(\lambda)=-1/4[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(4)}(\lambda)=0.145494177$

$P^{*(4)}_{0,3}(\lambda)=-1/5[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(5)}(\lambda)=0.116395341$

$P^{*(5)}_{0,3}(\lambda)=-1/6[P_{0,M-1}(0)+P_{0,M}(0)]S^{*(6)}(\lambda)=0.096996118$

$P_{0,1}(0)=[P_{0,2}(0)-\lambda P^*_{0,3}(\lambda)]/S^*(\lambda)=1.136305121$

$P^*_{0,2}(\lambda)=-[\lambda P^{*(1)}_{0,3}(\lambda)+P_{0,1}(0)S^{*(1)}(\lambda)]=0.709011646$

$P^{*(1)}_{0,2}(\lambda)=-1/2[\lambda P^{*(2)}_{0,3}(\lambda)+P_{0,1}(0)S^{*(2)}(\lambda)]=0.306007764$

$P^{*(2)}_{0,2}(\lambda)=-1/3[\lambda P^{*(3)}_{0,3}(\lambda)+P_{0,1}(0)S^{*(3)}(\lambda)]=0.187839157$

$P^{*(3)}_{0,2}(\lambda)=-1/4[\lambda P^{*(4)}_{0,3}(\lambda)+P_{0,1}(0)S^{*(4)}(\lambda)]=0.133604658$

$P^{*(4)}_{0,2}(\lambda)=-1/5[\lambda P^{*(5)}_{0,3}(\lambda)+P_{0,1}(0)S^{*(5)}(\lambda)]=0.103003882$

$P_{0,0}(0)=[P_{0,1}(0)-\lambda P^*_{0,2}(\lambda)]/S^*(\lambda)=1.161504089$

$P^*_{0,1}(\lambda)=-[\lambda P^{*(1)}_{0,2}(\lambda)+P_{0,0}(0)S^{*(1)}(\lambda)]=0.733301239$

$P^{(1)*}_{0,1}(\lambda)=-1/2[\lambda P^{*(2)}_{0,2}(\lambda)+P_{0,0}(0)S^{*(2)}(\lambda)]=0.307566316$

$P^{*(2)}_{0,1}(\lambda)=-1/3[\lambda P^{*(3)}_{0,2}(\lambda)+P_{0,0}(0)S^{*(3)}(\lambda)]=0.186966044$

$P^{*(3)}_{0,1}(\lambda)=-1/4[\lambda P^{*(4)}_{0,2}(\lambda)+P_{0,0}(0)S^{*(4)}(\lambda)]=0.132574339$

(단계 3 : $P_{i,0}(0)$ $(i=1,\ldots,K)$ 계산)

$P_{1,0}(0)=[P_{0,0}(0)-\lambda P^*_{0,1}(\lambda)]/S^*(\lambda)]=1.163976026$

$P^*_{0,0}(\lambda)=-[\lambda P^{*(1)}_{0,1}(\lambda)+P_{1,0}(0)S^{*(1)}(\lambda)]=0.735769166$

$P^{*(1)}_{0,0}(\lambda)=-1/2[\lambda P^{*(2)}_{0,1}(\lambda)+P_{1,0}(0)S^{*(2)}(\lambda)]=0.307584447$

$P^{*(2)}_{0,0}(\lambda)=-1/3[\lambda P^{*(3)}_{0,1}(\lambda)+P_{1,0}(0)S^{*(3)}(\lambda)]=0.18692573$

$$P_{2,0}(0)=[P_{1,0}(0)-\lambda P^*_{0,0}(\lambda)]/S^*(\lambda)=1.163986926$$

$$P^*_{1,0}(\lambda)=-[\lambda P^{*(1)}_{0,0}(\lambda)+P_{2,0}(0)S^{*(1)}(\lambda)]=0.735791307$$
$$P^{*(1)}_{1,0}(\lambda)=-1/2[\lambda P^{*(2)}_{0,0}(\lambda)+P_{2,0}(0)S^{*(2)}(\lambda)]=0.307566298$$

$$P_{3,0}(0)=[P_{2,0}(0)-\lambda P^*_{1,0}(\lambda)]/S^*(\lambda)=1.16395637$$

$$P^*_{2,0}(\lambda)=-[\lambda P^{*(1)}_{1,0}(\lambda)+P_{3,0}(0)S^{*(1)}(\lambda)]=0.735761917$$

(단계 4 : $P^*_{0,j}(0)$ $(j=1,\ldots,M-1)$, $P^*_{i,0}(0)$ $(i=0,\ldots,K-1)$ 계산)

$$P^*_{0,1}(0)=P_{0,0}(0)/\lambda=1.161504089$$
$$P^*_{0,2}(0)=P_{0,1}(0)/\lambda=1.136305121$$
$$P^*_{0,0}(0)=P_{1,0}(0)/\lambda=1.163976026$$
$$P^*_{1,0}(0)=P_{2,0}(0)/\lambda=1.163986926$$
$$P^*_{2,0}(0)=P^*_{3,0}(0)/\lambda=1.16395637$$

(단계 5 : $P^*_{K,0}(0)$ 계산)

$$P^{*(1)}_{0,3}(0)=[P_{0,2}(0)+P_{0,3}(0)]*S^{*(1)}(0)/\lambda+P_{0,2}(0)/\lambda^2=-0.581976707$$

$$P^{*(1)}_{0,2}(0)=[\lambda P^{*(1)}_{0,3}(0)+P_{0,1}(0)S^{*(1)}(0)]/\lambda+[\lambda P^*_{0,3}(0)+P_{0,1}(0)-P_{0,2}(0)]/\lambda^2=-0.581976707$$

$$P^{*(1)}_{0,1}(0)=[\lambda P^{*(1)}_{0,2}(0)+P_{0,0}(0)S^{*(1)}(0)]/\lambda+[\lambda P^*_{0,2}(0)+P_{0,0}(0)-P_{0,1}(0)]/\lambda^2=-0.581976707$$

$$P^{*(1)}_{0,0}(0)=[\lambda P^{*(1)}_{0,1}(0)+P_{1,0}(0)S^{*(1)}(0)]/\lambda+[\lambda P^*_{0,1}(0)+P_{1,0}(0)-P_{0,0}(0)]/\lambda^2=-0.581976707$$

$$P^{*(1)}_{1,0}(0)=[\lambda P^{*(1)}_{0,0}(0)+P_{2,0}(0)S^{*(1)}(0)]/\lambda+[\lambda P^*_{0,0}(0)+P_{2,0}(0)-P_{1,0}(0)]/\lambda^2=-0.581976707$$

$$P^{*(1)}_{2,0}(0)=[\lambda P^{*(1)}_{1,0}(0)+P_{3,0}(0)S^{*(1)}(0)]/\lambda+[\lambda P^*_{1,0}(0)+P_{3,0}(0)-P_{2,0}(0)]/\lambda^2=-0.581976707$$

$$P^*_{3,0}(0)=-\lambda P^{*(1)}_2(0)=0.581976707$$

(단계 6 : 안정상태 확률 계산)

$$\text{SUM}=1+\sum_{i=0}^{3}\sum_{j=0}^{2}P^*_{i,j}(0)=7.371705239$$

$P_{0,3}=1/SUM=0.135653823$

$P_{0,2}=P^*_{0,2}(0)/SUM=0.154144134$

$P_{0,1}=P^*_{0,1}(0)/SUM=0.157562471$

$P_{0,0}=P^*_{0,0}(0)/SUM=0.157897798$

$P_{1,0}=P^*_{1,0}(0)/SUM=0.157899277$

$P_{2,0}=P^*_{2,0}(0)/SUM=0.157895132$

$P_{3,0}=P^*_{3,0}(0)/SUM=0.078947365$

2.2. 손실 확률(Loss Probability)

고객(셀)손실 확률과 토큰 손실확률은 앞절에서 구한 임의시점에서의 입력버퍼내의 고객(셀)수에 대한 안정상태확률을 이용하면 쉽게 구할 수 있다. 셀 손실이 일어나는 경우는 고객(셀)이 시스템에 도착하였을 때 입력버퍼가 차 있을 때에만 가능하다. 고객(셀)이 도착하는 시점에서 입력버퍼내의 고객수와 임의시점에서의 입력버퍼내의 고객(셀)수는 입력과정이 포아송과정을 따르므로 PASTA(Poisson Arrivals See Time Average, Wolff[15])에 의하면 동일하다. 따라서 고객(셀)손실 확률은 $P_{K,0}$ 이다.

$$P_{cell\ loss}=P_{K,0}$$

토큰 손실확률은 다음의 식을 이용하여 구할 수 있다.

$$(1-P_{cell\ loss})\lambda = \frac{1}{D}(1-P_{token\ loss})$$

2.3. 임의의 고객(셀)에 대한 입력버퍼내 평균대기시간

임의의 고객(셀)에 대한 평균 입력버퍼내 대기시간은 입력버퍼내에 존재하는 고객수 분포와 Little의 공식으로부터 쉽게 구할 수 있다.

다음과 같은 확률과 기호를 정의하자.

L_q : 입력버퍼에 있는 평균 고객(셀)수

W_q : 임의의 고객(셀)에 대한 평균 대기시간

λ' : 고객(셀)의 유효 도착률(effective arrival rate) $(=\lambda(1-P_{cell\ loss}))$

앞절에서 구한 임의시점에서의 입력버퍼내의 고객(셀)수에 대한 안정상태확률로부터 입력버퍼내에 존재하는 평균대기고객수(L_q)는 다음과 같이 쉽게 구할 수 있다. 여기서는 토큰을 가지고 시스템을 빠져나가는 시간이 0이기 때문에 평균대기고객수는 평균시스템고객수와 같고 따라서 $L_q=L$ 이다.

$$L_q=\sum_{i=0}^{K} i\cdot P_{i,0}$$

Little의 공식으로부터 임의의 고객(셀)에 대한 평균 입력버퍼내 평균대기시간 (W_q)은 다음과 같이 구한다. 여기서도 평균대기시간과 평균체제시간은 같고 따라서 $W_q=W$ 이다.

$$W_q=L_q/\lambda'$$

3. 성능분석

본 장에서는 먼저 Leaky Bucket 시스템과 대기행렬시스템과의 차이점을 설명하고 도착률, 대기행렬용량, 토큰생성시간에 따라 셀 손실확률과 토큰 손실확률이 어떻게 변화하는지를 알아본 후, 셀 손실한계치를 만족시키는 최적 대기행렬

용량과 토큰 생성시간에 대하여 알아보려고 한다. Leaky Bucket 시스템에서의 토큰 생성시간은 통신시스템의 성질상 확정적분포를 따른다고 가정하였다.

Leaky Bucket 시스템과 일반적인 대기행렬시스템과의 차이점은 다음과 같다.

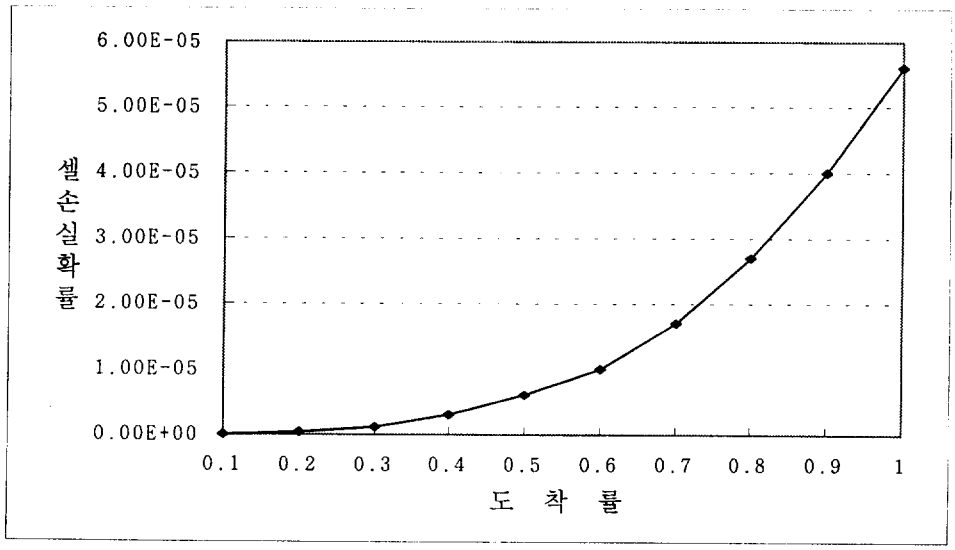
(1) 일반적인 대기행렬시스템에서는 고객이 처음 들어오면 그 때부터 서어버가 서비스를 제공하는 형태이다. 그러나 Leaky Bucket 시스템에서는 고객이 들어오지 않아도 서어버는 토큰이라는 물건을 만들어 내면서 실질적으로는 미래의 고객에 대해 서비스를 제공한다.

(2) 도착과정이 포아송과정을 따르고 토큰 생성시간이 일반적인 재생과정(renewal process)을 따른다고 가정하자. 이때 입력버퍼 크기(K)와 토큰풀의 크기(M)의 합이 같다면 셀 손실확률과 토큰 손실확률은 같다. 다만 입력버퍼가 커지면 입력버퍼내 평균 대기시간이 길어지고 토큰풀의 크기가 크면 Leaky Bucket 시스템을 통과하는 셀들의 버스트성이 커지게 된다(Berger [5]).

3.1. 도착률의 변화

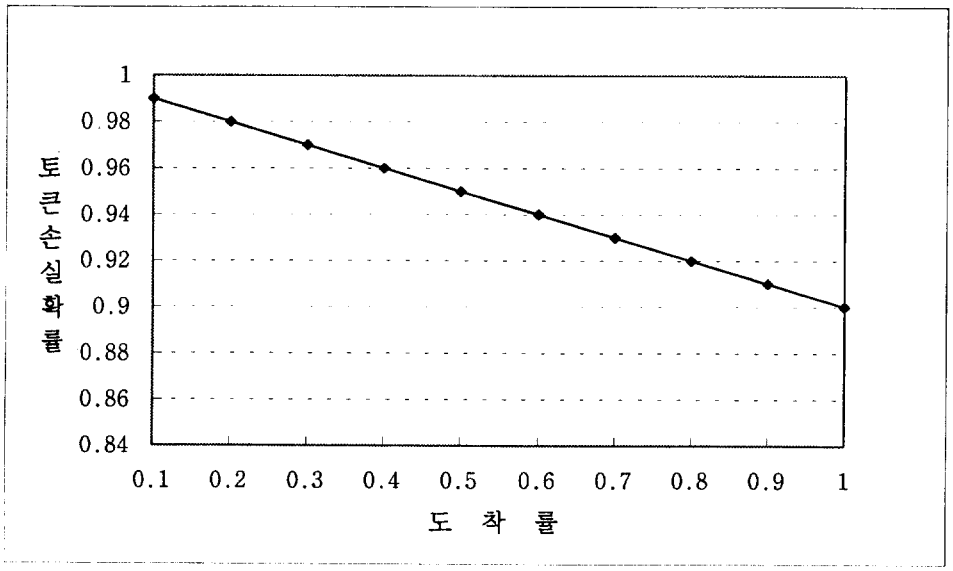
본 절에서는 입력버퍼, 토큰풀의 크기 및 토큰 생성시간을 고정시키고 도착률의 변화에 따른 셀 손실확률과 토큰 손실확률의 변화에 대하여 알아보고자 한다.

평균 토큰 생성시간이 비교적 작을때 (즉 토큰 생성률이 비교적 클때)를 가정해보자. 셀 손실확률은 도착률이 작을 때는 완만하게 증가하다가 도착률이 높아지면 급격하게 증가하고(〈그림 2〉, 〈그림 3〉) 토큰 손실확률은 도착률이 높아짐에 따라 완만하게 선형적으로 감소한다. 반면에 평균 토큰생성시간이 비교적 크다면 (즉



〈그림 2〉 도착률의 변화에 따른 셀 손실확률

토큰 생성시간 = 0.1, $K+M$ = 3



〈그림 3〉 도착률의 변화에 따른 토큰 손실확률
토큰 생성시간 = 0.1, $K+M$ = 3

토큰생성률이 비교적 작다면) 셀 손실확률과 토큰 손실확률은 토큰 생성시간이 작을 때보다 더 급격하게 변화한다.

대기행렬용량이 비교적 크다면 셀 손실확률은 대기행렬용량이 작은 것보다는 조금 줄어들고 토큰 손실확률은 대기행렬용량의 변화에 둔감하지만 토큰 생성시간의 변화에는 민감하다.

3.2. 입력버퍼 크기(K)와 토큰풀 크기(M)의 변화

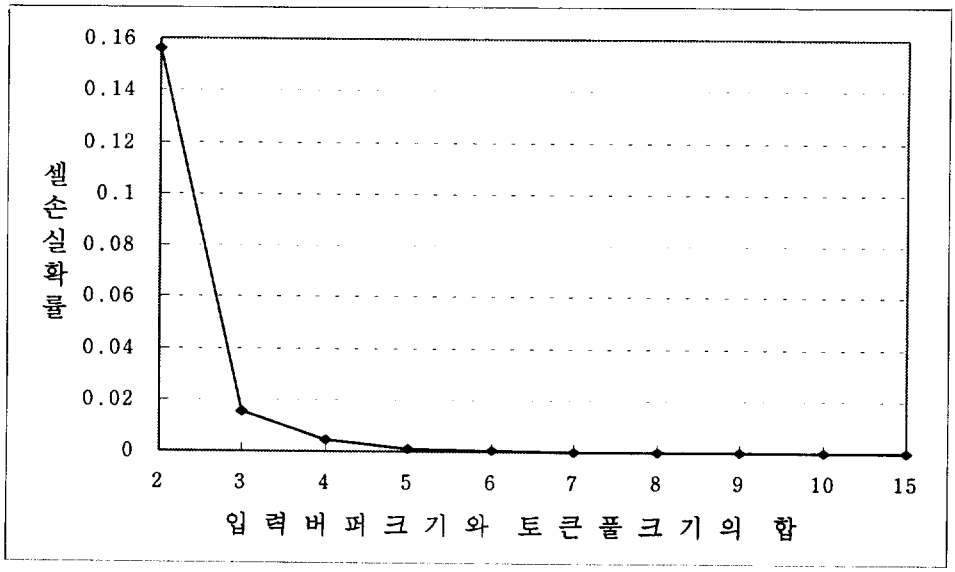
본 절에서는 도착률, 토큰 생성시간을 고정시키고, 입력버퍼 크기(K)와 토큰풀 크기(M)의 변화에 따른 셀 손실확률과 토큰 손실확률의 변화에 대하여 알아보고자 한다. 셀 손실확률과 토큰 손실확률은 입력버퍼와 토큰풀의 크기의 합의 변화에 의존한다.

도착률이 비교적 작다고 가정을 하면 다음과 같은 결과가 나타난다. 셀 손실확률은 대기행렬용량이 작을 때는 급격하게 감소하다가 대기행렬용량이 커질 때는 0에 가까워지고 토큰 손실확률은 대기행렬용량이 커짐에 따라 조금 줄어들다가 더 이상 변화가 생기지 않는다. 반면에 도착률이 비교적 크면 셀 손실확률과 토큰 손실확률은 도착률이 작을 때보다는 더 느린 속도로 변화한다.(〈그림 4〉, 〈그림 5〉)

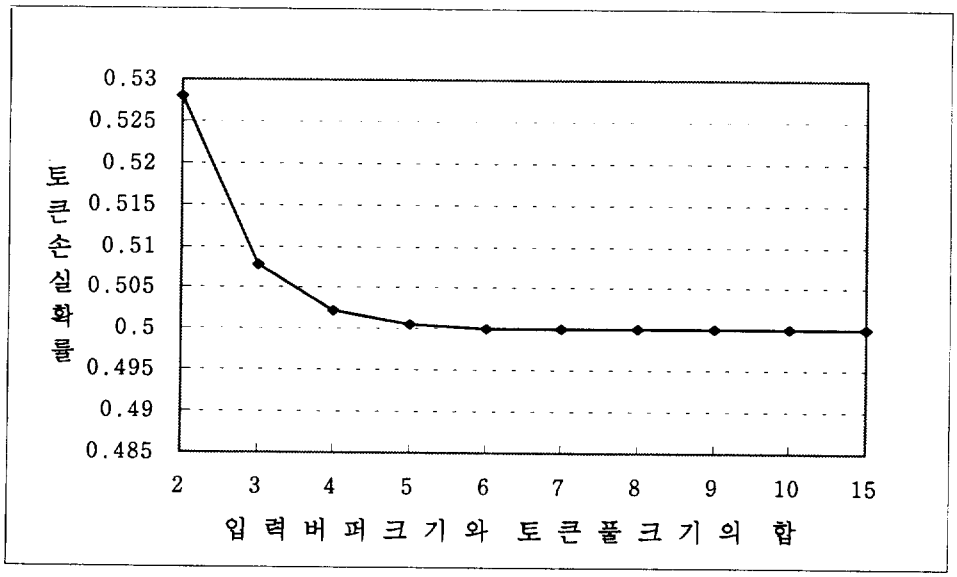
이와 같은 결과를 통해 대기행렬용량은 어떠한 한계치보다 크면 시스템의 성능을 향상시키는데 별로 도움이 되지 못하고 비용만 낭비한다는 것을 알 수 있다.

3.3. 토큰 생성시간의 변화

본 절에서는 도착률과 대기행렬용량(입력버퍼 크기+토큰풀 크기)를 고정시키고 토큰 생성시간 D의 변화에 따른 셀 손실확률과 토큰 손실확률의 변화에 대하여 알아보고자 한다.

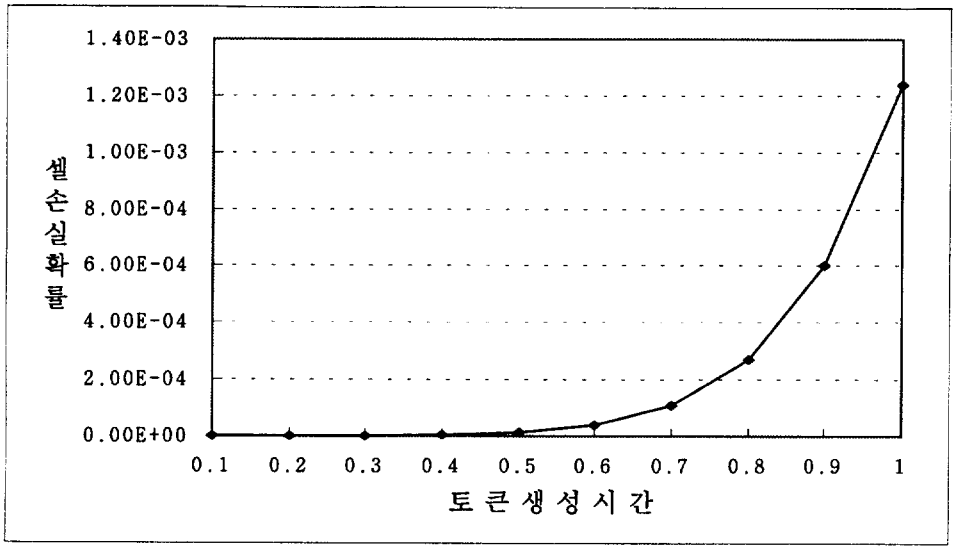


〈그림 4〉 *K*와 *M*의 변화에 따른 셀 손실확률
도착률 = 1, 토큰 생성시간 = 0.5

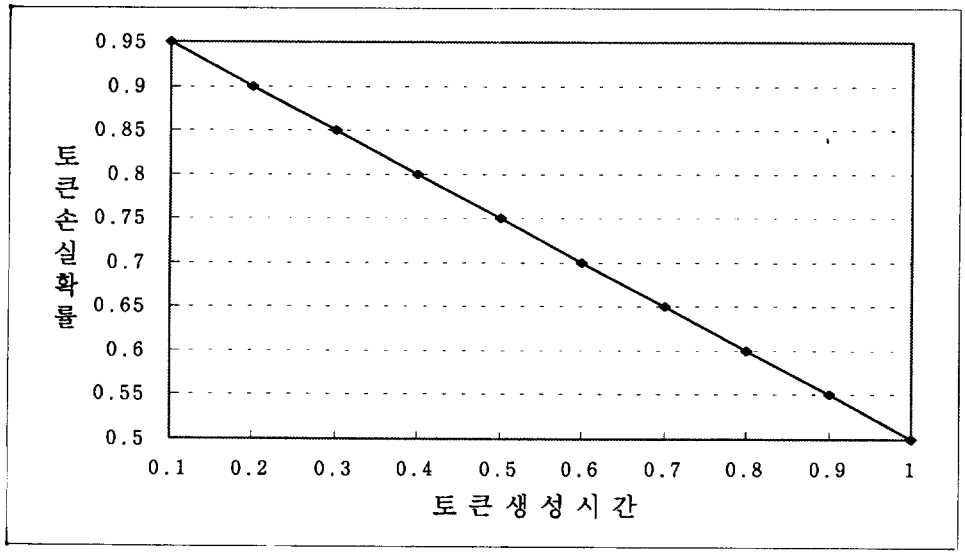


〈그림 5〉 *K*와 *M*의 변화에 따른 토큰 손실확률
도착률 = 1, 토큰 생성시간 = 0.5

도착률이 비교적 작다고 가정하자. 셀 손실확률은 토큰 생성시간이 작을 때는 완만하게 증가하다가 토큰 생성시간이 높아지면 급격하게 증가하고(〈그림 6〉, 〈그림 7〉) 토큰 손실확률은 토



〈그림 6〉 토큰 생성시간의 변화에 따른 셀 손실확률
도착률 = 0.5, $K+M$ = 5



〈그림 7〉 토큰 생성시간의 변화에 따른 토큰 손실확률
도착률 = 0.5, $K+M$ = 5

큰 생성시간이 커짐에 따라 완만하게 선형적으로 줄어든다. 반면에 도착률이 비교적 높다면 셀 손실확률과 토큰 손실확률은 도착률이 작을 때보다 더 급격하게 변화한다.

대기행렬용량이 비교적 크다면 셀 손실확률은 대기행렬용량이 작은 것보다는 조금 줄어들고

토큰 손실확률은 대기행렬용량의 변화에 둔감하지만 토큰 생성시간이 증가함에 따라 토큰 손실확률이 조금 줄어든다.

이와 같은 결과를 통해 분석을 해보면 토큰 생성시간을 조절하는 것은 셀 손실확률이나 토큰 손실확률에 큰 변화를 가져다주는 것을 알 수 있다. 따라서 시스템의 성능을 변화시키기 위하여 대기행렬용량을 변화시키는 것보다는 가능하면 토큰 생성속도를 변화시키는 것이 훨씬 효율적이라는 것을 알 수 있다.

3.4. 최적 대기행렬용량과 토큰 생성시간

본 절에서는 셀 손실확률 한계치를 만족시키는 최적 입력버퍼 크기와 토큰풀 크기의 합을 구하고, 토큰 생성시간의 구간을 구한다. 이러한 최적화문제를 풀기 위해서는 목적 함수의 성격을 알아야 하지만 목적함수가 어떤 성격을 갖고 있는지 알 수 없기 때문에 최적해를 찾을 수 없었다. 따라서 본 연구에서는 프로그램상에서 대기행렬용량과 토큰 생성시간의 모든 경우에 대해서 셀 손실확률의 한계치를 만족시키는 최적해를 구하였다. Leaky Bucket 시스템을 빠져나가는 셀들의 버스트성은 토큰 손실확률로써 나타내었다. 토큰 손실확률이 높다면 토큰풀에 토큰이 차 있을 확률이 높다는 것을 의미하며 이것은 Leaky Bucket에 들어오는 셀이 즉시 이 시스템을 빠져나갈 확률이 높다는 것을 의미한다. Leaky Bucket에 들어오는 셀이 즉시 빠져나간다면 Leaky Bucket 시스템으로 인해 셀들의 버스트성을 통제하지 못하기 때문에 버스트성은 커진다. 셀 손실확률의 한계치는 10^{-10}으로 정하였다.

3.4.1. 최적 대기행렬용량

셀 손실확률 한계치를 만족시키는 최적 대기행렬용량을 구하려고 한다. 도착률을 0.1, 0.5로 두고 토큰 생성시간에 따른 최적 대기행렬용량을 구하였다.

셀 손실확률 한계치를 만족시키는 최적 대기행렬용량은 토큰 생성시간이 크면 클수록 증가하는데 도착률이 작을 때는 천천히 증가하고, 도착률이 클 때는 급속도로 빨리 증가하는 경향이 있다(〈표 1〉, 〈표 2〉). 도착률이 작을 때는 토큰 손실확률이 비교적 크므로 셀들의 버스트성은 향상되지 않지만, 도착률이 클 때는 셀들의 버스트성이 많이 향상됨을 알 수 있다.

〈표 1.〉 셀 손실확률 한계치를 만족시키는 최적 대기행렬용량($K+M$)과 토큰 손실확률

도착률 = 0.1, 셀 손실확률 한계치 = 10^{-10}

D	최적대기행렬용량	토큰 손실확률	D	최적대기행렬용량	토큰 손실확률
D=0.1	4	0.990000	D=0.6	6	0.940000
D=0.2	5	0.980000	D=0.7	6	0.930000
D=0.3	5	0.970000	D=0.8	7	0.920000
D=0.4	6	0.960000	D=0.9	7	0.910000
D=0.5	6	0.950000	D=1.0	7	0.900000

〈표 2.〉 셀 손실확률 한계치를 만족시키는 최적 대기행렬용량($K+M$)과 토큰 손실확률

도착률 = 0.5, 셀 손실확률 한계치 = 10^{-10}

D	최적대기행렬용량	토큰 손실확률	D	최적대기행렬용량	토큰 손실확률
D=0.1	6	0.950000	D=0.6	12	0.700000
D=0.2	7	0.900000	D=0.7	13	0.650000
D=0.3	8	0.850000	D=0.8	15	0.600000
D=0.4	9	0.800000	D=0.9	16	0.550000
D=0.5	11	0.750000	D=1.0	18	0.500000

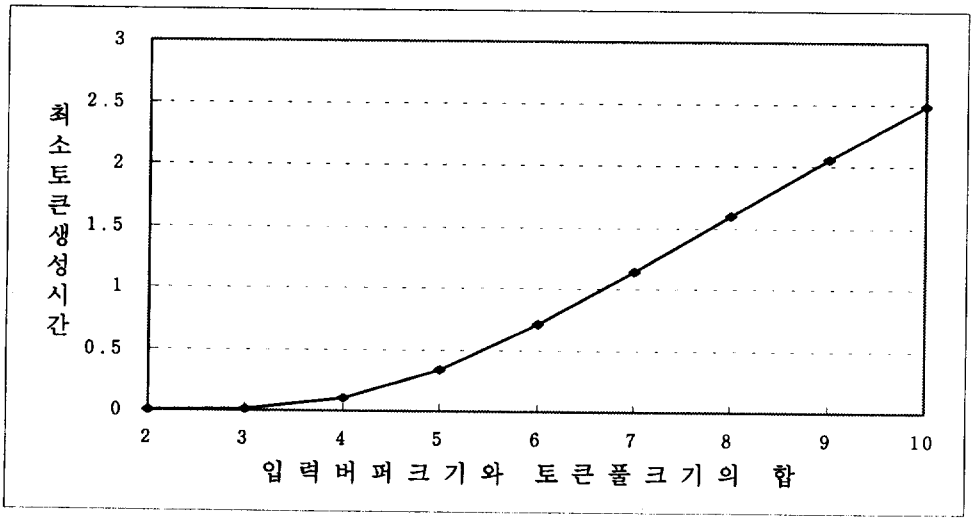
3.4.2. 최적 토큰 생성시간

셀 손실확률 한계치를 만족시키는 최적 토큰 생성시간의 구간을 구하려 한다. 도착률을 0.1, 0.5, 1.0으로 두고 대기행렬용량에 따른 최적 토큰 생성시간의 구간을 구하였다.

대기행렬용량이 커짐에 따라 최소 토큰 생성시간은 처음에는 완만하게 증가하다가 점차 직선의 형태로 바뀌고(〈표 3〉, 〈그림 8〉) 도착률이 커짐에 따라 최소 토큰 생성시간이 급격하게 감소하다가 완만하게 감소하는 것을 알 수 있다(〈그림 9〉).

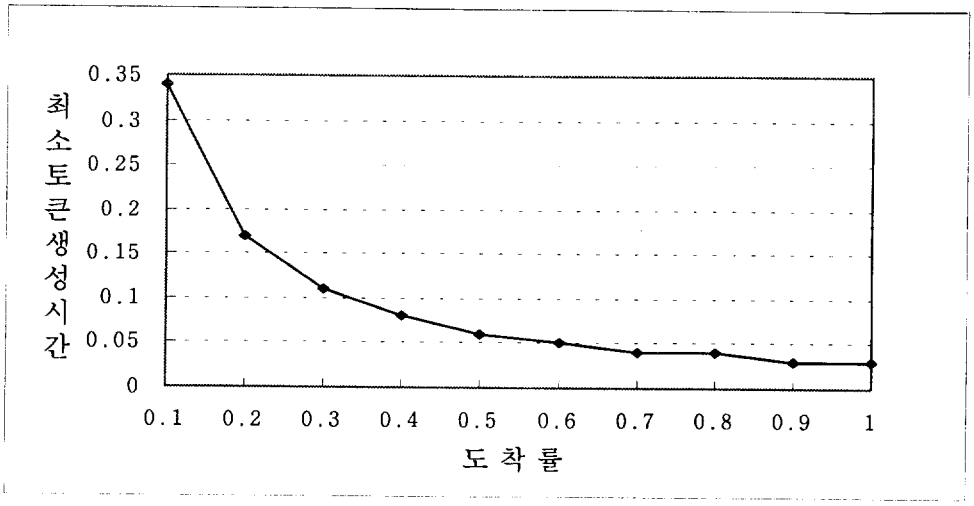
4. 결 론

본 논문에서는 입력버퍼가 있는 Leaky Bucket 시스템에 대하여 임의시점에서의 입력버퍼내의 고객(셀)수에 대한 안정상태확률, 평균 대기시간



〈그림 8〉 셀 손실확률 한계치를 만족시키는 최적 토큰생성시간 구간

도착률 = 0.1, 셀 손실확률 한계치 = 10^{-10}



〈그림 9〉 셀 손실확률 한계치를 만족시키는 최적 토큰생성시간 구간

$K+M = 5$, 셀 손실확률 한계치 = 10^{-10}

과 대기고객수를 구하였다. 또한 서비스 시간이 확정분포를 따르는 경우에 대해서 시스템의 특성 분석을 하였으며 최적 대기행렬용량(입력버퍼 크기+토큰풀 크기)과 최적 토큰 생성시간의 상·하한을 구하였다.

본 논문에서는 부가변수법을 적용하여 모형화를 시도하였으며 안정상태확률을 구하기 위해 순환방법을 사용하였다. 또한 평균 대기시간과 평균 대기고객수를 구하고 성능평가를 하였다.

4.1. 주요결과

주요 결과를 요약하면 다음과 같다.

(1) 도착률이 증가하는 경우에 셀 손실확률은 증가하고 토큰 손실확률은 감소한다. 일반적인 Leaky Bucket이 없는 시스템에서는 도착률이 증가할 때 다른 조건들이 모두 같다면 셀 손실이 많이 일어나고 버스트성도 커지지만 Leaky Bucket이 있는 시스템은 Leaky Bucket이 없는 시스템보다 비교적 버스트성이 좋아진다.

(2) 대기행렬용량(입력버퍼 크기+토큰풀 크기)이 증가하는 경우에는 셀 손실확률과 토큰 손실확률은 어떤 한계치까지만 감소한다. 도착률이 높다면 한계치에 빨리 도달하는 경향이 있다.

(3) 토큰 생성시간이 커지면 셀 손실확률은 처음에는 완만하게 증가하다가 점점 급격하게 증가하고, 토큰 손실확률은 선형적으로 감소한다. 도착률이 크면 셀 손실확률이 감소하는 속도와 토큰 손실확률이 증가하는 속도가 빨라진다.

(4) 최적 대기행렬용량은 도착률이 높아지면 커지고 또한 토큰 생성시간이 길어져도 커진다. 만약 도착률이 높다면 최적 대기행렬용량은 급격히 증가하고 또한 셀의 버스트성도 많이 좋아진다.

(5) 최적 토큰 생성시간의 구간은 도착률이 커지면 급격하게 줄어들다가 서서히 완만하게 줄어들고, 대기행렬용량이 커지면 완만하게 증가하다가 선형적으로 증가하는 것을 볼 수 있다.

(6) 만약 토큰 생성시간의 조절이 가능하다면

〈표 3〉 셀 손실확률의 한계치를 만족시키는 최적 토큰 생성시간구간

셀 손실확률 한계치 = 10^{-10}

도착률	대기행렬용량	최적 토큰생성시간 구간
0.1	M+K=2	0 < D < 0.0001
	M+K=3	0 < D < 0.0133
	M+K=4	0 < D < 0.10
	M+K=5	0 < D < 0.34
	M+K=6	0 < D <0.71
	M+K=7	0 < D < 1.14
	M+K=8	0 < D < 1.60
	M+K=9	0 < D < 2.06
	M+K=10	0 < D < 2.49
0.5	M+K=2	0 < D < 0.00004
	M+K=3	0 < D < 0.0026
	M+K=4	0 < D < 0.02
	M+K=5	0 < D < 0.06
	M+K=6	0 < D < 0.14
	M+K=7	0 < D < 0.22
	M+K=8	0 < D < 0.32
	M+K=9	0 < D < 0.41
	M+K=10	0 < D < 0.49
1.0	M+K=2	0 < D < 0.00002
	M+K=3	0 < D < 0.0013
	M+K=4	0 < D < 0.01
	M+K=5	0 < D < 0.03
	M+K=6	0 < D < 0.07
	M+K=7	0 < D < 0.11
	M+K=8	0 < D < 0.16
	M+K=9	0 < D < 0.20
	M+K=10	0 < D < 0.24

대기행렬용량을 늘리는 것보다는 토큰 생성시간을 조절하므로써 더 효율적인 제어가 가능하다는 것을 알 수 있다.

4.2. 추후 연구 과제

본 연구의 목적은 Leaky Bucket 시스템의 안정상태확률을 구하고 특성을 분석하는 것이었다. 이 모형을 실제 시스템에 적용하기는 몇 가지 난점이 있다. 일반적인 통신 시스템의 도착과정은 포아송과정에 잘 맞지 않는 경향이 있다. 따라서, 다른 도착과정에 대하여 연구가 필요하고 또한 고객이 집단으로 도착하는 집단 도착대기행렬 시스템에 관한 연구도 필요하다.

더 좋은 효율을 추구하기 위하여 Leaky Bucket 시스템이 여러 가지 형태로 바뀌어 가고 있는데 이러한 시스템에 대해서도 연구가 필요하다.

참 고 문 헌

[1] Anantharam, V. and Konstantopoulos, T., "Burst Reduction Properties of the Leaky Bucket Flow Contol Scheme in ATM Networks", IEEE Transactions On Communication, 42, No.12, 3085-3089, 1994.

[2] Baba, Y., "The $M^X/G/1$ Queue with Finite Waiting Room", Journal of the Operations Research Society of Japan, 27, No.3, 260-272, 1984.

[3] Bae, J.J. and Suda, T., "Survey of Traffic Control Schemes and Protocols in ATM Networks", Proceedings of the IEEE, 79, No.2, 170-189, 1991.

[4] Bala, K., Cidon, I. and Sohraby, K, " Congestion Control for High Speed Packet Switches Networks", IEEE INFOCOM '90, 520-526.

[5] Berger, A.W., "Performance Analysis of a Rate Control Throttle where Tokens and Jobs Queue", IEEE INFOCOM '90, 30-38.

[6] Butto, M., Cavallero, E. and Tonietti, A.,

"Effectiveness of the Leaky Bucket Policing Mechanism in ATM Networks", IEEE Journal on Selected Areas in Communications, 9, No.3 335-342, 1991.

[7] Cidon, I., and Gopal, I.S., "PARIS: An approach to Integrated High -Speed Private Networks", Int. J. Digital & Cabled Systems , 1, No.49, 77-86, 1988.

[8] Gallassi, G., Rigolio, G. and Fratta, L., "ATM: Bandwidth Assignment and Bandwidth Enforcement Policies", GLOBECOM '89, 1788-1793.

[9] Gupta, U.C. and Srini Vasa Rao, T.S.S, "Computing Steady State Probabilities in $\lambda(n)/G/K$ Queue", Working paper, Dept. of Mathematics, Indian Institute of Technology, June, 1993.

[10] Liu, Z. and Towsley, D., "Burst Reduction Properties of Rate-Control Throttles: Downstream Queue Behavior", IEEE Transactions On Networking, 3, No.1, 82-90, 1995.

[11] Rathgeb, E., "Modeling and Performance Comparison of Policing Mechanisms for ATM Networks", IEEE Journal on Selected Areas in Communications, 9, No.3, 325-324, 1991.

[12] Sidi, M., Liu, W.Z., Cidon, I. and Gopal, I., "Congestion Control through Input Rate Regulation", GLOBCOM '89, 1764-1773.

[13] Turner, J.S., "New Direction in Communications", IEEE Comm. Mag., Oct. 1986.

[14] Van Hoorn, M.H., "Algorithms for the State Probabilities in a General Class of Single Server Queueing Systems with Group Arrivals", Management Science, 27, No.10, 1178-1187, 1981.

[15] Wolff, R.W., "Poisson Arrivals See Time Average", Operations Research, 30, No.2, 223-231, 1986.

[16] Wu, G. and Mark, J.W. "Discrete Time Analysis of Leaky Bucket Congestion Control", IEEE ICC, 1196-1200, 1992.