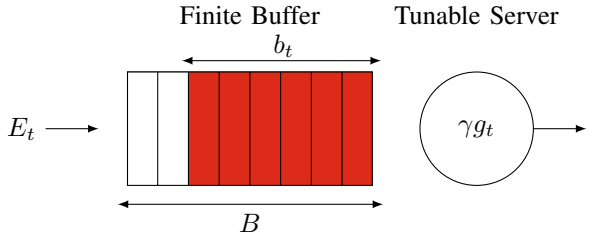
Performance Limits of Tunable Servers with Finite Buffer Capacities

***Abstract*—We consider the problem of a tune-able server with finite buffer size of $\bar{B}$. We provide the performance limits in a stochastic setting where the packet arrivals happen with some underlying probability distribution unknown to us. We first concentrate on a simple Bernoulli packet arrival process where at each step, either packets of size $\bar{B}$ arrives with a probability p, or no packet arrival takes place at all, independent of other time steps. We analyze the performance of a near optimal policy and give lower and upper bounds for the case of Bernoulli distribution of packet arrivals. We also provide a universal lower bound on the performance of any offline/online policy.**

***Keywords*—Renewal Reward, Expectation, Online Algorithm, Bernoulli Processes**

I. Introduction

We consider a single server with a tune-able speed. We assume that the cost associated with the serving of packets is a linear function of the number of packets being served. Let us say that the speed at which the server operates is given by $s = \gamma g_t$ where g_t represents the number of packets being served. Let E_t represent the number of packets arriving at each time t. The pictorial description of the problem is given below.



II. System Model

At each time step t, we assume that E_t is the number of packets arriving at the queue and stored in the buffer of maximum size $\bar{B}$. The problem is to minimize the cost of servicing the packets which is given as some convex function of the speed of servicing the packets. We assume the convex function is a quadratic function and the cost of servicing the packets at time t is given by $c_t = (\gamma g_t)^2$. The total cost incurred in servicing the packets is given by $\frac{1}{n}\sum_{t=1}^{n} c_t$. To enforce the servicing of packets, we add an additional constraint of keeping the probability of dropping packets at less than α. We formulate the problem as follows:

$$\begin{aligned} \underset{s}{\text{minimize}} \quad & \lim_{n\to\infty} \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} c(s_t)\right] \\ \text{subject to} \quad & P_{dropping} \leq \alpha \end{aligned} \tag{1}$$

This can be equivalently written as:

$$\begin{aligned} \underset{s}{\text{minimize}} \quad & \lim_{n\to\infty} \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} c(s_t)\right] \\ \text{subject to} \quad & \sum_{t=1}^{n} g_t \geq (1-\alpha)\sum_{t=1}^{n} E_t \end{aligned} \tag{2}$$

III. λ-Fraction Policy: An Approximate Online Policy

The inspiration for our policy comes from that used in **[Ayfer]** where they consider the problem of power control in an energy harvesting node, where they have to maximize a concave, increasing function. Their policy is to use a *fixed-fraction* of the power left in the battery at each time instant.

We follow a similar policy of servicing a *fixed-fraction* of the packets in the buffer at each time t. Let us call this *fixed-fraction* λ and our policy as λ-Fraction Policy.

$$\boxed{\lambda = \frac{(1-\alpha)p}{\alpha + (1-\alpha)p}} \tag{3}$$

IV. Bounding the Performance

We start by finding a lower bound on the cost function which is irrespective of any policy being followed by just taking the hard probability of packet drops into account. The proof follows by finding a lower bound on the expectation of the packets being serviced averaged over time and then use the Jensen's inequality to lower bound our cost function. For finding the upper bound on the cost function, we find the performance of our suggested policy for the case of Bernoulli distribution of packet arrivals. We then show that our policy is a feasible policy for other distributions.

A. Tight Lower Bound for Bernoulli Process:

a) Proposition:: For $\alpha << 1, p << 1$, there exists no online policy that has a cost which is lower than $\Theta\left(\frac{B^2p^2}{p+\alpha}\right)$

V. Present Work

Right now, we are attempting to tighten the lower bound. At the same time since the λ- policy is too fast for $0, x$ Bernoulli distributions, we are looking to reduce the number of packets being sent out.