Trading link utilization for queueing delays: an adaptive approach

Rade Stanojević, Robert Shorten
Hamilton Institute, NUIM, Ireland

Abstract

Understanding the relationship between queueing delays and link utilization for general traffic conditions is an important open problem in networking research. Difficulties in understanding this relationship stem from the fact that it depends on the complex nature of arriving traffic and the problems associated with modelling such traffic. Existing AQM schemes achieve a "low delay" and "high utilization" by responding early to congestion without considering the exact relationship between delay and utilization. However, in the context of exploiting the delay/utilization tradeoff, the optimal choice of a queueing scheme's control parameter depends on the cost associated with the relative importance of queueing delay and utilization. The optimal choice of control parameter is the one that maximizes a benefit that can be defined as the difference between utilization and cost associated with queuing delay. We present two practical algorithms, Optimal Drop-Tail (ODT) and Optimal BLUE (OB), that are designed with a common performance goal: namely, maximizing this benefit. Their novelty lies in fact that they maximize the benefit in an online manner, without requiring knowledge of the traffic conditions, specific delay-utilization models, nor do they require complex parameter estimation. Packet level ns2 simulations are given to demonstrate the efficacy of the proposed algorithms and the framework in which they are designed.

Key words: Internet routers, Buffer management, Delay/utilization tradeoff, Network utility

1. Introduction

Measurements from a number of sources suggest that traffic generated by TCP users accounts for 85-95% of the Internet traffic [5,7]. Van Jacobson's congestion control algorithm [14] is one of the main reasons for the robustness of the Internet and the prevention of the congestion collapse over last two decades. Given the success of TCP and the stability of the current Internet, it is unlikely that other, conceptually different, transport protocols will replace TCP in the near future.

Most Internet routers use FIFO Drop-Tail buffers. Current router buffers are generally sized by the rule-of-thumb given in the Villamizar&Song paper [30]: router buffers require approximately space for $B = \overline{RTT} \times C$ packets, where $\overline{RTT}$ is the "average" round trip time for connections that use the link and C is capacity of the link. Following this rule, most router buffers are designed in such a fashion that they result in up to $100ms$ to $250ms$ of queueing. This, together with TCP's mechanism of congestion avoidance, serves to ensure a high link utilization. In the last few years several studies related to buffer sizing for congested routers have appeared. It is claimed in [1] that the amount of buffer space required for high link utilization can in some circumstances be far less than that suggested by the Bandwidth-Delay-Product rule. However, it is also shown in this paper that the required buffering highly depends on the number of active flows using the link. In particular, briefly, assuming a single congested link topology, and N homogenous, unsynchronized, long TCP flows, with a "typical" [1] round trip time $\overline{RTT}$, then the buffer space required for a link utilization of $u \cdot C$ is given by:

$$B_{AKM}(u) = A(u)\frac{\overline{RTT} \times C}{\sqrt{N}}. \quad (1)$$

Here, $A : (0,1) \mapsto (0,\infty)$ is a real function for which $A(0.99) \approx 1$ and $A(0.9999997) \approx 2$. One should note that, although the bound (1) is derived in the context of drop-tail queues, it is also applicable to other AQM schemes as well. Namely, in order to ensure utilization of $u \cdot C$, one needs a physical buffer space for accommodating packets of N unsynchronized TCP flows, given by (1).

[1] It is suggested in [3] that in environments without synchronization, one should use harmonic mean of the RTTs of the active connections as "typical" RTT.

t	t_1	t_2	t_3	t_4
$aQd(t)(sec)$	0.1	0.02	0.005	0.001
$u(t)$	1.00	0.98	0.90	0.60

Table 1
Synthetic example of $aQd(t)$ and $u(t)$ for 4 different possible choices of parameter t.

Although the bound (1) yields important theoretical insights into the relation between link utilization and the required buffering it is not immediately applicable to size buffers in the real Internet routers for a number of reasons. Firstly, the bound (1) depends on the number of active users that are bottlenecked at the link, as well as their RTT distribution. These quantities vary, and are also usually hard to estimate. Secondly, the mathematical assumptions used in deriving (1) are not realistic and do not take into account the various and variable traffic mixes possible, the level of synchronization, the existence of non-TCP traffic, etc. Most importantly, while it is useful to know that delay and utilization are related in some manner, it is not immediately clear how to utilize this relationship in a meaningful manner. Specifically, an important practical question that arises in the queue management design is:

(*) "What is more important: low queueing delays or high utilization?"

This question arises particularly in the context of Service Level Agreements (SLA); we refer interested reader to the measurement study [27]. To illustrate the question $(*)$, consider the following hypothetical example. Suppose that for a given traffic mix, the relation between average queueing delays (aQd) and utilization (u) is given in Table 3:

In this table, t is some parameter that defines the queue management scheme. For example, t can be interpreted as available buffer space, or the per packet drop probability. Now, the important practical question is, which t should a network operator chose under this traffic mix? The answer to this question depends on the relative importance of utilization and queueing delays. To formalize this notion we can define the benefit $B(t)$ as the difference between utilization and cost a network operator is willing to pay that is an increasing function of queueing delays. Having defined this cost function, which specifies formally the tradeoff between utilization and delays, the problem then becomes that of choosing the optimal queueing scheme parameter t. This is a problem of maximizing the benefit $B(t)$ and can be solved in an optimization framework. In the Section 3 we will formalize the framework described above.

As we already noted, although there exist number of mathematical models [1–4] that can give us some insight into the delay-utilization relationship, it appears extremely hard to evaluate this relationship for general traffic environments. Moreover, even if one has reasonably accurate theoretical predictions between delay and utilization for a given traffic mix, these predictions would certainly be a function of traffic parameters such as the number of active flows, the number of active TCP flows, the proportion of TCP traffic, per flow responsiveness, the distribution of round trip times, the level of loss synchronization, the level of congestion on other links in the network, etc. From a measurement point of view, estimation of these quantities is very demanding and requires significant amount of computational and physical resources [6,16,17,29]. We summarize the discussion from this paragraph with the following statement to briefly describe a difficultly in applying the existing models on real-world Internet links.

(**) "It is highly nontrivial to predict or estimate in real-time, relation between queueing delays and utilization, for congested high-speed Internet links."

Having $(**)$ as the starting point, we will try to maximize overall the benefit $B(t)$ online rather than estimating the delay-utilization relationship. In this paper we propose two practical queue management schemes that have the same common goal: namely, maximizing the benefit $B(t)$, by controlling the parameter t online. In the first scheme, called Optimal Drop-Tail (ODT), t is the maximum available buffer space in the FIFO Drop-Tail queue, while in the second scheme, called Optimal BLUE (OB), the parameter t is the probability that an arriving packet is dropped.

The rest of the paper is organized as follows. Existing models of the delay-utilization relationship and AQM schemes are disscused in the Section 2. In Section 3 we define the optimization problem to be addressed and provide a theoretical analysis of the possible approaches to solving it. The queue management schemes Optimal Drop-Tail and Optimal BLUE are introduced in Sections 4 and 5 respectively. In Section 6 we provide detailed packet level ns2 simulations to show behavior of both ODT and OB. Finally we summarize our findings and discuss open issues in Section 7.

2. Previous work

In this section we discuss existing models for the delay-utilization relationship as well as Active Queue Management schemes that aim to reduce queueing delays.

Models. Over the last few years, a number of models have been proposed to estimate the relationship between queueing delays and utilization. Most of these consider the problem of sizing FIFO Drop-Tail buffers for achieving a certain level of link utilization under the assumption of a single bottleneck link servicing N long TCP transfers. In [1] the authors give a $O(\frac{1}{\sqrt{n}})$ bound (1). Another bound of this type is given in [2]. It is showed there (under the assumption of N unsynchronised homogenous TCP users with the same round trip time $\overline{RTT}$) that to achieve 100% of utilization one needs a buffer size of :

$$B_{AAP} = \frac{(\overline{RTT} \times C)^2}{2N(4N-1)^2} \approx \frac{(\overline{RTT} \times C)^2}{32N^3}. \quad (2)$$

The bound (2) is derived from a fluid model which assumes full unsynchronization (ie. only one source loses packets per congestion event). The authors [4] are even more optimistic and claim that if TCP users have bounded the maximal congestion window $maxcwnd\_$, then the needed queue size for achieving high throughput is of form of $O(log(maxcwnd\_))$. Such an approach assumes a low maximum $cwnd\_$, or equivalently a high loss rate, and does not allow high speed connections[10,19].

Another important problem that could result from extra small buffers is the problem of extreme unfairness between flows. To see this, consider a congested link, without per-flow management, with a loss rate p and an average queueing delay d_0. From the square root formula [25], the sending rate (in packets per second) of flow with base RTT given by RTT_b is equal to

$$r = \theta \frac{1}{\sqrt{p}(RTT_b + d_0)}. \quad (3)$$

Bearing in mind that a typical range for base RTT's spans a few microseconds to several seconds, and assuming that d_0 is in range of few microseconds (as suggested in all-optical framework in [31,4]), then this would imply unfairness between competing connections in the range $1 : 10^5$. In such situations, long-RTT connections would suffer heavily, and will not have any benefit in reducing queueing delays. Allowing a few milliseconds of queueing delays would decrease unfairness level to range $1 : 100$, which is several orders of magnitude more acceptable.

A very different approach has been presented in [3]. Namely, they suggest that an important performance metric in dimensioning router buffers is loss rate. By exploiting the window based nature of the TCP congestion control algorithm, they provide bound for buffer sizes that ensure loss rates lower than certain, prescribed, value.

As we already have noted, the insight obtained by theoretical analysis of proposed models is highly valuable for understanding the problem of interest, but there is significant gap between them and their application in real Internet links.

AQMs. Active queue management generally stands for a mechanism that has, as its ultimate objective, of keeping utilization as high as possible without incurring "large queueing delays". In TCP environments, the main cause of low utilization is synchronization. By breaking synchronization and responding early (but not "too early"), AQM schemes like RED, BLUE, PI and AVQ, aims to achieve high throughput together with low delays. However, no existing AQM takes in account the interaction between queueing delays and utilization. Low queueing delays and high utilization are mainly an ad hoc consequence of early response, rather than a formal performance goal.

3. Optimization framework

Lets go back to the example from the introduction illustrated in Table 1. For simplicity, assume for the moment that the parameter t is the available buffer space on the congested FIFO Drop-Tail queue; for buffer size equal to t_1 the average queue delay is $100ms$ and the utilization is 100%, for buffer size equal to t_2 the average queue delay is $20ms$ and the utilization is 98%, and so on. Which choice of t is optimal (among 4 possible in this example), depends on the "importance" of low queueing delays. To formalize this, one can identify the "importance" by the relative price between utilization and queueing delays. Let $P : [0, \infty) \mapsto [0, \infty)$ be a function that specifies relative price between utilization and delays. In other words, a queueing delay of d seconds has same price as utilization of $P(d)$. Formally, a price function is any function that satisfies the following definition.

Definition 1 *The function $P : [0, \infty) \mapsto [0, \infty)$ is a price function if it is twice differentiable, increasing and convex. In other words if:*
(a) $\forall d \in [0, \infty)\ \exists P''(d)$
(b) $\forall d \in [0, \infty)\ P'(d) \geq 0$
(c) $\forall d \in [0, \infty)\ P''(d) \geq 0$

The following simple technical lemma will be useful in later discussion.

Lemma 1 *Let $E \subset R$ be a segment ($E = [a_1, a_2]$ for some real a_1 and a_2). If $f : E \mapsto [0, \infty)$ is a twice differentiable, convex function and P an arbitrary price function, then $P \circ f : E \to [0, \infty)$ is convex as well.*

Proof The proof is straightforward. It is enough to prove that $(P(f(t))'' \geq 0$ for all $t \in E$.

$$(P(f(t))'' = (P'(f(t)) \cdot f'(t))' =$$
$$P''(f(t)) \cdot (f'(t)))^2 + P'(f(t)) \cdot f''(t) \geq 0.$$

□

Having defined a price function, the overall benefit, in the case given by the parameter t, can be written in the form:

$$B(t) = u(t) - P(aQd(t)). \quad (4)$$

The definition of benefit allows us to define a notion of optimal choice, as the value of t that maximizes the benefit. Formally:

Definition 2 *For a given price function P and set $\mathcal{T}$ of possible choices of t, an optimal Delay-Utilization(D-U) choice is any t_0 such that*

$$B(t_0) = \max\{B(t) \mid t \in \mathcal{T}\}, \quad (5)$$

if the maximum on the right hand side exist.

Comment. In the rest of the paper we will consider exclusively the following two cases: (1) the set $\mathcal{T}$ is finite; then the maximum in (5) clearly exists; (2) the set $\mathcal{T}$ is closed and bounded in metric space R, and $B : \mathcal{T} \mapsto \mathcal{R}$

is continuous function - in this case the maximum in (5) exists as well.

In the example given in Table 1, if we completely ignore the importance of low queueing delays, by setting $P(d) \equiv 0$ for all d, then the optimal D-U choice is given by t_1, as this maximizes the benefit $B(t) = u(t) - P(aQd(t)) = u(t)$ on the set $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$. For the price function $P(d) = 5 \cdot d$, the optimal D-U choice is t_2, and for the price function $P(d) = 100 \cdot d$, the optimal D-U choice is t_4. Linear price functions $P(d) = \gamma \cdot d$, are a simple way of specifying the relative price of queueing delays in sense that $a\%$ of utilization is equivalent with $\frac{a \cdot 0.01}{\gamma}\ sec = \frac{10 \cdot a}{\gamma} msec$ of queueing delays. Thus, a high γ gives high importance of low queueing delays, while a low γ gives priority to high utilization.

Throughout this paper we assume:

Assumption 1 *Under static traffic conditions the overall benefit given by (4) is a concave function of t.*

In the previous discussion, we have referred to t as a parameter which defines a queueing scheme. In later sections we will be concerned with the following two cases:

Case A. t_S defines the available buffer space, and packets are queued in Drop-Tail queue of size t_S.

Case B. t_p defines the drop probability, and each packet is dropped on arrival with probability t_p.

In the next two sections we present two queue management algorithms: Optimal Drop-Tail (ODT) and Optimal BLUE (OB). Both of these have a common performance goal: to maximize the overall benefit given by (4). ODT achieves this goal by adaptation of the available buffer space, while OB tunes the drop probability t_p in order to maximize $B(t_p)$.

Comment. Other approaches might be possible as well. For example, in the framework of Virtual Queue (VQ) schemes [12], t can be seen as virtual queue capacity. Following our optimization methodology, one can design an virtual queue management algorithm by adapting the virtual queue capacity subject to the performance goal that maximize the benefit $B(t)$ rather than keeping the utilization at certain level γ as it has been done in the Kunniyur and Srikant's AVQ [23] algorithm.

In the rest of this section we discuss the validity of the Assumption 1, the existance/uniqueness of optimal D-U choice and possible strategies for online solving optimization problem (5).

Assumption 1 is very hard to formally check. In a theoretical framework, this would require accurate models of various traffic mixes, and as we already noted, modelling such complex environments is highly nontrivial. Some results related to the convex relationship between utilization and buffer size in non-elastic traffic environments are developed in [21,22]. However, our empirical observations suggest that for the traffic mix that is consisted from the static number of TCP and UDP flows, Assumption 1 holds in

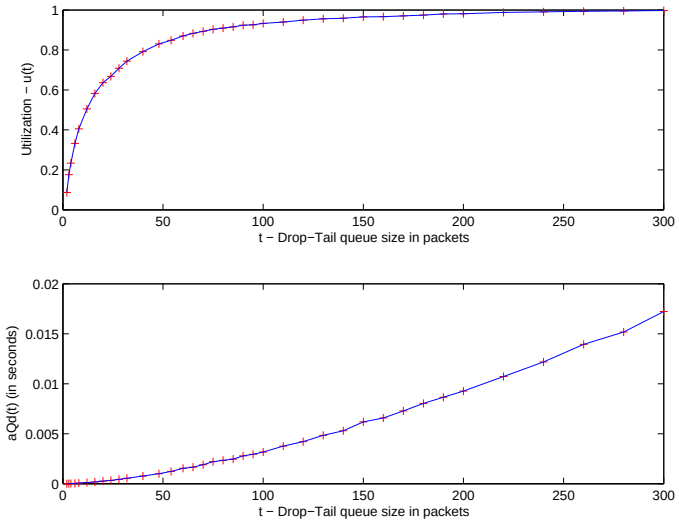

Fig. 1. t_S : Drop - Tail queue size vs. $u(t_S)$ (top) and $aQd(t_S)$ (bottom).

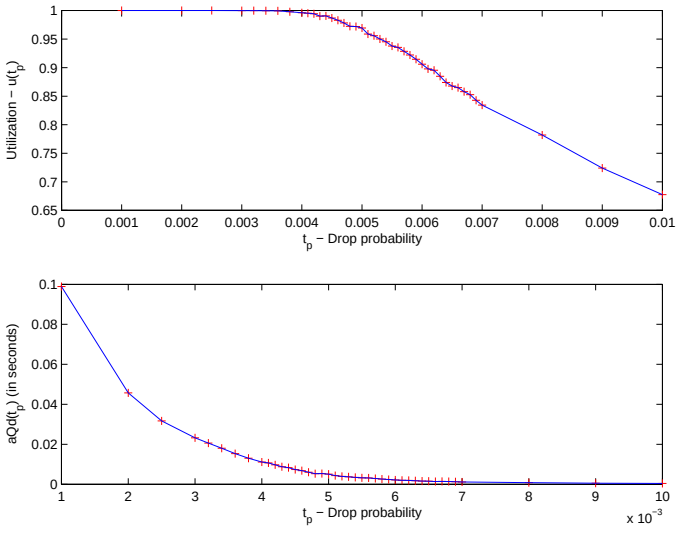

Fig. 2. t_p : Drop probability vs. $u(t_p)$ (top) and $aQd(t_p)$ (bottom).

both Case A and Case B. To illustrate this we run two sets of packet level ns2 simulations, and evaluate the utilization and the average queueing delay.

Simulation A. In the first set of simulations, we consider a FIFO Drop-Tail queue with a service rate of 10MBps and size of t_S packets. This queue is shared by 50 TCP connections with round trip times uniformly distributed in range $[20, 220] ms$ and with packet size of 1000 bytes. We varied t_S from 1 to 300 packets, and plotted $u(t_S)$ and $aQd(t_S)$ in Figure 1. The convexity of $aQd(t_S)$, by Lemma 1, implies the convexity of $P(aQd(t_S))$, and this together with concavity of $u(t_S)$ implies concavity of the benefit $B(t_S)$, for arbitrary price function P.

Simulation B. In the second set of simulations, we consider a queue with service rate of 10MBps, of size of 10000 packets (so that no packet is dropped because of overflow), such that every packet is dropped with probability t_p on the arrival to the queue. This queue is shared by the same

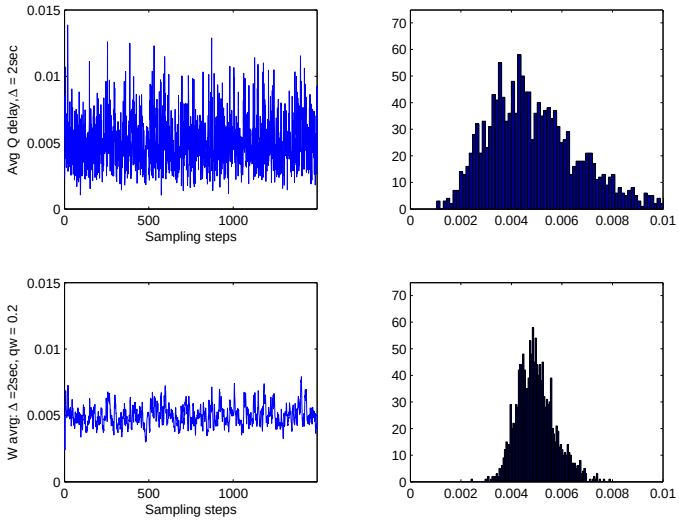

Fig. 3. $d_\Delta(k)$ (top), and $\overline{d}_\Delta(k)$ (bottom); $\Delta = 2sec$, $qw = 0.2$.

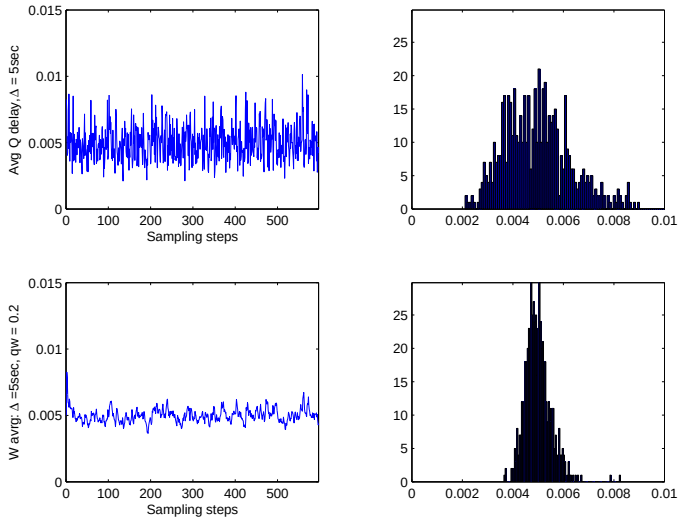

Fig. 4. $d_\Delta(k)$ (top), and $\overline{d}_\Delta(k)$ (bottom); $\Delta = 5sec$, $qw = 0.2$.

set of 50 TCP connections as in first simulation. We varied t_p in the range $[10^{-3}, 10^{-2}]$, and plotted $u(t_p)$ and $aQd(t_p)$ in Figure 2. As in the previous case, convexity of $aQd(t_p)$, together with concavity of $u(t_p)$ implies concavity of the benefit $B(t_p)$, for arbitrary price function P.

Convex optimization has been widely employed in the networking community. For example, optimization methods are essential in the analysis and design of distributed congestion control algorithms [28,18]. In our case, we need efficient algorithms for solving (5). A standard control strategy for solving (5) is given by

$$\dot{t} = g(t) \cdot B'(t),\ g(t) \geq \epsilon > 0, \tag{6}$$

or its discrete version:

$$t(k+1) = t(k)\left(1 + g(k)\frac{B(t(k)) - B(t(k-1))}{t(k) - t(k-1)}\right), \tag{7}$$

$$g(k) \geq \epsilon > 0.$$

The problem with employing one of these strategies in the present case is twofold. First, as we do not have explicit relationship between t and $B(t)$, we can not instantly compute the derivative $B'(t)$. Second, the signal to noise [2] ratio in measuring of both queueing delays and utilization can be very large especially in the neighborhood of the solution of (5). This would potentially imply low confidence in the estimation of $B'(t)$ in the neighborhood of the solution of (5). One approach to this problem is the use of larger sampling times and low pass filter for smoothing out the results. To illustrate the level of noise one can expect in queue measurements we ran the following ns2 simulation.

Simulation C. We consider the same setup of 50 TCP flows competing over a link with service rate of 10MBps, as in Simulation B. We drop each packet on arrival with constant probability $t_p = 0.005$. The quantities of interest are the following: $d_\Delta(k)$ is the average queueing delay in the k-th sampling period $[(k-1)\Delta, k\Delta]$, and the weighted average $\overline{d}_\Delta(k)$, with weighting factor qw, given by:

$$\overline{d}_\Delta(k) = (1 - qw) \cdot \overline{d}_\Delta(k-1) + qw \cdot d_\Delta(k).$$

Figures 3 and 4, depict measured values of $d_\Delta(k)$ (left top) and $\overline{d}_\Delta(k)$ (left bottom) from the simulation, for $\Delta = 2sec$ and $\Delta = 5sec$. The right hand side plots show the histograms of quantities depicted on left plots.

We observed a similar level of noise in measurements of link utilization. We also ran the same simulation over Drop-Tail instead of constant drop rate queues, and the level of noise in queueing delay and utilization measurements are approximately of the same order of magnitude as in Simulation C.

In the next two sections we will present two novel queue management algorithms for which the performance goal is given by maximization of the benefit defined by (4). The first one, Optimal Drop-Tail(ODT), achieves this goal, by adaptation of t_S - the available Drop-Tail queue space, while the second one, Optimal BLUE(OB), adapts t_p - the per packet drop probability. Both ODT and OB use a form of MIMD [3] algorithm, where at the end of each sampling period control variable t (that is t_S or t_p) is updated by the rule:

$$t(k+1) = t(k) \cdot m(k). \tag{8}$$

where $m(k)$ is either α or $1/\alpha$, for some α greater 1, and $m(k)$ determines direction in which t should go. Algorithms of this type cannot settle to constant value, but rather continuously search for the point on the grid $\mathcal{T}_\alpha = \{t(0) \cdot \alpha^n,\ n \in Z\}$, that maximizes $B(t), t \in \mathcal{T}_\alpha$. However, choosing α to be close to 1, the point on the grid $\mathcal{T}_\alpha$ that maximizes $B(t)$ will be close to the global optimal value.

[2] By definition $B(t)$ is function of average utilization $u(t)$ and average queueing delay $aQd(t)$. Instantaneous utilization (queueing delay) can be seen as random variable that is sum of $u(t)$ ($aQd(t)$) and appropriate zero mean random variable, that we refer to as noise.

[3] Multiplicative Increase Multiplicative Decrease.

4. Optimal Drop-Tail

Drop-Tail queueing is the scheme that is employed by the majority of the current Internet routers. Drop-Tail queues have a single parameter S that determines the available size [4] of the queue, and is usually manually configured by a network operator. The following simple observation is the basis for the spectrum of algorithms presented here. That, by controlling the value t_S - available queue size [5], the objective of achieving certain performance goals, given in terms of utilization and queueing delays, can be met.

By controlling t_S, one can control both utilization and queueing delays.

For example, if the performance goal is given by keeping the average utilization at a certain level λ, one can design a strategy for achieving that goal by controlling t_S. Similarly, if the performance objective is keeping the average queueing delay (at the times of congestion) at a prescribed level d_0, another control strategy can be designed for solving that problem. At this point we should note that by controlling t_S one can control not only utilization and queueing delays, but also other (important) performance metrics such as jitter and loss rate. Embedding them into an optimization framework could be done in straightforward manner, but is out of scope of the present paper.

Following the delay-utilization optimization framework developed in the previous section, the performance goal of interest will be the maximization of the benefit $B(t_S)$. We proceed by presenting an ODT algorithm, a strategy with that performance goal.

The ODT algorithm controls the variable t_S that represent available queue size. In other words, on every packet arrival, a packet is dropped, if by its enqueuing to the existing queue, the queue length would be greater than t_S, otherwise the packet is enqueued. The value t_S is updated once per sample time period (Δ) in the following manner:

$$t_S(k+1) = t_S(k) \cdot m(k), \tag{9}$$

where $m(k)$ is defined by:

$$m(k) = \alpha, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_S(k) - t_S(k-1)} \geq 0, \tag{10}$$

$$m(k) = \frac{1}{\alpha}, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_S(k) - t_S(k-1)} < 0. \tag{11}$$

Here, $\alpha > 1$ is a constant parameter, close to 1. The choice of α determines the responsiveness of the algorithm. Since t_S is either multiplied with α or divided by α, in each step k, $t_S(k) = t_S(0) \cdot \alpha^{l(k)}$, for some integer $l(k)$. By $\hat{B}(l(k))$ we denote the estimated value of $B(x)$ at the point $x = t_S(k) = t_S(0) \cdot \alpha^{l(k)}$. Algorithms of this type can be seen as a version of (7) that do not allow arbitrarily small steps. Strategies of the form of (7) are inappropriate in the our problem for the following two reasons. First, any algorithm of type (7) that allows very small changes in the parameter t_S would suffer from a high noise to signal ratio around global maximum of $B(t_S)$, and would require long time for accurate estimation of B in the neighborhood of the global maximum. Second, it has been proved in [26], using information-theoretical techniques, that any algorithm for finding an optimum using noisy observations of a benefit function has slow expected convergence. Namely, $O(\epsilon^{-4})$ queries have to be made before one can ensure ϵ-accuracy in the estimation of the optimum x^*. Under dynamic, Internet-like traffic conditions, frequent (small) changes of the traffic patterns might not allow such algorithms to converge, and can potentially cause undesirable large oscillations.

Algorithms of the form of (9) that do not converge to the certain value, but rather continuously search for the optimal value have been extensively used in the networking literature. Examples of such algorithms are AIMD [6] $cwnd\_$ control in TCP [14], AIAD algorithm for controlling the drop probability in BLUE[9] as well as MIMD algorithm for the adaptation of RED parameters in Self-Configuring RED [8].

Now we proceed by describing the technique for estimation of $\hat{B}(l(k))$.

In every sampling interval, we have that $t_S(k) = t(0)\alpha^{l(k)}$ for some integer $l(k)$, and this integer is uniquely determined. In other words the mapping between the set of all possible values of t_S, $\mathcal{T}_\alpha$, and set of integers given by $t(0)\alpha^m \mapsto m$ is bijective. This allows us to use the history at each possible value of $t_S(k)$ independently for computing the estimate $\hat{B}(l(k))$. For each possible integer m we will keep the value: $n_-(m)$ which is the number of sample intervals within previous W_0 sampling intervals for which $l(k)$ was equal to m. Thus at sampling interval k we have:

$$n_-(m) = \#\{k_1 : \ k_1 \in (k - W_0, k], \ l(k_1) = m\}.$$

Denote by $\tilde{B(k)}$ the instantaneous benefit in the k-th sampling interval : $[(k-1)\Delta, k\Delta]$, i.e.:

$$\tilde{B(k)} = \tilde{u}(k) - P(\tilde{d}(k)),$$

where $\tilde{u}(k)$ and $\tilde{d}(k)$ are the instantaneous utilization and queueing delay in the k-th sampling interval respectively, we will estimate $\hat{B}(l(k))$ using the following weighted average:

$$\hat{B}(l(k)) = \hat{u}(k) - P(\hat{d}(k)), \tag{12}$$

where:

$$\hat{u}(l(k)) = \frac{1}{n_-(l(k))}\hat{u}(l(k)) + (1 - \frac{1}{n_-(l(k))})\tilde{u}(k),$$

[4] Size can be configured in either bytes or packets.

[5] We write t_S instead S to distinguish cases between variable queue size (for which we use t_S) and constant queue size (for which we use S).

[6] Additive Increase Multiplicative Decrease.

and

$$\hat{d}(l(k)) = \frac{1}{n_-(l(k))}\hat{d}(l(k)) + (1 - \frac{1}{n_-(l(k))})\tilde{d}(k).$$

The rationale for the estimation technique given by (12) is the following. If some m has a large number of occupancies in the recent history of $l(k)$ then this indicates that the corresponding $t_S = t_S(0)\alpha^m$ is close to the optimal value and a finer estimation of the benefit is required. If m has a small number of appearances in the previous W_0 sampling periods we need less accuracy, but a faster response to changes in the traffic conditions. If m had no appearances in the previous W_0 sampling periods, all history will be forgotten in the future estimation of $\hat{B}(t_S(0)\alpha^m)$.

We use one additional correction step, that is rarely needed under static conditions but is important for improving accuracy for new values t_S that have had very few visits in the recent history. Namely, we do not allow non-monotonicity in the estimation of $\hat{u}$ and $\hat{d}$. Set $i = l(k)$. Then:

if $\hat{u}(i) > \hat{u}(i+1)$ or $\hat{u}(i) < \hat{u}(i-1)$ then

$$\hat{u}(i) = \frac{n_-(i-1)\hat{u}(i-1) + n_-(i)\hat{u}(i) + n_-(i+1)\hat{u}(i+1)}{n_-(i-1) + n_-(i) + n_-(i+1)}, \quad (13)$$

if $\hat{d}(i) > \hat{d}(i+1)$ or $\hat{d}(i) < \hat{d}(i-1)$ then

$$\hat{d}(i) = \frac{n_-(i-1)\hat{d}(i-1) + n_-(i)\hat{d}(i) + n_-(i+1)\hat{d}(i+1)}{n_-(i-1) + n_-(i) + n_-(i+1)}. \quad (14)$$

Recall that $n_-(m)$ represents the number of sampling periods k, in the recent history (driven by sliding window of size W_0), in which $l(k)$ was equal to m. We have a confidence in the estimate of $\hat{u}$ and $\hat{d}$ that is roughly proportional to n_-. This is exploited in (13) and (14).

Note that we need to store four sequences, aug (augmenting sequence of length W_0 for computing n_-), n_-, $\hat{u}$ and $\hat{d}$, for implementing this estimator. The size of the sequence aug is W_0, while the size of sequences n_-, $\hat{d}$ and $\hat{u}$ depend on the choice of α in a logarithmic fashion: if the physical buffer space is S_0 bytes, then the size of sequences $\hat{d}$, $\hat{u}$ and n_- should be $\log_\alpha S_0$ to cover the range from 1 to S_0 bytes. For example, with $\alpha = 1.01$, a sequence of size 2000 will cover the range from 1 to $S_0 = 1.01^{2000} \approx 439286205$ bytes with a granularity of 1%.

From the computational point of view, ODT is a very light scheme. Namely, for the computation of the instantaneous utilization $\tilde{u}(k)$, and the instantaneous queueing delay $\tilde{d}(k)$, we use three counters: $NmbBytes$ (number of processed bytes since last update), $NmbArrivals$ (number of packet arrivals since last update) and $TotQDelay$ (sum of potential queueing delays for all $NmbArrivals$ packets since last update) [7]. All of these three counters are updated once per packet and these updates are the only per packet operations required.

[7] Having this information $\tilde{u}(k)$ is computed as ratio $NmbBytes/\Delta$, while $\tilde{d}(k) = TotQDelay/NmbArrivals$, at the end of the k-th sampling period.

$P(d)$	price function
Δ	length of sampling period
α	MIMD parameter
W_0	history window

Table 2
Parameters of ODT, OB.

The input parameters for ODT are given in the Table 2. While in general $P(d)$ can be an arbitrary function that satisfies Definition 1, throughout this paper we will mainly use functions that are linear in d:

$$P_\gamma(d) = \gamma d, \ \gamma > 0. \quad (15)$$

If we restrict ourselves to price functions of this form then the parameter $P_\gamma(d)$ can be specified by a single scalar γ. A higher value of γ assigns more importance to low delays and vise versa. The sampling period time Δ should be chosen to cover several "typical" round trip times, in order to allow traffic to respond to change of t_S. Choosing Δ in range $[1sec, 5sec]$ usually satisfies this condition. The parameter α determines the responsiveness of ODT, and should be selected such that it allows doubling/halving of t_S within several seconds (up to one minute). The parameter W_0, is the one that determines importance of old measurements in the current estimation, a large W_0 is appropriate for static or very slowly varying environments, while a small W_0 is necessary for more dynamic conditions. In our experiments we use W_0 such that ΔW_0 is within an order of magnitude of one minute.

At this point we discuss the notion of variability in the traffic conditions. Measurements from [24] show that on typical 150Mbps+ links, basic IP parameters such as the number of active connections, the proportion of TCP traffic, the aggregate IP traffic, etc., do not change dramatically. Although we do not exclude the possibility that there can be drastic changes in the traffic mixes, our basic presumption in the design of ODT is that such events are rare enough to be considered as exception rather than rule. Thus, ODT is designed to search for an optimal solution in the "regular" intervals, during which traffic conditions vary slowly. In the cases of dynamic traffic conditions, one can perform self tuning of the parameters W_0 and Δ depending on the level of changes in the traffic conditions. However, for the reasons discussed above, present ODT algorithm does not incorporate this adaptation of the parameters.

The following theorem shows that, assuming that estimators $\hat{B}$ preserves order of B on the grid $\mathcal{T}_\alpha = \{t(0) \cdot \alpha^n, \ n \in Z\}$ the controller (9) runs system to the state that is close to global optima.

Theorem 1 *Let t^* be the point where global maximum of B is attained. Suppose that estimator $\hat{B}$ preserves the order on the grid $\mathcal{T}_\alpha$, ie. for all $m_1, m_2 \in Z$:*

$$\hat{B}(m_1) \geq \hat{B}(m_2) \Leftrightarrow B(t(0)\alpha^{m_1}) \geq B(t(0)\alpha^{m_2}). \quad (16)$$

Then there exist m_0 such that for all positive integers r:

$$t(m_0 + 2r) = t(m_0 + 2r + 2) = \bar{t}$$

$$t(m_0+4r+1)=\bar{t}\alpha, \quad and \quad t(m_0+4r-1)=\frac{\bar{t}}{\alpha},$$

and the relative error between $\bar{t}$ *and* t^* *satisfies:*

$$\frac{\bar{t}-t^*}{t^*} \leq \alpha - 1. \tag{17}$$

Proof Suppose without loss of generality that $t(0) < t^*$. Then by Assumption 1 and (16) there exist positive integer k_0 such that for all $k < k_0$:

$$t(k+1) = \alpha t(k) > t(k)$$

and

$$t(k_0) > t^*.$$

Now, we can distinguish two cases:

1st Case: $B(t(k_0)) \geq B(t(k_0-1))$. Then $t(k_0+1) = \alpha t(k_0)$. By concavity $B(t(k_0+1)) < B(t(k_0))$ and therefore $t(k_0+2) = \frac{t(k_0+1)}{\alpha} = t(k_0)$, and $t(k_0+3) = \frac{t(k_0+1)}{\alpha}$.

2nd Case: $B(t(k_0)) < B(t(k_0-1))$. Then $t(k_0+1) = \frac{t(k_0)}{\alpha} = t(k_0-1)$ and $t(k_0+2) = \frac{t(k_0-1)}{\alpha}$. From concavity of B we get $t(k_0+3) = t(k_0-1)$ and $t(k_0+4) = \alpha t(k_0-1)$.

By taking $\bar{t} = t(k_0)$ and $m_0 = k_0$ in the first case, or $\bar{t} = t(k_0-1)$ and $m_0 = k_0 - 1$ in the second case, we conclude the first part of the Theorem. To obtain the inequality (17), note that $\bar{t} \in (\frac{t^*}{\alpha}, t^*\alpha)$ which is equivalent to

$$\frac{\bar{t}-t^*}{t^*} \in (\frac{1-\alpha}{\alpha}, \alpha - 1) \subset (-(\alpha-1), \alpha-1).$$

□

5. Optimal BLUE

BLUE is an active queue management algorithm that maintains a single internal variable p_m which is used for calculating the drop probability of arriving packets: each packet is dropped with probability p_m on arrival. Roughly speaking, the performance goal of BLUE is to keep the loss probability as low as possible such that no buffer overflow occurs. The variable p_m is updated once per interval of length $freeze\_time$, in an Additive Increase - Additive Decrease (AIAD) fashion with parameters δ_1 and δ_2: if during the previous $freeze\_time$ interval no buffer-overflow losses has occurred then p_m is reduced by δ_2, otherwise p_m is increased by δ_1. By using a strategy of this type, BLUE searches for the "correct" rate at which it should drop packets. As we can see, there is no formal objective in terms of utilization or delays. However, we use the idea of controlling the drop probability t_p (that is same as p_m in the original BLUE) in order to maximize overall benefit $B(t_p)$. The first step in the design of such scheme is the following observation.

By controlling drop probability, one can control both utilization and queueing delays.

Low drop probabilities keep both queueing delays and utilization large, and high drop probabilities keep both queueing delays and utilization low; see Simulation B in the Section 3. By specifying a price function $P(d)$ that defines a relative price between delays and utilization, our performance goal in the design of Optimal BLUE is maximization of the benefit:

$$B(t_p) = u(t_p) - P(aQd(t_p)).$$

The quantities of interest for the calculation of the benefit are the average values of utilization and the queueing delays, and can be seen as the expected values of appropriate random variables. Because of large noise in the estimation of these quantities, it is helpful to use filtering in conjunction with an algorithm that continuously searches for the optimum. OB will use same strategy for controlling of t_p as ODT uses for controlling the available queue size t_S. We update t_p once per sample period of length Δ in an MIMD fashion:

$$t_p(k+1) = t_p(k) \cdot m(k), \tag{18}$$

where $m(k)$ is defined by:

$$m(k) = \alpha, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_p(k) - t_p(k-1)} \geq 0 \tag{19}$$

$$m(k) = \frac{1}{\alpha}, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_p(k) - t_p(k-1)} < 0. \tag{20}$$

Here $l(k)$ denotes an integer such that $t_p(k) = t_p(0) \cdot \alpha^{l(k)}$, and $\hat{B}(l(k))$ is the estimated value of the benefit $B(x)$ at the point $x = t_p(k) = t_p(0) \cdot \alpha^{l(k)}$. The method for estimating $\hat{B}(l(k))$ is same as in ODT and is given by (12). We also exclude non-monotonic estimation of $\hat{u}$ and $\hat{d}$. Recall that $u(t_S)$ and $aQd(t_S)$ are increasing functions of the available queue size t_S, while $u(t_p)$ and $aQd(t_p)$ are decreasing functions of t_p(see Figures 1 and 2). Because of this (13) is executed if $\hat{u}(i) < \hat{u}(i+1)$ or $\hat{u}(i) > \hat{u}(i-1)$, and (14) is executed if $\hat{d}(i) < \hat{d}(i+1)$ or $\hat{d}(i) > \hat{d}(i-1)$; $i = l(k)$.

We use an MIMD strategy for the control of the drop probability t_p rather than the original BLUE-like AIAD, because of scalability reasons. Namely, with an MIMD strategy, if the algorithm runs in a low drop probability regime, absolute changes in t_p per step are smaller than absolute changes in t_p per step in higher drop probability regimes. On the other hand, in AIAD schemes, absolute changes per step are constant and are given by AI and AD parameters.

The memory requirements of OB are same as ODT. Storing four sequences, $aug, n_-, \hat{u}$ and $\hat{d}$, requires just several kilobytes. Since all these quantities are used just once per Δ, then they can be stored off-chip and their size is not important. From the computational point, OB requires one more operation per packet: random drop. This, makes OB roughly equivalent to RED from the level of the computational resources they require. The parameters of OB are the

same as in ODT and are given in the Table 2. Comments related to the selection of the parameters of ODT applies to the OB as well.

6. Simulations

Simulation D. Our first set of simulations illustrate the dynamics of t_S and t_p under static conditions of 50 TCP flows with RTT's uniformly distributed in range $[20, 220]msec$ and with packet sizes of 1000 bytes. We run ODT and OB with parameters $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The price function used in both cases is $P_{10}(d) = 10 \cdot d$. Initially: $t_S(0) = 100Kbytes$, $t_p = 0.002$. The off-line (see Simulation E) optimal values are approximately $t_S^* \approx 130Kbytes$ and $t_p^* \approx 0.0048$.

Simulation E. The second set of simulations shows how close the average queueing delays and average utilization are to the optimal values, in static conditions with a constant number TCP flows, for both ODT and OB. We ran the same set of 50 TCP flows, with RTT's uniformly distributed in range $[20, 220]ms$ and packet sizes of 1000 bytes, as in Simulations A and B (Section 3). By running a sequence of long simulations in Section 3 we obtained plots given in Figures 1 and 2, that represent the dependance between t_S, (t_p respectively), the average utilization, and the queueing delays. Having these graphs, we can, in an off-line manner, find the value of t_S (t_p) that maximizes benefit $B(t_S)$ (resp $B(t_p)$). The red stars on Figures 7 (ODT) and 8 (OB) show these off-line optimal values for this optimization problem (5) (defined by price functions $P_\gamma(d) = \gamma \cdot d$) for $\gamma = 2, 10, 20$. Having $P_\gamma(d)$ as a parameter of the scheme, we ran ODT and OB. The other simulation parameters were: $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The blue crosses on Figures 7 and 8 represent the long-run (5 minutes) averages of queueing delays and utilization for $\gamma = 2, 10, 20$. The numerical results for this simulation are shown in the Table 3.

It is important to notice here that in TCP environments, OB outperforms ODT in terms of maximization the benefit B. This is because a Drop-Tail queue that achieves average queueing delay d_0 achieves less throughput than a queue with on-arrival random dropping, with the same average queueing delay d_0. The reason for this lies in the phenomenon of (partial) synchronization of losses that is a feature of Drop-Tail queues. Figure 9 illustrates this difference for queues servicing 50 TCP flows, and is based on data from the simulations A and B.

Remark. At this point we note an important practical issue related to applicability of our algorithms in the existing Internet routers. The only two pieces of information they require are the "achieved utilization" and the "average queueing delay" experienced during one sampling period. At the end of each sampling period we set new values of the available buffer space (t_S in ODT) or drop probability (t_p in OB). While most routers allow configuration of the available buffer space, we are not aware of option for configurability of the drop probability.

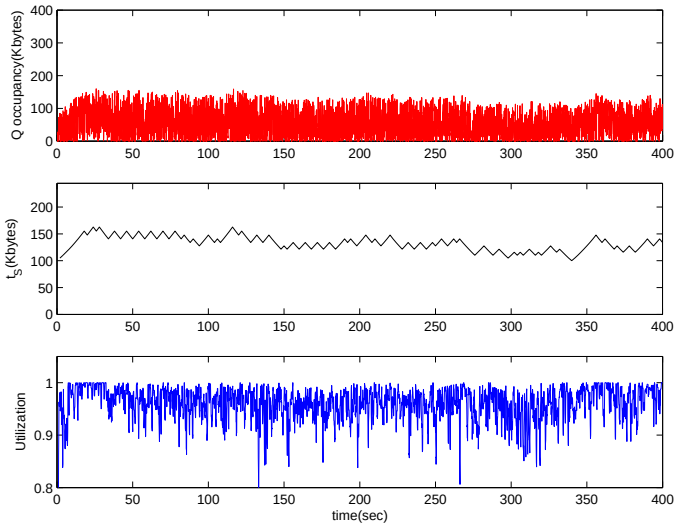

Fig. 5. Simulation D. Queue occupancy, available buffer space(t_S), and utilization for ODT servicing 50 TCP flows.

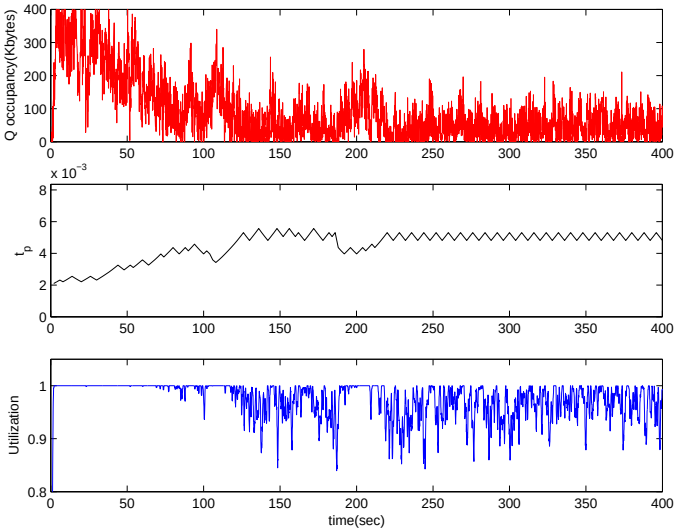

Fig. 6. Simulation D. Queue occupancy, drop probability(t_p), and utilization for OB servicing 50 TCP flows.

Simulation F. In this simulation we present the behavior of ODT and OB in the case of mixtures of TCP and UDP traffic. In this simulation, the same set of 50 TCP flows that were defined previously compete for a bandwidth on $10Mbyte/sec$ link, with 50 UDP flows that have exponentially distributed on and off periods. The on-periods have a mean of $1000ms$, and the off-periods have mean of $3000ms$. The sending rate for the on-periods is $1000Kbit/sec$. The aggregate UDP arrival rate has a mean of $1.4867Mbyte/sec$ which is approximately 14.9% of the link's service rate. A histogram, given in Figure 10, shows the distribution of the aggregate UDP sending rate sampled on $100ms$ intervals.

The ODT and OB parameters are the same as in previous simulations: $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The price function used in both cases is $P_{10}(d) = 10 \cdot d$. Initially: $t_S(0) = 100Kbytes$ and $t_p(0) = 0.002$.

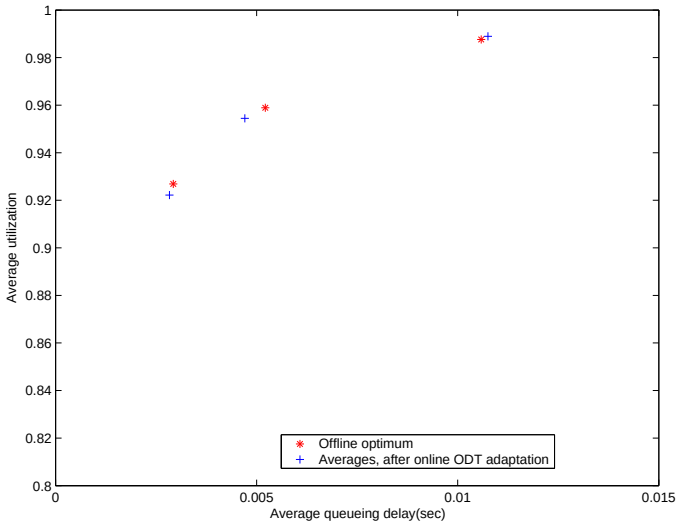

Fig. 7. ODT: Off-line vs. online average queueing delays and utilization. Price functions $P_\gamma(d)$, for $\gamma = 2, 10, 20$.

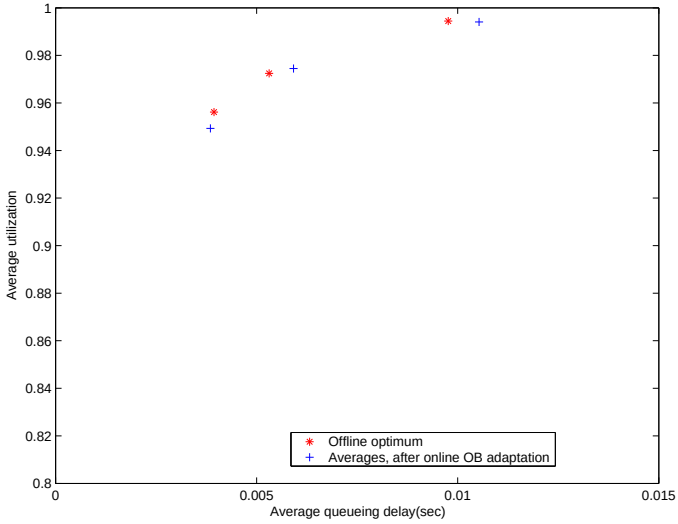

Fig. 8. OB: Off-line vs. online average queueing delays and utilization. Price functions $P_\gamma(d)$, for $\gamma = 2, 10, 20$.

Scheme, γ	$aQd(sec)$	u	B_γ
ODT, $\gamma = 2$, online	0.01075	0.9890	0.9675
DT, $\gamma = 2$, off-line	0.01058	0.9876	0.9664
ODT, $\gamma = 10$, online	0.00471	0.9544	0.9074
DT, $\gamma = 10$, off-line	0.00521	0.9589	0.9067
ODT, $\gamma = 20$, online	0.00283	0.9222	0.8655
DT, $\gamma = 20$, off-line	0.00293	0.9269	0.8683
OB, $\gamma = 2$, online	0.00975	0.9941	0.9730
CDP, $\gamma = 2$, off-line	0.00973	0.9945	0.9749
OB, $\gamma = 10$, online	0.00591	0.9745	0.9153
CDP, $\gamma = 10$, off-line	0.00531	0.9725	0.9193
OB, $\gamma = 20$, online	0.00385	0.9493	0.8723
CDP, $\gamma = 20$, off-line	0.00394	0.9562	0.8774

Table 3
Numerical results: off-line optima and online ODT and OB averages. The last column represents $B_\gamma(t) = u(t) - \gamma \cdot aQd(t)$. (CDP - stands for Constant Drop Probability queuing)

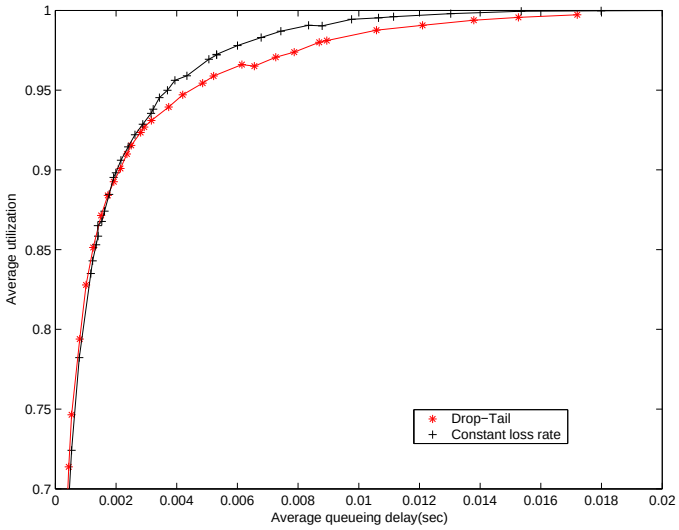

Fig. 9. Average queueing delays vs. average utilization for drop tail and constant loss rate queues.

Figures 11 and 12 depict plots of utilization, queue occupancy and $t_S(t_p$ resp.). We observed a slightly larger fluctuation in t_S and t_p compared to Simulation D. This is consequence of the stochastic nature of the non-responsive traffic.

Simulation G. In the following simulation we show the behavior of ODT and OB in the case of sudden changes in the traffic conditions. Two sets of 50 TCP flows in this simulation compete for bandwidth on a $10Mbyte/sec$ link. The first set is active during whole simulation in the interval $[0, 900]$ seconds; connections from the second set are active only in the interval from $[300, 600]$ seconds. Thus, at time $t_1 = 300sec$, the number of TCP connections is doubled while at time $t_2 = 600sec$, the number of connections is halved, as is depicted in the Figure 13. The ODT and OB parameters are the same as in previous simulations: $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The price function used in both cases is $P_{10}(d) = 10 \cdot d$. Initially: $t_S(0) = 50Kbytes$ and $t_p(0) = 0.002$.

Figures 14 and 15 illustrate the behavior of ODT and OB in this case. Although ODT and OB are not designed for environments with sudden changes we see that they adjust their parameters to the sudden traffic changes. However, "convergence" to the optimal region takes 1-2 minutes in the present simulation.

Simulation H. Here we demonstrate how other performance metrics are impacted by changes in queueing delay. We concentrate on fairness and loss rate. We use Jain's Fairness Index (JFI) [15] as a fairness indicator and is defined as follows. For set of users $u_1, \ldots, u_k$ let $r = (r_1, \ldots, r_k)$ be vector of their achieved average rates during the measurement interval. Then

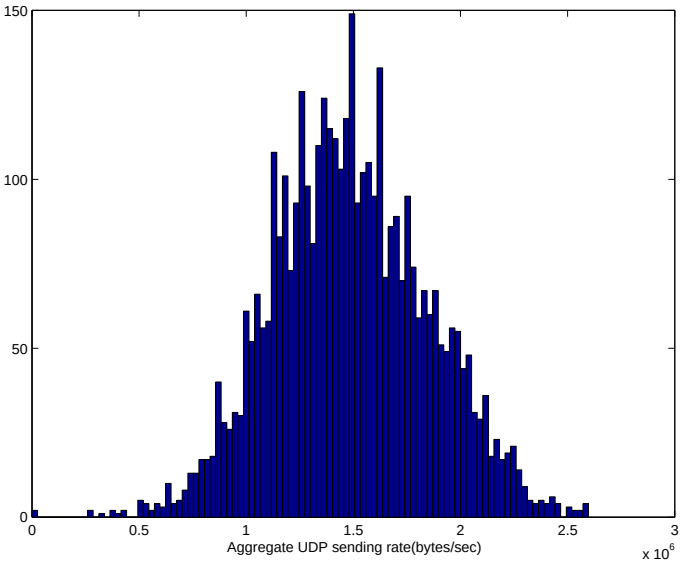

Fig. 10. Simulation F. Histogram of aggregate UDP sending rate; sampling intervals $100ms$.

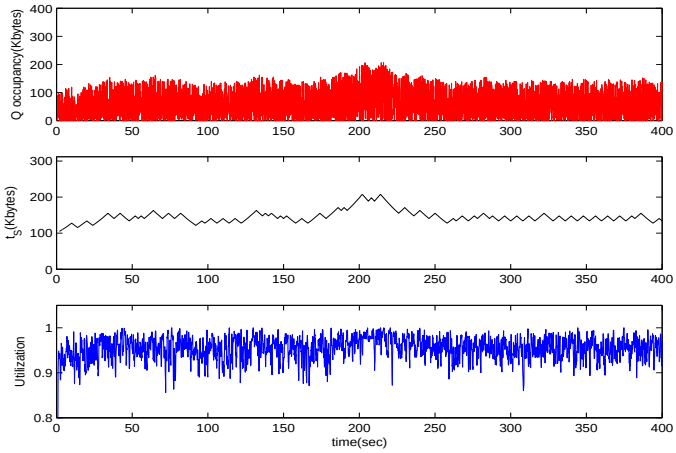

Fig. 11. Simulation F. Queue occupancy, available buffer space(t_S), and utilization for ODT servicing 50 TCP flows and 50 on-off UDP flows.

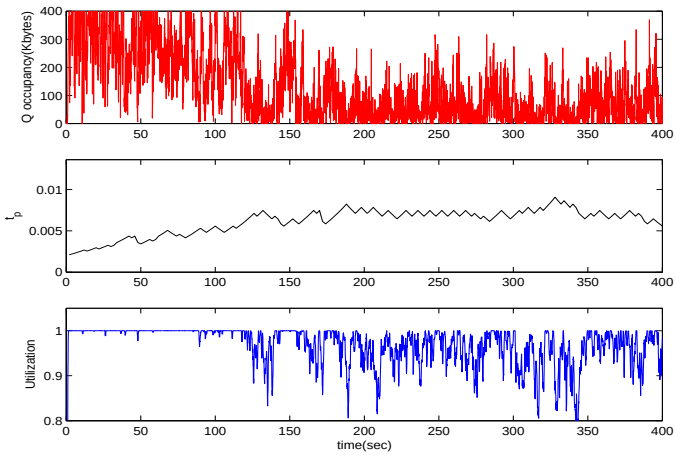

Fig. 12. Simulation F. Queue occupancy, drop probability(t_p), and utilization for OB servicing 50 TCP flows and 50 on-off UDP flows.

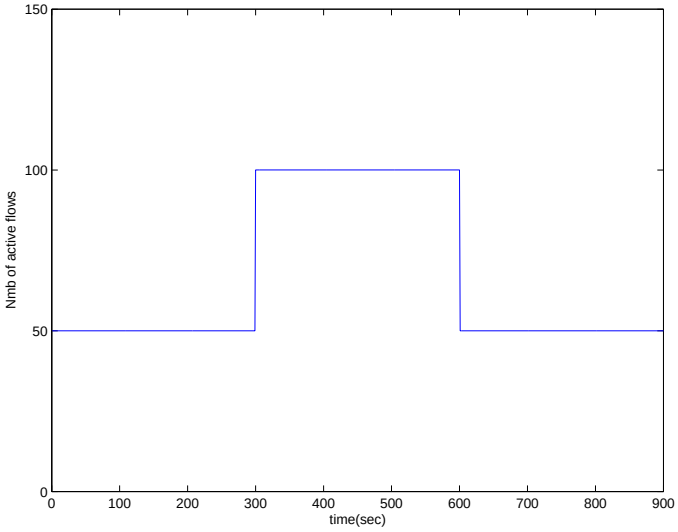

Fig. 13. Simulation G. Number of active TCP flows in time.

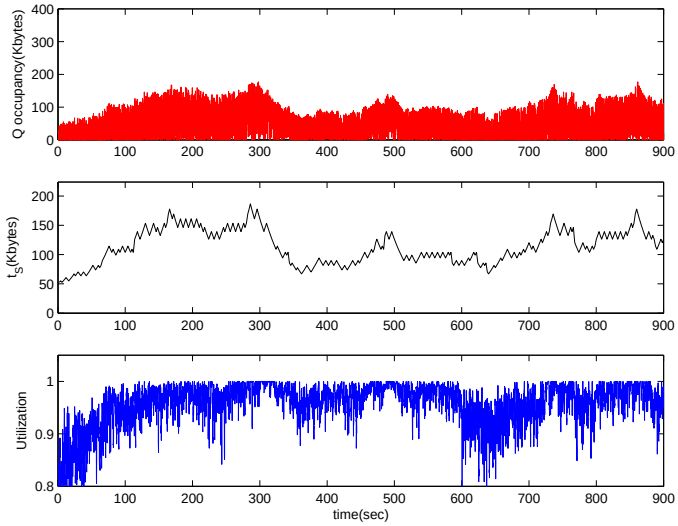

Fig. 14. Simulation G. Queue occupancy, available buffer space(t_S), and utilization for ODT servicing a varying number of TCP flows.

$$JFI(r) = \frac{\left(\sum_{i=1}^{N} r_i\right)^2}{N \sum_{i=1}^{N} r_i^2}. \tag{21}$$

The simulation setup is same as in Simulations A and B and consists of 50 TCP flows serviced by 10MBps link with RTT's uniformly distributed in $[20, 200]ms$. A basic observation is that the performance of TCP-like congestion control algorithms, whose dynamics depend on round-trip time, is significantly affected by queueing delays. By increasing the queueing delay, the aggressiveness of TCP senders is increased, implying lower loss rates. From a fairness perspective, larger queueing delays decrease bias against long-RTT connections. Indeed, for two TCP connections, with round trip times RTT_1, RTT_2, $RTT_1 < RTT_2$, bottlenecked at a single link with queueing delay d_0, the ratio of their expected rates [8] is $\frac{RTT_1+d_0}{RTT_2+d_0}$. Increas-

[8] This follows from the square root formula (3).

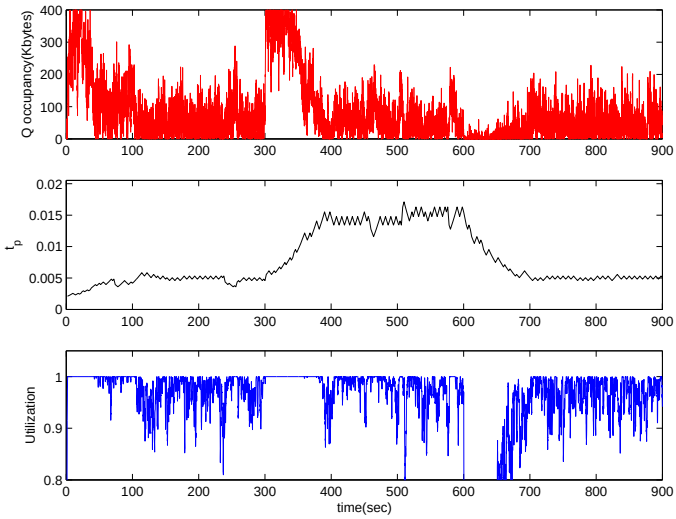

Fig. 15. Simulation G. Queue occupancy, drop probability(t_p), and utilization for OB servicing a varying number of TCP flows.

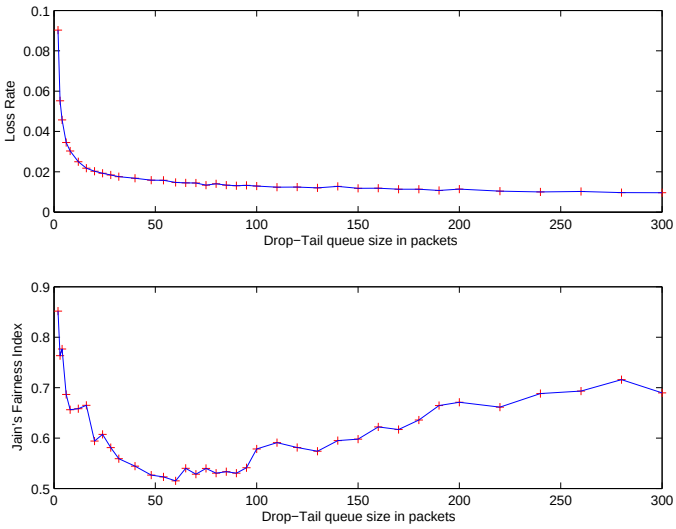

Fig. 16. Simulation H. Loss rates (top) and JFI (bottom) for 50 TCP flows serviced by Drop-Tail queues of different sizes.

ing, d_0 leads this ratio to a value closer to one. Figure 16 presents the dependance between available space in FIFO Drop-Tail queue and loss rate and JFI. Each '+' corresponds to a 5 minute average. We note that for very small queue sizes (< 50 packets), loss rates are large and TCP dynamics is dominated by timeouts. In this regime the square root formula is not valid and fairness is impacted mainly by timeout mechanism. Corresponding average queueing delays and utilization are depicted in Figure 1.

Figure 17 depicts the JFI for the same set of 50 flows, on link where each packet is dropped on arrival with constant probability t_p. The corresponding average queueing delays and utilization are depicted in Figure 2. Note that for t_p in range $[0.005, 0.01]$, queueing delays are small ($< 5ms$) compared to the RTT's and that the loss rate is small enough implying good accuracy of the square root formula. As result of this, the JFI is roughly constant in this range.

Simulation I. Our final simulation compares ODT and OB with the queueing strategies that does not explicitly exploit the tradeoff between the queueing delays and the link utilization. The key observation is the fact that ODT and OB have direct control over both parameters and can therefore work on the "knee" of the curve. The simulation setup is the following. We run $N = 1, 10, 100$ TCP flows over a link with speed $C = 10MBps$. The RTT of the flows are taken randomly in the range $[100, 300]ms$ and the ODT and OB parameters are the same as in the previous simulations; in particular we set $\gamma = 10$. We compare ODT and OB with: droptail queue worth $Q_{max} = 10ms$ of buffering (DT small); droptail queue with $Q_{max} = 200ms$ buffering (DT large) that corresponds to the buffer-delay product [30]; and adaptive RED queue that aims to achieve high utilization and low queueing delay by dropping packets early and avoiding TCP synchronization[1]. The results are shown in Table 8. While DT-small can achieve very small queueing delays, it consequently harms the utilization of the egress link. Large buffer droptail queue (DT large) in contrast achieves 100% utilization while at the same time it results in very large queuing delays. Adaptive RED has results that lie between the above two but however does not directly try to optimize for the queueing delay versus utilization tradeoff. Both ODT and OB manage to optimize the relevant cost function in an efficient manner.

Scheme	$N = 1$	$N = 10$	$N = 100$
ODT	0.88/58ms	0.95/16ms	0.99/3ms
OB	0.89/60ms	0.96/14ms	0.99/3ms
DT small	0.76/1ms	0.85/3ms	0.95/5ms
DT large	1.00/105ms	1.00/161ms	1.00/187ms
ARED	0.81/22ms	0.99/31ms	1.00/54ms

Table 4
Numerical comparison between different queueing strategies. Each entry is a pair of average utilization/average queueing delay (in milliseconds).

7. Summary

In this paper we have addressed the problem of utilizing the tradeoff between queueing delays and link utilization. By specifying the relative importance of queueing delays and utilization, an optimal choice of a queue management parameter is the one that maximizes the overall benefit defined by (4).

The optimization problem (5) assumes a linear dependance between utilization and benefit, and completely neglected other important performance metrics such as jitter, loss probability, and fairness. In fact, one can define the general overall benefit of the queueing scheme controlled by parameter t as:

$$B_G(t) = V(u(t)) - P_1(aQd(t)) - P_2(j(t)) - P_3(L(t)) - \\ -P_4(f(t)), \quad (22)$$

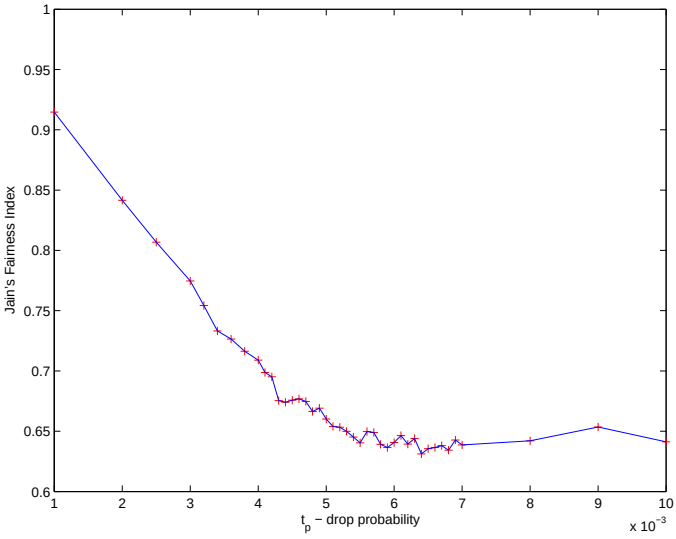

Fig. 17. Simulation H. JFI for 50 TCP flows serviced by queue with constant drop probability.

where $j(t)$ is jitter, $L(t)$ is the loss rate, $f(t)$ is a fairness indicator, $V(u(t))$ is the value of the utilization $u(t)$, and $P_i, i = 1, 2, 3, 4$ are appropriate price functions. We again emphasize the importance of fairness in TCP environments where long-RTT connections could heavily suffer from low queueing delays at the congested links. The embedding of jitter and loss rate into current framework can be done in straightforward manner. However, including fairness into the optimization framework, would be much more challenging as we are not aware of any, computationally light, estimation technique that would faithfully indicate level of fairness. One possible approach to estimate level of the fairness could be by counting runs [9] as suggested in [20]. However, a runs counter would give estimate of Jain's fairness index [15] only for flows that have experienced runs and Jain's fairness index is just one possible fairness indicator.

In this paper we implemented two strategies for settling the underlying optimization problem: in one, the control variable is the available queue space, while another controls the drop probability. Queueing schemes that use different control parameters are possible as well. For example, virtual queue capacity could be powerful in the control of delay/utilization, and therefore could be basis for ODT/OB-like scheme.

From the theoretical point of view, an important open issue is convexity (concavity) of the average utilization/Q-delays/ loss-rates as function of control parameter t (available buffer space, random drop probability, virtual queue capacity, etc). While some results exist for the nonelastic traffic [21,22], in the case of elastic traffic, arrival process depends on the control parameter, which makes modelling of the corresponding tradeoff curve much more challenging.

[9] Run is event where arriving packet belongs to the same flow as some, previously arrived packet.

References

[1] G. Appenzeller, I. Keslassy, N. McKeown. "Sizing router buffers". Proceenings of the ACM SIGCOMM '04, Portland, Oregon, USA, 2004.

[2] K. Avrachenkov, U. Ayesta, A. Piunovskiy. "Optimal choice of the buffer size in the Internet routers". Proceedings of the IEEE CDC, pp. 1143-1148, Seville, Spain, 2005.

[3] A. Dhamdhere, H. Jiang, C. Dovrolis. "Buffer sizing for congested internet links". Proceedings of IEEE INFOCOM, Miami, FL, USA, 2005.

[4] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown,T. Roughgarden. "Part III: routers with very small buffers". ACM Computer Communication Review 35(3): 83-90 (2005).

[5] C. Estan, G. Varghese. "New directions in traffic measurement and accounting". Proceedings of ACM SIGCOMM '02, Pittsburgh, Pennsylvania, USA, 2002.

[6] C. Estan, G. Varghese, M. Fisk. "Bitmap algorithms for counting active flows on high speed links". Proceedings of 3rd ACM SIGCOMM conference on Internet measurement. Miami Beach, Florida, USA, 2003.

[7] W. Fang, L. Peterson. "Inter-as traffic patterns and their implications". Proceedings of IEEE GLOBECOM'99, Rio de Janeiro, Brazil, 1999.

[8] W. Feng, D. D. Kandlur, D. Saha, K. G. Shin. "A Self-Configuring RED Gateway". Proceedings of INFOCOM, New York, NY, USA, 1999.

[9] W. Feng, K.G. Shin, D.D. Kandlur, D. Saha. "The BLUE active queue management algorithms". *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, 513-528, August 2002.

[10] S. Floyd. "HighSpeed TCP for Large Congestion Windows". RFC 3649, Experimental, December 2003.

[11] S. Floyd, V. Jacobson. "Random early detection gateways for congestion avoidance". *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397 -413, Aug. 1993.

[12] R. J. Gibbens, F. P. Kelly. "Distributed Connection Acceptance Control for a Connectionless Network", 16th International Teletraffic Conference, Edimburgh, June 1999, pp. 397-413.

[13] C. Hollot, V. Misra, D. Towsley, W.B. Gong. "Analysis and design of controllers for AQM routers supporting TCP flows" *IEEE Transactions on Automatic Control*, pp. 945-959 June, 2002.

[14] V. Jacobson. "Congestion avoidance and control". Proceedings of SIGCOMM, 1988.

[15] R. Jain. "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling". John Wiley and Sons, INC., 1991.

[16] H Jiang , C. Dovrolis. "Passive estimation of TCP round-trip times". ACM SIGCOMM Computer Communication Review, v.32 n.3, p.75-88, July 2002.

[17] S. Jaiswal, G. Iannaccone, C. Diot, D. F. Towsley. "Inferring TCP connection characteristics through passive measurements". Proceedings of INFOCOM, Hong Kong, China, March 2004.

[18] F.P. Kelly, A.K. Maulloo, D.K.H. Tan. "Rate control for communication networks: shadow. prices, proportional fairness and stability". Journal of the Operational Research Society, Vol. 49 (3), pp. 237-252, March 1998.

[19] T. Kelly. "Scalable TCP: Improving Performance in Highspeed Wide Area Networks". ACM SIGCOMM Computer Communication Review Volume 33 , Issue 2, April 2003.

[20] M. Kodialam, T. V. Lakshman, S. Mohanty. "Runs bAsed Traffic Estimator (RATE): A Simple, Memory Efficient Scheme for Per-Flow Rate Estimation". Proceedings of IEEE INFOCOM, Hong Kong, China, March 2004.

[21] K. Kumaran, M. Mandjes. "The buffer-bandwidth trade-off curve is convex". Queueing Systems, 38 (2001), no. 4, 471–483.

[22] K. Kumaran, M. Mandjes, A. Stolyar. “Convexity properties of loss and overflow functions”. Operations Research Letters, 31 (2003), no. 2, 95–100.

[23] S. Kunniyur, R. Srikant. “Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management”. *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, 286-299, April 2004.

[24] Online: http://pma.nlanr.net/Special/.

[25] J. Padhye, V. Firoiu, D. F. Towsley, J. F. Kurose. “Modeling TCP Reno performance: a simple model and its empirical validation”. *IEEE/ACM Transactions on Networking*, Volume 8 , Issue 2 ,April 2000.

[26] B. Pearlmutter. “Bounds on query convergence”. Preprint 2005. Online: http://arxiv.org/abs/cs.LG/0511088.

[27] J. Sommers, P. Barford, A. Greenberg, W. Willinger. “A SLA perspective on the. router buffer sizing problem. ACM SIGMETRICS Performance Evaluation Review, vol 35(4), 2008.

[28] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkḧauser, 2004.

[29] B. Veal, K. Li, D. Lowenthal. “New Methods for Passive Estimation of TCP Round-Trip Times”. Proceedings of PAM, Boston, MA, USA, March 2005.

[30] C. Villamizar, C. Song. “High Performance TCP in ANSNET”. ACM Computer Communication Review, 24(5), 1994

[31] D. Wischik and N. McKeown. “Part I: Buffer sizes for core routers”. ACM Computer Communication Review 35(3): 75-78 (2005).

[32] Y. Zhang, L. Breslau, V. Paxson, S. Shenker. “On the Characteristics and Origins of Internet Flow Rates”. Proceedings of ACM SIGCOMM, Aug. 2002, Pittsburgh, PA.