Computer Networks 54 (2010) 2562–2574

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

ABS: Adaptive buffer sizing for heterogeneous networks

Yueping Zhang [a,*], Dmitri Loguinov [b]

[a] NEC Laboratories America, Inc., Princeton, NJ 08540, United States
[b] Department of Computer Science, Texas A&M University, College Station, TX 77843, United States

ARTICLE INFO

Article history:
Received 6 October 2008
Received in revised form 21 November 2009
Accepted 6 April 2010
Available online 11 April 2010
Responsible Editor: S. Mascolo

Keywords:
Buffer sizing
Heterogeneous traffic
PID control
Gradient descent

ABSTRACT

Most existing criteria [3,5,9] for sizing router buffers rely on explicit formulation of the relationship between buffer size and characteristics of Internet traffic. However, this is a non-trivial, if not impossible, task given that the number of flows, their individual RTTs, and congestion control methods, as well as flow responsiveness, are unknown. In this paper, we undertake a completely different approach that uses control-theoretic buffer size tuning in response to traffic dynamics. Motivated by the *monotonic* relationship between buffer size and loss rate and utilization, we design a mechanism called Adaptive Buffer Sizing (ABS), which is composed of two Integral controllers for dynamic buffer adjustment and two gradient-based components for intelligent parameter training. We demonstrate via ns2 simulations that ABS successfully stabilizes the buffer size at its *minimum* value under given constraints, scales to a wide spectrum of flow populations and link capacities, exhibits fast convergence rate and stable dynamics in various network settings, and is robust to load changes and generic Internet traffic (including FTP, HTTP, and non-TCP flows). All of these demonstrate that ABS is a promising mechanism for tomorrow's router infrastructure and may be of significant interest for the ongoing collaborative research and development efforts (e.g., GENI and FIND) in reinventing the Internet.

1. Introduction

One of the key components of Internet routers is the I/O buffer, which is closely linked to various critical performance metrics, including packet loss rate, end-to-end delay, and utilization level. On one hand, router buffers should be large enough to accommodate transient bursts in packet arrivals and hold enough packets to maintain high link utilization. On the other hand, large buffers in turn leads to increased queuing delays and may potentially cause instability of TCP in certain scenarios [27]. Clearly, optimally determining the required buffer size is of immense importance for router manufactures when configuring their routers for the future high-speed Internet and significantly affects the ability of large Internet service providers (ISP) to deliver and guarantee competitive Service Level Agreements (SLA) [34].

As today's Internet rapidly grows in scale and capacity, it becomes widely recognized that the classic Bandwidth-Delay-Product (BDP) [36] rule for sizing router buffers is no longer suitable for the future Internet. In addition, the Internet is foreseeing a disruptive evolution driven by focused collaborative efforts such as the NSF Global Environment of Network Innovations (GENI) and Future Internet Network Design (FIND) initiatives. This imposes significant challenges as well as great opportunities for all most every corner of Internet technologies, including the next-generation infrastructure for router buffer management. As a consequence, there has emerged in the research community a surge of renewed interest [3,5,6,9,10,16,18,19,23,29,30,36,38,39] in the buffer-sizing problem during the last five years. However, these results present vastly different, even contradictory, views on how to optimally dimension the buffer of a router interface. In addition, all these results rely on certain assumptions of the incoming Internet traffic

[*] Corresponding author. Tel.: +1609 951 2647; fax: +1 609 951 2480.
E-mail addresses: yueping@nec-labs.com (Y. Zhang), dmitri@cs.tamu.edu (D. Loguinov).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.04.003

and may have limited applications to and exhibit undesirable behavior in other traffic models. In contrast, Shorten et al. [35] take a completely different approach and models the buffer-sizing problem as the Lur'e problem. Under this model, they proposed a dynamic buffer sizing algorithm called Adaptive Drop Tail (ADT). However, the control parameter K depends on the underlying Lur'e formulation and can hardly be obtained without off-line calculation. Thus, it still remains open to develop a *model-independent* buffer-sizing mechanism that is able to ideally allocate buffers under different traffic patterns.

In this paper, we achieve the goal of buffer sizing by proposing a new buffer management infrastructure, where the router adapts its buffer size to the dynamically changing incoming traffic based on one or more performance constraints. We first formulate buffer sizing as the following problem. Let B be the total size of router's memory and $b_l(t)$ be the amount of buffer allocated to link l at time t. Then, the problem becomes determining the optimal buffer size for each link l under the constraint that $\sum_l b_l(t) \leqslant B$. We then propose that this problem can be alternatively solved by leveraging the *monotonic* relationship between buffer size b_l and various performance metrics (e.g., utilization u, loss rate p, and queuing delay q). Rigorously proving this relationship is very difficult and out of scope of the paper. Instead, we provide an intuitive explanation of this result using a simple yet generic congestion control model.

Utilizing this result, we design a buffer management mechanism, called *Adaptive Buffer Sizing* (ABS), which dynamically determines the minimum buffer size satisfying the target performance constraints based on real-time traffic measurements. ABS consists of two sub-controllers ABS_u and ABS_p, each of which employs an Integral controller that adapts to dynamics of input traffic by regulating the buffer size based on the error between the measured and target values of utilization and loss rate, respectively. However, we observe that the naive Integral controller ABS_u drives buffers of non-bottleneck routers to infinity. We successfully address this problem by introduce a damping component, such that the resulting ABS_u quickly converges buffers to their equilibrium values in both bottleneck and non-bottleneck routers.

Another challenge is how to tune integral gains for optimal performance. Improper parameter settings may lead to undesirable system behavior, such as slow convergence and persistent oscillations. We solve this problem by associating with each sub-controller a gradient-based parameter training component, which is capable of automatically adapting parameters to achieve their *optimal* values under the current ingress traffic. We evaluate the resulting controller in ns2 simulations and demonstrate that ABS is able to deal with generic Internet traffic consisting of HTTP sessions, different TCP variants, and non-TCP flows, is robust to changing system dynamics, and is scalable to link capacities and flow populations, all of which make the concept of ABS an appealing and practical buffer sizing framework for future Internet routers.

The rest of the paper is organized as follows. In Section 2, we explore the monotonic relationship between b, p, and u. In Section 3, we present ABS. In Section 4, we examine ABS in a variety of scenarios via ns2 simulations. In Section 5, we review related work on buffer sizing. In Section 6, we conclude our work and point out directions for future work.

2. Motivation

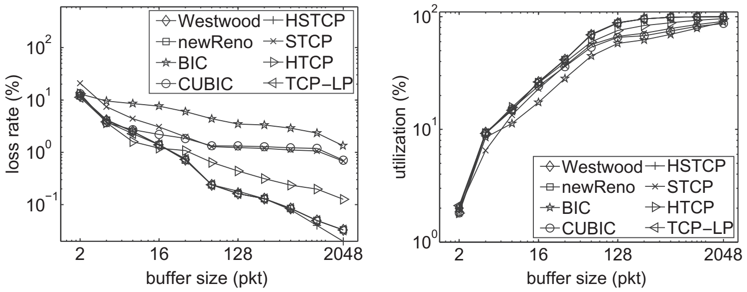
While a comprehensive modeling of Internet traffic and its relationship with buffer size b remains open, we show in this subsection that there are strong indications that there exists a *monotonic* relationship between b and two key performance metrics, loss rate p and utilization u. Due to the extreme difficulty of the problem, we do not seek to present a rigorous proof of this monotonic relationship, but provide an intuitive explanation experimentally via ns2 simulations and analytically using a simple congestion control model. This monotonic relationship serves as motivation and foundation of our adaptive buffer sizing scheme proposed in the following section.

2.1. Simulation illustration

We first empirically examine the impact of the buffer size on the performance of different congestion control protocols using ns2 simulations. To accomplish this goal, we utilize the framework developed in [37], which incorporates into ns2 the Linux-2.6.16.3 implementations of several proposed TCP variants, including newReno [12], BIC [40], CUBIC [31], HSTCP [11], HTCP [26], STCP [24], Westwood [17], and TCP-LP [25]. The simulation setup is composed of one bottleneck link of capacity 100 mb/s and ten sources with packet size 1500 bytes and RTTs uniformly distributed in [30, 50] ms. We set buffers of access links to be 2500 packets and verify that no packet is lost at these links in all simulations. The plots of loss rate p and utilization u under different bottleneck buffer sizes are given in Fig. 1, from which we can see that p and u respectively monotonically decreases and increases as b grows. Note that the loss curves of CUBIC and STCP in Fig. 1(a) appear to be flat when b is between 64 and 1024 packets; however, we verified numerically that they are actually monotonically decreasing.

2.2. Intuition

This monotonic relationship is not surprising. It is intuitive clear that a larger buffer can absorb more bursts in packet arrivals, thereby reducing the frequency of packet drops. In addition, assuming the buffer is always depleted between events of packet drops, it takes longer time for sources to saturate a larger buffer so that to experience the next packet loss. Thus, the average loss rate is a decreasing function of the buffer size. At the same time, a larger buffer can hold more packets and thus maintain the bottleneck link at full utilization for a longer time when senders back off in response to congestion. Therefore, the average utilization level proportionally scales with the buffer size. However, the above reasonings assume a depleted queue after each packet loss and may not be obvious otherwise. Thus, it is desirable if we can obtain a more generic



(a) loss rate

(b) utilization

Fig. 1. Effect of buffer size b on loss rate p and utilization u of several TCP variants.

explanation with less restrictive assumptions. We next seek to achieve this goal with a simple congestion control model.

2.3. Simple model

We note that the goal of this section is not to present a comprehensive congestion control model that is able to formulate generic Internet traffic, but to intuitively explain results observed in the previous section using a simple, but illustrative, model. We start with the definition of p and u in mathematical terms. Consider a scenario where traffic passes through a single-channel FIFO queue of capacity b and service rate C. Let $L(t)$ and $A(t)$ respectively denote the number of lost and admitted packets by time t. Then, p is defined as the *long-term* average loss rate $p = \lim_{t\to\infty} L(t)/(L(t)+A(t))$, $u = \lim_{t\to\infty} A(t)/Ct$ as the average utilization, and $\lambda = \lim_{t\to\infty} A(t)/t$ as the average input rate.

Using these definitions, we next examine the effect of buffer size on traffic that reacts to congestion using a simple model. Denoting by $W_i(n)$ the congestion window size of flow i during the n-th RTT, we can model a generic congestion control algorithm as following[1]:

$$W_i(n) = \begin{cases} W_i(n-1) + \alpha_i(W_i(n-1)) & \text{no loss}, \\ W_i(n-1) - \beta_i(W_i(n-1)) & \text{loss}, \end{cases} \tag{1}$$

where $\alpha_i(\cdot)$ and $\beta_i(\cdot)$ are non-negative functions and each discrete time step corresponds to one RTT. This simple model subsumes a wide spectrum of loss-based congestion control protocols, including AIMD (e.g., Reno [1] and Westwood [17]), MIMD (e.g., Scalable TCP [24]), and many other existing TCP variants (e.g., BIC [40], TCP-LP [25], and HSTCP [11]). Note that delay-based schemes (e.g., FAST [22] and Vegas [7]) generally are not sensitive to buffer size b as long as it is kept larger than the stationary queue length q^* of the system. However, when $b < q^*$ these methods experience packet losses and their responses can also be modeled by (1). Moreover, since (1) allows different response functions $\alpha_i(\cdot)$ and $\beta_i(\cdot)$ for different flows i, this model applies to scenarios where the traffic is generated by a *mixture* of protocols.

Assuming N sources with homogeneous RTTs access a single link of capacity C and letting $q(n)$ be the queue length at time n, we can model the queuing dynamics as:

$$q(n) = \min((q(n-1) + X(n) - C)^+, b), \tag{2}$$

where b is the buffer size and $X(n) = \sum_{i=1}^{N} W_i(n)$ is the total number of arrivals during the n-th RTT. Assuming that $W_i(n)$ of each source i is bounded above by W_{max}, the system dynamics can be represented by a discrete Markov chain with state $S_n = [W_1(n), W_2(n), \ldots, W_N(n)]$ and state space $O : [1, 2, \ldots, W_{max}]^N$.

Let $Z(n)$ be the number of dropped packets during the n-th RTT: $Z(n) = (q(n-1) + X(n) - C - b)^+$. Define $\upsilon(n) = Z(n)/X(n)$ as the average loss rate during this RTT. Assuming packet loss rate of each flow is independent of each other, we have $Pr\{W_i(n+1) = W_i(n) + \alpha_i(W_i(n))\} = (1-\upsilon(n))^{W_i(n)}$ and $Pr\{W_i(n+1) = W_i(n) - \beta_i(W_i(n))\} = 1-(1-\upsilon(n))^{W_i(n)}$. Based on this, we can derive the transition probability between any pair of states. Furthermore, it is clear that the transition probability depends only on the previous state, which implies that series $\{S_n\}$ is a Markov chain. Then, the following result is easy to obtain.

Theorem 1. *The Markov chain defined by* (1) and (2) *always converges to a stationarity distribution.*

According to Theorem 1, for any fixed N and starting from any initial state, the Markov chain defined by (1) and (2) always converges to its steady state. Thus, we omit the transient phase in the rest of the section and only examine the queuing process under traffic generated by a stationary Markov chain. In such a scenario, the following result shows that packet loss rate p scales inversely proportionally to the buffer size b.

Theorem 2. *Loss probability p in a finite queue fed by traffic governed by a stationary Markov chain defined by* (1) and (2) *monotonically decreases in queue size b.*

Proof. Under the stationary Markov chain defined by (1) and (2), denote by S_n the state at time n, by M the number of states, and by $\pi = [\pi_1, \pi_2, \ldots, \pi_M]$ the stationary probability vector of each state (i.e., $\pi_i = Pr\{S_n = i\}$). Further let $A_{i,j}(k)$ be the probability that the chain goes from state i to j and the next arrival has k packets, i.e., $A_{i,j}(k) = Pr\{X(n+1) = k, S_{n+1} = j | S_n = i\}$. Then, define A_k as the probability matrix

[1] We assume $W_i(n)$ are rounded to integers during calculations and omit the corresponding ceiling function for brevity.

whose (i,j)-th element is $A_{i,j}(k)$. Using these variables, we can represent the traffic density ρ as $\rho = \boldsymbol{\pi}\sum_{k=1}^{\infty} k\boldsymbol{A}_k\boldsymbol{e}$, where e is a column vector with all elements equal to one.

We next express loss probability of a finite buffer of size b in terms of the queue length distribution of an infinite buffer, whose queuing process $q'(n)$ is given by:

$$q'(n) = (q'(n-1) + X(n-1) - c)^{+}. \tag{3}$$

Consider the steady state probability matrix $\boldsymbol{\Delta}$ of the infinite buffer, where the (k,i)-th element $\Delta_{k,i}$ is:

$$\Delta_{k,i} = \lim_{n\to\infty} Pr\{X(n) = k, S_n = i\}. \tag{4}$$

Then, according to [21, Theorem 4], under arrivals governed by the same stationary Markov chain, loss probability p of a finite buffer of size b is represented by:

$$p = \frac{(1-\rho)\sum_{k=b+1}^{\infty}\boldsymbol{\Delta}_k\mathbf{A}_0\mathbf{e}}{\rho\sum_{k=0}^{b}\boldsymbol{\Delta}_k\mathbf{A}_0\mathbf{e}}, \tag{5}$$

where $\boldsymbol{\Delta}_k$ is the k-th row vector of $\boldsymbol{\Delta}$. Since the Markov chain is stationary, all variables in the last equation are constant. Thus, it follows that loss rate p monotonically decreases as buffer size b increases. □

Moreover, it is clear that utilization u scales inversely to loss rate p according to a general TCP model of the form $r = c/p^d$, where r is the throughput and c and d are constants [40]. This holds for various TCP flavors including Reno, BIC, HSTCP, and STCP. Further notice that according to Theorem 2, loss rate p scales inversely to b. This implies that utilization u monotonically increases in b. In addition, it is rather obvious that queuing delay q scales proportionally to buffer size b. Thus, in the rest of the paper we assume the monotonic relationship between u and b and q and b.

Finally, we should emphasize that although system (1) is generic enough to represent the increase/decrease behavior of a wide class of congestion control algorithms, it is by no means comprehensive. To make the model complete, one should additionally consider factors such as heterogeneous delay, slow start, timeouts, and retransmissions. The pure purpose of this model is to provide an intuitive explanation of the monotonic relationship between b and u, p. This result motivates us to design a dynamic buffer-sizing mechanism presented below.

3. Adaptive buffer sizing (ABS)

In this section, we describe a dynamic buffer sizing framework that is adaptive to dynamics and uncertainties of input traffic while maintaining the system under target performance constraints such as loss rate p^* and utilization u^*. As an example of this framework, we start with a simple mechanism, progressively identify and overcome its underlying drawbacks, and eventually arrive at the final controller that we call ABS.

3.1. General consideration

To design a buffer sizing mechanism, first it is necessary to understand how buffers are managed in current commercial routers. The memory system in a Cisco 3600 series router [8], for instance, is composed of the main processor memory, shared (packet) memory, flash memory, Nonvolatile Random Access Memory (NVRAM), and Erasable Programmable Read Only Memory (EPROM). Among them, we are particularly interested in shared (packet) memory, which is used for packet buffering by the router's network interfaces. Specifically, each interface is associated with an Input Hold Buffer (IHB), which resides in the system buffer of shared memory and is used to store packets for transfer between fast switching and process switching code. For each packet arriving into an interface, the interface driver writes it into an IHB. An incoming packet is immediately dropped if the IHB reaches its maximum size, which is static and does not grow or shrink based on demands. Our goal in this paper is redesign IHB such that its size adapts to dynamics of the incoming traffic.

We note that it is important to distinguish the framework of dynamic buffer sizing from the large class of AQM algorithms (e.g., RED [13], REM [4], and PI [20]). These methods operate within a given buffer size b_l and aim to stabilize the queue occupancy (or queuing delay) at a certain target level, which is a portion of the selected buffer size b_l. Thus, AQM is unable to solve issues associated with incorrectly sized router buffers.

To better see this, we test several TCP variants under REM using ns2 simulations. Recall that an REM-enabled router dynamically updates its packet dropping/marking probability by monitoring the discrepancy between the aggregate input rate $y(t)$ and link capacity C and the difference between the current queue length $q(t)$ and its target value q^*. In the steady state, the system achieves $y(t) = C$ and $q(t) = q^*$. In our simulations, we use a simple "dumbbell" topology with a single REM ($q^* = 50$ pkts) link of capacity $C = 10$ mb/s shared by 20 TCP sessions. We use marking at the REM router and enable ECN at end-users. As seen from Table 1, if we set buffer size of the bottleneck link to $b = 100$ pkts, which is greater than REM's target queue size $q^* = 50$ pkts, REM successfully maintain the queue size close to q^* while achieving 100% utilization and 0% packet loss for all TCP variants. However, if we set buffer size b below q^*, the bottleneck link suffers significant under-utilization and prohibitively high loss rate.

In contrast, dynamic buffer-sizing mechanisms focus on determining the optimal size of the physical buffer. This way, the router can efficiently allocate its available buffers among different memory-sharing interfaces, hereby achieving pre-agreed QoS requirements, shrinking the required space of the main memory, and reducing the system

Table 1
Performance of different TCP variants with REM ($q^* = 50$ pkts) under different buffer sizes.

	$b = 100$ pkts			$b = 10$ pkts		
	q (pkts)	p (%)	u (%)	q (pkts)	p (%)	u (%)
Reno	56.14	0.00	100.00	5.41	9.88	84.78
BIC	52.88	0.00	100.00	5.27	9.04	86.52
CUBIC	52.48	0.00	100.00	4.92	7.84	87.22
HSTCP	56.73	0.00	100.00	4.96	9.60	86.59
STCP	54.38	0.00	100.00	5.40	12.48	83.74
HTCP	57.77	0.00	100.00	4.60	8.20	87.46
Westwood	54.61	0.00	100.00	4.99	10.08	84.50

cost and board space. Dynamic buffer sizing schemes can overcome the problem of improper buffer sizing that AQM is unable to solve or may be combined with AQM methods to achieve desired performance. In existing Internet routers where memory is already fixed, the proposed approach is also valuable since it guarantees the minimum queuing delay in each interface under predetermined performance constraints. Additionally, ABS lends ISPs and router manufactures great freedom in choosing preferred constraints when configuring their routers.

3.2. Controller design

Although the underlying differential/difference equations describing the effect of router buffer size on Internet traffic remain unknown, it follows from the last section an important property of this relationship – *monotonicity*. This implies that for any given feasible loss rate p^* (or utilization u^*) under stationary input traffic, there exists a *unique* buffer size b^* such that the resulting system achieves p^* (or u^*). In addition, this monotonic relationship gives us a hint of the correct direction in which we should adjust the buffer size. Specifically, assuming target loss rate p^* and its actual value $p(n)$ measured at time n, the router buffer size $b(n)$ should increase if $p(n) > p^*$ and decrease otherwise. Analogously, given u^* and $u(n)$, $b(n)$ should decrease if $u(n) > u^*$ and increase otherwise. This result allows us to develop simple yet robust controllers to adaptively size router buffers to satisfy given system constraints.

One natural candidate for achieving this goal is the Integral controller. First, consider the controller under the loss rate constraint, in which case $b_p(n)$ denotes the buffer size at time n and $e_p(n) = p(n) - p^*$. Then, the time-domain Integral controller can be represented by the following difference equation:

$$b_p(n) = b_p(n-1) + I_pT(p(n) - p^*), \tag{6}$$

where T is the sampling interval and I_p is the integral gain. Similarly, we obtain the control equation of $b_u(n)$ under the utilization constraint:

$$b_u(n) = b_u(n-1) - I_uT(u(n) - u^*), \tag{7}$$

where I_u is the integral gain. It is noteworthy to point out that since $p(n)$ monotonically decreases with $b(n)$ while $u(n)$ increases with $b(n)$, controllers (6) and (8) have opposite signs before their respective integral gain.

However, the last controller invites a serious problem if deployed in non-bottleneck routers. This is because a non-bottleneck router is always under-utilized regardless of its buffer size. Thus, if u^* is set to be above the maximally achievable utilization level of the link, the router always have $u(n) < u^*$ and drives its buffer size to infinity. We overcome this problem by modifying (7) as follows:

$$b_u(n) = b_u(n-1) - I_uT(u(n) - u(n-1))(u(n) - u^*). \tag{8}$$

Compared to (7), the extra term $u(n) - u(n-1)$ in (8) is to damp the effect of $(u(n) - u^*)$. Specifically, in the steady state of a non-bottleneck router, we must have $u(n) - u(n-1) = 0$, which forces the second term of (8) to converge to zero and prevents $b_u(n)$ from diverging to infinity.

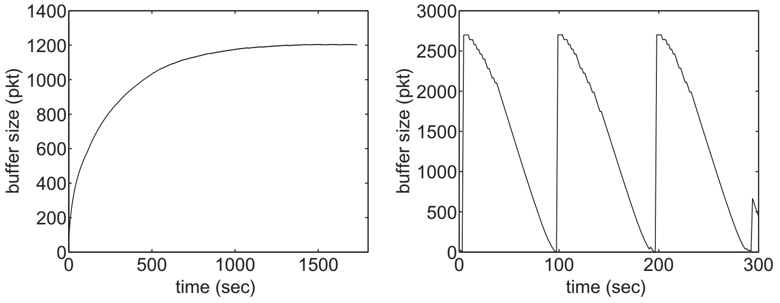
Now, we have two buffer sizes $b_u(n)$ and $b_p(n)$ based on the utilization and loss rate constraints, respectively. Similar to BSCL [9] , the minimum buffer size $b(n)$ satisfying both requirements should be the larger of $b_p(n)$ and $b_u(n)$, i.e.,

$$b(n) = \max(b_u(n), b_p(n)). \tag{9}$$

We call the hybrid controller (6)–(9) Adaptive Buffer Sizing (ABS) scheme and sub-controllers (6) and (8) ABS$_p$ and ABS$_u$, respectively. We note that in practice buffer size allocated by ABS is always bounded from above by the size of physical memory. Specifically, denoting by $M(n)$ the amount of physical memory available for allocation at time n, buffer size $b(n)$ at time n should not exceed the sum of its current value $b(n-1)$ and the additional memory $M(n)$. This translates (9) into $b(n) = \min(\max(b_u(n), b_p(n)), b(n-1) + M(n))$. It is easy to see that the previous equation only imposes a physical cap to (9) and has no effect on the system's stability (which is verified using ns2 simulations in Section 4.5). Thus for ease of presentation, in the rest of this paper except Section 4.5, we assume that the physical memory is over provisioned and buffer size requirement calculated by ABS can always be achieved.

The above two constraints, *small loss* and *high utilization*, both follow the same philosophy – maximizing the buffer size subject to the given performance constraints. We call this class of objectives type-A. All other type-A objectives (such as small jitter and high R score) can be easily incorporated into ABS. In contrast, there exists another class of objectives (which we call type-B), such as *small queuing delay*, which try to minimize the buffer size to satisfy the constraints. Type-A and -B objectives are contradicting and cannot be satisfied simultaneously. In this paper, we focus on automatically satisfying type-A objectives. Optimizing type-B objectives is out of scope of this paper and in general easy to achieve (e.g., by simply setting $buffer\ size = C \times max_{delay}$). In practice, the network operator has to decide to either impose a hard upper bound on buffer size to achieve small queuing delay or apply ABS to automatically tune the buffer to achieve type-A objectives. Furthermore, it is recently proposed by Gorinsky et al. [28] that we can let flows to decide whether to achieve type-A or -B objectives based on their own performance incentives. This way, in the future Internet an ISP can implement both objectives (type-A and -B), while the flow should decide what it wants. For all flows interested in A and all ISPs deploying it, ABS is fully autonomous.

Note that ABS does not rely on comprehensive prior knowledge of the system being controlled, but adapts the controller according to errors of the output signals $u(n)$ and $p(n)$. As a consequence, ABS is expected to work in practical network settings and be robust to real Internet traffic (more on this below). This controller works very well in many cases. However, its main limitation lies in the difficulty in choosing the optimal gain parameters I_u and I_p. Specifically, if they are chosen too small, the system may suffer from a sluggish convergence rate to the equilibrium; however, if they are set too large, the system may exhibit exceedingly aggressive adaptation behavior and persistently oscillate around, instead of converging to, the stationary point. This phenomenon is illustrated in



(a) $u^* = 95\%$, $p^* = 0.5\%$

(b) $u^* = 70\%$, $p^* = 5\%$

Fig. 2. ABS ($I_u = I_p = 3000$ and $T = 200$ ms) without parameter training in a network with a single link of capacity 10 mb/s and 20 TCP flows.

Fig. 2, where 20 TCP flows share an ABS-equipped bottleneck link of capacity 10 mb/s. We set integral gains $I_p = I_u = 3000$ and control interval $T = 200$ ms. As seen in the figures, ABS is stable and converges the buffer size to 1200 packets when $u^* = 95\%$ and $p^* = 0.5\%$. However, when $u^* = 70\%$ and $p^* = 5\%$ the system becomes unstable and the buffer size periodically oscillates between 1 and 2700 packets.

Due to the lack of the knowledge of the differential equations describing the system, it is unlikely that any off-line pre-training of the controller's parameters I_u and I_p can be effective. Even if such a method could exist, parameters trained for a particular system setting may immediately become inappropriate as the traffic dynamics evolve. We next seek to overcome this issue by designing a parameters tuning mechanism that is able to adaptively converge the control parameters to their optimal values for the current system configuration.

3.3. Adaptive parameters training

It is clearly a non-trivial task to find the *optimal* parameters for controlling such a complex system as the Internet, which is especially the case provided that the system has an *unknown* underlying model and dynamically changes over time. However, we manage to achieve this goal using a simple scheme, which combines the *output error* [2] method and the *gradient descent* [33] technique. In what follows, we explain our method in the context of ABS$_u$ and note that the mechanism for ABS$_p$ can be obtained similarly.

Denote by $I_u(n)$ the instantaneous value of integral gain I_u at time n. Then, rewrite ABS$_u$'s control Eq. (8) as:

$$b_u(n) = f_u(u^*, b_u(n-1), u(n), I_u(n)), \tag{10}$$

where function $f_u(\cdot)$ is given by the right-hand side of (8). Suppose that the router, at the end of the n-th control interval, sets its buffer size to $b_u(n)$ based on (10) and observes that link utilization becomes $u(n+1)$ during the next interval. Then, we know that if we set $u^* = u(n+1)$ as the target utilization, $b_u(n)$ must be the *optimal* output of controller (10) under the same traffic pattern and given buffer size $b_u(n-1)$ and utilization $u(n)$. This is equivalent to saying that under the optimal control gain I_u^*, we must have the following equation:

$$b_u(n) = f_u\big(u(n+1), b_u(n-1), u(n), I_u^*\big). \tag{11}$$

Thus, at every control step, we get the exact value of the inverse function of the controlled plant [2]. Hence, it remains to adaptively adjust $I_u(n)$ to achieve its optimal value I_u^*, which translates into the following optimization problem.

First, define

$$b'_u(n) = f_u(u(n+1), b_u(n-1), u(n), I_u(n)), \tag{12}$$

as the actual controller's output under the current integral gain $I_u(n)$. This value is not used to decide the buffer size, but is applied to the following calculation: $F_u(n) = b'_u(n) - b_u(n)$, which is the difference between the actual and optimal outputs. Then, the optimal control gain I_u^* under the current traffic is the one that minimizes $F_u(n)$. To find this optimal parameter, we use the gradient descent algorithm.

According to the gradient-decent method, since function $F_u(n)$ is differentiable at $I_u(n)$, it decreases fastest along the direction of its gradient $\triangledown F_u(I_u(n))$, which is the derivative of $F_u(n)$ with respect to $I_u(n)$. Thus, at every control step, if the router updates parameter $I_u(n)$ as follows:

$$I_u(n+1) = I_u(n) - \gamma \triangledown F_u(I_u(n)), \tag{13}$$

with step size γ (which is set to 1 in the paper), then we have that sequence $F_u(I_u(1)) \geqslant F_u(I_u(2)) \geqslant \cdots$, which eventually converges to zero. In this case, $I_u(n)$ reaches I_u^*.

Invoking (8), we simply have:

$$\begin{aligned}\triangledown F_u(I_u(n)) &= \frac{dF_u(n)}{dI_u(n)} \\ &= T(u(n+1) - u(n))(u(n+1) - u^*). \end{aligned} \tag{14}$$

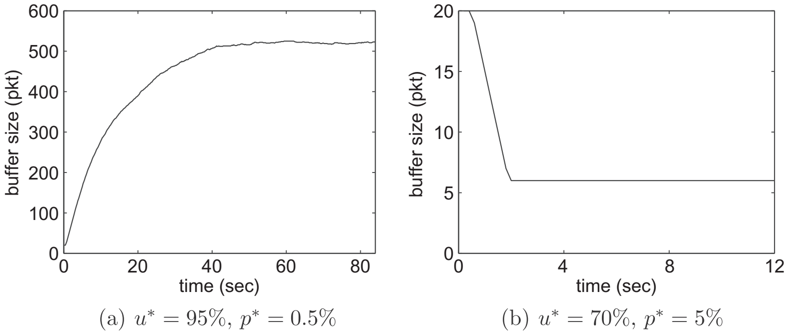
Combining the last two equations, we arrive at the following parameter tuning rule for $I_u(n)$:

$$I_u(n+1) = I_u(n) - \gamma T(u(n+1) - u(n))(u(n+1) - u^*). \tag{15}$$

Following the above techniques, we can derive the following parameter training rule for $I_p(n)$ in ABS$_p$:

$$I_p(n+1) = I_p(n) - \gamma T(p^* - p(n+1)). \tag{16}$$

So far, we have finished the design process of ABS, which now consists of two basic Integral controllers (6) and (8) and two parameter training components (15) and (16). Note that the resulting system is independent of the exact model of the controlled plant and highly adaptive to the



(a) $u^* = 95\%$, $p^* = 0.5\%$ (b) $u^* = 70\%$, $p^* = 5\%$

Fig. 3. ABS with parameter training in a network with a single link of capacity 10 mb/s and 20 TCP flows.

plant's changing system dynamics. In addition, the proposed parameter training mechanism is not limited to our particular case, but applicable to other systems with multiple parameters to be optimized.

To examine performance of the resulting controller, we rerun simulations in Fig. 2. The simulation results are illustrated in Fig. 3, from which we can see that in both cases ABS successfully converges the buffer size to its stationary value and exhibits much faster convergence speed compared to its original version shown in Fig. 2. We note that since traffic loads at Internet routers change slowly over time, buffer sizing schemes are able to utilize long sampling intervals to filter out noise in measurements and achieve more accurate approximation of the systems average behavior. Thus, the convergence rate of ABS should not be confused with that of a congestion control or AQM protocols, whose control actions are usually performed at the time-scale of milliseconds. However, as we demonstrate later in the paper, ABS is in fact very responsive and works well under highly bursty Internet traffic. Finally, we observe in both simulations that gain parameters I_u and I_p indeed converge to their respective optimal value.

4. Performance

We next demonstrate via ns2 simulations performance of ABS under a wide range of flow populations and link capacities, dynamically changing traffic loads, synthetic web traffic, and mixture of TCP and non-TCP flows.

4.1. Implementation

ABS admits a very simple implementation and incurs minimal computational overhead. Specifically, the router maintains two counters S_1 and S_2 to respectively record the amount of data enqueued and dropped by the router during the current sampling interval. For each incoming packet k with size s_k, either S_1 or S_2 is incremented by s_k depending on whether the packet is admitted. Thus, there is only one addition per packet. At the end of the n-th interval, the router computes loss rate using $p(n) = S_1/(S_1 + S_2)$ and utilization using $u(n) = (S_1 + S_2)/(CT)$ where C is the link's capacity. Then, the router calculates the gain parameters based on (15) and (16) and decides its buffer size according to (6)–(9). Since these operations are performed once every control interval (which is set to 20 ms in the paper), the incurred overhead is negligible.

In practice, dynamic buffer sizing may encounter some implementation issues. For instance, one such problem is *memory fragmentation*, which occurs when the router frequently allocates and releases differently sized memory blocks and as a result the memory space contains a lot of small unused pieces. This problem can be mitigated by increasing granularity of memory allocation, i.e., allocating in fixed-size chunks of memory. Sizes of chunks can be 2048 bytes, 4096 bytes, or other values depending on the system. However, for purpose of demonstration, we do not include this mechanism in simulations shown below.

4.2. Scalability

Next, we compare performance of existing buffer sizing mechanisms (i.e., BDP, Stanford model, BSCL, and ABS) under different link capacities C and flow populations N. Note that due to lack of publicly available implementation and unspecified control parameter K, we do not include ADT in this comparison study. We use a "dumbbell" topology with N long-lived TCP flows, whose RTTs are randomly distributed in $[30, 30 + 2N]$ ms. As suggested in [9], we use the harmonic average RTT R_e for the BDP rule. For the Stanford model, we use equation $b = 2R_eC/\sqrt{N}$. In BSCL, we set the loss synchronization factor α to 0.6. In both BSCL and ABS, we set $u^* = 98\%$ and $p^* = 2\%$.[2]

We first fix link capacity $C = 16$ mb/s and vary N between $[2, 1024]$. The simulation results are illustrated in Table 2, in which data are averaged over the second half of each simulation to eliminate initial transient effects. As shown in the table, when the number of flows is small, both the BDP and Stanford rules are not very successful in achieving their design goal (i.e., high link utilization). As N becomes large, both methods do achieve high utilization, but in the expense of high loss rates. This is especially evident in the Stanford model, whose loss rate is 13.45% when there are 1024 flows. Capability of controlling loss rate is improved in BSCL; however, it still cannot achieve the target loss rate $p^* = 2\%$ and suffer from low link utilization when the number of flows is small. In contrast, ABS

[2] Conclusion drawn in this section should not change if we set u^* and p^* to other values.

Table 2
Performance of existing buffer sizing strategies in a single-link network with different numbers of flows N (C = 10 mb/s, u^* = 98%, and p^* = 2%).

N	BDP			Stanford			BSCL			ABS		
	b (pkts)	p (%)	u (%)	b (pkts)	p (%)	u (%)	b (pkts)	p (%)	u (%)	b (pkts)	p (%)	u (%)
2	110	0.08	94.3	155	0.06	95.8	55	0.11	89.6	287	0.03	97.9
4	116	0.18	94.3	116	0.18	94.3	47	0.32	85.5	232	0.09	97.8
8	128	0.42	93.5	91	0.55	88.5	45	0.83	78.6	229	0.23	97.9
16	150	1.05	94.9	75	1.6	87.8	45	1.8	83.4	227	0.74	97.9
32	190	1.76	98.3	68	2.88	92.8	45	3.24	89.8	151	2.04	97.9
64	261	2.76	99.9	66	4.85	95.8	94	4.51	97.6	443	2.00	100.0
128	383	4.06	100.0	68	7.08	97.5	358	4.18	100.0	941	2.05	100.0
256	595	5.11	100.0	75	9.38	98.8	922	4.24	100.0	2006	2.03	100.0
512	965	5.69	100.0	86	11.42	99.8	2106	4.19	100.0	4210	2.00	100.0
1024	1618	6.47	100.0	102	13.45	100. 0	4569	3.94	100.0	8613	2.03	100.0

achieves its design goal under all flow populations. Specifically, when $N \leqslant 32$ and utilization is the primary constraint, ABS successfully maintains link utilization at close to its target value u^* = 98%. As N grows and the loss rate constraint becomes dominant, ABS is still able to effectively allocate buffer such that the average loss rate is within a close neighborhood of p^* = 2%.

It is worth noting that as seen from Table 2, when N = 1024, ABS converges the buffer size to 8613 packets. This buffer size translates into a queuing delay of 10 s, which is prohibitively high for most applications. However, this is not a problem of ABS, but a consequence of an unrealistic choice of p^* given the particular network setting with a single 10 mb/s link shared by 1024 long-lived TCP flows. In practice, router manufactures and ISPs are free to adjust u^* and p^* according to the type of service they agree to provide and the actual traffic situation. To avoid exceedingly large queuing delay, they can increase the link capacity or enforce a predetermined upper bound of buffer size to prevent queuing delay from growing beyond a certain threshold value.

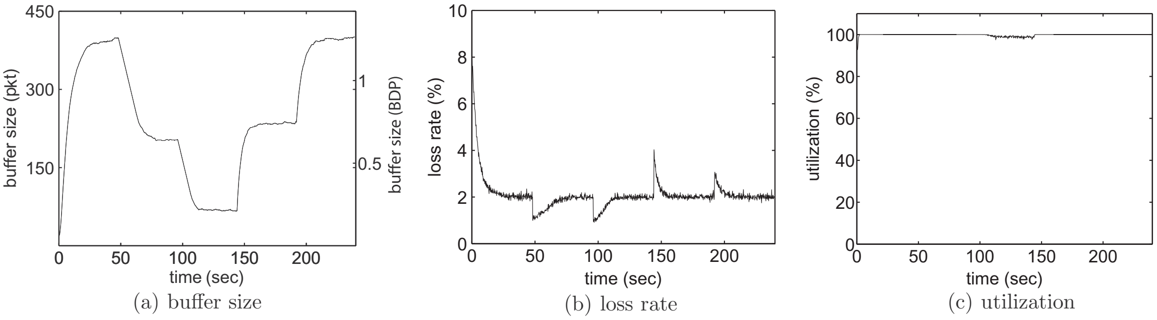
We next set N = 16 and examine scalability of these methods to link capacities. As seen from Table 3, the BDP and Stanford rules result in significant packet loss under small link capacities ($C \leqslant 4$ mb/s). Although they achieve both low loss rate and high utilization when C is large (e.g., $C \geqslant 256$ mb/s), the allocated buffer sizes are over provisioned compared to those of ABS. BSCL experiences less loss rate than the BDP and Stanford models, but it still does not lead to a buffer size that satisfies the target loss rate and utilization constraints. ABS again demonstrates the best performance among all these methods, maintaining buffer size within the target performance constraints for all link capacities.

4.3. Response to load changes

The volume of traffic perceived by any Internet router is not constant, but exhibits burstiness at different timescales due to various reasons such as users' demand, route changes, and load balancing. Thus, stability and responsiveness in the presence of load changes is crucial for any buffer sizing scheme purported to operate in practical routers. Hence, we next examine ABS in such a scenario. We still use a "dumbbell" topology where a single bottleneck link of capacity 10 mb/s is shared by 60 heterogeneous TCP flows. The target utilization u^* = 90% and loss rate p^* = 2%. As shown in Fig. 4, after all flows start simultaneously at the beginning, both $b(n)$ and $p(n)$ are quickly brought to a close neighborhood of their respective stationary value. At time 48 s, 20 flows depart from the system. As a consequence of the reduced traffic load, packet loss rate $p(n)$ immediately drops to 1.1%, which allows the router to release memory space to meet p^* in this new scenario. After another 48 s, 20 more flows left and again ABS quickly shrinks the buffer. At time 144 and 192 s, these two sets of departed flows respectively rejoin the system and ABS is forced to increase the buffer size. It can be observed from the plots that during the entire simulation, $b(n)$ demonstrates quick responses to load changes, experiences small oscillations in both the transient and steady states, and exhibits smooth transitions be-

Table 3
Performance of existing buffer sizing strategies in a single-link network with different link capacities C (N = 16, u^* = 98%, and p^* = 2%).

C (mb/s)	BDP			Stanford			BSCL			ABS		
	b (pkts)	p (%)	u (%)	b (pkts)	p (%)	u (%)	b (pkts)	p (%)	u (%)	b (pkts)	p (%)	u (%)
2	19	9.7	99.2	10	12.7	97.5	76	3.7	99.9	148	2.08	99.9
4	38	4.8	98.4	19	6.6	95.1	53	4.0	99.1	131	2.03	99.7
8	75	2.4	96.1	38	3.03	91.9	17	4.3	79.5	120	1.85	97.9
16	150	1.05	94.9	75	1.6	87.8	45	1.8	83.4	227	0.74	97.9
32	300	0.4	97.3	150	0.6	91.2	100	0.8	86.2	333	0.35	97.9
64	600	0.13	98.5	300	0.2	94.3	211	0.3	90.9	496	0.15	97.9
128	1200	0.05	99.2	600	0.08	96.4	432	0.09	93.0	780	0.06	97.8
256	2400	0.021	99.4	1200	0.03	98.1	875	0.04	95.2	1164	0.028	97.9
512	4799	0.003	99.6	2400	0.008	98.6	1760	0.01	97.7	1759	0.012	98.1
1024	9597	0.0003	99.6	4799	0.0007	98.2	3531	0.0009	96.6	3860	0.0008	97.9



(a) buffer size (b) loss rate (c) utilization

Fig. 4. ABS under changing traffic loads (u^* = 90% and p^* = 2%).

tween neighboring states. Note that during the entire simulation, loss rate is the dominant constraint and therefore buffer size is mainly driven by ABS$_p$. This can be seen from Fig. 4(c), where utilization is always above its target value 90%.

4.4. Web traffic

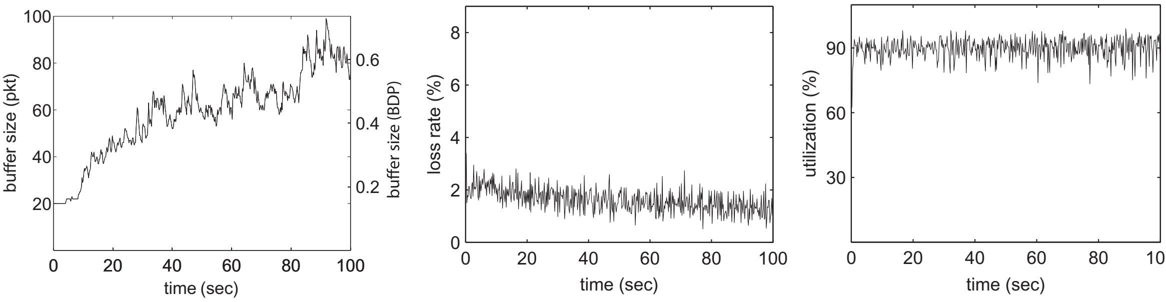
All scenarios considered so far have only long-lived TCP flows. However, the real Internet traffic is composed of a mixture of connections with a wide range of transfer lengths, packet sizes, and RTTs [14]. Thus, to obtain a better understanding of ABS, we next test it in more diverse scenarios.

Consider a network with a single link of capacity 10 mb/s shared by 20 persistent FTP flows in the presence of background web traffic generated by 100000 HTTP sessions. Each HTTP session downloads n_p pages with inter-page time t_p seconds, where n_p is uniformly distributed in [10, 2000] and t_p is exponentially distributed with mean 1 s. Each page contains n_o objects where n_o is uniformly distributed in [1,5]. The inter-object time t_o is exponentially distributed with mean 0.01 s. Sizes of objects follow the Pareto distribution with mean μ = 10 KB and shape parameter α = 1.2. We set the target utilization u^* = 90% and loss rate p^* = 2%.

The simulation results are shown in Fig. 5. As observed from Fig. 5(a), ABS's behavior in this scenario differs from that of previous simulations in that the buffer size does not converge to a particular value, but fluctuates between 40 and 100 packets due to bursty ingress traffic. Note that this phenomenon does not indicate that ABS is incapable of effectively controlling short-lived web-like traffic, but actually demonstrates that this mechanism is adaptive and responsive in highly dynamic scenarios. This can be clearly seen from Fig. 5, where utilization $u(n)$ is maintained within a close neighborhood of its target value u^* = 90% and loss rate $p(n)$ is kept below p^* = 2%.

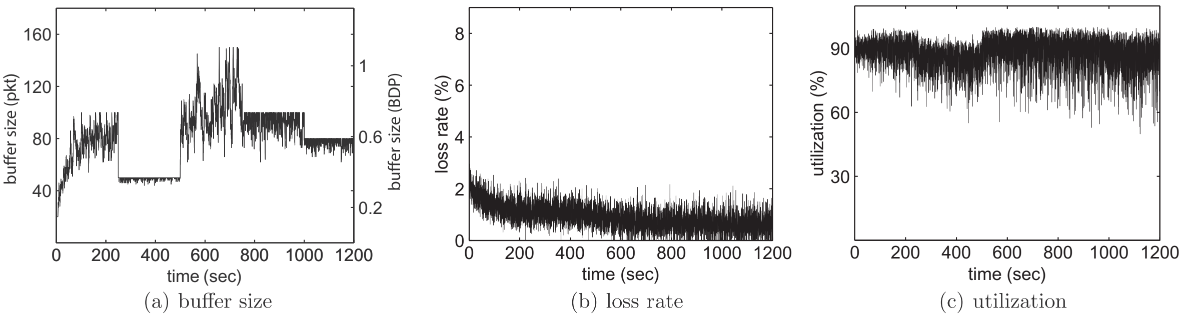
4.5. ABS under memory bound

As explained in Section 3.2, with under-provisioned physical memory, buffer assignment calculated by ABS is always bounded from above by the amount of available memory. To study behavior of ABS under limited memory, we conduct the following simulation. We use the same networking setting and traffic load as those in Section 4.4. Instead of providing unlimited memory resource, we impose a memory bound that changes over time to emulate fact that in practice the amount of available memory to a particular router port is time-varying. Specifically, we set the memory bound to 100, 50, 150, 80, and 60 packets for time intervals [0, 250), [250, 500), [500, 750), [750, 1000), and [1000, 1200], respectively. As seen from Fig. 6(a), ABS works as expected under physical memory cap and does not exhibit any stability issues. However, due to insufficient buffering, end-users experience higher packet loss and exhibit more aggressive back-off and retransmission behavior as a result of TCP's congestion avoidance mechanism. This explains more oscillatory utilization demonstrated in Fig. 6(c) compared to that in Fig. 5(c).



(a) buffer size (b) loss rate (c) utilization

Fig. 5. ABS in a single link of capacity 10 mb/s shared by 20 FTP and 100000 HTTP flows (u^* = 90% and p^* = 2%).



(a) buffer size (b) loss rate (c) utilization

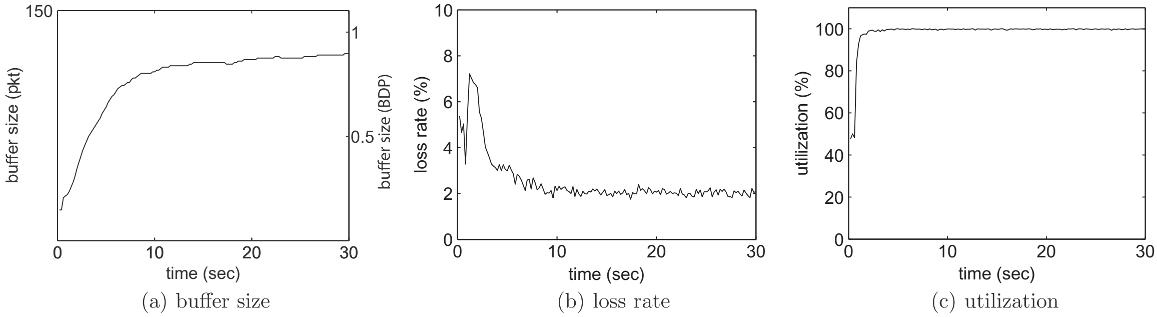
Fig. 6. ABS under limited available memory ($u^* = 90\%$ and $p^* = 2\%$).

4.6. Mixture of TCP and non-TCP traffic

Recall that analysis of real Internet traffic traces has demonstrated that although TCP is the predominant transport protocol, a non-eligible portion of Internet traffic is contributed by non-TCP protocols [15]. Thus, in this subsection we test ABS in a more diverse environment with 20% UDP background traffic and examine its effectiveness in the presence of unresponsive CBR traffic. Consider a scenario where 20 FTP, 20 HTTP, and 20 UDP flows compete for resources of a single link of capacity 10 mb/s. Traffic parameters of HTTP flows are the same as the last subsection and each UDP flow transmits packets at a constant rate 0.1 mb/s. We set reference values of the bottleneck router to be $u^* = 90\%$ and $p^* = 2\%$. The simulation result is shown in Fig. 7, where ABS is dominated by the loss constraint and $p(n)$ quickly reaches and subsequently remains in a close neighborhood of p^*. At the same time, utilization is always kept above its required minimum value.

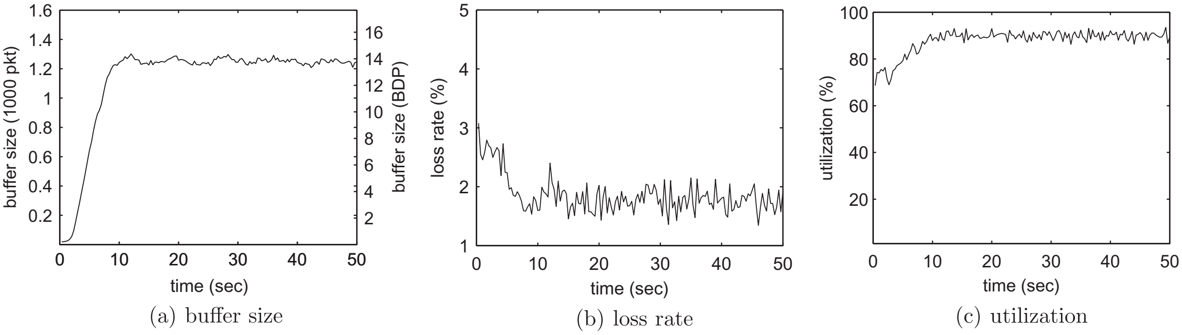
According to Section 2, the monotonic effects of buffer size $b(n)$ on loss rate $p(n)$ and utilization $u(n)$ should also hold for traffic generated by a set of different congestion control protocols. Thus, we next test ABS under a mixture of TCP variants. Specifically, we preserve values of p^* and u^*, increase the link capacity to 100 mb/s, and synthesize the ingress traffic with 10 Reno, 10 HSTCP, 10 STCP, 10 HTCP, and 10 Westwood flows with RTTs uniformly distributed within [40,60] ms. As illustrated in Fig. 8, ABS successfully maintains $u(n)$ around its target value u^* and keeps $p(n)$ below $p^* = 2\%$.

Thus, examples provided in this and the preceding subsections clearly demonstrate ABS's excellent capability of



(a) buffer size (b) loss rate (c) utilization

Fig. 7. ABS in a single link of capacity 10 mb/s shared by 20 FTP, 20 HTTP, and 20 UDP flows ($u^* = 90\%$ and $p^* = 2\%$).



(a) buffer size (b) loss rate (c) utilization

Fig. 8. ABS in a single link of capacity *100* mb/s shared by *10* Reno, *10* HSTCP, *10* STCP, *10* HTCP, and *10* Westwood flows ($u^* = 90\%$ and $p^* = 2\%$).

regulating the buffer size under different traffic patterns and transport protocols, making the concept of an ABS-like dynamic scheme a highly versatile and appealing buffer sizing mechanism for real Internet routers.

4.7. Multi-link topology

We next extend out study to multi-link networks and see whether interactions between multiple ABS routers can produce undesirable effects. Towards this end, consider a two-link "parking lot" topology with three sets of flows. Each set is composed of 20 FTP, 10 HTTP, and 10 UDP (with constant rate 0.1 mb/s) flows. These three sets of flows respectively pass through the first link, second link, and both links. Capacities of these two links are respectively 50 and 20 mb/s. Constraint values are $u_1^* = 95\%$ and $p_1^* = 1\%$ for the first link and $u_2^* = 75\%$ and $p_2^* = 5\%$ for the second link. As seen from Fig. 9, two ABS routers do not intervene each other and successfully maintain utilization at their respective target level.

Based on simulations conducted in this subsection, we have demonstrated that ABS achieves its design goal – regulating buffer size $b(n)$ to satisfy the pre-specified performance constraints. Furthermore, ABS is shown to be stable in the presence of dynamically changing loads and robust to a diverse mixture of long and short TCP flows and even non-TCP traffic. All of these properties make ABS a highly appealing buffer sizing scheme for real Internet routers.

5. Related work

It is commonly suggested that the buffer size b of a bottleneck router should be at least the product of the output link's capacity C and the average round-trip time R of all incoming TCP sessions, i.e., $b \geqslant CR$. This rule-of-thumb is commonly attributed to Villamizar and Song [36] and is deployed in most current large commercial routers [3]. However, the huge amount of memory space required by this rule becomes progressively unrealistic as link speed of the Internet evolves to the magnitude of multiple giga-bps and even tera-bps.

As pointed out by Appenseller et al. [3], this classic principle is applicable in scenarios where only synchronized long-lived TCP flows are present. However, Internet core routers are usually utilized by hundreds of thousands of heterogeneous flows, in which case synchronization rarely happens and the aggregate window size process converges to a Gaussian process [3]. Based on this result, they prove that when router buffers are sized according to $b = CR/\sqrt{N}$, link utilization is lower bounded by 98.99%. This result deviates from the rule-of-thumb in that their suggested buffer sizes scale inversely to the number of flows, indicating that all current backbone routers are over-buffered and their memory space and costs can be substantially reduced.

The small-buffer criteria are extended by Enachescu et al. [10], who suggest that buffers be as small as 10–20 packets in core routers provided the packet arrival process follows a Poisson distribution. This assumption is enforced by introducing *Paced TCP* [10], where senders evenly spread out-going packets over an RTT. This result is further extended to the model of combined input–output queue [6] and later supported in [30,39].

Although the assumption of totally asynchronous flows and Poisson arrivals are sound for backbone routers, as pointed out in [9], generic Internet routers are usually accessed by partially synchronized flows. In this case, the minimum buffer requirement is shown to be [9]:

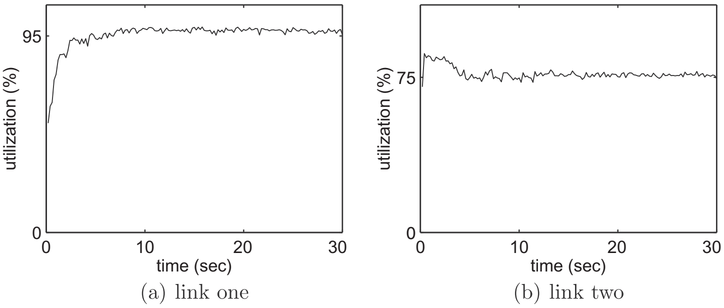
$$b = \frac{p(N)CR_e - 2SN(1 - p(N))}{2 - p(N)}, \tag{17}$$

where R_e is the harmonic mean of the RTTs, S is the MTU, $p(N) = 1 - (1 - 1/N)^{L_N}$ is the fraction of flows that see at least one packet loss, and L_N is the average number of dropped packets during a congestion event.

Besides saturating a given link, Dhamdhere et al. [9] propose that minimizing packet loss rate should also be taken into account when sizing router buffers. To accomplish this goal, they develop a buffer management rule based on *Flow Proportional Queuing* (FPQ), in which the loss rate is kept within a threshold value p by increasing the RTTs (or the buffer size) of the flows proportionally to N. Letting R_p and R_q^* respectively be the propagation and required queuing delays of the link, the proposed buffer sizing equation is

$$b = CR_q^* = K_p^* N - CR_p, \tag{18}$$

where $K_p^* = 0.87/\sqrt{p^*}$ and p^* is the target loss rate. If we consider both utilization and loss constraints, buffer size should be the larger of (17) and (18) and the resulting



(a) link one (b) link two

Fig. 9. ABS in a "parking lot" topology ($u_1^* = 95\%$ and $u_2^* = 75\%$).

mechanism is called *Buffer Sizing for Congested Link* (BSCL) [9].

Note that buffer sizing rule (18) suggests that the bottleneck buffer should linearly increase with N, which is in sharp contrast to the aforementioned small-buffer criteria.

Compared to the above schemes that seek to derive an explicit model of buffer size and Internet traffic, another class of methods tries to solve this problem by utilizing a certain implicit relationship between them. Specifically, Shorten et al. [32] propose a method called *Adaptive AIMD*, which is shown to adapt to any buffer size in the path. In addition, Shorten et al. [35] formulate the relationship between buffer size and utilization as a sector-bounded nonlinearity and develop an adaptive buffer sizing scheme called *Adaptive Drop Tail* (ADT), whose control equation is given by $b(n) = b(n-1) + K(u^* - u(n))$, where unspecified parameter K needs to satisfy $K \in (0, 2/k_2)$ to achieve stability and k_2 is the sector nonlinearity upper bound. However, it is unclear how k_2 is obtained for a given traffic condition and how it can be calculated in real-time as the system's dynamics change. Clearly, ADT has to resolve these issues before being used in real Internet routers.

6. Conclusion

In this paper, we designed and implemented a dynamic buffer sizing scheme, called ABS, that stabilizes the buffer size to its minimum value satisfying given utilization and/or loss constraints. ABS is composed of two Integral controllers ABS_u and ABS_p, each of which is equipped with a parameter training component using a gradient-based technique to achieve the optimal control gains. Besides stability and optimality, an appealing feature of ABS is its robustness to generic Internet traffic composed of long, short, and non-TCP flows. Thus, ABS can significantly benefit router manufactures and ISPs by improving their routers' performance, reducing system cost, and providing QoS guarantees.

We finally note that the emphasis of the paper is not demonstrating superiority of a particular controller, but advocating a new buffer management methodology and presenting the possibility of optimally sizing router buffers using a simple yet robust controller without comprehensive knowledge of Internet dynamics. This controller actually may be replaced by more advanced candidates (e.g., nonlinear PID and variable structure control). Our future work involves designing simpler ABS-like mechanisms, analyzing ABS in its transient phase, studying its stability in more complicated congestion control models, and implementing and testing it in hardware routers under real Internet traffic.

References

[1] M. Allman, V. Paxson, W. Stevens, TCP congestion control, IETF RFC 2581, April 1999.

[2] H.C. Andersen, A. Lotfi, A.C. Tsoi, A new approach to adaptive fuzzy control: the controller output error method, IEEE Trans. Syst. Man Cybern. 27 (4) (1997) 686–691.

[3] G. Appenseller, I. Keslassy, N. McKeown, Sizing router buffers, in: Proc. ACM SIGCOMM, August 2004, pp. 281–292.

[4] S. Athuraliya, S.H. Low, V.H. Li, Q. Yin, REM: active queue management, IEEE Networks 15 (3) (2001) 48–53.

[5] K. Avrachenkov, U. Ayesta, A. Piunovskiy, Optimal choice of the buffer size in the internet routers, in: Proc. IEEE CDC, December 2005, pp. 1143–1148.

[6] N. Beheshti, Y. Ganjali, R. Rajaduray, D. Blumenthal, N. McKeown, Buffer sizing in all-optical packet switches, in: Proc. OFC/NFOEC, March 2006.

[7] L. Brakmo, S. O'Malley, L. Peterson, TCP vegas: new techniques for congestion detection and avoidance, in: Proc. ACM SIGCOMM, August 1994, pp. 24–35.

[8] Cisco 3600 Series Routers. Available at: <http://www.cisco.com/en/US/products/hw/routers/ps274/>.

[9] A. Dhamdhere, H. Jiang, C. Dovrolis, Buffer sizing for congested internet links, in: Proc. IEEE INFOCOM, March 2005, pp. 1072–1083.

[10] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, T. Roughgarden, Routers with very small buffers, in: Proc. IEEE INFOCOM, April 2006.

[11] S. Floyd, High-speed TCP for large congestion windows, IETF RFC 3649, December 2003.

[12] S. Floyd, T. Henderson, The newReno modification to TCP's fast recovery algorithm, IETF RFC 2582, April 1999.

[13] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Networking 1 (4) (1993) 397–413.

[14] S. Floyd, E. Kohler, Internet research needs better models, ACM SIGCOMM Comput. Commun. Rev. 33 (1) (2003) 29–34.

[15] M. Fomenkov, K. Keys, D. Moore, K. Claffy, Longitudinal study of internet traffic from 1998–2003, in: Proc. WISICT, January 2004, pp. 1–6.

[16] Y. Ganjali, N. McKeown, Update on buffer sizing in internet routers, ACM SIGCOMM Comput. Commun. Rev. 36 (5) (2006) 67–70.

[17] M. Gerla, M.Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, TCP westwood: congestion window control using bandwidth estimation, in: Proc. IEEE GLOBECOM, November 2001, pp. 1698–1702.

[18] S. Gorinsky, A. Kantawala, J. Turner, Link buffer sizing: a new look at the old problem, in: Proc. IEEE ISCC, Junuary 2005, pp. 507–514.

[19] S. Gorinsky, A. Kantawala, J. Turner, Simulation perspectives on link buffer sizing, Simulation 83 (3) (2007) 245–257.

[20] C.V. Hollot, V. Misra, D. Towsley, W.-B. Gong, On designing improved controllers for AQM routers supporting TCP flows, in: Proc. IEEE INFOCOM, April 2001, pp. 1726–1734.

[21] F. Ishizaki, T. Takine, Loss probability in a finite discrete-time queue in terms of the steady state distribution of an infinite queue, Queueing Syst.: Theory Appl. 31 (3-4) (1999) 317–326.

[22] C. Jin, D. Wei, S.H. Low, FAST TCP: motivation, architecture, algorithms, performance, in: Proc. IEEE INFOCOM, March 2004, pp. 2490–2501.

[23] C. Kellett, R. Shorten, D.J. Leith, Sizing internet router buffers, active queue management, and the Lur'e problem, in: Proc. IEEE CDC, December 2006, pp. 650–654.

[24] T. Kelly, Scalable TCP: improving performance in high-speed wide area networks, Comput. Commun. Rev. 33 (2) (2003) 83–91.

[25] A. Kuzmanovic, E. Knightly, TCP-LP: a distributed algorithm for low priority data transfer, in: Proc. IEEE INFOCOM, April 2003, pp. 1691–1701.

[26] D. Leith, R. Shorten, H-TCP protocol for high-speed long distance networks, in: Proc. PFLDnet, February 2004.

[27] S.H. Low, F. Paganini, J. Wang, S. Adlakha, J. Doyle, Dynamics of TCP/RED and a scalable control, in: Proc. IEEE INFOCOM, June 2002, pp. 239–248.

[28] M. Podlesny, S. Gorinsky, RD network services: differentiation through performance incentives, in: Proc. ACM SIGCOMM, August 2008, pp. 255–266.

[29] R. Prasad, C. Dovrolis, M. Thottan, Router buffer sizing revisited: the role of the output/input capacity ratio, in: Proc. ACM SIGCOMM CoNEXT, December 2007.

[30] G. Raina, D. Towsley, D. Wischik, Part II: control theory for buffer sizing, ACM SIGCOMM Comput. Commun. Rev. 35 (3) (2005) 79–82.

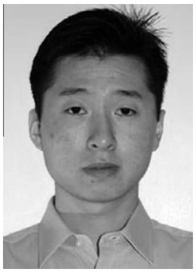
[31] I. Rhee, L. Xu, CUBIC: a new TCP-friendly high-speed TCP variant, in: Proc. PFLDnet, February 2005.

[32] R.N. Shorten, D.J. Leith, On Queue Provisioning, Network Efficiency and the Transmission Control Protocol, Technical Report, Hamilton Institute, 2006.

[33] J.A. Snyman, Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms, Springer, 2005.

[34] J. Sommers, P. Barford, A. Greenberg, W. Willinger, An SLA perspective on the router buffer sizing problem, ACM SIGMETRICS Perform. Eval. Rev. 35 (4) (2008).

[35] R. Stanojevic, R.N. Shorten, C.M. Kellett, Adaptive tuning of drop-tail buffers for reducing queueing delays, IEEE Commun. Lett. 10 (7) (2006) 570–572.
[36] C. Villamizar, C. Song, High performance TCP in the ANSNET, ACM SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 45–60.
[37] D.X. Wei, P. Cao, NS-2 TCP-Linux: an NS-2 TCP implementation with congestion control algorithms from Linux, in: Proc. WNS2, October 2006.
[38] D. Wischik, Buffer requirements for high-speed routers, in: Proc. ECOC, September 2005, pp. 23–26.
[39] D. Wischik, N. McKeown, Part I: buffer sizes for core routers, ACM SIGCOMM Comput. Commun. Rev. 35 (2) (2005) 75–78.
[40] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control (BIC) for fast, long distance networks, in: Proc. IEEE INFOCOM, March 2004, pp. 2514–2524.



Yueping Zhang received a B.S. degree in computer science from Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001 and a Ph.D. degree in computer engineering at Texas A&M University, College Station, USA, in 2008. He is currently a Research Staff Member at NEC Laboratories America, Inc. His research interests include congestion control, delayed stability analysis, Active Queue Management (AQM), router buffer sizing, peer-to-peer networks, Internet measurement, datacenter networking, and IP service management.

Dmitri Loguinov received the B.S. degree (with honors) in computer science from Moscow State University, Russia, in 1995 and the Ph.D. degree in computer science from the City University of New York, New York, in 2002.

Between 2002 and 2007, he was an Assistant Professor in the Department of Computer Science at Texas A&M University, College Station. He is currently a tenured Associate Professor and Director of the Internet Research Lab (IRL) in the same department. His research interests include peer-to-peer networks, video streaming, congestion control, Internet measurement and modeling.