Adaptive Tuning of Drop-Tail Buffers for Reducing Queueing Delays

Rade Stanojević, Robert N. Shorten
The Hamilton Institute, National University of Ireland, Maynooth

***Abstract*— Internet router buffers are used to accommodate packets that arrive in bursts and to maintain high utilization of the egress link. Such buffers can lead to large queueing delays. We propose a simple algorithm, Active Drop-Tail (ADT), which regulates the queue size, based on prevailing traffic conditions, to a minimum size that still allows for a desired (high) level of utilization. Packet level *ns-2* simulations are provided to show that Adaptive Drop-Tail achieves significantly smaller queues than current approaches at the expense of 1-2% of the link utilization.**

***Index Terms*— Router buffers, Drop-Tail queues, queueing delays, TCP.**

I. Introduction

TRADITIONALLY, router buffers have been provisioned according to the delay-bandwidth product (DBP) rule: namely, one chooses the buffer size as $q = B \times T$, where B is the rate of the link served by the router, and T is the "typical" [1] round trip time (RTT) experienced by connections utilizing the buffer. This amount of buffering allows for 100% utilization of the egress link under all traffic conditions. Following this rule, most router buffers are designed to have 100-250ms of buffering. This, together with the TCP mechanism of congestion avoidance, ensures high link utilization. In the last few years, several studies related to buffer sizing at a congested router have occurred [1], [2], [3], [4]. For example, it is claimed in [2] that the amount of buffer space needed for high utilization of a link is highly dependent on the number of active flows on the link. Namely, they claim that, if N is the number of active TCP flows, the buffer space required for 99% utilization of the link capacity is $B_{AKM} = \frac{B \times T}{\sqrt{N}}$.

Having small buffers is attractive as it reduces the amount of memory, required physical space, energy consumption, and price of the router. From our point of view, however, the main advantage of having small buffers is the reduction in queueing delays and jitter. In the current Internet the average number of hops on a random path is about 13 [5]. For a single flow with that many hops it is possible to expect several congested links on the path. Thus buffering of several hundreds ms at each router would imply very large queueing delays.

The $\frac{1}{\sqrt{N}}$ bound in [2] depends on the number of active users accessing a bottleneck link and the distribution of RTTs. Since, for any congested link, these quantities vary and are difficult to estimate, a network operator must provide as much buffer space as is necessary for the least possible number of active users in order to keep utilization high at all times. Assuming that a router uses a drop-tail queue of sufficient size to ensure high utilization in the low number of users regime, queueing delays will be much larger than necessary in regimes with a large number of users. Motivated by this observation we develop a simple algorithm, called Active Drop-Tail (ADT), which keeps the available buffer space as small as possible while maintaining a certain level of utilization. Thus, for a large number of users the available queue size will be low, and for a low number of users will be as large as necessary to achieve high utilization.

II. Active Drop-Tail algorithm

We propose a discrete-time queue management algorithm, called Active Drop-Tail (ADT), that tries to "find" the smallest queue size that allows the outgoing link to operate at a given desired utilization u. ADT has two stages: (i) estimating the (average) throughput of the link and (ii) adjusting the available buffer space once per sampling period based on this measurement. In order to estimate the throughput, at each sampling time we first compute the current throughput as the number of bytes enqueued normalized by the length of the sampling interval. We then compute a weighted average of the throughput. We will control the available buffer space based on this weighted average.

ADT maintains an internal variable q_{ADT} that corresponds to the number of packets that can be accommodated in the buffer. The basic idea is to modify[2] q_{ADT} based on the estimated average throughput. If the average throughput is less than the desired utilization, q_{ADT} is increased to allow more buffering, yielding higher utilization. On the other hand, if the average throughput is greater than the desired utilization, we decrease q_{ADT} to regulate utilization to the desired level. For the purposes of adjusting q_{ADT} we use a Multiplicative Increase - Multiplicative Decrease (MIMD) strategy. While other control strategies are possible, our simulations show that a simple MIMD approach works well to reduce the queue size needed to maintain a certain level of utilization. Assuming that network conditions do not vary quickly[3], the MIMD parameter should be chosen to allow q_{ADT} to be halved/doubled in a few seconds up to

[1] In [1], it is suggested that in order to ensure full utilization of the link one should use $B \times T_h$ of buffering, where T_h is the harmonic mean of round trip times for all bottlenecked connections passing through the link. In this paper we use the term "typical" RTT for harmonic mean T_h

[2] The idea of adapting available buffer space has been exploited in [6], but in a very different context.

[3] Measurements [7] show that, on typical 150Mbps+ links, basic IP parameters (such as the number of active connections, proportion of TCP traffic, aggregate IP traffic, etc.) do not change dramatically in short time periods.

one minute. ADT deals with arriving packets in the same fashion as drop-tail with q_{ADT} as the queue limit, i.e. if, at the moment of packet arrival, the queue has size less than $q_{ADT} - 1$, then packets the packet is enqueued. Otherwise it is dropped[4]. Pseudo-code describing the ADT algorithm is given in the table below. The parameters of ADT are as follows. ρ is an averageing parameter in $(0, 1)$ and is used to calculate the weighted average of the throughput. $c > 1$ is the MIMD parameter for controlling q_{ADT}. $SamplePeriod$ and u are the sampling period and desired utilization respectively.

ON EVERY PACKET ARRIVAL:
IF $(now - LastUpdate) > SamplePeriod$
. CURTHR := $\frac{NmbBEnq(now) - NmbBEnq(LastUpdate)}{now - LastUpdate}$
. THR := ρ CURTHR + (1-ρ)THR
. IF $(\frac{\text{THR}}{ServiceRate} < u)$
. $q_{ADT} := q_{ADT} \cdot c$
. OTHERWISE
. $q_{ADT} := \frac{q_{ADT}}{c}$
. END
. $q_{ADT} := min(q_{ADT}, SizeOfBuffer)$
. $LastUpdate := now$
END

In the above table $NmbBEnq(t)$ denotes the number of enqueued bytes in the interval $[0, t]$, and now is the current sample time. The parameter ρ is used to filter possible transient effects, such as a flash crowd or a sudden decrease in arrival traffic, and should be chosen such that weighted averaging is performed over several congestion epochs[5]. While in most cases congestion epochs last less than a few seconds, some extreme situations (e.g., high bandwidth-low number of users) might require a lower choice of ρ. Such situations can be easily identified and dealt with by adapting ρ online: due to space restrictions we do not describe the adaptation of ρ here other than to note that it can be done in a straightforward manner.

Comment 1: In the case of a noncongested link, as long as the average arrival rate is less than the desired utilization u, the ADT algorithm will keep q_{ADT} at $SizeOfBuffer$; i.e. at the actual buffer size. However, in noncongested links queueing delays are negligible, and there is no need for ADT.

Comment 2: ADT is designed for loss based TCP. Delay-based TCP proposals, like Vegas or FAST, require a certain level of queueing delay as a congestion signal. Adaptation of available buffer space in such circumstances is not beneficial.

III. EVALUATION

In this section we present *ns2* packet level simulation results on the performance of ADT[6]. Throughout this section we use the following set of ADT parameters: $\rho = 0.1$, $SamplePeriod = 0.3sec$, $c = 1.01$, and desired utilization $u = 0.99$.

[4]We assume that the buffer is configured in packets. However one can easily change the algorithm to track q_{ADT} and queue size in bytes instead packets.

[5]A congestion epoch is the time between two consecutive packet losses.

[6]ns2 software used in these simulations can be found at http://www.hamilton.ie/person/rade/ADT/.

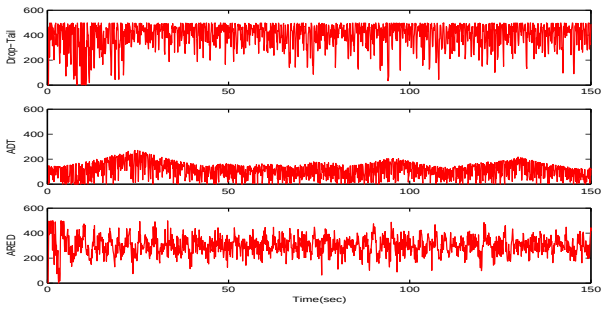

Fig. 1. Queue sizes for Drop-Tail, ADT-0.99, and Adaptive RED

A. ADT, Drop-Tail, and Adaptive RED : Our first *ns2* simulations are for 1000 TCP users with RTTs uniformly distributed in the interval 40 to 440ms, and a packet size 1500B, all competing through a single bottleneck link with capacity $200Mb/sec$. Figure 1 shows the queue occupancy for a Drop-Tail (DT) queue with a buffer size of 500 packets ($30msec$ of buffering), an ADT queue, and an Adaptive RED [8] queue. The following table contains the average utilization (AU), loss rate (LR), average queueing delay (aQd), Jain's fairness index (JFI)[9] and average goodput (AG) for each of these three disciplines. In order to show how ADT adapts the available buffer space in the case of a congested reverse path we performed a simulation with 1000 TCP connections in both directions competing over ADT queues.

.	AU(%)	LR(%)	aQd(ms)	JFI	AG(%)
DT	99.99	4.73	23.78	0.58	98.75
ADT	99.03	4.87	6.66	0.56	97.72
ARED	100.00	4.90	17.99	0.79	98.92
ADT(R)	99.04	4.73	17.14	0.70	94.54

As we can see from the previous table, the utilization of Drop-Tail and Adaptive RED is 100% while the utilization of ADT is 99.03%. The loss rates of all three queueing disciplines are similar and are mainly driven by the congestion control algorithms of the users. A stated secondary goal of most RED-like schemes is to keep queueing delays small. However, having designed ADT specifically to minimize queueing delay, we see that the average queueing delay of ADT, in this example, is three times less than that of Adaptive RED. We note that even with reverse path traffic and heavy congestion in both directions (i.e., ACK accumulation and larger bursts) ADT is able to adapt the available queue size in order to achieve the desired utilization (although because of ACK losses goodput is significantly reduced).

B. Queue size, number of users, and loss rate : Here we investigate the relationship between queue size, number of users, and loss rate. In order to compare our results with the $\frac{1}{\sqrt{N}}$ bound from [2] we consider N TCP flows with RTTs uniformly distributed in $[80, 100]ms$[7] with a packet size of $1500B$. All flows compete via a bottleneck link of

[7]In deriving their $\frac{1}{\sqrt{N}}$ bound the authors of [2] assume uniform RTTs. For this reason we chose the RTTs in the range $[80, 100]ms$, to emulate the condition of "almost the same" RTT.

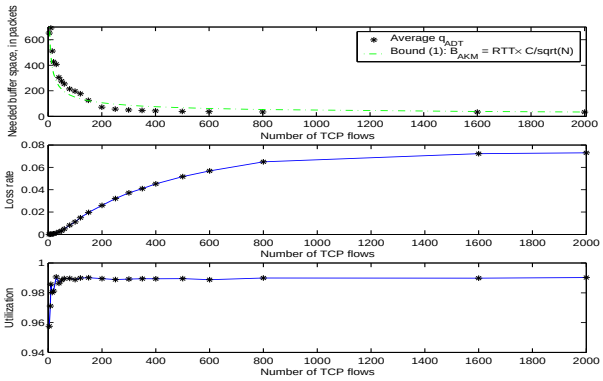

Fig. 2. Average q_{ADT} (top), loss rates (middle), and utilization (bottom).

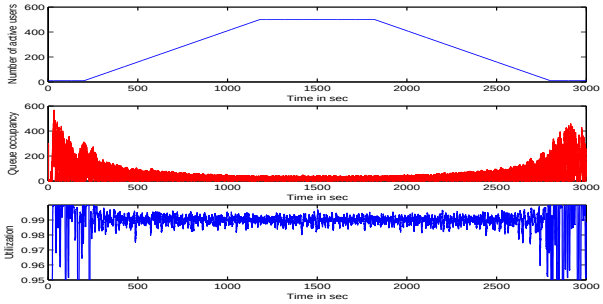

Fig. 3. Utilization and q_{ADT} with changing number of active TCP users.

$200Mb/sec$ with an ADT queue. We vary N from 4 to 2000 and evaluate loss rate and average q_{ADT}. In Figure 2 (top) we compare the average queue size for a fixed number of users with the estimate given in [2]. We observed close fit to the theoretical bound from [2]. As we can see from Figure 2 (middle) the loss rate increases slowly with N. This is a consequence of the time-out mechanism of TCP.

C. ADT and a varying number of users : We now allow the number of users to vary with time[8]. We consider a bottleneck link with capacity $100Mb/s$ where the number of active TCP users varies from 10 to 500 and back to 10 again, with one user becoming active or inactive every two seconds as depicted in Figure 3 (top). The RTT's are randomly chosen uniformly from the interval 10 to $300ms$. As we can see in Figure 3 (middle and bottom), q_{ADT} falls below 40 packets for large numbers of users, while for a small number of active users the required queue size is over 400 packets.

D. Selection of parameters : ADT is highly robust to the choice of its parameters. To illustrate this we ran a set of N TCP connections, over a $100Mb/s$ link and varied the ADT parameters as follows: $SampleTime$ in range $[100, 1000]msec$; $c \in \{1.002, 1.01, 1.05\}$. We fixed the weighted average factor $\rho = 0.1$. We evaluated two cases, low number of users: $N = 20$, and high number of users: $N = 200$. Recall, that the performance goal of ADT is to regulate utilization at a prescribed level; in our case $u = 0.99$.

[8]As we deal with slowly varying environments, in our simulation we vary the number of active users in a continuous manner rather than abruptly.

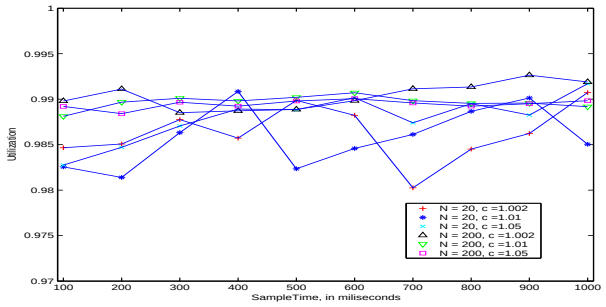

Fig. 4. Average utilization for different choices of parameters.

The ability of ADT to achieve this objective for different choices of parameters is depicted in Figure 4.

IV. Conclusions

In this letter we have presented a simple algorithm, ADT, for keeping queueing delays small, while maintaining a certain desired utilization. Via packet level simulations, we have shown that, for networks serving a large number of TCP flows, by allowing a 1-2% underutilization of a bottleneck link, we can realise smaller average queueing delays than in other queueing disciplines. We point out that this algorithm is easy to implement in current routers, requiring a minimum amount of processing power. Finally, we note that ADT strives to adjust the available buffer space to accommodate bursts through the network buffers irrespective of their generating mechanisms.

References

[1] A. Dhamdhere, H. Jiang, C. Dovrolis. "Buffer sizing for congested internet links". Proceedings of IEEE INFOCOM, Miami, FL, March 2005.
[2] G. Appenzeller, I. Keslassy, N. McKeown. "Sizing router buffers". Proceedings of ACM SIGCOMM '04, Portland, Oregon, USA, 2004.
[3] D. Wischik and N. McKeown. "Part I: Buffer sizes for core routers". ACM SIGCOMM Computer Communications Review, 2005.
[4] 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.
[5] Traffic measurements. [Online]. Available: http://www.caida.org/tools/measurement/skitter/RSSAC/
[6] J. Sun, K.T. Ko, G. Chen, S. Chan, M. Zukerman. "Adaptive Drop-Tail: A simple and efficient active queue management algorithm". Proceedings of International Teletraffic Congress, Berlin, Germany, 2003.
[7] Online: http://pma.nlanr.net/Special/.
[8] S. Floyd, R. Gummadi, S. Shenker. "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management". August 2001, online: http://www.icir.org/floyd/papers/adaptiveRed.pdf.
[9] R. Jain. "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling". John Wiley and Sons, INC., 1991.