Obtaining High Throughput in Networks with Tiny Buffers

Neda Beheshti*, Yashar Ganjali†, Ashish Goel‡, Nick McKeown*
*Department of Electrical Engineering, Stanford University
{nbehesht, nickm}@stanford.edu
†Department of Computer Science, University of Toronto
yganjali@cs.toronto.edu
‡Department of Management Science and Engineering, Stanford University
ashishg@stanford.edu

Abstract— **In this paper we explore whether a general topology network built up of routers with very small buffers, can maintain high throughput under TCP's congestion control mechanism. Recent results on buffer sizing challenged the widely used assumption that routers should buffer millions of packets. These new results suggest that when smooth TCP traffic goes through a *single* tiny buffer of size $O(\log W)$, then close-to-peak throughput can be achieved; W is the maximum window size of TCP flows.**

In this work, we want to know if a *network of many routers* can perform well when all buffers in the network are made very small, independent of the structure of the network. This scenario represents a real network where packets go through several buffering stages on their routes. Assuming the ingress TCP traffic to a network is paced, we first prove that all routers can get by with very small buffers, if the network has a tree structure. For networks with general topology, we propose a simple active queue management policy called *Bounded Jitter Policy* (BJP), and show that under the proposed policy each flow will preserve its smooth pattern across the network. Logarithmic size buffers would therefore be enough in every router of the network.

I. Introduction

Until recently, Internet routers were widely believed to need large buffers. A typical core router today, can buffer millions of packets, and the buffer size grows linearly as the capacity of the network links increases over time. Recent results on buffer sizing challenged this widely used assumption, and suggested that a smooth and well-multiplexed TCP traffic makes the routers achieve high throughput even when the buffers are made very small [1], [2], [4]. It is shown that implementing buffers of size $O(\log W)$ packets suffices to have close-to-peak throughput when the packets in each TCP flow are sufficiently spaced out [1]. Here, W is the maximum congestion window size of the TCP flows going through the buffer.

The *tiny buffer* results [1] assumes that traffic is paced, either by implementing paced TCP at source (source-paced traffic), or as a result of having very slow access links (link-paced traffic). If this paced traffic goes through one buffering point, then tiny buffers are enough to achieve high throughput. In a real packet-switched network, however, packets go through several buffering points on their routes from source to destination. Even a single router in today's Internet, usually has more than one buffering stage. A combined input output queued (CIOQ) router, for example, has buffers both on ingress and egress ports. Going through many buffers – even if they are not congestion points – can change traffic pattern along the links.

In this work, we explore whether we can obtain a core network of very small buffers and yet keep the utilization high on every link. To answer this question, we consider a network of routers where packets go through at most n routers on their paths. We show that if TCP traffic is paced as it enters the network, then the size of every buffer in the network can be reduced to $O(\log W + \log n)$ without losing more than a small fraction of throughput on any link in the network. The condition of having paced traffic at ingress ports is automatically satisfied when traffic entering the core is aggregated from slow access networks.

To study a network of routers, we first consider a tree-structured network, and show that such structure does not add to the burstiness of traffic as packets move forward in the network. More precisely, we show that any arbitrary buffer of such network would have lost no less throughput, had the arrival traffic been directly injected into that buffer (without going through the prior stages of buffering). A router in this network would therefore require buffers as big as what a single isolated router with the same input traffic requires: when the ingress TCP traffic is non-bursty, shrinking buffers of all routers to a logarithmic size maintains high throughput on every link in the network.

Next, we consider a network with general topology. In a network with arbitrary traffic matrix and topology, the characteristics of individual flows might change as they get aggregated with other flows on a link, or when they share queues with some cross-traffic. As a result, a smooth traffic at the source, or at ingress ports of the network, may not preserve its advantageous non-bursty attribute as it traverses across the network. Examples in [10] show network settings where delay jitter increases as traffic goes through more hops toward its destination, despite having only smooth (constant-bit-rate or Poisson) traffic generated at sources. Increased burstiness necessitates bigger buffers to absorb the bursts.

To eliminate the possibility of having increased burstiness caused by cross traffic, and consequently the need to have bigger buffers, we propose a simple active queue management policy, which we refer to as *Bounded Jitter Policy* (BJP). We show that by implementing BJP at each router of a general topology network, all packets will experience an almost fixed delay at each router. As a result, the inter-arrival time of packets in a flow remains unchanged as the flow traverses across the network. The packet arrival sequence of an arbitrary flow at any intermediate router will therefore be a delayed copy of its arrival sequence at the ingress port of the network: if the traffic is spaced out at the ingress port, it will remain spaced out and smooth at any node in the network.

Under BJP, a fixed delay can be achieved at the expense of some small additional drops compared with a buffer of the same size under FIFO policy. In return for dropping slightly more packets, our proposed policy guarantees that each flow will preserve its characteristic all along its route in the network. Consequently, if the ingress traffic of the network is sufficiently paced, then tiny buffers of logarithmic size will be sufficient to achieve high throughput between any arbitrary pair of ingress-egress ports.

II. Background: buffer sizing in a single router

Internet routers need buffers to hold packets during times of congestion, and to resolve contention. The size of this buffer directly impacts the packet drop rate, as well as the delay of packets going through the router. Historically, the buffer sizes in Internet core routes have been set based on a rule-of-thumb which says in order to achieve 100% link utilization, the buffer size B should be equal to the bandwidth-delay product $C \times RTT$ [6]. Here, RTT is the effective two-way propagation delay of packets going through the router, and C is the capacity of the bottleneck link. For a 10Gb/s link for example, and an average two-way propagation delay of 100ms the required buffer size would be roughly $1,000,000$ packets. According to this rule, the required buffer size grows linearly with the link capacity over time.

The bandwidth-delay-product rule assumes there is a single long-lived TCP flow going through the bottleneck link. The congestion control mechanism of TCP adjusts the transmission rate in response to the drop indications from the network: the sender controls the outgoing traffic by limiting the window size, *i.e.*, the number of outstanding (unacknowledged) packets at any time. In additive-increase and multiplicative-decrease (AIMD) congestion avoidance mode, the window size increases additively upon receiving an ACK packet, and is halved when the source detects packet loss. Using this mechanism, TCP controls the average transmission rate of data over a round-trip-time interval. Nevertheless, the source has no control over the data rate of smaller time intervals: the number of allowed outstanding packets might all be sent out in a burst, at the beginning of the round-trip time. Moreover, when packet loss is detected in the AIMD congestion avoidance mode, the transmitter reduces the window size, and stops sending out any packet until the number of unacknowledged packets fits into the reduced window size. These idle time-intervals add up to the burstiness of the traffic, and hence, increase variations in the buffer occupancy.

The rule-of-thumb for buffer sizing aims at keeping the bottleneck link busy at all times, so as to maximize the network throughput. The required buffer size is determined by the shape of the flow's TCP window-size dynamics. The window-size follows the well-known sawtooth shape, and has a distance from peak to trough of $C \times RTT$. Therefore, this much of buffering is required to ride out reductions in window-size and to make sure the bottleneck buffer never goes empty and the link is never idle.

Recently, Enachescu *et al.* [1], and Raina *et al.* [2] suggested that buffer sizes can be reduced drastically to $B = O(\log W)$. [1] This is equivalent to only $20 - 50$ packets for today's typical networks. There are two underlying assumptions for the tiny buffers result. First, the maximum throughput which we will achieve is 85-90% rather than 100%, *i.e.*, we must be willing to sacrifice 10-15% of link utilizations. Second, network traffic must be smooth. This smoothness can happen as a result of implementing Paced TCP [5] at all sources, which spreads packets over time rather than injecting bursts of traffic. Smoothness also can result from having slow access links in the network. When the capacity of access links is much slower than the core links, packets are automatically spaced out when they arrive at the core of the Internet.

III. How big buffers need to be in a network of routers

The dynamics of packet backlog in a buffer, and consequently the required buffer size for achieving a desired throughput, are imposed by the characteristics of the arrival traffic. A bursty traffic needs big buffers for absorbing the bursts, while a smooth traffic, a Poisson traffic for example, with the same average arrival rate needs much smaller buffers.

In this work we consider a core network, and assume that the ingress traffic of the network is non-bursty TCP traffic. This model can represent a backbone network in today's Internet: if the core links are sufficiently faster than the links in access networks, then as packets enter the core network, they automatically get paced (without a need to change TCP sources). But does the traffic stay non-bursty on every link of the core network?

In a network with an arbitrary traffic matrix and topology, the arrival pattern of packets in individual flows might change as they get aggregated with other flows on a link, and as they share queues with some cross-traffic. As a result, a smooth traffic at the source, or at ingress ports of the network, may become bursty as it traverses across the network. While there has been no thorough analytical work on how an arbitrary network of FIFO buffers will change the pattern of input traffic, evidences show that the network can add to the burstiness of the traffic. Grossglauser et al. have studied cases where delay jitter increases as traffic goes through more hops toward its destination [10]. In their setting, the generated traffic is smooth; flows generated at network sources follow either the constant-bit-rate model or the Poisson model, but the traffic doesn't preserve its smooth pattern across the network.

In the following subsections we will explore whether under the assumption of having smooth traffic at ingress ports, the network can maintain high throughput when all buffers are made very small. To answer this question, we first study networks with tree structure. In Section III-A we show that any buffer in this type of network drops no more packets than an isolated buffer of the same size and with the same incoming traffic. Networks with arbitrary topology are studied in Section III-B. We show that the logarithmic buffer size rule holds in all buffers of such network, if a simple packet scheduling mechanism is implemented in network's routers. As will be shown, implementing this policy makes the arrival sequence of any flow at all intermediate nodes look almost like a time-delayed copy of the flow's arrival sequence at the ingress node. If the ingress traffic is assumed to be Poisson for example, then by scheduling packets according to the proposed policy the arrival traffic to each node will be a sub-sequence of a Poisson process.

In both tree-structured and general topology networks, we find the sufficient size for buffers to obtain a small drop rate across the network when the ingress traffic to the network is Poisson. Section III-C explains how these results can be applied to size buffers in networks with TCP traffic.

For a given network and traffic rate matrix, we define the offered load on each link to be the aggregate injection rate of flows sharing that link. The *load factor* ρ of the network is defined to be the maximum, over all links in the network, of the offered loads. In our analysis, we assume that the network is over-provisioned, i.e., $\rho < 1$, and that the packets are all of the same size. Throughout the paper, we assume that flows go through only one buffering stage inside each router. Hence, we set the number of buffering points equal to the number of routers, and refer to them interchangeably. This can simply be generalized by setting n, used in our equations, to be the number of buffers rather than the number of routers.

A. Tree-Structured Networks

Fig. 1 shows a network of routers with a tree structure. We show that any arbitrary router of such network would have dropped more packets, had the arrival traffic been directly injected into the router (without going through the prior nodes of the tree).

Consider the two networks shown in Fig. 1, one with buffers in a tree structure, and the other with only one single buffer. Both networks have the same arrival sequence, i.e., a packet enters the first network at time t, if and only if at time t there is an arrival

[1] Enachescu *et al.* have also shown that a constant buffer size cannot achieve high throughput as the capacity of the network grows.

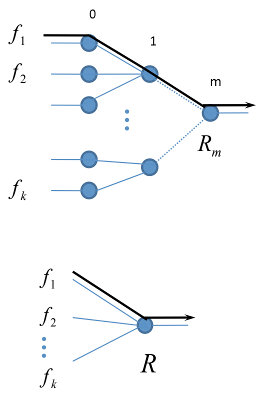

Fig. 1. Tree-structured network.

in the second network. In the tree-structured network, packets enter the leaves first. After being processed at each buffering node, the packets are transferred to the next buffering point until they depart the system. Assume that all physical links in both networks have capacity of $c = 1$ packet per unit of time, and that the propagation delay on the links is negligible. We will discuss later how the latter assumption can be relaxed without changing the results.

We consider a router R_m at distance m from the ingress ports of the tree network (leaves of the tree), and compare the occupancy of its buffer to the buffer occupancy of the single router R. Assuming that buffers in both routers are of size B, we show that R_m does not drop more packets than R.

Lemma 1. For any $t > m$, the number of drops at R_m in interval $[0, t)$ is less than or equal to the number of drops at R in interval $[0, t - m)$.

Proof: First, we assume that all routers have buffers of infinite size. We compare the queue occupancy in router R_m at time t, and the queue occupancy in router R at time $t - m$, which are denoted by $q_m(t)$ and $q(t - m)$ respectively. Both systems (shown in Fig. 1) have the same input sequences. An arriving packet in the first system, goes through exactly m FIFO queues – each with departure rate of 1 packet per unit time – before getting into R_m. Whereas, packets in the second system are directly forwarded to R. Let $A_m(t - \tau, t)$ be the number of packet arrivals in R_m during the interval $[t - \tau, t)$. Define τ_0 as follows:

$$\tau_0 = \max_{\tau}\{0 \le \tau \le t; A_m(t - \tau, t) - \tau \ = q_m(t)\} \qquad (1)$$

τ_0 corresponds to the largest possible time interval in which the backlog $q_m(t)$ is built up. Picking the maximum of such intervals indicates that there is no arrival to R_m during the interval $[t - \tau_0 - 1, t - \tau_0)$; otherwise τ_0 should have been replaced by $\tau_0 + 1$, as it is assumed to be maximized in equation 1. Note that when a packet enters the tree-structured system, it takes at least m time units for the packet to reach the m-th buffer, as the capacity of each link is exactly 1 packet per unit of time. This indicates that all packets arriving to R_m in interval $[t - \tau_0, t)$ have been injected to the system during the time interval $[t - \tau_0 - m, t - m)$. Because the whole system is work-conservative, if one of those packets had arrived before $t - \tau_0 - m$, there should have been an arrival during the interval $[t - \tau_0 - 1, t - \tau_0)$.

Consider the single router R in the second system. Since the arrivals to both systems are exactly the same, router R should have received the same number of packets during the interval $[t - \tau_0 - m, t - m)$, and the queue occupancy at time $t - m$ satisfies $q(t - m) \ge A_m(t - \tau, t) - \tau$. Therefore, the number of packets in router R at time $t - m$ is no less than $q_m(t)$.

Now assume that all buffers in both systems have a finite size B. We will use a similar argument to show that for any given $t \ge 0$, the number of drops in R during the interval $[0, t - m)$ is greater than or equal to the number of drops in R_m during the interval $[0, t)$. In particular, we will show that for every t, there exists some $t_0 \ge 0$ such that the number of drops in R_m during $[t_0, t)$ is smaller than or equal to the number of drops in R during $[t_0 - m, t - m)$.

Let q'_m be the pseudo queue occupancy in router R_m, defined as $q'_m(t) = \max_\tau A_m(t - \tau, t) - \tau, \quad 0 \le \tau \le t$. Here, $A_m(t - \tau, t)$ denotes the number of packets arrived (not necessarily stored) in R_m during the interval $[t - \tau, t)$. Define τ_0 as follows:

$$\tau_0 = \max_{\tau'}\{0 \le \tau' \le t; A_m(t - \tau', t) - \tau' \ = q'_m(t)\}$$

Assume that during the interval $[t - \tau_0, t)$, k of these arriving packets are dropped from R_m at times $t - \tau_0 \le t_1, t_2, ..., t_k \le t$.

Let $S_m(I)$ be the number of packets stored in R_m during some time interval I. Note that at each time t_i, where a drop event occurs at router R_m, we have $S_m(t - \tau_0, t_i) - \tau_0 > B - 1$. Therefore, $A_m(t - \tau_0, t_k) - \tau_0 > B + k - 1$. Since there is no arrival to R_m during the interval $[t - \tau_0, t - \tau_0 - 1)$, all $A_m(t - \tau_0, t_k)$ packets must have arrived in R during the interval $[t_k - \tau_0 - m, t_k - m)$, and hence at least k of these packets must have dropped from R during this interval. The total number of drops in R_m until any time t, can not exceed the corresponding number in R until time $t - m$.

■

Now let us relax the condition that every flow which arrives to R_m has gone through exactly m previous buffering stages. Denote by m_i the number of prior buffers that flow f_i has gone through. Assume that each flow has a stationary arrival sequence, independent from other flows. Coupling $\{f_i(t)\}$ with $\{f_i(t - m_i)\}$ will result in the same tail probability in both R and R_m when the buffers are of unlimited size, and the same drop probability when the size of buffers is limited.

Assuming that the ingress traffic is Poisson, and that the network has a load factor $\rho < 1$, the drop rate at each queue will be bounded above by ρ^B. This makes the overall drop rate of any packet to be less than $n\rho^B$, if each packet goes through at most n routers. The following theorem immediately follows.

Theorem 1: In a tree-structured network with Poisson ingress traffic, packet drop rate of ϵ can be achieved if every router can buffer

$$B \ge \log_{1/\rho}(\frac{n}{\epsilon})$$

packets, where n is the maximum number of buffering nodes on each route.

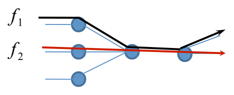

Fig. 2. An example of network with general topology.

B. General-Topology Networks

Fig. 2 is an example of a network with general topology, where a flow can merge with other flows by sharing a number of links and their corresponding buffers. But flows can also diverge on different routes after sharing a number of common links.

To analyze a queueing network of arbitrary topology, first consider a network in which each router delays every single packet which goes through it for exactly D units of time. Clearly, in such network, the inter-arrival time of packets in each flow will remain intact as the flow is routed across the network. In particular, if the network's ingress traffic is Poisson, then the input traffic to each intermediate router will continue to be Poisson. However, delaying every packet by some fixed amount of time is not feasible when the output link capacity is limited, and the inter-arrival time of successive arriving packets is very small. Packets coming in a burst can not all be sent out in a burst when the output link capacity is limited. Therefore, we need to allow some small variations in the amount of delay packets experience in a buffer. In the following, we define a simple active queue management policy which we call the Bounded Jitter Policy (BJP).

Informally, the BJP scheme tries to delay each packet for D units of time, but it also allows a cumulative slack of $\Delta \leq D$ units of time. We call Δ the "jitter bound" of the scheme. More exactly, a packet that enters the network at time t_0 is allowed to depart from the ith router anytime during the interval $[t_0 + iD - \Delta, t_0 + iD]$. In order to implement this scheme, we need to time-stamp packets at each router. The time-stamp of each packet is initialized to be zero as the packet enters the network. At each router on the path, the time-stamp is updated and incremented by D units of time.

Consider a router in the network, and assume that a packet with time-tamp t arrives at the router. The router tries to delay the packet for D units of time. If this exact delay can not generated, i.e., if another packet has already been scheduled for departure at time $s = t + D$, then the packet will be scheduled to depart the buffer at the largest available time-slot in the interval $[s - \Delta, s]$. If there is not any available time-slot in this interval, then the packet is dropped.

In the remainder of this section we study the buffer size requirements in a network where queues are managed by BJP. We assume that the ingress traffic of the network is Poisson, and show that under the BJP scheme, high throughput can be achieved with buffers of logarithmic size.

Theorem 2: In a network of arbitrary topology with Poisson ingress traffic, packet drop rate of ϵ can be achieved if the size of every buffer in the network is B, and

$$B \geq 4 \log_{1/\alpha}\left(\frac{n}{(1-\alpha)\epsilon}\right)$$

packets, where n is the maximum number of routers on any route, $\alpha = \rho e^{1-\rho}$, and $\rho < 1$ is load factor of the network.

Proof: Under BJP scheme, an arriving packet is dropped only if it can not be scheduled for in-time departure from the queue, i.e., if there is no free time-slot available in time interval $[s - \Delta, s]$. In this case, there must be exactly Δ packets in the queue whose scheduled departure times are in $[s - \Delta, s]$. Let p_m denote the probability of having an interval I_m of m consecutive busy (scheduled) time-slots-corresponding to the departure times of m packets at this router, followed by an available (nonscheduled) time-slot. As explained before, BJP tries to send out a given packet in a time-slot that is as close as possible to (and not later than) the packet's stamped departure time. Therefore, if the $m + 1$st time-slot is available, then the departure time-stamps of these m packets must fall within the same interval I_m.

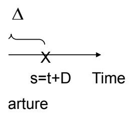

Fig. 3. A packet that arrives at time a, is scheduled based on its arrival time-stamp $t \geq a$. The departure time-stamp of this packet will be a fixed delay D added to the arrival time-stamp. The system schedules the packet to depart as close to the departure time-stamp as possible.

The packet drop probability, P_{drop}, is therefore bounded as follows:

$$P_{drop} \leq \sum_{m=\Delta}^{2\Delta} p_m < \sum_{m=\Delta}^{\infty} P\{A_{[m]} \geq m\} \tag{2}$$

Note that before the arrival time of a packet with time-stamp t, no packet in the router can be scheduled to depart later than $t + D + \Delta$, otherwise the jitter bound of some packet would have been violated. Therefore, m can not exceed 2Δ in equation 2.

The jitter bound Δ should be made large enough to make the drop probability less than the desired value ϵ. Let ρ be the arrival rate of the Poisson traffic. The following set of inequalities hold [7]:

$$\begin{aligned} \frac{e^{-m\rho}(m\rho)^m}{m!} &< P\{A_{[m]} \geq m\} \\ &< (1 - \frac{\rho m}{m+1})^{-1} \frac{e^{-m\rho}(m\rho)^m}{m!} \\ &< \frac{1}{1-\rho} e^{m(1-\rho)} \rho^m \end{aligned}$$

Substituting each term of the geometric sum in equation 2 by its upper bound, we have

$$P_{drop} < \frac{(\rho e^{1-\rho})^\Delta}{(1-\rho)(1-\rho e^{1-\rho})} \leq \frac{(\rho e^{1-\rho})^\Delta}{(1-\rho e^{1-\rho})^2}$$

Let n be the maximum number of FIFO buffers along the path of each flow. To make the overall drop rate smaller than ϵ, it suffices to make the drop rate at each queue Q smaller than ϵ/n, and hence, choose Δ such that

$$P_{drop} < \frac{\alpha^\Delta}{(1-\alpha)^2} \leq \epsilon/n \tag{3}$$

where $\alpha = \rho e^{1-\rho}$, and ρ is the aggregate rate of all flows sharing Q. This results in the following logarithmic constraint on the size of the jitter:

$$\Delta \geq 2 \log_{1/\alpha}\left(\frac{n}{(1-\alpha)\epsilon}\right) \tag{4}$$

Let the size of buffers, B, be equal to $D + \Delta$. Note that with such scheduling, there will be no drop incidence due to full occupancy of a buffer. No packet in a queue can be delayed more than $D + \Delta$ time-slots, without violating the delay policy. Therefore, at any time, the number of packets in a queue will be less than or equal to $D + \Delta$.

Equation 3 shows that the drop rate at each router is only a function of the slack size Δ, and does not depend on the value of the fixed delay D. The only constraint that the scheduling policy imposes is that $\Delta \leq D$. Therefore, the size of buffers can be made as small as 2Δ. This completes the proof of theorem 2.

■

C. Networks with smooth TCP incoming traffic

The results of previous sections provide bounds on the required buffer size for achieving a desired throughput under Poisson packet arrival. In this section, we study the throughput of the network in the case of having TCP traffic, where the sources reduce their transmission rates in response to packet drops in the network. We show that when the packet in every underlying flow of the ingress TCP traffic is smooth, then buffers of logarithmic size suffice for having high throughput.

In section II we explained the intuition behind the delay-bandwidth rule, and pointed out that a single TCP flow can get very bursty. It is important to note that requiring big buffers is not necessarily caused by the AIMD dynamics of TCP's congestion control mechanism. Traffic generated by TCP sources can be made very smooth without any modifications to the AIMD mechanism. Consider TCP sources which adjust their transmission rates by following the AIMD dynamics, but rather than sending out packets in bursts, their traffic is spread out over a round trip time interval (Note that still the average transmission rate over a round trip time is equal to what TCP dictates). Intuitively, this pacing should prevent bursts and hence remove the need for large buffers.

In our analysis, we assume that packet injection time of every flow (at the rate dictated by the AIMD mechanism of TCP) follows a Poisson model. As shown in [1], when the time gap between successive packets of any flow is at least $O(\log W)$, then the throughput will be the same as what the Poisson model gives us.

Consider a particular (but arbitrary) link l with bandwidth C packets per unit of time, and assume that N long-lived TCP flows share this link. Flow i has time-varying window size $W_i(t)$, and follows TCP's AIMD dynamics. In other words, if the source receive an ACK at time t, it will increase the window size by $1/W_i(t)$. If the flow detects a packet loss, it will decrease the congestion window by a factor of two. In any time interval [t, t') when the congestion window size is fixed, the source will send packets as a Poisson process at rate $W_i(t)/RTT$. This is different from regular TCP, which typically sends packets as a burst at the start of the window.

We will assume that the window size is bounded by W. Implementations today typically have a bound imposed by the operating system (Linux defaults to W=64 kbytes). We will make the simplifying assumption that the round trip time of all flows sharing link l is RTT. Having a different propagation delay for each flow leads to the same results, but the analysis is more complicated. We also assume that the network load factor ρ (as defined before) is less than one, which implies that $\rho_l = NW/RTT < C$, where ρ_l is the offered load on link l. The effective utilization, θ_l, on link l is defined as the achieved throughput divided by ρ_l. Under the above assumptions, the following theorem holds:

Theorem 3: To achieve an effective utilization of θ on each link in the network, buffers of size

$$B \geq 4 \log_{1/\alpha}\left(\frac{nW}{(1-\alpha)(1-\theta)}\right)$$

packets suffices under the BJP scheme.

Proof: It can be shown that if the overall drop rate of each packet in the network is bounded as

$$P_{drop} < \frac{2(1-\theta)}{W^2}$$

then, an effective utilization of θ will be achieved [1]. Using the result of theorem 2, it suffices that

$$\frac{n\alpha^{\Delta}}{(1-\alpha)^2} \leq \frac{2(1-\theta)}{W^2}$$

Therefore, it is enough for the buffer size B to satisfy

$$B = 2\Delta \geq 4 \log_{1/\alpha}\left(\frac{nW}{(1-\alpha)(1-\theta)}\right)$$

■

The above theorem states that with a fixed offered load and link utilization, the buffer size on a network link grows with the logarithm of W. Since the network is assumed to be over-provisioned, the maximum window size is limited by $C * RTT/N$, where N is the number of flows sharing that link. Consequently, as the capacity of the core links grows over time, the buffer size only needs to be increased logarithmically.

IV. Conclusions

Our results suggest that a network of routers with logarithmic size buffers can achieve high throughput under TCP traffic, as long as the ingress traffic is sufficiently paced. If the network has a tree structure, then no modification in needed in the routers. In a network with general topology, where flows might be intervened by other cross-traffic flows on their paths, implementing a simple active queue management policy in buffers makes the arrival traffic of each router behave as if it is directly coming from ingress ports of the network. The smoothness of TCP traffic is thus preserved across the network, and the logarithmic size buffers suffice.

Our results assume that packets are sufficiently spaced out as they enter the core network. Slow access links or slightly modifying TCP to space out packet injections (Paced TCP) can make this happen in a network. If having tiny buffers is inevitable in a network – in an all-optical network for instance – in which neither of the above conditions hold, one can use traffic shapers at the edge to space out packets entering the network. In this case, all the results mentioned in this paper will apply.

References

[1] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, *Routers with very small buffers*, In Proceedings of the IEEE INFOCOM'06, Barcelona, Spain, April 2006.

[2] G. Raina, D. Towsley, and D. Wischik, *Part II: Control theory for buffer sizing*, ACM/SIGCOMM Computer Communication Review, 35(3):7982, July 2005.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. *Sizing router buffers*, In Proceedings of SIGCOMM '04, pp. 281-292, New York, NY, USA, 2004.

[4] R. S. Prasad, C. Dovrolis, and M. Thottan. *Router Buffer Sizing Revisited: The role of the input/output capacity ratio*, In Proceedings of the ACM CoNext conference, New York, December 2007

[5] A. Aggarwal, S. Savage, and T. Anderson, *Understanding the performance of TCP pacing*, In Proceedings of the IEEE INFOCOM, pages 1157-1165, Tel-Aviv, Israel, March 2000.

[6] C. Villamizar and C. Song. "High performance TCP in ANSNET," ACM Computer Communication Review, 24(5):45-60, 1994

[7] B. Klar, *Bounds on tail probabilities of discrete distributions*, Probability in the Engineering and Informational Sciences, 14(2):161-171, April 2000

[8] R. Motwani, and R.Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995

[9] F. T. Leighton, B. M. Maggs, and S. B. Rao, *Packet routing and job-shop scheduling in O(congestion+dilation) steps*, Combinatorica, Vol. 14, No. 2, pp. 167-180, 1994.

[10] M. Grossglauser and S. Keshav. *On CBR Service*, In Proceedings of INFOCOM'96, pp129-137, March 1996.