Queue Length Based Internet Congestion Control

Marios Lestas *Member IEEE*, Andreas Pitsillides, *Senior Member IEEE*,
Petros Ioannou, *Fellow IEEE*, George Hadjipollas *Student Member IEEE*

Abstract— **In this paper we present new queue length based Internet congestion control protocol which is shown through simulations to work effectively. The control objective is to regulate the queue size at each link so that it tracks a reference queue size chosen by the designer. To achieve the latter, the protocol implements at each link a certainty equivalent proportional controller which utilizes estimates of the effective number of users utilizing the link. These estimates are generated online using a novel estimation algorithm which is based on online parameter identification techniques. The protocol utilizes an explicit multi-bit feedback scheme and does not require maintenance of per flow states within the network. Extensive simulations indicate that the protocol is able to guide the network to a stable equilibrium which is characterized by max-min fairness, high utilization, queue sizes close to the reference value and no observable packet drops. In addition, it is found to be scalable with respect to changing bandwidths, delays and number of users utilizing the network. The protocol also exhibits nice transient properties such as smooth responses with no oscillations and fast convergence.**

I. Introduction

TCP congestion control has served the Internet remarkably well as this has evolved from a small scale network to the largest artificially deployed system. However, TCP is known to exhibit undesirable properties such as low utilization in the presence of large bandwidth delay products and random packet losses ([1], [2]). It has also been shown analytically that as bandwidth delay products increase TCP becomes oscillatory and prone to instability ([3]).

These observations have triggered intense research activity on Internet Congestion Control which has led to TCP enhancements and new congestion control protocols. Due to the fundamental performance limitations of protocols which employ binary and implicit feedback, in the last few years there has been renewed interest in the design of network assisted congestion control protocols which employ explicit, multi-bit feedback. Three of the most promising proposals (XCP [4], RCP [5] and ACP [6]) share a common objective: to match at each link the input data rate to the link capacity and at the same time ensure zero queue sizes. In this paper, we consider a different design objective: to regulate at each link the queue size so that it tracks a reference queue size chosen by the designer. This approach pushes the system to its limits as it forces the network to achieve full network utilization and takes advantage of all network resources including queues. It also reduces the number of states that need to be maintained at each link as the integral action is offered for free by the queueing dynamics. Despite many proposals for queue length based congestion control protocols in the context of the ABR service in ATM networks ([7], [8]) , in Internet congestion control the only notable attempt is reported in [9]. The scheme computes the feedback signal sent to the TCP users by using a combination of a logarithmic function and an additive increase and multiplicative decrease policy. The problem with this approach is that it is based on ad-hoc nonlinear techniques which may cause stability problems as the delays and number of users within the network increase.

This work was partly funded by the RPF VIDEO and the UCY ADVIDEO projects .

Marios Lestas, Andreas Pitsillides and George Hadjipollas are with the Computer Science Department, University of Cyprus, Nicosia, Cyprus, `lestas@cs.ucy.ac.cy`, `andreas.pitsillides@ucy.ac.cy`, `hpollas@ucy.ac.cy`

Petros Ioannou is with the Electrical Engineering Department, University of Southern California, Los Angeles, California, `ioannou@usc.edu`

In this paper, we present a new, queue length based Internet congestion control protocol which is shown through simulations to work effectively. The proposed protocol utilizes an explicit multi-bit feedback scheme similar to the one in [4] and does not require maintenance of per flow states within the network. It implements at each link a certainty equivalent, proportional controller which utilizes estimates of the effective number of users utilizing the link. These estimates are generated online using a novel estimation algorithm which is presented in detail in [10] and is based on online parameter identification techniques. Extensive simulations indicate that the protocol is able to guide the network to a stable equilibrium which is characterized by max-min fairness, high utilization, queue sizes close to the chosen reference value and no observable packet drops. In addition, it is found to be scalable with respect to changing bandwidths, delays and number of users utilizing the network. The protocol also exhibits nice transient properties such as smooth responses with no oscillations and fast convergence. In section II we present the protocol in detail and in section III we evaluate its performance through simulations. Finally in section IV we offer our conclusions and future research directions.

II. The Protocol

In this section, we describe the main features of the protocol. Some of the functionalities of the protocol are inspired from the work in [4].

A. The packet header

The packet carries a congestion header which consists of 3 fields as shown in Fig. 1 . The H_rtt field carries the current round trip time estimate of the source which has generated the packet. The field is set by the user and is never modified in transit. It is read by each router and is used to calculate the control period. The $H\_feedback$ field carries the sending rate which the network requests from the user which has generated

H_rtt (sender's rtt estimate)
H_feedback (desired sending rate)
H_congestion (congestion bit)

Fig. 1. Congestion header

the packet. This field is initiated with the user's desired rate and is then updated by each link the packet encounters in its path. At each link, the value in the field is compared with the desired sending rate value and the smallest value is stored in the $H\_feedback$ field. In this way, a packet as it traverses from source to destination it accumulates the minimum sending rate it encounters in its path. The $H\_congestion$ bit is a single bit which is initialized by the user with a zero value and is set by a link if the input data rate at that link is more that 95% of the link capacity. In this way, the link informs its users that it is on the verge of becoming congested so that they can apply a delayed increase policy and avoid excessive instantaneous queue sizes and packet losses.

B. The sender

As in TCP, each link maintains a congestion window $cwnd$ which represents the number of outstanding packets and a smoothed estimate of the current round trip time $srtt$ which is calculated using an exponentially weighted moving average filter.

The initial congestion window value is set to 1 and is never allowed to become less than this value because this would cause the source to stop sending data. On packet departure, the $H\_feedback$ field in the packet header is initialized with the desired sending rate of the application and the H_rtt field stores the current estimate of the round trip time. If the source does not have a valid estimate of the round trip time the H_rtt field is set to zero.

The congestion window is updated every time the sender receives an acknowledgement. When a new acknowledgement is received, the value in the $H\_feedback$ field, which represents the sending rate requested by the network in bytes per second, is read and is used to calculate the desired congestion window as follows:

$$desired\_window = \frac{H\_feedback \times srtt}{size} \quad (1)$$

where size is the packet size in bytes. We multiply with the $srtt$ to transform the rate information into window information and we divide by the packet size to change the units from bytes to packets. The desired window is the new congestion window requested by the network. We do not immediately set the $cwnd$ equal to the desired congestion window because this abrupt change may lead to bursty traffic. Instead we choose to gradually make this change by means of a first order filter. The smoothing gain of this filter depends on the state of the $H\_congestion$ bit in the acknowledgement received. If this is equal to 1, which indicates congestion in the source destination path, we apply a less aggressive increase policy. The congestion window is updated according to the following equation:

$$cwnd = cwnd + \frac{0.1}{cwnd}(desired\_window - cwnd) \quad (2)$$

if $desired\_window > cwnd$ and $H\_congestion = 1$ and

$$cwnd = Pr[cwnd + \frac{1}{cwnd}(desired\_window - cwnd)] \quad (3)$$

otherwise. The projection operator Pr[.] is defined below and guarantees that the congestion window does not become less than 1.

$$Pr[x] = \begin{cases} x & \text{if } x > 1 \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

C. The receiver

The receiver is identical to the XCP receiver. When it receives a packet it generates an acknowledgement in which it copies the congestion header of the packet.

D. The router

The router maintains at each output queue a value p, which represents the sending rate it desires from all users traversing the link. The desired rate is updated every time a control timer expires. The timer is set to to expire every average round trip time d. The average round trip time is initiated with a value of 0.05 and is also updated every time the control timer expires. To calculate the average round trip time, the router records upon packet arrival, the value stored in the H_{rtt} field of the packet header and divides the sum of the values recorded in one control period with the period.

In order to calculate the desired sending rate, the router requires two more variables at each output queue: the persistent queue size q and the rate of incoming packets y. The persistent queue size q is computed by taking the minimum queue seen by the arriving packets during the last propagation delay. The propagation delay is unknown at the router and is thus estimated by subtracting the local queueing delay from the average RTT. The local queueing delay is calculated by dividing the instantaneous queue size with the link capacity.

The rate of incoming packets y, is equal to the number of packets entering the queue in one control period divided by the control period. To calculate the number of received packets, the router maintains a variable which is incremented with the packet size every time the queue receives a packet. When the control timer expires, the link calculates y by adding the received number of packets and then dividing with the control period. It then resets the received number of bytes.

The above variables are used to calculate the desired rate p every control period. At each link, the objective is to regulate the queue size q so that it tracks a reference queue size q_{ref} chosen by the designer. To achieve the latter, we calculate the desired sending rate p using a modified version of the algorithm proposed in ([11]). The algorithm is as follows:

$$p(k) = Pr[\frac{C - \frac{1}{d(k)}(q(k) - 2q_{ref})}{\hat{N}(k)}] \tag{5}$$

where $\hat{N}(k)$ represents an estimate of the number of users utilizing the link which is calculated online. The projection operator is defined below:

$$Pr[x] = \begin{cases} 1 & \text{if } x < 1 \\ C & \text{if } x > C \\ x & \text{otherwise} \end{cases} \tag{6}$$

It guarantees that the desired sending rate is greater than 1 and smaller than the link capacity. The lower bound is imposed since we know priori that the protocol will send a packet, only if it has at least one byte to send. Values greater than the link capacity are not feasible. The control algorithm (5) basically applies proportional action. The delay term $d(k)$ is added to maintain stability in the presence of delays. A novel part of the proposed scheme is that at each link, the estimate of the number of users utilizing the link $\hat{N}(k)$ is generated online using an algorithm which is presented in detail in [10] and is based on online parameter identifiaction techniques. The estimation algorithm is as follows:

$$\hat{N}(k+1) = Pr[\hat{N}(k) + \delta\hat{N}(k)], \quad \hat{N}(0) = 10 \tag{7}$$

where

$$\delta\hat{N}(k) = \frac{\gamma[y(k) + \frac{q(k-1)}{d(k)} - \hat{N}(k)p(k-1)]p(k-1)}{1 + p^2(k-1)} \tag{8}$$

The projection operator Pr[.] is defined in (4). The projection operator guarantees that the number of flows traversing the link is never allowed to be less than 1. Values less than one are obviously not feasible. γ is a design parameter which affects the convergence properties of the algorithm. We choose γ to be equal to 0.1. Note that the initial value of the estimated number of flows $\hat{N}$ is equal to 10. We choose this value to ensure a relatively conservative policy when initially updating the desired sending rate.

The desired sending rate calculated at each link is used to update the $H\_feedback$ field in the packet header. On packet departure, the router compares the desired sending rate with the value stored in the $H\_feedback$ field and updates the field with the minimum value. In this way, a packet as it traverses from source to destination it accumulates the minimum of the desired sending rates it encounters in its path.

The last function performed by the router at each link is to notify the users traversing the link of the presence of congestion so that they can apply a delayed increase policy. On packet departure the link checks whether the input data rate is larger than 0.95 the link capacity. In this case it deduces that the link is congested and sets the $H\_congestion$ bit in the packet header.

III. Performance Evaluation

Our objective has been to develop a window based protocol which does not require maintenance of per flow states within the network and satisfies all the design objectives of congestion control protocols. In this section, we demonstrate through simulations that the proposed protocol satisfies its design objectives to a very good extent.

A. Scalability

It is important for congestion control protocols to be able to maintain their properties as network characteristics change. We thus investigate the scalability of the proposed protocol with respect to changing link bandwidths, propagation delays and number of users utilizing the network.

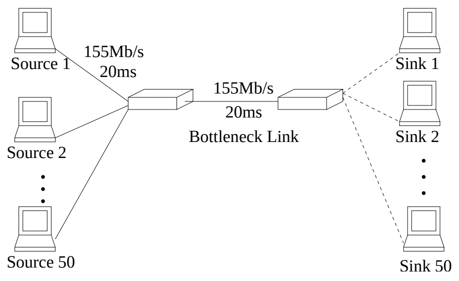
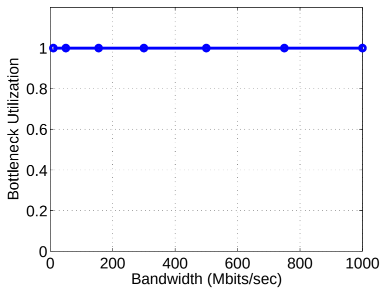


Fig. 2. Single bottleneck link topology used to investigate the scalability of the protocol with respect to changing link capacities, delays and number of users.

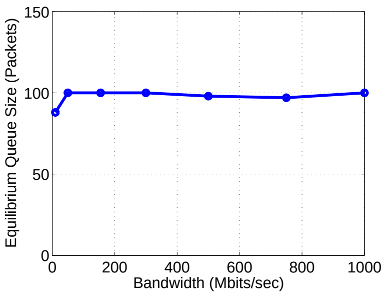
We conduct our study by considering the single bottleneck link network shown in Fig. 2. In the basic setup, 50 users share the bottleneck link through access links. The bandwidth of all links in the network is set equal to 155Mb/sec and the propagation delay is set equal to 20msec. The access links have different propagation delays. The propagation delay of the access link of the first user is set equal to the same value as that of the bottleneck link and the propagation delays of the access links of the rest of the users differ by increments of 0.5msec. In this way we create an asynchronous network. As mentioned above, the purpose of this study is to investigate the scalability of the protocol with respect to changing bandwidths, delays and number of users utilizing the network. We consider bandwidths in the range 10Mbits/s-1Gbit/sec, delays in the range 10msec-1sec and number of users in the range 1-1000. The performance metrics that we use in this study are the equilibrium utilization and the equilibrium queue size at the bottleneck link. The *equilibrium* values are calculated by averaging the values recorded after the system has converged to its equilibrium state. We do not report packet drops, as in all simulations we do not observe any. In addition, we do not show fairness plots, as in all simulations the network users are assigned the same sending rate at equilibrium, which implies that max-min fairness is achieved in all cases. The dynamics of the protocol and its ability to perform well in more complex network topologies are investigated in separate studies later in this section.

In our simulations, we consider persistent FTP sources. The packet size is equal to 1000 bytes and the buffer size of all links is set equal to the bandwidth delay product. The reference queue size q_{ref} is chosen to be equal to 100 packets. The simulation time is not constant. It varies depending on the round trip propagation delay. We simulate for a sufficiently long time to ensure that the system has reached an equilibrium state. It is highly unlikely that in an actual network the network users will enter the network simultaneously. So, in all scenarios, the users enter the network with an average rate of one user per round trip time.

Effect of Capacity: We first evaluate the performance of the proposed protocol as we change the link bandwidths. We fix the number of users to 50, we fix the propagation delays to 20msec and we consider link bandwidths in the range 10Mbits/s-1Gbit/s. Plots of the bottleneck utilization and the average queue size versus the link capacity are shown in Fig. 3.



(a) Utilization vs Capacity

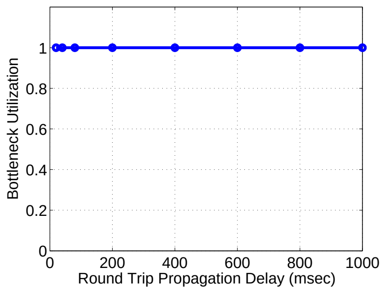


(b) Average Queue Size vs Capacity

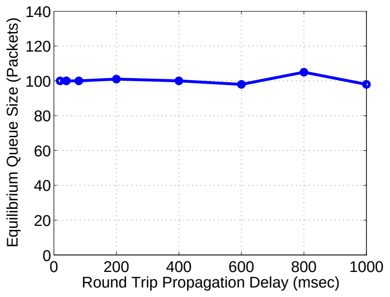
Fig. 3. The protocol achieves full network utilization and experiences no drops as the capacity increases. The equilibrium queue size is always close to 100 which is the reference value.

We observe that the proposed scales well with increasing bandwidths. The protocol achieves full network utilization (100%) at all bandwidths. Moreover, the queue size always converges to an equilibrium value which is close to 100 as required.

Effect of Delays: We then investigate the performance of the protocol as we change the propagation delay of the links. Any change in the link propagation delay causes a corresponding change in the round trip propagation delay of all source destination paths. We fix the link bandwidths to 155Mbits/s, we fix the number of users to 50 and we consider round-trip propagation delays in the range 10ms-1sec. It must be noted that each user of the network has different round trip propagation delay, since the propagation delay of each access link is different. So, when we refer to the round trip propagation delay of a particular simulation we refer to the minimum round trip propagation delay among the network users. Plots of the bottleneck utilization and the average queue size versus the round trip propagation delays are shown in Fig 4.



(a) Utilization vs Delay



(b) Average Queue Size vs Delay

Fig. 4. The protocol achieves full network utilization and experiences no drops as the round trip propagation delay increases. The equilibrium queue size is close to 100 at all delays as required.

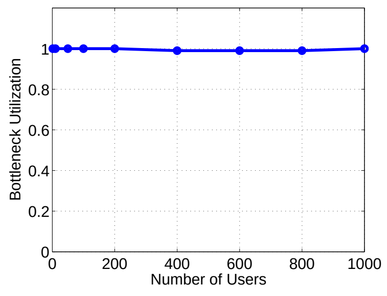
The results are similar to the results obtained when investigating the effect of changing capacities. Fig. 4 (a) demonstrates that the protocol achieves full network utilization at all delays and that the queue size at equilibrium is close to 100 as required.

Effect of the Number of Users We finally investigate the performance of the proposed protocol as we increase the number of users utilizing the single bottleneck link network in Fig. 2. We consider different number of users in the range 1-1000. Plots of the bottleneck utilization and the average queue size versus the number of users are shown in Fig. 5.

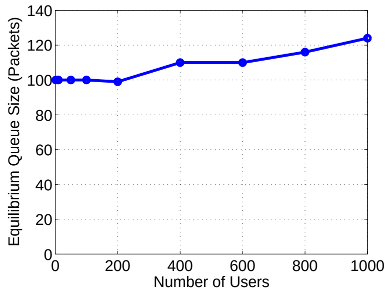
We observe that the protocol achieves full network utilization in all cases. However, we also observe that the equilibrium queue size starts deviating from the reference queue size as the number of users increases. The reason for this is that as the number of users increases, the queue size experiences oscillations of increasing magnitude with a corresponding shifting of the mean value towards a higher value. However, this increase in the value of the queue size at equilibrium is relatively small and no packet losses are observed.

B. The Dynamics of the protocol

To fully characterize the performance of the proposed protocol, apart from the properties of the system at equilibrium, we need to investigate its transient properties. The protocol must generate smooth responses which are well damped and converge fast to the desired equilibrium state. To conduct our study we consider the following dynamic scenario. 30 users originally utilize the single bottleneck link network shown in Fig. 2. At 30 seconds 20 of theses users stop sending data



(a) Utilization vs Number of Users



(b) Average Queue Size vs Number of Users

Fig. 5. The protocol achieves full network utilization and experiences no packet drops in all cases. However, the equilibrium queue size increases with increasing number of users.

simultaneously. So the number of users utilizing the network is reduced to 10. At 45 seconds, however, 40 additional users enter the network thus causing the number of users to increase to 50.

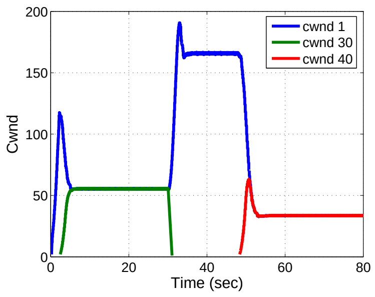
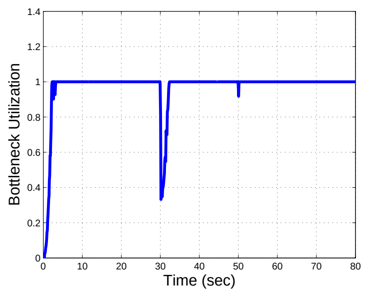


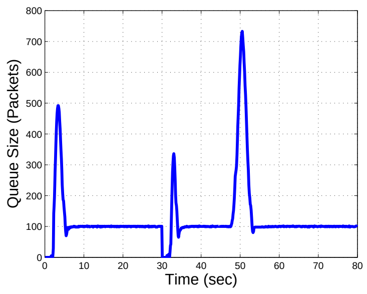
Fig. 6. Time response of the congestion window of three users. User 1 utilizes the network throughout the simulation, user 30 stops sending data at 30 seconds and user 40 enters the network at 45 seconds. We observe smooth and fast responses with no oscillations.

In Fig. 6 we present the time responses of the congestion window of a representative number of users. User 1 utilizes the network throughout the simulation, user 30 stops sending data at 30 seconds and user 40 enters the network at 45 seconds. The transient behavior of the other users is very similar to the ones shown in Fig 6. We observe that the protocol achieves smooth responses which converge fast to the desired equilibrium with no oscillations. However, in some cases, they experience overshoots. When user 1 starts sending data it converges fast to its max-min fair allocation. Since the users gradually enter the network, the max-min allocation gradually decreases. This is why the congestion window of user 1 experiences a large overshoot before settling down to its equilibrium value. Note, however, that once the desired sending rate calculated at the bottleneck link has settled down to an equilibrium value, a new user, such as user 30, converges fast to the max-min allocation value with no overshoots. When the 20 users suddenly stop sending data at 30 seconds the flow of data through the bottleneck link drops thus causing an instantaneous underutilization of the link. The link identifies this drop in the input data rate and reacts by increasing its desired sending rate. This causes user 1 to increase its congestion window. The time response in Fig 6 indicates fast convergence to the new equilibrium value with no oscillations. However, the response does experience a small overshoot before settling down to its equilibrium value. When 40 new users enter the network at 45 seconds, the max-min fair sending rate decreases. The controller at the bottleneck link iteratively calculates this rate and communicates this information to the end users. This causes user 1 to decrease its congestion window and user 40 which has just entered the network to gradually increase its congestion window to the equilibrium value. We observe from Fig. 6 that user 1 converges fast to the new equilibrium value with no undershoots or oscillations. We also observe that the time response of the congestion window of user 40 experiences a small overshoot before settling down to its equilibrium value. This is due to the fact that the user sets its sending rate equal to the desired sending rate calculated at the bottleneck link while the latter is still decreasing.

The next thing we investigate is the transient behavior of the utilization and the queue size at the bottleneck link. In Fig. 7 we show the time responses of the utilization and the queue size at the bottleneck link.



(a) Utilization vs Time



(b) Queue Size vs Time

Fig. 7. Time response of the instantaneous utilization and the queue size at the bottleneck link. Utilization converges fast to a value which is close to 1. The queue size experiences instantaneous increases when new users enter the network but at equilibrium the queue size is equal to the reference value.

We observe that the link utilization converges fast to a value which is close to 1. When the 20 users leave the network, the flow of data suddenly decreases thus causing an instantaneous decrease in the utilization. However, the system reacts quickly by increasing the sending rate of the remaining users, thus achieving almost full utilization in a very short period of time.

The time response of the queue size indicates that the latter converges to a value which is close to 100. This is the main objective of the congestion control protocol. However, in the transient periods during which new users enter or leave the network, the queue size experiences an instantaneous increase. It might seem strange that we observe increasing queue sizes when users leave the network. This is caused by the fact that the remaining users, while they increase their sending rate to take up the slack created, they experience overshoots. However, careful choice of the control parameters at the links and the delayed increase policy that we apply at the sources ensure that these overshoots do not exceed the buffer size and thus do not lead to packet drops.

C. *A multi-link example*

Until now we have evaluated the performance of the proposed protocol in a single bottleneck link network topology. Our objective in this section is to investigate how the protocol performs in a more complex network topology. We consider the parking lot topology shown in Fig. 8.

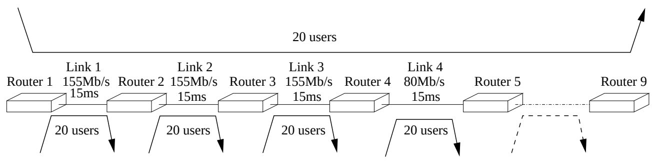
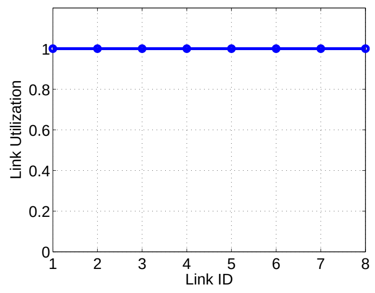


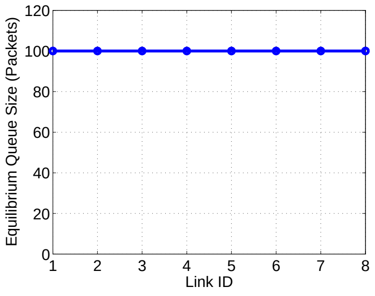
Fig. 8. A parking lot network topology

The network consists of 8 links which are connected in series. All links have a bandwidth of 155Mbits/sec except link 4 which has a bandwidth of 80Mbits/sec. The propagation delay of all links is set equal to 15msec. 20 users utilize the network by traversing all 8 links. Moreover, each link in the network is utilized by an additional 20 users which have single hop paths as shown in Fig. 8. In this way, all links in the network are bottleneck links and link 4 is the single bottleneck link for the 20 users which traverse the whole network. In fig. 9, we show on separate graphs the equilibrium utilization and the equilibrium queue size recorded at each link.

Since all links in the network are bottleneck links for some flows, we do expect them to be fully utilized. Indeed, we observe that the proposed protocol achieves full utilization at all links. In addition, at all links the equilibrium queue size is equal to the reference queue size as required.



(a) Utilization at each link



(b) Queue at each link

Fig. 9. The protocol achieves full utilization at all links and experiences no packet drops. In addition, the equilibrium queue size is equal to 100 as required.

IV. Conclusions

In this paper we present new Internet congestion control protocol whose objective is to regulate the queue size at each link so that it tracks a reference queue size chosen by the designer. We demonstrate through simulations that the protocol meets its design objectives to a very good extent. Our next objective is to further evaluate its performance in more complex topologies and in the presence of realistic weblike traffic. We also aim at establishing its properties analytically in networks of arbitrary topology.

References

[1] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[2] S. Floyd and V. Jacobson. Connections with multiple congested gateways in packet-switched networks. *Comput. Commun. Rev.*, 21(5):30–47, August 1991.

[3] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J.C. Doyle. Dynamics of TCP/RED and a scalable control. In *Proc. IEEE INFOCOM*, volume 1, pages 23–27, June 2002.

[4] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for high-bandwidth-delay products. In *Proc. ACM SIGCOMM*, August 2002.

[5] N. Dokkipati, M. Kobayashi, Rui Zhang-Shen, and Nick McKeown. Processor sharing flows in the internet. In *Proc. Thirteenth Intenrational Workshop on Quality of Service 2005,*, June 2005.

[6] M. Lestas, A. Pitsillides, P. Ioannou, and G. Hadjipollas. Adaptive congestion protocol: A new congestion control protocol with learning capability. *Computer Networks*. Submitted for publication.

[7] L. Benmohamed and S. M. Meerkov. Feedback control of congestion in packet switching networks: The case of a single congested node. *IEEE/ACM Transactions on Networking*, 1(6):693–708, December 1993.

[8] S. Chong, R. Nagarajan, and Y. Wang. Designing stable ABR flow control with rate feedback and open loop control:first-order control case. *Performance Evaluation*, 34(4):189–206, December 1998.

[9] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Explicit window adaptation: A method to enhance TCP performance. In *Proc. INFOCOM*, pages 242–251, 1998.

[10] M. Lestas, A. Pitsillides, P. Ioannou, and G. Hadjipollas. A new estimation scheme for the effective number of users in internet congestion control. *IEEE/ACM Transactions on Networking*. Submitted for publication.

[11] G. Veciana C. F. Su and J. Walrand. Explicit rate flow control for ABR services in ATM networks. *IEEE/ACM Transactions on Networking*, 8(3):350–361, June 2000.