Adaptive Congestion Protocol: A Congestion Control Protocol with Learning Capability

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

Abstract

There is strong evidence that the current implementation of TCP will perform poorly in future high speed networks. To address this problem many congestion control protocols have been proposed in literature which, however, fail to satisfy key design requirements of congestion control protocols, as these are outlined in the paper. In this work we develop an Adaptive Congestion Protocol (ACP) which is shown to satisfy all the design requirements and thus outperform previous proposals. 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, small queue sizes 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. ACP does not require maintenance of per flow states within the network and utilizes an explicit multi-bit feedback signalling scheme. To maintain stability it implements at each link a novel estimation algorithm which estimates the number of users utilizing the network. Using a simple network model, we show analytically the effectiveness of the estimation algorithm. We use the same model to generate phase portraits which demonstrate that the ACP protocol is stable for all delays.

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. The schemes which have emerged from this research effort have different degrees of complexity in terms of the feedback mechanism with which congestion information is transferred from the network to the end users. The current implementation of TCP ([4]) utilizes a feedback scheme which is binary and implicit as it uses packet losses to infer congestion. Floyd has suggested a modification to TCP which introduces a single ECN bit in the packet header to enable the explicit notification of the end users of the presence of congestion ([5]). Such a modification has been shown to improve TCP performance but fails to deliver schemes which satisfy all the design objectives ([6]). Binary feedback schemes have fundamental performance limitations.

These limitations have motivated researchers to explore ways with which to enable explicit multi-bit feedback. One idea has been to use the receiver window field in the packet header to transfer explicit congestion information ([7], [8]). This approach is attractive since it is transparent to the TCP protocol and requires no standardization. However, a single field in the packet header may not be enough to convey the necessary congestion information. A feedback mechanism which has been recently proposed in [9] introduces multiple fields in the packet header which are used to transfer multiple variables between the

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

Marios Lestas and Petros Ioannou are with the Electrical Engineering Department, University of Southern California, Los Angeles, California, `lestas@usc.edu`, `ioannou@usc.edu`

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

network and the end users. The performance implications of these explicit feedback schemes on TCP congestion control have been explored in a number of studies. Some of these studies are based on the assumption that fair rate information is available at the network routers ([7], [10]). Availability of this information can be materialized by leveraging any of the algorithms which have been proposed in the context of the ABR service in ATM networks (e.g. [11], [12] [13]). However, these algorithms require maintenance of per flow states within the network and are thus not amenable for implementation in TCP/IP networks. Attempts to develop algorithms which do not require maintenance of per flow states within the network include the queue length based approach in [8], the XCP protocol presented in [9] and the RCP protocol presented in [14]. All these approaches have distinct disadvantages. The scheme proposed in [8] generates feedback signals using queue length information only. However, it is well known that such an approach offers limited control space and thus leads to significant oscillations and degradation in performance, in networks with high bandwidth delay products. The RCP protocol has been designed with the objective of minimizing the completion time of the network flows. In order to achieve the latter, it applies a rather aggressive policy when increasing or decreasing the sending rate of the network users. However, as shown in this paper, such an aggressive policy can cause underutilization of the network for large periods of time. XCP constitutes the most promising approach as it achieves high network utilization, smooth and fast responses, scalability with respect to changing bandwidths, delays and the number of users utilizing the network, small queue sizes and almost no packet drops. However, it has been shown in [15] that the scheme fails to achieve fairness in scenarios with multiple congested links. The question that arises is the following: Assuming that we can use an explicit multi-bit feedback scheme, is it possible to develop a window based congestion control protocol which satisfies all the design objectives and does not require maintenance of per flow states within the network? In this paper we demonstrate that the answer to this question is yes.

We develop ACP (Adaptive Congestion Protocol) which is a new congestion control protocol with learning capability. This learning capability enables the protocol to adapt to dynamically changing network conditions and maintain stability. ACP can be characterized as a dual protocol where intelligent decisions are taken within the network. The main control architecture is in the same spirit as the one used by the ABR service in ATM networks. Each link calculates at regular time intervals a value which represents the sending rate it desires from all users traversing the link. A packet as it traverses from source to destination it accumulates, in a designated field in the packet header, the minimum of the desired sending rates it encounters in its path. This information is communicated to the user which has generated the packet through an acknowledgement mechanism. The user side algorithm then gradually modifies its congestion window in order to match its sending rate with the value received from the network. The user side algorithm also incorporates a delayed increase policy in the presence of congestion to avoid excessive queue sizes and reduce packet drops. The justification for this policy is discussed in section II.

At each link, the desired sending rate is calculated using both queue length and rate information. The algorithm does not require maintenance of per flow states within the network. Previous experience in the design of such link algorithms ([16], [17], [18], [19], [20]) has shown that in order to maintain stability in the presence of delays, the control parameters of the algorithm need to be normalized with the number of users utilizing the network. This is an unknown time varying parameter. Algorithms which have been proposed to estimate this parameter are based on point wise division in time ([21], [22], [23]). This approach, however, is known to lack robustness and lead to erroneous estimates. In this work, we use on-line parameter identification techniques to derive an estimation algorithm which is shown through analysis and simulations to work effectively.

We evaluate the performance of the proposed protocol using simulations. Extensive simulations indicate that the proposed protocol satisfies all the design objectives. The scheme guides the network to a stable equilibrium which is characterized by high network utilization, max-min fairness, small queue sizes and almost no packet drops. It is scalable with respect to changing delays, bandwidths and number of users utilizing the network. It also exhibits nice dynamical properties such as smooth responses and fast convergence. In our simulations we use realistic traffic patterns which include both bulk data transfers and short lived flows. Finally, we address stability issues of the ACP protocol by using phase plane analysis. We use a non-linear network model to generate phase portraits which demonstrate that ACP is stable for all delays.

The paper is organized as follows. In section II we describe the reasoning behind our design choices, in section III we present in detail the congestion control protocol, in section IV we evaluate the performance of ACP and finally in section V we present our conclusions and our future research directions.

II. Design Guidelines

Our objective has been to design an effective window based congestion control protocol assuming availability of a feedback mechanism which allows the explicit exchange of information between the end users and the network. An effective congestion control protocol must satisfy some basic requirements. It must guide the system to a stable equilibrium point which is characterized by high network utilization, max-min fairness, small queue sizes and no packet drops ([24]). It also needs to be scalable with respect to changing bandwidths, delays, and number of users. Finally, it must exhibit nice dynamical properties such as smooth responses with fast convergence and no overshoot.

XCP constitutes the most notable attempt to fulfil the above objectives. The protocol achieves most of the specifications but fails to achieve max-min fairness at equilibrium in the case of multiple congested links ([15]). XCP has been shown in [9] to outperform all other TCP proposals so our objective has been to develop a protocol which outperforms XCP.

At each link in the network, XCP tries to match the input data rate to the link capacity and at the same time maintain small queue sizes. It achieves this by explicitly dictating to each user utilizing the link the amount by which the congestion window must be increased. However, this does not guarantee that all users utilizing the link will share the same sending rate. XCP addresses this problem by implementing a fairness controller which, however, proves to be ineffective in the case of multiple congested nodes. In order to fix this problem ACP adopts a different design approach. Each link in the network, instead of calculating desired increments of the sending rate of the users traversing the link, it calculates the desired sending rate of the users directly. In a way similar to XCP, the desired sending rate is made available to the end users through an explicit feedback mechanism. A packet as it traverses from source to destination it accumulates, in a designated field in the packet header, the minimum of the desired sending rates it encounters in its path. This information is communicated to the user which has generated the packet through an acknowledgement mechanism. The user then gradually modifies its congestion window in order to match its sending rate with the value received from the network. Since the above mechanism guarantees that all users bottlenecked at a particular link share the same sending rate, fairness is achieved automatically.

Having decided on the control architecture the next step is to design the algorithm which calculates the desired sending at each link. At each link, the objective is to match the input data rate to the link capacity and to maintain small queue sizes. So, the algorithm which updates the desired sending rate must use both queue length and rate information. Pure rate information does not guarantee bounded queue sizes and pure queue length information offers a limited control space and thus leads to oscillations in environments with high bandwidth delay products. Ignoring a projection operator which imposes hard bounds on the desired sending rate a continuous time version of the link algorithm is the following:

$$\dot{p} = \frac{1}{\hat{N}}[\frac{k_i}{d}(0.99 * C - y) - \frac{k_q}{d^2}q], \quad p(0) = p_0 \tag{1}$$

where p is the desired sending rate, C is the link capacity, y is the input data rate, q is the queue size, k_i and k_q are design parameters, d is the calculated average propagation delay and $\hat{N}$ is an estimate of the number of users traversing the link. The main idea is to integrate the excess capacity and add a queue length factor in order to ensure that at equilibrium the queue size converges to zero. Similar ideas have been used in previous attempts to design congestion control schemes ([25], [9], [14]). The main novelty of our approach is the way with which we estimate the number of flows online in order to maintain stability. We use online parameter identification techniques to derive the estimation algorithm and we implement a certainty equivalent controller ([26]).

Equation (1) is a continuous time representation of the algorithm. In practice, each link updates the desired sending rate every control period. The choice of this period needs careful consideration as it affects the stability and transient properties of the congestion control protocol. ACP chooses its control period using the same mechanism as XCP. Each user utilizes a designated field in the packet header to inform the routers of its current round trip time estimate. Each router then calculates the average round trip time over all packets traversing the link and sets the control period equal to this value. We choose this particular control period to counter the destabilizing effect of delays. Based on fundamental control theory principles, as delays increase within the network we slow down the response time of our controllers in order to maintain stability.

However, simply by choosing the control period as described above, we cannot guarantee stability in the presence of delays. Previous work ([9]) has shown that in order to maintain stability the excess capacity term must be divided with the time delay and the queue size term must be divided with the square of the propagation delay as shown in equation (1). Note that, in a discrete time implementation of the algorithm since we use a varying control period which is equal to the delay we only need to divide the queue term with the delay to maintain stability.

Previous experience in the design of explicit rate based schemes has also shown that in order to maintain stability in the presence of delays, the control parameters need to be normalized with the number of users utilizing the network. This is an unknown time varying parameter which needs to be estimated. Many estimation algorithms have been proposed in literature with different degrees of implementation complexity. Algorithms which do not require maintenance of per flow states within the network are based on point-wise division in time. Assuming that all users traversing the link, send data with the desired sending rate p, the input data rate y satisfies the following relationship:

$$y = Np \tag{2}$$

where N is the number of flows traversing the link. Since both y and p are known at the link, N can be estimated by dividing y with p. However, such a point-wise division in time is known to lack robustness and can lead to erroneous estimates. In this work, we treat (2) as a linear parametric model of the unknown parameter N and we use online parameter identification techniques to derive an estimation algorithm which is shown through analysis and simulations to work effectively.

As described earlier, each user in the network receives, through explicit feedback, the minimum of the desired sending rates a packet encounters in its path. Since we aim at implementing a window based protocol the rate information must be transformed into window information. We do this by multiplying the desired sending rate with a measure of the propagation delay. Such a measure is obtained by calculating the minimum of the round trip time estimates observed throughout the session. In order to avoid the generation of bursty traffic we do not immediately change the congestion window to the desired value calculated. Instead, we make this change gradually in one round trip time by means of a first order filter.

Even with such a gradual increase policy, excessive queue sizes and packet losses may be observed during transients. The problem arises when a particular link is congested (the input data rate is close to the link capacity) and new users traversing the link enter the network. In this case, the new users adopt the desired sending rate of the link in one round trip time. If the desired sending rate is large, the net effect is a sudden increase in the input data rate at the link. Since, the link is already congested, this increase can lead to large instantaneous queue sizes and packet drops. To alleviate this problem we apply a delayed increase policy at the source in the case of congestion. When a link is congested it sets a designated bit in the header of all incoming packets. In this way the users traversing the link are notified of the presence of congestion and react by applying a delayed increase policy. When they have to increase their congestion window they decrease the smoothing gain of the first order filter by a factor of 10 so that they increase at a much slower rate, thus giving time to the link to detect the new users, adapt its desired sending rate and avoid packet losses.

III. The Protocol

A. The packet header

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

Fig. 1. ACP congestion header

In a way similar to XCP the ACP 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 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 ACP sender

As in TCP, ACP maintains a congestion window $cwnd$ which represents the number of outstanding packets and an estimate of the current round trip time rtt. In addition to these variables ACP calculates the minimum of the round trip time estimates which have been recorded, $mrtt$. This is a good measure of the propagation delay of the source destination path and is used to transform the rate information reaching the sender to window information.

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:

$$desired\_window = \frac{H\_feedback \times mrtt}{size} \tag{3}$$

where size is the packet size in bytes. We multiply with the $mrtt$ 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:

$$cwnd = \begin{cases} cwnd + \frac{0.1}{cwnd}(desired\_window - cwnd) & \text{if } desired\_window > cwnd,\ H\_congestion = 1 \\ Pr[cwnd + \frac{1}{cwnd}(desired\_window - cwnd)] & \text{otherwise} \end{cases} \tag{4}$$

where the projection operator Pr[.] is defined as follows:

$$Pr[x] = \begin{cases} x & \text{if } x > 1 \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

The projection operator guarantees that the congestion window does not become less than 1.

C. The ACP receiver

The ACP 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 ACP router

At each output queue of the router, the objective is to match the input data rate y to the link capacity C and at the same time maintain small queue sizes. To achieve this objective the router maintains for each link, a value which represents the sending rate it desires from all users traversing the link. The desired sending rate is denoted by p and is updated every control period. The router implements a per link control timer. The desired sending rate and other statistics are updated every time the timer expires. The control period is set equal to the average round trip time d. The average round trip time is initialized with a value of 0.05 and is updated every control period. On packet arrival the router reads the H_rtt field in the packet header and updates the variables which are used to calculate the average round trip time.

The router calculates at each output queue the input data rate y. For each link, the router maintains a variable which denotes the number of received bytes. This variable is incremented with the packet size every time the queue associated with the link receives a packet. When the control timer expires, the link calculates the input data rate by dividing the received number of bytes with the control period. It then resets, the received number of bytes.

The router also maintains at each output queue the persistent queue size q. The 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 above variables are used to calculate the desired rate p every control period using the following iterative algorithm:

$$p(k+1) = Pr[p(k) + \frac{1}{\hat{N}(k)}[k_i(0.99 * C - y(k)) - \frac{1}{d(k)}k_q q(k)]], \quad p(0) = 0 \tag{6}$$

where k_i and k_q are design parameters, $\hat{N}$ represents an estimate of the number of users utilizing the link and the projection operator is defined as follows:

$$Pr[x] = \begin{cases} 0 & \text{if } x < 0 \\ C & \text{if } x > C \\ x & \text{otherwise} \end{cases} \tag{7}$$

The projection operator guarantees that the desired sending rate is non-negative and smaller than the link capacity. Values outside this range are not feasible. The design parameters k_i and k_q are chosen to be 0.1587 and 0.3175 respectively. In Appendix B we show using phase plane analysis that this choice of the design parameters guarantees that the ACP protocol is stable for all delays. There are several things to note about the link algorithm (6). The basic idea is to integrate the excess capacity and to add a queue size term to guarantee that at equilibrium the queue size converges to zero. Previous work has shown that in a continuous time representation of the algorithm, in order to maintain stability, the excess capacity term must be divided with the time delay and the queue size term must be divided with the square of the propagation delay. However, when transforming the continuous time representation to the discrete time representation of equation (6), we multiply both terms with the time delay and so we end up dividing only the queue term with the delay to maintain stability. Note also that we slightly underutilize the link at equilibrium by setting the virtual capacity equal to 99% of the true link capacity. We do this to reserve bandwidth resources which can be used to accommodate statistical fluctuations of the bursty network traffic. This prevents excessive instantaneous queue sizes.

Previous experience in the design of link algorithms for congestion control has shown that to maintain stability we need to normalize the control parameters with the number of users utilizing the network. A novel part of this work is that we use online parameter identification techniques to derive an algorithm which estimates the unknown parameter online. The derivation is based on a fluid flow model of the network and is presented in Appendix A together with the properties of the algorithm. Here we present a discrete time implementation of the algorithm.

$$\hat{N}(k+1) = Pr[\hat{N}(k) + \frac{\gamma[y(k) - \hat{N}(k)p(k)]p(k)}{1+p^2(k)}], \quad \hat{N}(0) = 10 \tag{8}$$

where the projection operator Pr[.] is defined as follows:

$$Pr[x] = \begin{cases} x & \text{if } x > 1 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

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.

IV. 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. These objectives have been outlined in section II. In this section, we demonstrate through simulations that ACP satisfies these objectives to a very good extent. We also conduct a comparative study and demonstrate how ACP fixes the performance problems encountered by XCP and RCP. We conduct our simulations on the ns-2 simulator. In our simulations we mainly consider bulk data transfers but we also evaluate the performance of the protocol in the presence of short web like flows.

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 ACP with respect to changing link bandwidths, propagation delays and number of users utilizing the network.

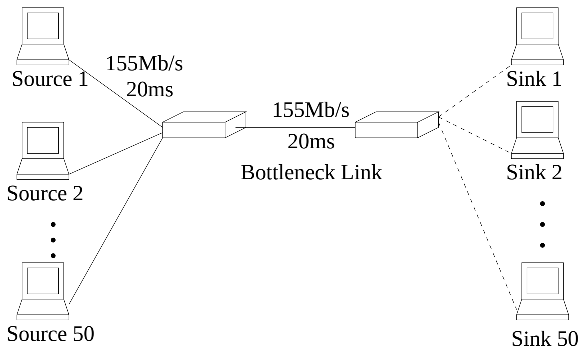


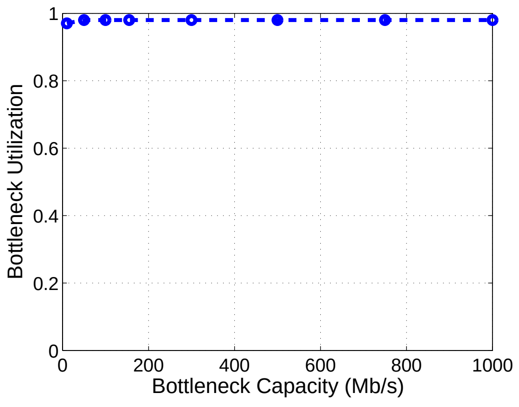
Fig. 2. Single bottleneck link topology used to investigate the scalability of ACP with respect to changing link capacities, delays and number of users.

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 their propagation delay is set equal to 20msec. As mentioned above, the purpose of this study is to investigate the scalability of ACP with respect to changing bandwidths, delays and number of users utilizing the network. When investigating the scalability of the protocol with respect to a particular parameter, we fix the other parameters to the values of the basic setup and we evaluate the performance of the protocol as we change the parameter under investigation. 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, is the average utilization of the bottleneck link and the queue size of the buffer at the bottleneck link. We consider two measures for the queue size: the average queue size and the equilibrium queue size. The *average queue size* is calculated over the entire duration of the simulation and thus contains information about the transient behavior of the system. The *equilibrium queue size* is calculated by averaging the queue length 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 in later sections.

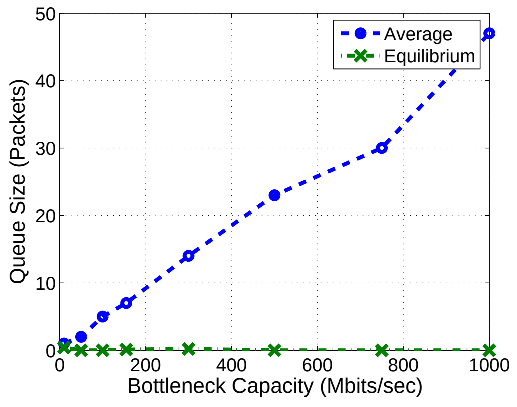
In our simulations, we consider persistent FTP sources. The packet size is equal to 1000 bytes and the buffer size of all links is set equal to the bandwidth delay product. The simulation time is not constant. It varies depending on the round trip propagation delay. We simulate for a sufficiently long 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 ACP 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.

We observe that ACP scales well with increasing bandwidths. The protocol achieves high network utilization ($\approx 98\%$) at all bandwidths. Moreover, the queue size always converges to an equilibrium value which is close to zero. The average queue size remains very small but we do observe an increasing pattern. The reason for this, becomes apparent when we investigate the transient properties of the protocol. In the



(a) Utilization vs Capacity

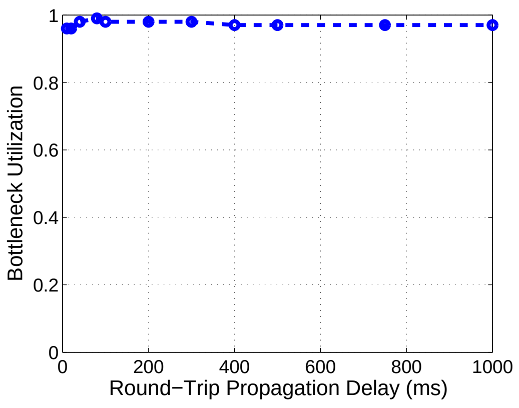


(b) Average Queue Size vs Capacity

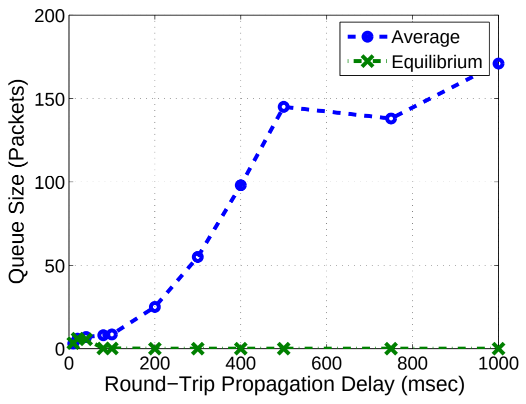
Fig. 3. ACP achieves high network utilization and experiences no drops as the capacity increases. The average queue size increases with increasing capacity due to larger instantaneous queue sizes in the transient period. However, at all capacities, the queue size at equilibrium is close to zero.

transient period, during which the users gradually enter the network, the queue size at the bottleneck link experiences an instantaneous overshoot, before settling down to a value which is close to zero. As the bandwidth increases the maximum value of this overshoot increases, thus causing the average queue size to increase as well. However, in all cases the queue size at equilibrium is small as required.

Effect of Delays: We then investigate the performance of ACP 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. 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

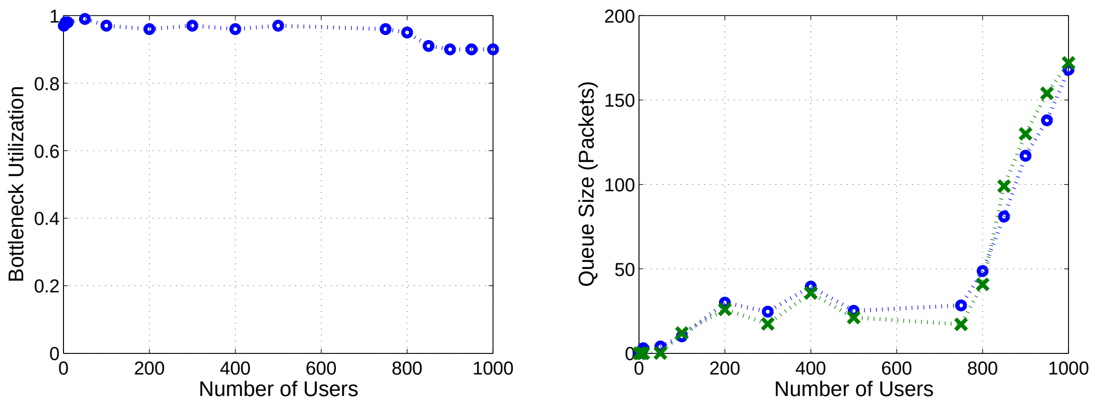


(b) Average Queue Size vs Delay

Fig. 4. ACP achieves high network utilization and experiences no drops as the round trip propagation delay increases. The average queue size increases with increasing propagation delay due to larger instantaneous queue sizes in the transient period. However, at all delays, the queue size at equilibrium is close to zero.

The results are similar to the results obtained when investigating the effect of changing capacities. Fig. 4 (a) demonstrates that the protocol achieves high network utilization at all delays. The equilibrium queue size remains very small, however, the average queue size increases. This trend, as in the case of capacities, is due to the increasing instantaneous queue size in the transient period. As the propagation delays increase, the maximum of the overshoot observed in the transient period increases, thus causing an increase in the average queue size. Although, the average queue size increases the queue size at equilibrium is close to zero as required.

Effect of the Number of Users We finally investigate the performance of ACP 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.



(a) Utilization vs Number of Users (b) Average Queue Size vs Number of Users

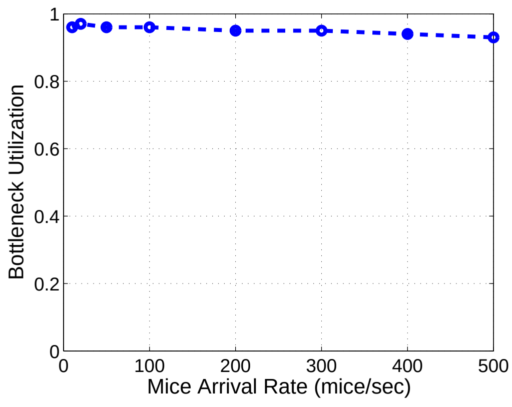
Fig. 5. ACP achieves high network utilization and experiences no packet drops as the number of users increases. At high number of users, the utilization drops slightly and the average queue size increases. The reason is that the fair congestion window is small (close to 1). Since the congestion window can only take integer values both the utilization and queue size oscillate thus causing a slight degradation in performance.

We observe that up to approximately 800 users the protocol satisfies the control objectives as it achieves high network utilization and small queue sizes. However, unlike the previous two cases, the equilibrium queue size is not close to zero. It exhibits similar behavior to the behavior of the average queue size. The reason for this is that as the number of users increases the queue size experiences oscillations. These oscillations dominate the overshoots observed during the transient period and so the equilibrium queue size calculated is very close to the average queue size. The oscillatory behavior at equilibrium is caused by the fact that the congestion window can only take integer values. When the fair congestion window is not an integer (which is the common case), the desired sending at the link is forced to oscillate about the equilibrium value, thus causing oscillations of the input date rate and the queue size. As the number of users increases, these oscillations grow in amplitude and at some point they cause a significant degradation in performance. We observe in Fig. 5 that when the network is utilized by more than 800 users, the utilization drops to about 90% and the average queue size increases. The reason is that at such a high number of users, the fair congestion window is close to 1. Since the congestion window can only take integer values, it oscillates between 1 and 2. These oscillations of the congestion window cause both the utilization and the queue size to oscillate. This behavior causes a decrease in the observed average utilization and an increase in the observed average and equilibrium queue size.

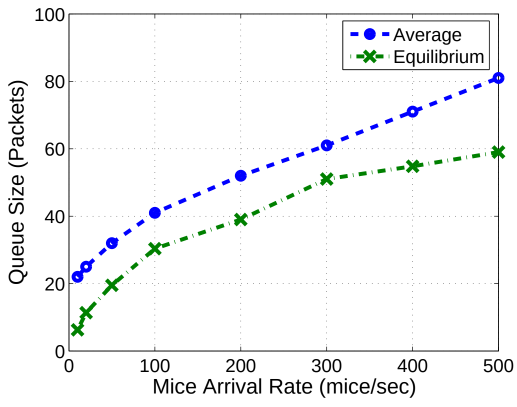
B. Performance in the presence of short flows

In our performance analysis so far we have only considered persistent FTP flows which generate bulk data transfers. Internet traffic, however, consists of both short and long flows. The set of flows is dominated by a relatively few elephants (long flows) and a very large number of mice (short flows). Elephants, although smaller in number, account for the biggest percentage of the network traffic. Short flows account for a smaller percentage which however, cannot be ignored. In this section, we evaluate the performance of ACP in the presence of short web like flows.

We consider the single bottleneck link network shown in Fig. 2. The bandwidth of each link is set equal to 155Mbits/sec and the round trip propagation delay is equal to 80msec. 50 persistent FTP flows share the single bottleneck link with short web like flows. Short flows arrive according to a Poisson process. We conduct a number of tests where we change to mean of this arrival process to emulate different traffic loads. The transfer size is derived from a Pareto distribution with an average of 30 packets. The shape of this distribution is set to 1.35.



(a) Utilization vs mice arrival rate



(b) Average queue size vs mice arrival rate

Fig. 6. ACP achieves high network utilization and maintains small queue sizes as the arrival rate of short web-like flows increases. Note that 500 users per second corresponds to a link load of 75%. In simulations, the transfer size of short flows is derived from a Pareto distribution with an average of 30 packets and a shape factor equal to 1.35

In Fig. 6 we show plots of the utilization and the average queue size at the bottleneck link versus the mean arrival rate of the short flows. We observe that as we increase the arrival rate, the utilization drops slightly whereas both the average queue size and the equilibrium queue size increase. The important thing is that the queue size remains small and no packet drops are observed. It must be noted that 500 users per second corresponds to a link load of 75%. Experiments have shown that short flows account for about 20% of the traffic. In this regime, the utilization recorded at the bottleneck link is 96% which is satisfactory.

C. Fairness

Our objective in this work has been to develop a congestion control protocol which at equilibrium achieves max-min fairness. In this section we investigate the effectiveness of ACP to achieve max-min fairness in a scenario where the max-min fair sending rates change dynamically due to changes in the network load.

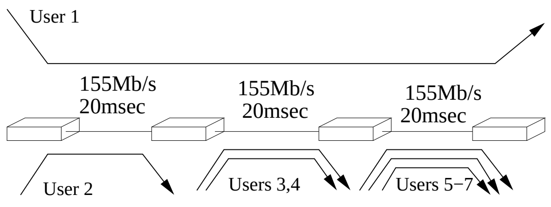


Fig. 7. A three link network used to investigate the ability of ACP to achieve max-min fairness. The first two users utilize the network throughout the simulation, users 2 and 3 start sending data at 20 seconds and users 5-7 start sending data at 40 seconds.

We consider the three link network shown in Fig. 7. The bandwidth of each link is set equal to 155Mbits/s and the propagation delay of each link is set equal to 20msec. 7 users utilize the network at different time intervals. At the beginning only users 1 and 2 utilize the network. The path of the first user traverses all three links while the path of the second user traverses the first link only. During the time that only these two users are active, the first link is the bottleneck link of the network and the fair sending rate for the two links is 77.5Mbits/s. At 20 seconds users 3 and 4 enter the network. Both users traverse the second link which becomes the bottleneck link for users 1, 3 and 4. User 2 is still bottlenecked at the first link since this is the only link that it utilizes. Note that at 20 seconds, user 2 increases its window to take up the slack created by user 1 sharing the bandwidth of link 2 with the other 2 users. At 40 seconds users 5-7 start sending data through the third link which now becomes the bottleneck link for users 1, 5, 6 and 7. User 2 is bottlenecked at the first link whereas users 3 and 4 are still bottlenecked at the second link.

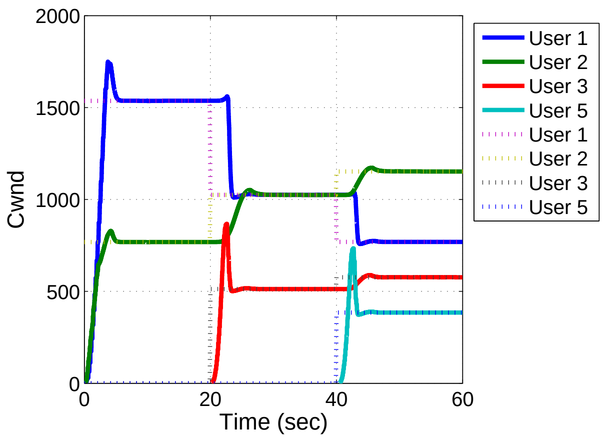


Fig. 8. Time response of the congestion window of a representative number of users compared with the theoretical max-min values. The theoretical values are denoted by dotted lines.

In Fig. 8 we show the time responses of the congestion window of a representative number of users. These responses are compared with the theoretical max-min allocation values at each time. The actual responses are denoted by solid lines whereas the theoretical values are denoted by dotted lines. We observe that at equilibrium, the actual values match exactly the theoretical values which implies that max-min fairness is achieved at all times. One thing to notice, is that during the first 20 seconds, the congestion windows of users 1 and 2 are different, despite the fact that their theoretical max-min sending rates in this period are the same. There is no inconsistency between the two observations. The two users experience different round trip propagation delays as they travel different number of hops. Although their sending rates are identical, the different round trip times generate different congestion windows. This demonstrates the ability of ACP to achieve fairness in the presence of flows with different round trip times and number of hops. Also note that the response of user 4 equals the response of user 3 and the response of users 6 and 7 are equal to the response of user 5 and are thus not shown.

Another interesting observation is the overshoot in the response of user 3. This is a result of the second link becoming a bottleneck link only when users 3 and 4 enter the network. During the time that only users 1 and 2 utilize the network, the two users are bottlenecked at the first link, and so the input data rate in the second link is consistently less than the capacity. This causes the algorithm which updates the desired sending rate at the link, to consistently increase the desired sending rate. Basically, the link asks for more data, the users do not comply because they are bottlenecked elsewhere and the link reacts by asking for even more data. The desired sending rate, however, does not increase indefinitely. A projection operator in the link algorithm causes the desired sending rate at the second link to converge to the link capacity. When users 3 and 4 enter the network the second link becomes their bottleneck link. Their sending rate thus becomes equal to the desired sending rate computed at the link. Since the desired sending rate is originally equal to the link capacity, the congestion windows of the two users experience an overshoot before settling down to their equilibrium value. This can be observed in Fig. 8. Despite this overshoot the system does not experience any packet drops. The above setting can be used to emulate the case where network users cannot comply with the network's request because they do not have enough data to send. The above shows the ability of ACP to also cope with this case.

D. The Dynamics of ACP

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 evaluate the transient behavior of ACP, we consider the single bottleneck link network shown in Fig. 2 and we generate a dynamic environment where users enter and leave the network at different times. In such an environment, we investigate the dynamics of the user sending rates, we examine the queuing dynamics at the bottleneck link and we also evaluate the performance of the estimator which is used to track the number of users utilizing the network.

To conduct our study we consider the following scenario. 30 users originally utilize the single bottleneck link network shown in Fig. 2. At 30 seconds 20 of theses users stop sending data 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.

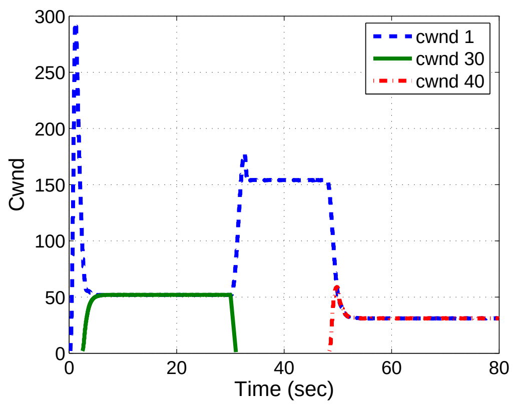


Fig. 9. Time response of the congestion window of three users. User 1 utilizes the network throughout the simulation, user 30 stops sending data at 30 seconds and user 40 enters the network at 45 seconds. We observe smooth and fast responses with no oscillations.

In Fig. 9 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 9. 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 9 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. This slight overshoot is caused by the feedback delays and the pure integral action of the congestion controller. It can be avoided by introducing proportional action. However, such a modification would increase the complexity of the algorithm without significantly improving the performance and is thus avoided. 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. 9 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. 10 we show the time responses of the utilization and the queue size at the bottleneck link.

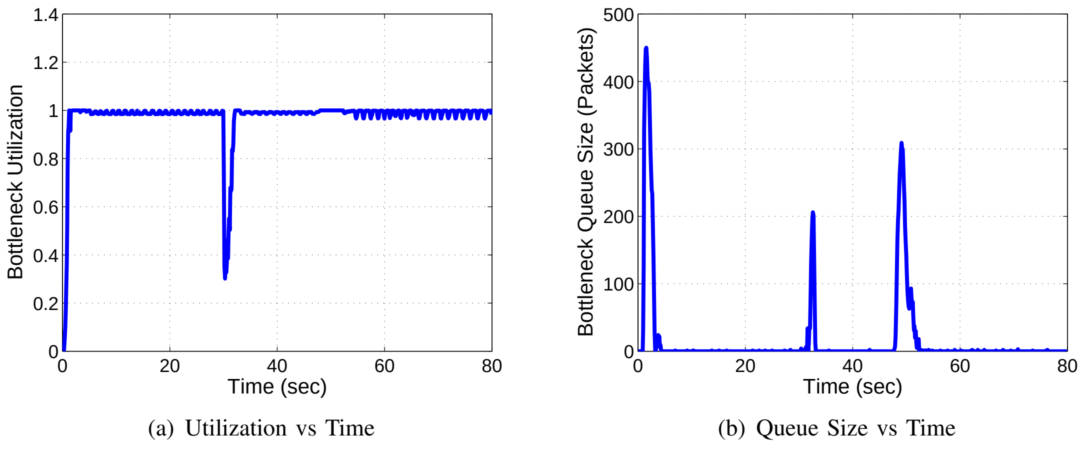


Fig. 10. Time response of the instantaneous utilization and the queue size at the bottleneck link. Utilization converges fast to a value which is close to 1. There is an instantaneous drop when the 20 users leave the network but the protocol manages to recover quickly. The queue size experiences instantaneous increases when new users enter the network but at equilibrium the queue size is almost zero.

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 0. This is what is required by the congestion control protocol in order to avoid excessive queueing delays on the long run. 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. It must be noted that the maximum queue size recorded in the transient period, increases as the bandwidth delay product increases. This is why in our study of the scalability properties of ACP, the average queue size increases as we increase the bandwidths and the delays. 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.

A distinct feature of the proposed congestion control strategy is the implementation at each link of an estimation algorithm which estimates the number of flows utilizing the link. These estimates are required to maintain stability in the presence of delays. Here, we evaluate the performance of the proposed estimation algorithm. In the scenario that we have described in the previous subsection, the number of users utilizing the single bottleneck link network changes from 30 to 10 at 30 seconds and it becomes 50 at 45 seconds. So, we evaluate the performance of the proposed estimation algorithm by investigating how well the estimator tracks these changes. In Fig. 11 we show the time response of the output of the estimator.

We observe that the estimator generates smooth responses with no overshoots or oscillations. In addition, the estimator tracks the changes in the number of users and produces correct estimates at equilibrium.

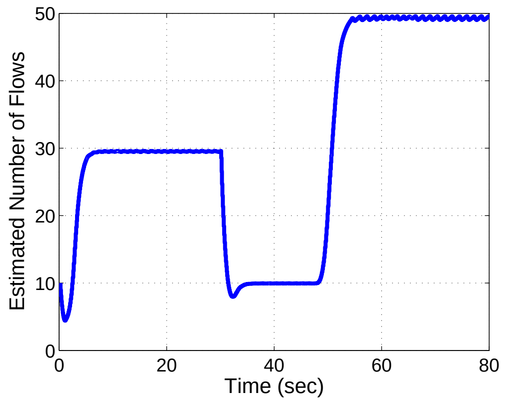


Fig. 11. Time response of the estimated number of users utilizing the bottleneck link. We observe almost perfect tracking at equilibrium and fast responses with no overshoots.

E. *A multi-link example*

Until now we have evaluated the performance of ACP in simple network topologies which include 1, 2 or 3 links. Our objective in this section is to investigate how ACP performs in a more complex network topology. We consider the parking lot topology shown in Fig. 12.

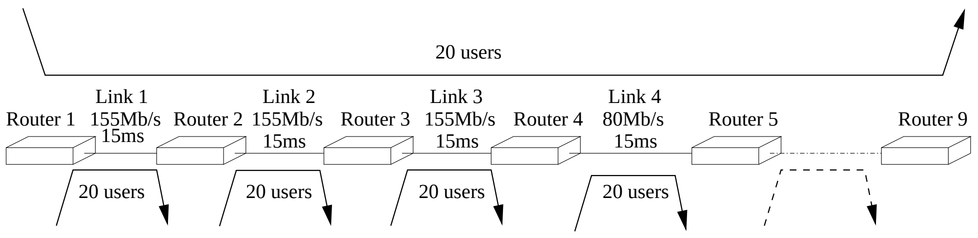


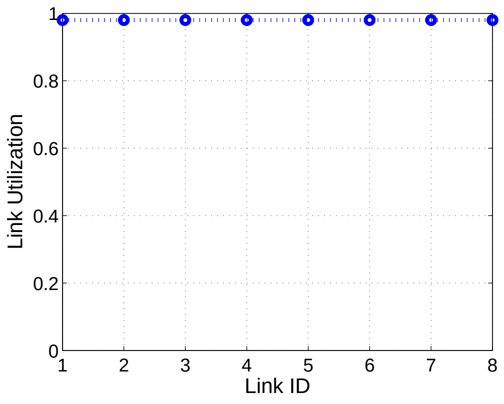
Fig. 12. A parking lot network topology

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. 12. 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. We evaluate the performance of ACP by examining the utilization and the average queue size observed at each link. We do not report packets drops, as we do not observe any. In fig. 13, we show on separate graphs the utilization achieved at each link and the average and equilibrium queue size recorded at the link.

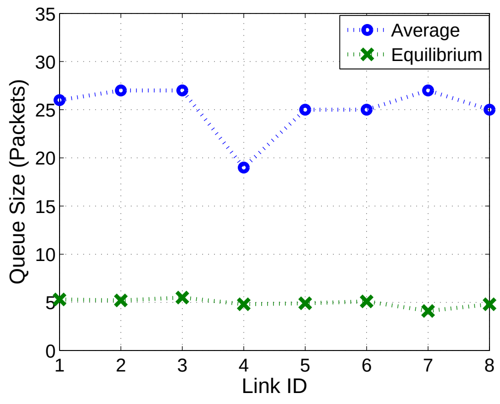
Since all links in the network are bottleneck links for some flows, we do expect them to be fully utilized. Indeed, we observe that ACP achieves almost full utilization at all links. In addition, both the equilibrium queue size and the average queue size remain small. At link 4 we observe smaller average queue size. This is due to its smaller bandwidth delay product. This is consistent with our observations in previous sections.

F. *Comparison with XCP*

Our objective in this work has been to develop a congestion control protocol which does not require maintenance of per flow states within the network and satisfies all the design objectives. An explicit congestion control protocol (XCP) which has been recently developed in [9], satisfies most of the design objectives but fails to achieve max-min fairness in the case of multiple congested links. It has been shown



(a) Utilization at each link



(b) Queue at each link

Fig. 13. ACP achieves high utilization at all links and experiences no packet drops. In addition it manages to maintain small queue sizes.

through analysis and simulations that when the majority of flows at a particular link are bottlenecked elsewhere, the remaining flows do not make efficient use of the residual bandwidth ([15]). In this section, we consider a topology where the above problem is evident and we demonstrate that ACP fixes this problem and achieves max-min fairness.

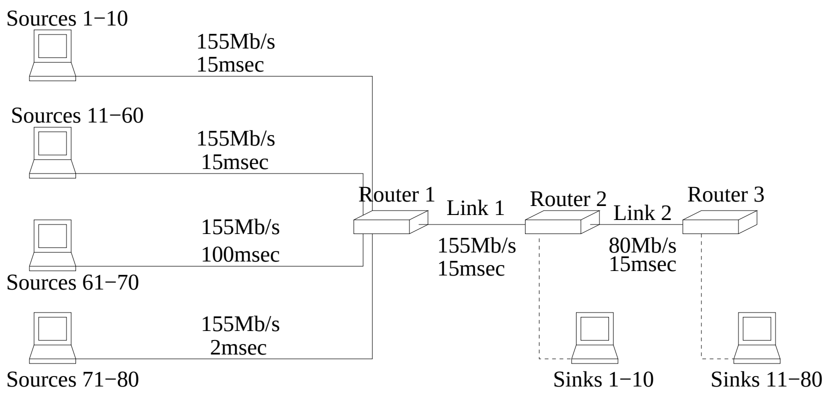


Fig. 14. A two link network, used to investigate the ability of ACP to achieve max-min fairness at equilibrium. We consider a simulation scenario which involves users with heterogeneous round-trip times.

We consider the two link network shown in Fig. 14. Link 1 has a bandwidth of 155Mbits/sec whereas link 2 has a bandwidth of 80Mbits/sec. 80 users access the network though 155Mbits/sec access links. The access links of the first 60 users have a propagation delay of 15msec, the access links of the next 10 users have a propagation delay of 100msec and the propagation delay of the last 10 users are set to 2msec. We have chosen a variety of propagation delays to investigate the ability of ACP to achieve fairness in the presence of flows with multiple round trip times. The first 10 users of the network have connection sinks at the first router and the rest of the users have connection sinks at the second router. This has been done to ensure that both links are bottleneck links for some flows. The first 10 users are bottlenecked at link 1 whereas the remaining users are bottlenecked at link 2.

We simulate the above scenario using both XCP and ACP users. In table I we compare the theoretical max-min congestion window values with the equilibrium values achieved by ACP and XCP. We observe that ACP matches exactly the theoretical values, whereas XCP does not. XCP fails to assign max-min sending rates to the first 10 users which utilize link 1 only. This is consistent with the findings in [15]. The other users traversing link 1 are bottlenecked at link 2 and so the 10 users which are bottlenecked at link 1 do not make efficient use of the available bandwidth. This inefficiency causes underutilization of link 1. This is demonstrated in Fig. 15 where we plot the time response of the utilization achieved at link 1 by

TABLE I

THEORETICAL MAX-MIN FAIR VALUES, COMPARED WITH THE EQUILIBRIUM VALUES ACHIEVED BY ACP AND XCP

Users	Round-Trip Time (ms)	Max-Min Bandwidth (Mb/s)	Max-Min Cwnd	ACP Cwnd	XCP Cwnd
1-10	60	7.5	56	56	40
11-60	90	1.14	13	13	13
61-70	260	1.14	37	37	37
71-80	62	1.14	9	9	9

the ACP and the XCP users. Obviously XCP causes underutilization of the link, whereas ACP achieves almost full utilization of the link at equilibrium. This example demonstrates that ACP outperforms XCP in both utilization and fairness. Another thing to note in table I is the ability of ACP to achieve max-min fairness despite the presence of flows with a variety of round trip times.

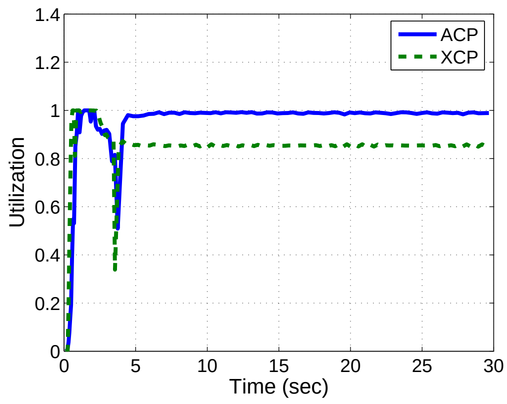


Fig. 15. Time response of the utilization at the first link achieved by ACP and XCP. We observe that ACP achieves higher utilization.

V. COMPARISON WITH RCP

RCP and ACP were developed independently based on similar design principles. However, the design objectives of the two protocols are different. The main objective of RCP is to minimize the duration of the network flows whereas the main objective of ACP is to optimize network centric performance metrics such as fairness, utilization, queue sizes and packet drops. Although RCP and ACP were motivated by the same design ideas, they implement different algorithms at both the sources and the links. At each source RCP applies a rather aggressive increase policy where it immediately adopts the desired sending rate received from the network as the current sending rate of the source. This is done to ensure that flows with small file sizes finish their sessions quickly. However, such an aggressive increase policy, especially for new users, must be accompanied by an aggressive decrease policy in the case of congestion, to avoid packet losses. However, as we will see later such an aggressive decrease policy can cause RCP to underutilize the network for a significant time period. ACP on the other hand, applies a more conservative policy both when increasing and when decreasing the source sending rate. This conservative policy ensures no packet losses and high network utilization. However, it does take several round trip times for each source to converge to its max-min fair sending rate and this can cause larger duration of flows with small file sizes.

RCP and ACP also have fundamental differences in the implementation of the algorithm which updates the desired sending rate. RCP implements a non-linear congestion controller whereas ACP implements a certainty equivalent controller. The properties of the RCP controller have been established by linearizing the non-linear equations in a small neighborhood about the stable equilibrium point. However, the linear model is a poor approximation of the non-linear model in some regions of the state space. These model inaccuracies can cause the RCP algorithm to deviate significantly from the predicted behavior and perform poorly in some scenarios. Specifically, when the desired sending rate experiences a large undershoot, the controller is very slow in recovering thus causing underutilization of the network for large time intervals. ACP on the other hand, implements a certainty equivalent controller at each link. The controller is designed assuming that the number of users utilizing the link is known. In practice the latter is an unknown time varying parameter. We utilize online parameter identification techniques to estimate this parameter online. We then replace the known parameter in the control algorithm with its estimate to yield the certainty equivalent controller.

RCP performs poorly when the network experiences sudden changes in the traffic load. Such sudden changes can cause RCP to underutilize the network for significant time periods. In this section we demonstrate this behavior of RCP and we show that ACP continues to perform well in the scenarios where RCP performs poorly. We consider the single bottleneck link network of Fig. 2. The bandwidth of each link is set to 155Mbit/sec and the round trip propagation delay is set equal to 80msec. The network is initially utilized by only one user. At 15 seconds a second user enters the network. This represents a 100% increase in the traffic load at the bottleneck link. We simulate both ACP and RCP networks.

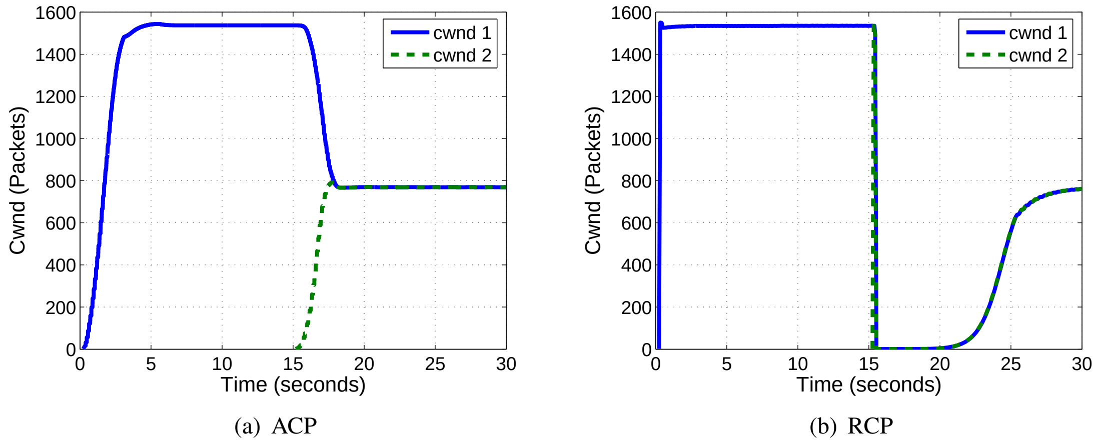


Fig. 16. Time responses of the congestion window of the network users for ACP and RCP. Observe that the second RCP user converges very slowly to its equilibrium value.

In Fig. 16 we show the time responses of the congestion window of users 1 and 2 for ACP and RCP. We observe that ACP generates smooth responses which gradually converge to their equilibrium values. The sending rate of user 1 converges to the bandwidth of the bottleneck link and then gradually decreases to half of this value when user 2 enters the network. It takes several round trip times for the congestion windows to converge to their equilibrium values. RCP on the other hand adopts a more aggressive response policy. Note how quickly user 1 originally converges to its equilibrium value. However, when user 2 enters the network its sending rate is set equal to the sending rate of user 1. This causes excessive queue sizes at the bottleneck link. The aggressive decrease policy which RCP adopts, then causes the desired sending rate calculated at the link to decrease to the minimum value allowed by the control algorithm, which is one packet. When this happens, the desired sending rate does not recover quickly. It remains close to 1 for approximately 5 seconds and converges to the equilibrium value in 15 seconds. This slow response is a result of the nonlinear control algorithm which RCP utilizes to calculate the desired sending rate. The nonlinearity causes slow responses when the desired rate experiences large undershoots. This problem is exacerbated as we increase the link bandwidths. This behavior of RCP can cause underutlization of the network in significant time periods. In Fig. 17 we show the time responses of the utilization of the bottleneck link achieved by ACP and RCP.

We observe that RCP underutilizes the network for a significant amount of time when the second user enters the network, whereas ACP achieves almost full utilization in that period.

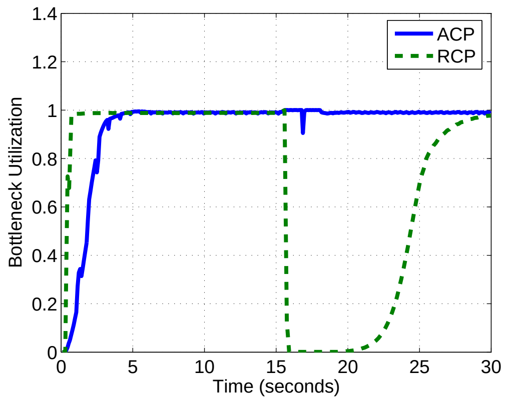


Fig. 17. Time response of the utilization achieved at the bottleneck link by ACP and RCP. We observe that RCP underutilizes the network for a significant amount of time when the second user enters the network, whereas ACP achieves almost full utilization in that period.

VI. Conclusions

Our main contribution in this paper is to develop an Adaptive Congestion control Protocol (ACP) which is shown through simulations and analysis to satisfy all the design requirements as outlined in the paper and thus outperform previous TCP proposals. ACP is a window based protocol which does not require maintenance of per flow states within the network. It utilizes an explicit multi-bit feedback signalling scheme to convey congestion information from the network to the end users and vice versa. A distinct feature of the protocol is the implementation at each link of an estimation algorithm which is derived using on line parameter identification techniques. The algorithm generates estimates of the number of users utilizing the link which are used to tune the control parameters in order to maintain stability. This feature, enables the protocol to adapt to dynamically changing network conditions. 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, small queue sizes 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. Apart from its practical significance, this work also demonstrates the effectiveness of formal control techniques in general and adaptive control techniques in particular in delivering efficient solutions in a highly complex networked system such as the Internet. Our next objective is to verify the properties of ACP analytically in networks of arbitrary topology.

Appendix

A. Derivation of the Estimation Algorithm

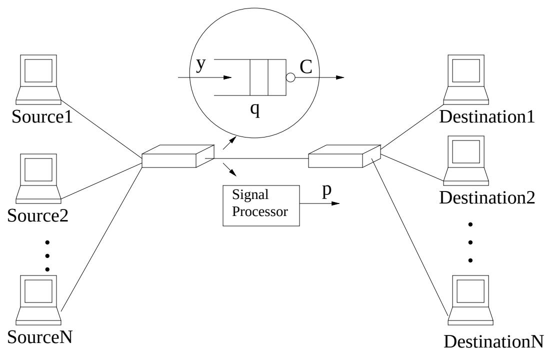


Fig. 18. Single bottleneck link network used for analysis.

We consider the single bottleneck link network shown in Fig. 18. It consists of N users, which share a common bottleneck link through high bandwidth access links. At the bottleneck link we assume that there exists a buffer, which accommodates the incoming packets. The rate of data entering the buffer is denoted by y, the queue size is denoted by q and the output capacity is denoted by C. At the bottleneck link, we implement a signal processor, which calculates the desired sending rate p. This information is communicated to the network users, which set their sending rate equal to p. We ignore time delays and the queuing dynamics and we assume that the desired sending rate p is larger than some positive constant λ. This is true in practice because the congestion window of the users is never allowed to be less than one packet. Since the sending rate of all users is equal to p, the input data rate is given by the following equation.

$$y = Np \tag{10}$$

The number of users N is the unknown parameter that needs to be estimated. The input data rate y and the desired sending rate p can be measured at the bottleneck link and so equation (10) constitutes a linear static parametric model of the unknown parameter. Using online parameter identification techniques [26], we derive the following estimation algorithm:

$$\epsilon = \frac{y - \hat{N}p}{m_s^2} \tag{11}$$

$$\dot{\hat{N}} = Pr[\gamma \epsilon p], \quad \hat{N}(0) = N_0 \geq 1 \tag{12}$$

where $\hat{N}$ is the number of flows estimated online, $m_s^2 = 1 + p^2$ and the projection operator is defined as follows:

$$\dot{x} = Pr[\alpha] = \begin{cases} \alpha & \text{if } x > 1 \\ \alpha & \text{if } x = 1 \text{ and } \alpha \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

We use the projection operator since we know priori that the the number of users cannot be less than 1. The properties of the above estimation algorithm are summarized in the following theorem:

Theorem 1: Equations (10)-(12) guarantee that $\hat{N}(t)$ converges exponentially fast to the unknown parameter N.

Proof: When $N = 0$, no data is traversing the link and $\hat{N}$ is not updated. So, we only consider values of N greater or equal to 1. When $\hat{N} = 1$, $\dot{\hat{N}} \geq 0$. Since we choose the initial condition of $\hat{N}$ to be greater than 1, the above implies that $\hat{N}$ cannot become less than 1. We can thus ignore the projection operator in our analysis.

Substituting equations (10) and (11) in equation (12), we can establish that:

$$\frac{d\tilde{N}}{dt} = -\gamma \tilde{N} \frac{p^2}{1 + p^2} \tag{14}$$

where $\tilde{N} = \hat{N} - N$. We can solve the above equation to get:

$$\tilde{N}(t) = \tilde{N}_0 e^{-\gamma \int_0^t \frac{p^2}{1+p^2} dt} \leq \tilde{N}_0 e^{-\gamma \frac{\lambda^2}{1+\lambda^2} t} \tag{15}$$

The above equation establishes that $\tilde{N}(t)$ converges exponentially fast to 0 which implies that $\hat{N}(t)$ converges exponentially fast to the desired parameter N.

■

B. ACP Stability Analysis

We consider the network model described in Appendix A. In this section we also account for the link propagation delays and the queuing dynamics. We assume that all network users have the same round-trip propagation delay which we denote by τ. Assuming that the sending rate of all users is equal to the same delayed value of the desired sending rate, the input data rate at the link is given by the following equation:

$$y = Np(t-\tau) \tag{16}$$

To account for the queuing dynamics we model the queue as a simple integrator with saturation as follows:

$$\dot{q} = \begin{cases} y - C & \text{if } y - C \geq 0 \\ y - C & \text{if } y - C < 0 \text{ and } q > 0 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

We assume that the estimation algorithm at the bottleneck link generates accurate estimates of the number of users utilizing the link. If in addition, we ignore the projection operator which bounds the desired sending rate, a continuous time version of the link side algorithm is as follows:

$$\dot{p} = \frac{1}{N}[\frac{k_i}{\tau}(C-y) - \frac{k_q}{\tau^2}q] \tag{18}$$

We define the variable $x(t) = y(t) - C$. We substitute the latter in equations (16)-(18) to obtain the following set of differential equations:

$$\dot{x} = -\frac{k_i}{\tau}x(t-\tau) - \frac{k_q}{\tau^2}q(t-\tau), \quad x(0) = x_0 \tag{19}$$

$$\dot{q} = \begin{cases} x & \text{if } x \geq 0 \\ x & \text{if } x < 0 \text{ and } q > 0 \\ 0 & \text{otherwise} \end{cases}, \quad q(0) = q_0 \tag{20}$$

These equations describe the dynamics of the ACP system and have been also used to describe XCP ([9]) and RCP ([14]). In [9], the authors linearize equation (20) and use Nyquist analysis, to obtain a set of values for k_i and k_q which guarantee that the linear system is stable. The values that we have chosen for ACP lie outside this set. Linear analysis thus predicts that ACP is unstable. Extensive simulations, however, demonstrate that ACP is stable. In this section we show using phase plane analysis, that if we describe ACP using the non-linear equations (19)-(20), the system is stable.

A phase portrait is a plot that shows a representative sample of trajectories for a given system. For the ACP system, we generate trajectories by simulating on Matlab equations (19) and (20) . We use the phase portraits that we obtain to demonstrate that the ACP system is stable for a variety of initial conditions x_0, q_0 and round trip propagation delays τ. We first investigate the stability properties of ACP as we change the initial conditions x_0, q_0. We set the propagation delay τ equal to 1 sec and we choose the design parameters k_i and k_q to be equal to 0.1587 and 0.3175 respectively, as in ACP. We consider a family of initial conditions in the range 0-100 and for each initial condition we generate one trajectory. In Fig. 19 we show the phase portrait that we have obtained. We observe that all trajectories converge to the origin which is the unique equilibrium point of the system.

We then investigate the stability properties of the ACP system as we change the time delays. We fix the initial conditions of x and q to 100 and 0 respectively, we set the design parameters k_i and k_q equal to 0.1587 and 0.3175 respectively and we consider values of τ in the range 100-600 msec. For each value of τ we generate a state trajectory. The resulting phase portrait is shown in Fig. 20. We observe that all trajectories converge to the equilibrium point. We also observe that as we increase the delays the maximum value of the queue size increases.

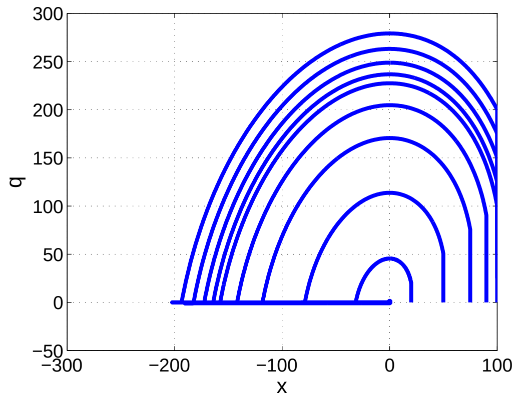


Fig. 19. ACP trajectories when the propagation delay τ is set equal to 1 sec. Each trajectory corresponds to a different set of initial conditions for the state variables. All trajectories converge to the origin.

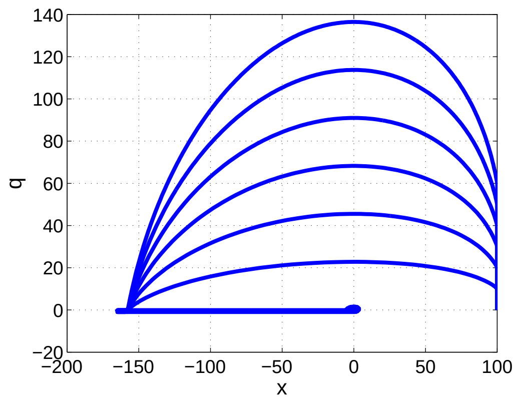


Fig. 20. ACP trajectories for time delays in the range 100-600msec. All trajectories converge to the origin.

We have obtained state trajectories for a much larger set of initial conditions and round trip propagations delays which we do not show here for clarity of presentation. In all cases the ACP system was found to be stable. We thus conjecture that the ACP system described by equations (19) and (20) is globally asymptotically stable for all delays when $k_i = 0.1587$ and $k_q = 0.3175$. Analytical proof of this conjecture using Lyapunov-Krasovskii functionals is the topic of current research.

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] W. Stevens. TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms. *RFC 2001*, January 1997.

[5] K. K. Ramakrishnan and S. Floyd. The addition of explicit congestion notification (ECN) to IP. *RFC 3168*, September 2001.

[6] S. Floyd. TCP and explicit congestion notification. *Comput. Commun. Rev.*, 24(5):10–23, October 1994.

[7] S. Karandikar, S. Kalyanaraman, P.Bagal, and B. Packer. TCP rate control. *Comput. Commun. Rev.*, 30(1):45–58, January 2000.

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

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

[10] A. Karnik and A Kumar. Performance of TCP congestion control with explicit rate feedback. *IEEE/ACM Transactions on Networking*, 13(1):108–120, February 2005.

[11] A. Charny, K. K. Ramakrishnan, and A. Lauck. Time scale analysis and scalability issues for explicit rate allocation in ATM networks. *IEEE/ACM Transactions on Networking*, 4(4):569–581, August 1996.

[12] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan. An efficient rate allocation algorithm for ATM networks providing max-min fairness. In *Proc. 6th IFIP Int. Conf. High Performance Networking, HPN'95*, Spain, September 1995.

[13] A. Arumbalam, X. Chen, and N. Ansari. An intelligent explicit rate control algorithm for ABR service in ATM netowks. In *Proc. ICC'95*, volume 1, pages 200–204, Montreal, June 1997.

[14] 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.

[15] S. H. Low, L. L. H. Andrew, and B. P. Wydrowski. Understanding XCP: Equilibrium and fairness. In *Proc. IEEE INFOCOM*, volume 2, pages 1025–1036, March 2005.

[16] 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.

[17] C. E. Rohrs and R.A. Berry. A linear control approach to explicit rate feedback in ATM networks. In *Proc. IEEE INFOCOM'97*, volume 1, pages 277–282, March 1997.

[18] T. Basar, E. Altman, and R. Srikant. A distributed globally convergent algorithm for fair, queue-length-based congestion control. In *Proc. IEEE Conf. Decision and Control*, volume 1, pages 622–627, December 2001.

[19] Y. Zhao, S. Q. Li, and S. Sigarto. A linear dynamic model for design of stable explicit-rate ABR control systems. In *Proc. IEEE INFOCOM'97*, volume 1, pages 283–292, March 1997.

[20] E. Altman, T. Basar, and R. Srikant. Robust rate control for ABR sources. In *Proc. IEEE INFOCOM'98*, volume 1, pages 166–173, March 1998.

[21] C. Fulton, S. Li, and C. S. Lim. An ABR feedback control scheme with tracking. In *Proc. IEEE INFOCOM'97*, volume 2, pages 805–814, April 1997.

[22] M. K. Wong and F. Bonomi. A novel explicit rate congestion cotnrol algortithm. In *Proc. IEEE GLOBECOM'98*, volume 4, pages 2432–2439, November 1998.

[23] Y. Zhang, D. Leonard, and D. Loguinov. Jetmax: Scalable max-min congestion control for high-speed heterogeneous networks. In *Proc. IEEE INFOCOM*, April 2006.

[24] Paganini F., Z. Wan, Doyle J.C, and Low S. H. Congestion control for high performance stability and fairness in general networks. *IEEE/ACM Transactions on Networking*, 13(1):43–56, February 2005.

[25] F. Paganini. On the stability of optimization-based flow control. In *Proceedings of 2001 American Control Conference.*, volume 6, pages 4689 –4694, 2001.

[26] P. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice Hall, 1996.