Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Congestion control for small queues: analysis and evaluation of a new protocol

Niels Möller Karl Henrik Johansson

Abstract— **A new congestion control protocol is presented, analyzed and experimentally evaluated. It consists of the standard inner-loop ACK-clock and a novel outer-loop adjusting the window size based on congestion signaling from the network. The aim of the new protocol is to maintain the efficiency and fairness properties of TCP, but with significantly smaller bottleneck queues and thereby it takes the sharing with real-time traffic into account. Stability properties of the protocol is proved using a recent fluid-flow traffic model. Experimental comparisons with New Reno and Vegas illustrate the advantages of the new protocol with respect to throughput, delay, utilization, and fairness.**

I. Introduction

To a first approximation, Internet traffic can be divided into two types: TCP traffic and real-time traffic. The TCP traffic includes services such as web-browsing and peer-to-peer file sharing, and it is dominating the current Internet. In a recent study of the traffic mix in Deutsche Telekom's ADSL network [1], more than 90% of the traffic was using TCP, and a significant proportion thereof was peer-to-peer file-sharing. Real-time traffic includes services such as voice over IP and online gaming. The quality of real time services degrade severely if they share a bottleneck link with TCP traffic, which is based on additive–increase multiplicative–decrease (AIMD), since TCP makes the queues grow and thereby causes a queuing delay on the order of 200 ms.

The main objectives of congestion control are to avoid network overload, ensure that bottlenecks are fully utilized, share resources between flows in a fair manner, and react to changes in network load as well as in the network topology. TCP New Reno usually achieves these objectives, with some well-known utilization problems for network paths with very large capacity or delay, and for wireless links that are characterized by a significant packet loss probability unrelated to congestion, or highly variable delay. To the list of objectives, we add in this paper the requirement that queuing delays should be kept small at all links, including bottlenecks. TCP New Reno does not attempt to minimize the queue size. Bottleneck routers will drop packets due to overflow of the router buffer. Furthermore, the buffer size is usually large, the old rule of thumb for buffer sizing corresponds to a queuing delay on the order of 200 ms when the buffer is close to full [2].

This work was supported by the Swedish Research Council and the Swedish Strategic Research Foundation.

N. Möller and K. H. Johansson are with the ACCESS Linnaeus Centre, School of Electrical Engineering, Royal Institute of Technology, SE-100 44 Stockholm, Sweden. {niels|kallej}@ee.kth.se

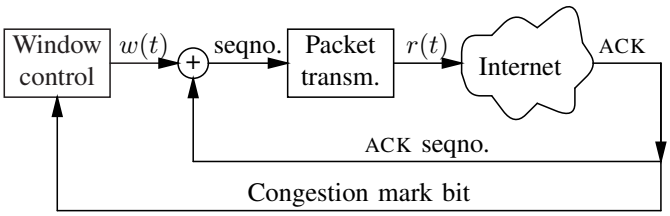

Fig. 1. Control structure for the congestion control protocol. The window size $w(t)$is a control signal which determines the difference between the sequence numbers of transmitted packets (seqno.) and the sequence numbers in received ACKs (ACK seqno.), i.e., the amount of in-flight data. The sending rate $r(t)$ is determined indirectly via the ACK-clock inner-loop.

The main contribution of this paper is a new congestion control mechanism that takes the sharing with real-time traffic into account. We aim for much smaller queues than TCP, resulting in smaller queuing delays and delay fluctuations. In addition, we are able to almost eliminate packet losses due to congestion. We describe the effect that real-time cross traffic has on the queuing dynamics, and take this effect into account when analyzing the stability of the congestion control mechanism.

A simple approach to reducing queuing delay is to reduce the buffer size by one or a few orders of magnitude, compared to the old rule of thumb [3], [4]. Small buffers may however lead to lead to high packet loss rates and high variability in TCP throughput [5]. Another, more sophisticated, approach is to use AQM, where RED is the most widely implemented scheme. Good tuning of the parameters of AQM schemes, and RED in particular, is challenging [6], [7]. Both these approaches work best when there are a large number of TCP flows sharing the bottleneck, and they are less useful when there are only one or a couple of flows, e.g., in a scenario where two users share an ADSL line.

The protocol proposed in this paper uses the cascaded control structure shown in Fig. 1. The signaling structure is the same as when using AQM together with ECN [8]. With the standard ECN mechanism, sources are required to treat the arrival of a congestion mark in the same way as a packet loss, and react by halving the window size. The new protocol uses a softer, additive, update to the window size in response to a congestion mark. Using a feedback signal from the network that is added to the source's window size is similar in spirit to XCP [9], although we use much lighter signaling.

The new protocol is also related to delay-based congestion control, in particular TCP Vegas [10], in that small queue sizes are a design objective. However, delay feedback is used only in the inner-loop, i.e., the ACK-clock, while the outer-

loop adjusts the window size based on explicit feedback from the network. So in the cascaded control structure, the two loops use different feedback signals. Another important difference is that the new protocol has a per-link tuning knob. If a particular link needs a higher average queue size, e.g., due to a traffic mix with larger fluctuations in the arrival rate, or a time varying capacity, the average queue size can be increased by local tuning.

A cascaded control structure is also used in [11]: The controller adjusts the window size, and then the corresponding sending rates and queue sizes are determined by the window sizes. The novelty in this paper is that we take into account the dynamic properties of the relationship between window sizes and queue sizes. The control structure is similar also to that in primal-dual congestion control schemes, although our approach is window-based rather than rate-based. One important difference is that we have two different signals that are fed back from the network to the sources: delay and congestion indication. For this reason, the protocol does not fit easily into the utility maximization framework. This problem is shared with the family of loss-delay based congestion control protocols [12]–[14].

The outline of the paper is as follows. Section II describes the new protocol. A fluid-flow model for the protocol in a single bottleneck scenario is developed in Section III. It is showed in Section IV that the closed-loop model is well-posed and that it has a unique asymptotically stable equilibrium. Fairness is also discussed. Experimental evaluation through `ns2` simulations is presented in Section V, where the performance of the new protocol is compared with TCP New Reno and TCP Vegas. Section VI gives the conclusions and some further work.

II. Protocol design

We are looking for a new congestion control mechanism, which addresses the main shortcoming of TCP, i.e., the large queuing delays at bottlenecks. We intend to have no packet losses in normal operation, and instead rely on congestion feedback generated by the network. The proposed scheme is structurally very similar to standard TCP/AQM, with some subtle but crucial differences in the handling at both end hosts and routers.

The new protocol is window-based. The ACK-clock is a simple and efficient per-packet rule which controls the sending rate. Since this inner-loop is stable, for arbitrary propagation delay in the network [15], [16], we keep the ACK-clock inner-loop, and just modify the window control. This gives the cascaded control structure in Fig. 1.

The proposed protocol is fairly simple. For each received ACK, Δw is added to the current window size, where Δw depends on the congestion mark bit in that ACK:

$$\Delta w := \begin{cases} \dfrac{m^2}{w} & \text{if congestion bit clear} \\ \dfrac{m^2}{w} - m & \text{if congestion bit set} \end{cases}$$

where m is the packet size. Each router sets the congestion mark bit of forwarded packets stochastically with probability

$$p(t) = \frac{q(t)}{q(t) + q_0} \tag{1}$$

where $q(t)$ is the queue size and q_0 a design parameter. In the following we describe the motivation for the additive increase, additive decrease, and congestion feedback in some more detail.

A. Additive increase

The lack of feedback in the absence of congestion, and the desire to have efficient utilization of resources, requires some mechanism that increases the sending rates in the absence of congestion. This growth rate is going to be one of the system parameters, and for stability reasons, it makes sense to scale it down with the RTT. Hence, we keep the additive increase part of TCP in our new protocol: in the absence of congestion, increase the window size by m bytes per RTT.

With the flow analogy in mind, additive increase can be thought of as a pressure applied by the sources, which forces more fluid into the system, filling up the pipes, and when the pipes are full, the pressure causes queues to grow. For an effective congestion control, we must design a way for the queues to impose a back-pressure on the fluid.

B. Additive decrease

With the multiplicative decrease mechanism of TCP New Reno, feedback is a relatively rare event, and the response to each feedback event is strong.

The long interval between feedback events is a consequence of the AIMD mechanism, and the average interval is long also when using AQM and ECN in routers. We aim for more frequent signalling in the new protocol, with an average of one feedback event per RTT. To get there, we need a softer response to each event: For each received ACK carrying a congestion indication, the window size is decreased by one packet. With this rule, senders will no longer operate in open-loop for long time periods.

Under the one-packet decrease rule, when a router decides to set a mark bit, the effect is that one packet less will be arriving an RTT later. For comparison, with multiplicative decrease, the corresponding flow will halve its window. Since the window size is unknown, in this case both the timing and the *magnitude* of the effect is unknown. With the one-packet decrease rule, only the RTT is uncertain. This makes the system more predictable, from a router's point of view. For this reason, we expect that using additive decrease will make it easier to design a protocol that is robust.

C. Congestion feedback

The amount of signalling that can be sent from the network to the sources is limited. We will assume that we have one bit of information in the packet header, analogous to the standardized ECN bits. The difference to ECN is that the response we are defining for the sender is different, and also the rules for setting the bit are different.

One consequence of feedback based on packet marking is that the congestion signal generated by a bottleneck, i.e., the mark probability p, is distributed to flows using that bottleneck in proportion to the rate of each flow. For loss-based congestion control, such as TCP New Reno, the signal loss rate per unit time has the same property. We use the marking probability Eq. (1). We let the marking probability be determined by the *instantaneous* queue size, since that is the state variable we want to control. The queue dynamics provides about the right amount of low-pass filtering of the arrival rate: The queue absorbs small fluctuations, and only if a disturbance is large enough to result in a significant change in the queue size, it is relevant for congestion control.

We have one design parameter, q_0. We will see that q_0 should be chosen to be of the same order as the bandwidth-delay product $c\tau$, and that the equilibrium size of the queue will be significantly smaller than q_0.

III. System model

In this section we develop a fluid-flow model for the new protocol in a scenario with a single bottleneck. The model analyzed in next section. The capacity of the bottleneck is c and the roundtrip propagation delay (excluding queuing delay) is τ. The bottleneck is shared with constant rate real-time cross traffic, such that a proportion γc of the capacity is available, with $0 < \gamma \leq 1$. The state variables are the queue size $q(t)$ and the window size $w(t)$.

We first consider the inner-loop, the ACK-clock. In Fig. 1, open up the outer-loop, and view the window size $w(t)$ as an input to the system, and the queue size $q(t)$ at the bottleneck as the system output. Let $r(t)$ denote the sending rate. Before its relation to window size and queue size is discussed, we write down the queue dynamics:

$$\dot{q}(t) = \begin{cases} r(t) - \gamma c & q(t) > 0 \\ \max(0, r(t) - \gamma c) & q(t) = 0 \end{cases} \tag{2}$$

To capture the influence of the parameter γ, we use the *joint link model*

$$r(t) = \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t)$$

The first term can be interpreted as the rate at which data is ACKed, while the second terms are additional packets that are sent or omitted following a change of the window size. Substituting into (2) we get

$$\dot{q}(t) = \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c \tag{3}$$

This model unifies the *integrator model* which puts $r = w/(\tau + q/c)$, which is used in much of the TCP literature [7], [17], and the *static model* $q(t) = w(t) - c\tau$, which takes the ACK-clock into account [18], but which does not handle real-time cross traffic.

It is important that (3) is *stable*. Both the static gain and the convergence time constant grow as γ is decreased, i.e., if the level of real-time cross traffic is increased. To capture this influence is the main advantage of the joint link model. We refer to [16], [19], for further discussion and validation.

The outer-loop, i.e., the window controller, is derived from the window update law together with the router's marking probability (1). Additive increase contributes a growth m per RTT. The rate of ACKed data is $w(t-\tau)/(\tau + q(t-\tau)/c)$, with a proportion $p(t-\tau)$ of ACKs being marked. This gives

$$\dot{w}(t) = \frac{1}{\tau + q(t-\tau)/c} \left(m - \frac{q(t-\tau)\, w(t-\tau)}{q_0 + q(t-\tau)} \right) \tag{4}$$

Substitution into (3) gives the closed-loop queue dynamics

$$\dot{q}(t) = \begin{cases} \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c & q(t) > 0 \\ \max\left(0, \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c\right) & q(t) = 0 \end{cases} \tag{5}$$

This system of non-linear time-delayed differential equations for w and q is further analyzed in the next section.

IV. Analysis

Due to the non-negativity constraint $q(t) \geq 0$, the right hand side of (5) is discontinuous at the border, and hence not Lipschitz. Since the standard results on existence and uniqueness of solutions does not hold in this setting, we derive the following result.

Theorem 1: Assume that $f : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^n$ is continuous, and that $\phi : [-1, 0] \to \mathbb{R}^n$ is non-negative, bounded, measurable, and right-hand continuous. Then there exists a unique non-negative function $x : [-1, \infty) \to \mathbb{R}^n$, continuous and right-hand differentiable for $t \geq 0$, such that for each $k = 1, \ldots, n$,

$$\dot{x}_k(t) = \begin{cases} f_k(t, x(t-1)) & x_k(t) > 0 \\ \max(0, f_k(t, x(t-1))) & x_k(t) = 0 \end{cases}$$

for $t \geq 0$, with initial condition $x(t) = \phi(t)$ for $-1 \leq t \leq 0$.

Proof: See [16]. ■

It is clear that if the initial conditions for q and w are sufficiently regular, Theorem 1 can be applied to Eqs. (4), (5), and hence our congestion control model is well posed.

For any positive parameters, Eq. (5) has a unique equilibrium, given by the solution w^* and q^* of the equations

$$mq_0 = q^*(w^* - m) \quad w^* = \gamma(q^* + c\tau) \tag{6}$$

It is clear that $q^* > 0$ and $w^* > \max(m, \gamma c\tau)$. These equilibrium equations can be solved explicitly. Both q^* and w^* grow with q_0, asymptotically as $O(\sqrt{q_0})$. The choice $q_0 = c\tau$ gives $w^* = \gamma c\tau + m$ and $q^* = m/\gamma$.

Next we study the system behavior close to the equilibrium. We will use the following lemma.

Lemma 2: Let A, B, C, and D be positive constants. Assume that $B < C \leq D < 1$, $A \leq 1$ and that $A + B < \pi/2$. Then the dynamical system

$$\dot{x}(t) = -\begin{pmatrix} A & B \\ A - D & B + C \end{pmatrix} x(t-1)$$

is globally asymptotically stable.

Proof: The idea of the proof is to consider the eigenvalues of the matrix

$$M = \begin{pmatrix} A & B \\ A-D & B+C \end{pmatrix}$$

It can be shown that they are real, distinct, and in the interval $0 < \lambda < \pi/2$. Then M can be diagonalized, reducing stability to the stability of two scalar systems. The details will appear in [20]. ■

The stability of the closed-loop system is stated next.

Theorem 3: Let $c > 0$ be the total capacity of the bottleneck, and let γc, $0 < \gamma \leq 1$, be the available capacity. Let $\tau > 0$ be the propagation delay, and $m > 0$ the packet size. If $q_0 \geq c\tau$, then the system (4) and (5), describing the queueing and window dynamics, is asymptotically stable.

Proof: Let q^* and w^* denote the equilibrium values, i.e., the solutions to Eq. (6). Put $q(t) = q^* + \widetilde{q}(t)$ and $w(t) = w^* + \widetilde{w}(t)$, and assume that $\widetilde{q} > -q^*$, so that the queue does not underflow. Ignoring higher order terms, the linearized system is

$$\dot{\widetilde{w}}(t) = \frac{1}{\tau + q^*/c}\left(-\frac{q^*\widetilde{w}(t-\tau)}{q_0+q^*} - \frac{q_0 w^*}{(q_0+q^*)^2}\widetilde{q}(t-\tau)\right)$$

$$\dot{\widetilde{q}}(t) = \frac{1}{\tau + q^*/c}\left(\frac{q_0\widetilde{w}(t-\tau)}{q_0+q^*} - \left(\gamma + \frac{q_0 w^*}{(q_0+q^*)^2}\right)\widetilde{q}(t-\tau)\right)$$

Scale time, by the change of variables

$$x(t) = \begin{pmatrix} \widetilde{w}(\tau\, t) \\ \widetilde{q}(\tau\, t) \end{pmatrix}$$

Then the system is of the form

$$\dot{x}(t) = -\begin{pmatrix} A & B \\ A-D & B+C \end{pmatrix} x(t-1)$$

where

$$A = \frac{\tau}{\tau + q^*/c}\,\frac{q^*}{q_0+q^*} \qquad B = \frac{\tau}{\tau + q^*/c}\,\frac{q_0 w^*}{(q_0+q^*)^2}$$

$$C = \gamma\frac{\tau}{\tau + q^*/c} \qquad D = \frac{\tau}{\tau + q^*/c}$$

All four constants are in the open interval $(0, 1)$, and we see that $C \leq D$. Since $w^* = \gamma(c\tau + q^*)$, the assumption $q_0 \geq c\tau$ implies that

$$B = \gamma\frac{\tau}{\tau + q^*/c}\,\frac{q_0}{q_0+q^*}\,\frac{c\tau+q^*}{q_0+q^*} \leq \gamma\frac{\tau}{\tau + q^*/c}\,\frac{q_0}{q_0+q^*}$$

Then $B < C$ and that $A + B \leq D < 1$. Asymptotic stability of the linearized system now follows from Lemma 2. Then the equilibrium of the non-linear system is locally asymptotically stable (see [21], Theorem 4.6). ■

On fairness of the new protocol, note that when generalizing the model to several flows and bottlenecks, it is worth noting that if several flows share the same set of bottlenecks, they will all experience the same mark probability p. The window update law implies that in equilibrium $m_k - w_k^* p^* = 0$ for each flow k. Hence $w_k^* = m_k/p^*$, so the window sizes of all flows, measured in number of packets, is the same. This equilibrium property is shared with TCP New Reno, and implies that in this case, the rate of flow k is proportional to m_k/RTT_k. This indicates that fairness properties of the new protocol will be close to those of TCP New Reno. Fairness for a general network topology is analyzed in more detail in [16].

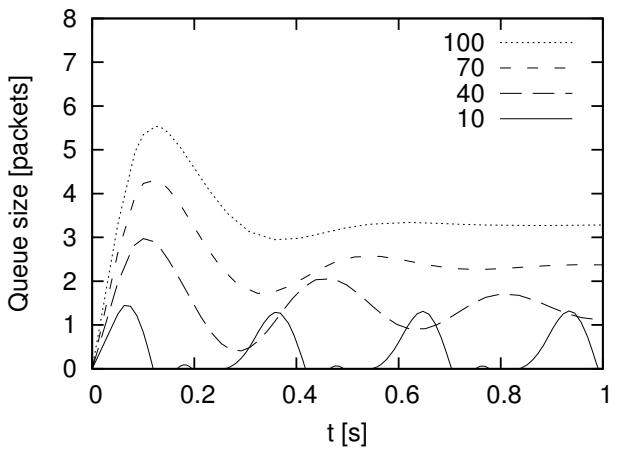

Fig. 2. The evolution of the queue size, for $q_0 = 10$, 40, 70, and 100. The behavior agrees well with the presented stability analysis.

V. Experimental evaluation

Experimental evaluation through `ns2` simulations is presented in this section. The performance of the new protocol is compared with TCP New Reno and TCP Vegas. First, however, tuning of the protocol parameter q_0 is discussed through simulation of the model developed in Section III.

A. Fluid-flow simulation

To get a better understanding of the dynamics, and the design parameter q_0, we have performed a series of simulations based on the fluid-flow system model (5). We consider a 10 Mbit/s link, with a roundtrip propagation delay of 50 ms, and with 30% non-responsive cross-traffic (i.e., $\gamma = 0.7$). The packet size is 1500 bytes. The bandwidth delay product is 42 packets. The initial conditions are $q(t) = 0$ and $w(t) = w^*$ for $t \leq 0$.

In Fig. 2, we see the evolution of the queue size for some values of the design parameter q_0. By Theorem 3, the system is stable when $q_0 \geq 42$ packets. We see that increasing q_0 makes the system less oscillatory, and the equilibrium queue size gets larger. For $q_0 = 40$ packets, the system converges slowly. For the smallest value, $q_0 = 10$ packets, the system is unstable and converges to a limit cycle. The amplitude of this limit cycle is quite small, in comparison to the transients for stable choices of q_0. So from a practical perspective, the main problem caused by this instability is a waste of resources, since the queue gets empty which implies underutilization of the bottleneck link, not large delay and delay fluctuations degrading the quality of real-time applications.

	New Reno	Vegas	New protocol
Loss rate (%)	4.27	0.00	0.02
Utilization (Mbit/s)	1.64	1.88	1.90
TCP throughput (Mbit/s)	1.02	1.39	1.40
Queue average (packets)	7.26	2.90	3.77
std. dev. (packets)	7.12	2.46	3.41
Forward delay average (ms)	114.66	75.78	80.84
std. dev. (ms)	40.11	13.74	19.61

TABLE I

RESULTS FOR THE SINGLE LINK, SINGLE FLOW SCENARIO.

	New Reno		Vegas		New protocol	
Flow	#0	#1	#0	#1	#0	#1
Throughput	0.11	1.49	0.88	0.71	0.12	1.48
Window	4.49	13.05	25.48	4.40	2.94	7.06
dev.	2.01	3.57	10.08	0.75	0.75	1.26
Delay	323.05	91.29	270.83	62.56	181.30	44.63
dev.	46.30	26.29	51.95	24.14	23.59	12.49

TABLE III

RESULTS FOR THE "PARKING-LOT" TOPOLOGY, PER-FLOW VALUES.

B. Comparison with other TCP versions

The fluid flow model neglects important packet-level features of the proposed mechanism, in particular, it does not capture the limited information about the mark probability p. To see how the mechanism behaves at the packet level, it has been implemented in the `ns2` simulator. The implementation has a few small changes to a standard implementation:

- Two new bits in the `ns2` packet header, analogous to the ECN and ECN echo bits, have been added.
- The `Agent/TCPSink` class is extended to copy the new bit from data packets to the corresponding ACKs.
- The `Queue/DropTail` class is extended to mark packets according to Eq. (1).
- The `Agent/TCP/Newreno` class is modified to examine the echoed mark bit in each received ACK and implement the new additive decrease rule.

Neither the slow start phase or New Reno's reaction to packet losses is modified at all. To force the TCP sender to leave the slow start phase, the initial slow start threshold in all the simulations were set to a value close to the product of the RTT and the fair share.

We compare the results to TCP New Reno and TCP Vegas. We are interested both in end-to-end properties, such as throughput and delay, and per-link properties such as queue size, utilization and loss rate.

1) Single link, single flow: The first scenario uses a single TCP flow over a single bottleneck, with $c = 2$ Mbit/s, $\tau = 100$ ms, and a bandwidth delay product of 16 packets. The buffer size is 21 packets, and q_0 is 32 packets. The simulation runs for 60 s, with a file transfer starting at $t = 0.1$ s. The bottleneck is shared with Poisson cross-traffic, 20% of the capacity, increased to 40% during the interval 20–40 s. The response to this step is illustrated in Fig. 3.

Table I shows average properties. Compared to TCP New Reno, the new mechanism almost eliminates packet losses, while at the same time the TCP throughput is increased and the queueing delay is reduced. The performance is similar to the performance with TCP Vegas.

2) "Parking-lot" topology: In the next scenario, we have three bottleneck links in series. The links are still 2 Mbit/s, with both the buffer size and q_0 set to 20 packets. Over each link, there is one TCP flow, and 20% Poisson cross-traffic. These short flows have 20 ms RTT excluding queueing. We also add one long RTT flow, traversing all three bottlenecks, with an RTT of 150 ms excluding queueing.

Average per-link quantities are summarized in Tab. II. We see that Vegas reduces queueing delay a bit compared to New Reno. While it reduces packet losses, there is still a significant packet loss (different for the three links). The new mechanism reduces queueing further, and it eliminates packet losses. Average per-flow quantities are summarized in Tab III. Flow #0 is the long flow, and flow #1 is one of the short flows. We see that the end-to-end delay and delay jitter is reduced considerably. If we compare the throughput between the long flow and the short flows, we see that the sharing is similar for the new protocol and New Reno; a short flow gets ≈ 13 times the throughput of the long flow. For Vegas, the picture is very different. Vegas aims to give all flows the same throughput, and in this example, the long flow gets slightly larger throughput than a short flow.

VI. CONCLUSIONS

We have proposed a new congestion control protocol, consisting of the standard ACK-clock based inner-loop, and a novel outer-loop that adjusts the window size. Routers mark packets with a probability depending on their instantaneous queue size. In the absence of congestion, sources use additive increase. When a marked ACK is received, the source reduces its window size by the size of the corresponding data packet.

The rule of reducing window size by one packet on the reception of a marked ACK is a subtle change from standard ECN processing, which has two important benefits: The frequency of packet marks can be much higher than with standard AQM/ECN schemes, on average, a flow in equilibrium will get receive one marked ACK per RTT. In this way, sources are provided with more information about the network state. Furthermore, from a router's point of view, the effect of the decision to mark a packet becomes much more predictable, since it does not depend on the unknown window size of the corresponding flow.

The proposed marking scheme uses a single parameter, q_0, per link. The value should be on the same order as the expected value of the bandwidth-delay product of flows using the link. Tuning of this parameter appear to be fairly easy. Larger values result in a larger equilibrium queue size and a more stable system. For small values of q_0, the system is unstable, but this instability corresponds to small oscillations of the queue size and queue underflow, which indicates that the penalty for choosing a too small q_0 is reduced link utilization, not large queue oscillations.

The proposed protocol has been implemented in `ns2`, and simulated over a single link topology, and over a "parking-

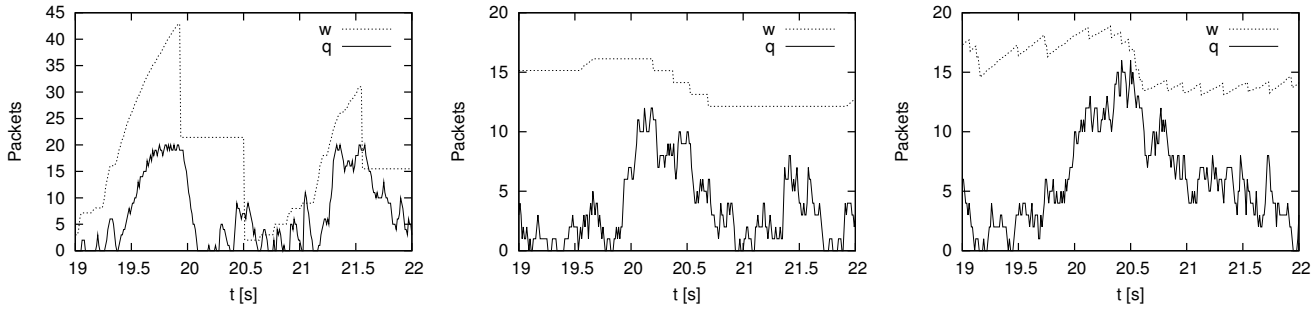

Fig. 3. Window size (w) and bottleneck queue size (q) for a short segment of a simulation with a single bottleneck and a congestion controlled flow sharing the bottleneck with Poisson cross-traffic. At $t = 20$ s, the cross-traffic intensity is increased from 20% to 40% of link capacity. Left to right: New Reno, Vegas, and the new protocol. Note the different scale on the vertical axis.

	New Reno			Vegas			New protocol		
Loss	1.96	1.99	1.67	0.25	0.14	0.09	0.00	0.00	0.00
Util.	2.00	2.00	2.00	1.99	1.99	1.99	1.99	1.99	1.99
Queue	11.98	12.07	11.94	8.78	8.45	8.58	4.37	4.43	4.33
dev.	4.32	4.37	4.35	4.02	3.92	3.71	2.15	2.16	2.09

TABLE II

RESULTS FOR THE "PARKING-LOT" TOPOLOGY, PER-LINK VALUES.

lot" topology with several bottlenecks. The simulations show that the proposed protocol maintains full utilization and small queueing delays. Furthermore, sharing of capacity between flows is close to what flows would get with TCP New Reno.

The fundamental difference from common TCP/AQM schemes is that the feedback from the network is applied additively to the window. This is a promising approach for making congestion controlled traffic, including peer-to-peer file-sharing of large files, coexist nicely with real-time traffic.

The development of the new congestion control protocol is not complete, but suggests some further work. Interesting problems include the equilibrium and dynamical properties when a large number of flows share a bottleneck. In TCP New Reno, the first transition from the slow start state to congestion avoidance state happens at the first packet loss. With no packet losses, we need a slow-start that grows the window size quickly when the network is uncongested, and uses received packet marks to switch to congestion avoidance before the bottleneck queue grows large.

REFERENCES

[1] G. Haßlinger, J. Mende, R. Geib, T. Beckhaus, and F. Hartleb, "Measurement and characteristics of aggregated traffic in broadband access networks," in *Proceedings of International Teletraffic Congress: Managing Traffic Performance in Converged Networks*, ser. LNCS, vol. 4516. Ottawa: Springer, June 2007, pp. 998–1010.

[2] R. Bush and D. Meyer, "Some Internet architectural guidelines and philosophy," RFC 3439, December 2002.

[3] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *SIGCOMM*. Portland: ACM, September 2004.

[4] Y. Ganjali and N. McKeown, "Update on buffer sizing in Internet routers," *Computer Communication Review*, vol. 36, no. 5, pp. 67–70, October 2006.

[5] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer sizing for congested Internet links," in *Proceedings of IEEE INFOCOM*, 2005.

[6] C. Chrysostomou, A. Pitsillidesa, L. Rossidesa, M. Polycarpoub, and A. Sekerciogluc, "Congestion control in differentiated services networks using fuzzy-RED," *Control Engineering Practice*, vol. 11, no. 10, pp. 1153–1170, October 2003.

[7] C. Hollot, V. Mistra, D. Towsley, and W. B. Gong, "A control theoretic analysis of RED," in *IEEE Infocom 2001*, 2001.

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

[9] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," in *ACM Sigcomm*, August 2002.

[10] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[11] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, October 2000.

[12] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of IEEE INFOCOM*, Barcelona, April 2006.

[13] S. Liu, T. Basar, and R. Srikant, "TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks," in *Proc. First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, Pisa, October 2006.

[14] R. King., R. Baraniuk, and R. Riedi, "TCP-Africa: an adaptive and fair rapid increase rule for scalable TCP," in *Proceedings of IEEE INFOCOM*, vol. 3, March 2005, pp. 1838–1848.

[15] N. Möller, K. H. Johansson, and K. Jacobsson, "Stability of window-based queue control with application to mobile terminal download," in *Mathematical Theory of Networks and Systems*, Kyoto, July 2006.

[16] N. Möller, "Window-based congestion control—modeling, analysis and design," Ph.D. dissertation, KTH, Stockholm, January 2008.

[17] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, pp. 28–43, February 2002.

[18] J. Wang, D. X. Wei, and S. H. Low, "Modelling and stability of FAST TCP," in *Proceedings of IEEE Infocom*, Miami, March 2005.

[19] K. Jacobsson, H. Hjalmarsson, and N. Möller, "ACK-clock dynamics in network congestion control – an inner feedback loop with implications on inelastic flow impact," in *IEEE Conference on Decision and Control*, 2006.

[20] N. Möller and K. H. Johansson, "Congestion control for small queues: analysis and evaluation of a new protocol," Extended paper submitted for journal publication, 2008.

[21] A. Halanay, *Differential equations—Stablity, Oscillations, Time Lags*, ser. Mathematics in Science and Engineering. Academic press, 1966, vol. 23.