On the fair coexistence of loss- and delay-based TCP

Łukasz Budzisz*, Rade Stanojević*, Arieh Schlote†, Robert Shorten* and Fred Baker‡
*Hamilton Institute, NUI Maynooth, Ireland
e-mail: [Lukasz.Budzisz, Rade.Stanojevic, Robert.Shorten]@nuim.ie
† Institut für Mathematik, Universität Würzburg, Germany
e-mail: arieh.schlote@stud-mail.uni-wuerzburg.de
‡Cisco Systems
e-mail: fred@cisco.com

Abstract—**Delay-based TCP variants continue to attract a large amount of attention in the networking community. Potentially, they offer the possibility to efficiently use network resources while at the same time achieving low queueing delay and virtually zero packet loss. One major impediment to the deployment of delay-based TCP variants is their inability to coexist fairly with standard loss-based TCP. In this paper we propose a simple strategy to make the fair coexistence possible and to ensure that delay-based flows will revert back to the delay-based operation when loss-based flows are no longer present. Analytical and ns-2 simulation results are presented to validate the proposed algorithm.**

I. Introduction

A longstanding question in the networking community is whether queueing delay may be used as a reliable basis for network congestion control [1], [2]. Despite much work on this topic [3]–[19], the case for delay-based congestion control remains compelling. Potentially, allocation of network bandwidth between competing sources can be achieved with low (zero) packet loss, with very low queueing delay, and with full utilisation of network links. Networks which exhibit this property are said to operate at the *knee of the throughput-delay* curve [20]. Motivated by these and other potential benefits, delay-based congestion control remains an active area of research and new algorithms continue to be developed. Recent examples include: Fast TCP [3], [21]; Microsoft Compound [22] (partially based on delay); more recent delay-based additive increase multiplicative decrease (AIMD) variants [23]–[25]; and this present work which is an outcome of a Cisco-funded project to investigate delay-based congestion control in harsh network environments.

Traditionally, a number of arguments are usually put forward that question the use of delay in congestion control applications. These include: the difficulty in obtaining delay estimates from network measurements [1]; network sampling issues [1], [2], [26]; the inability of existing delay-based algorithms to maintain a low standing queue [26]; and the inability of delay-based flows to coexist fairly with loss-based flows in mixed environments. These items have been the subject of much discussion [1], [2], [26] which we do not repeat here. Rather, we focus the specific issue of coexistence. We wish to develop delay-based algorithms that coexist fairly with loss-based counterparts, e.g., standard TCP. Note that the issue here is not just simple coexistence; after all, delay-based flows may simply switch to a loss-based mode once a packet loss is detected thereby solving the fairness problem. The issue that makes coexistence difficult is that delay-based flows must revert back to delay-based operation when loss-based flows are no longer present.

Our attempt to solve the aforementioned coexistence problem is motivated by recent work concerning Active Queue Management (AQM) emulation from end-hosts using delay measurements (PERT) [25], and its subsequent extension [27], where the original algorithm was modified, to address particularly coexistence issues. The strategy followed in this extension was to make PERT adapt in heterogeneous environments to achieve some form of coexistence. For convenience we refer to this modification as mPERT in this paper. While we believe this work, and its extension, to be of great promise, a number of issues nevertheless remain unresolved before, in our opinion, delay-based AIMD can be deployed successfully.

A key question to be addressed concerns that of *incremental deployment*. Namely, it is very important that any delay-based AIMD results in networks that not only operate at the knee of the delay-throughput curve, but also perform well in the presence of normal loss-based TCP flows (referred to as mixed or *heterogeneous* environments). In this latter context it is also important that network properties such as loss rate and bottleneck utilisation are not adversely affected when delay based flows are deployed in mixed environments.

A further issue concerns whether delay-based flows can accurately infer the presence of loss-based TCP flows. In prior work [11], mode switches have been used to switch delay-based flows into a loss-based mode once the presence of conventional TCP flows are detected. While detection of loss-based flows can be easily inferred from packet losses, it is not trivial to ensure that the delay-based flows revert back to delay-based behaviour when the loss-based flows are no longer present (we refer to this as *on/off switching*).

Our starting point in this paper is the recently proposed algorithm mPERT [27] that is designed to ensure coexistence of loss-based and delay-based TCP flows. Study of this algorithm reveals that several features might make incremental deployment difficult. In particular, we show that mPERT may lead to excessive network loss rates in the presence of loss-based TCP flows, and that the aforementioned problem of on/off switching is not fully resolved by the proposed extension to PERT [27]. The first issue, that of increased loss rate, is caused by a positive feedback loop that is at the heart of the mPERT. As loss rates increase, the mPERT flows become more aggressive, thereby increasing the loss rate further, and leading finally beyond the limit of the applicability of the square-root formula [28] on which mPERT is based. A further consideration is the operation of mPERT in many known situations in which square-root formula is not valid (e.g., networks with drop-tail buffers, networks with high-speed TCP, etc.). The other issue that raises our concern about mPERT, is that of on/off switching. As long as the network buffers remain nearly full, mPERT will operate in its loss-based mode with the associated high loss rates.

Given these basic observations, we suggest here an alternative strategy to ensure gentle coexistence between loss- and delay-based AIMD flows. In particular, our principal contribution in this paper is to propose a new strategy that allows delay-based (PERT-like) TCP flows to coexist fairly with loss-based flows when they are present, without any increase in network loss rate, and without any assumptions based on the square root formula [28]. Furthermore, our modification allows delay-based AIMD flows to revert to delay-based behaviour whenever loss-based flows are no longer present.

Our paper is structured as follows. In Section II we describe the PERT and mPERT algorithms. We then present a number of experimental results that outline some of the problems described above. Next, in Section III we introduce an alternative strategy for coexistence that is suitable for deployment in heterogeneous environments, along with a number of simulations that illustrate its efficacy. Finally, in Section IV we provide mathematical analysis that describes the ability of the proposed solution to switch between loss-based and delay-based operation regimes.

II. Prior work: The PERT algorithm

Our work is motivated by the recently published PERT algorithm. While much work has been published on delay-based congestion control [3]–[10], [12]–[19], PERT provides an initial focus for our work because of its recent publication (in this sense it constitutes state-of-the art), because of its great potential, and because of its (in modified form) ability to coexist with conventional TCP flows.

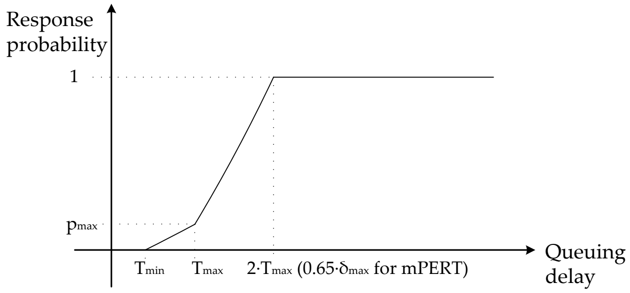


Fig. 1. PERT's back-off probability function

The basic idea behind Probabilistic Early Response TCP (PERT) proposed by Reddy et al. in [25], is very simple and involves responding to delay in a probabilistic rather that deterministic manner. By judiciously selecting the manner of the probabilistic response the designers of PERT are able to emulate, from end-hosts, the behaviour of a range of AQM's. To facilitate such AQM emulation, each PERT flow probes the network for congestion as a normal AIMD flow, but reduces its congestion window in a probabilistic manner that depends on the estimated network delay. We refer to this back-off mechanism as a *back-off policy*. The authors of PERT demonstrate that (in principle) any AQM can be emulated by selecting the back-off policy appropriately. A sample back-off policy, chosen by Reddy et al. to emulate a RED AQM in [25] is depicted in Fig. 1. As can be seen from this figure the back-off policy is characterised by a non-zero response probability above a minimum delay threshold (T_{min}) that is linearly increasing to the probability p_{max} at the T_{max} threshold (as in RED).

PERT, and its various embodiments, is examined experimentally in [25]. The paper discusses issues such as estimation of delay (δ), estimation of maximum (RTT_{max}) and minimum RTT (RTT_{min}), operation of PERT in single and multiple bottleneck environments, and convincingly demonstrate both that PERT can be easily implemented (i.e., all signals can be estimated), and that networks in which it is implemented, perform as would be expected if packets were marked by router based AQM's. Our purpose here is not to repeat this work; rather we wish to consider how this basic idea may fit networks in which standard, loss-based TCP flows are for time-to-time present. In particular, we would like the delay-based flows to coexist with loss-based counterparts when loss-based flows are present, and to revert to delay-based operation (to achieve low network queueing delays) when the loss-based flows are off.

Such an extension to PERT has already been proposed in [27]. Roughly speaking, in this work, Kotla and Reddy propose to sense the presence of loss-based TCP flows by observing whether the network queueing delay exceeds some threshold. Once a loss-based flow is detected, the modified PERT (mPERT) flows compensate for the higher back-off

```
On receipt of each ACK:
Estimate the current queueing delay, $\delta_{current}$
Set $p = g(\delta_{current})$ (function shown in Fig. 1)
Pick a random number $rand$, uniformly from 0 to 1
if $rand < p$ and only once per RTT then
    reduce $cwnd$ by $\beta * cwnd$
    $(0 \leq \beta = \frac{\delta_{current}}{\delta_{current}+\delta_{max}} \leq 0.5)$
else
    increment $cwnd$ by $\alpha / cwnd$
    $(1 \leq \alpha = f(\delta_{current}) \leq \alpha_{max} = 32)$
end if
```

Fig. 2. pseudo-code for mPERT algorithm

rates that they experience by adjusting the rate at which they probe for new bandwidth (i.e., additive increase factor, α) and by adjusting the manner in which they reduce their congestion window (i.e., multiplicative decrease factor, β). These adjustments are made based on the estimated network loss rate, and are outlined in pseudo-code form in Fig. 2.

Specifically, mPERT assumes that there are no loss-based flows in the network as long as the delay stays below 50% of maximum observed queueing delay (δ_{max}). Once this threshold is exceeded, mPERT anticipates the presence of a loss-based flow and to be able to compete with the loss-based counterpart, starts increasing the TCP congestion window variable ($cwnd$) upon receipt of each ACK by $min(1 + p^{'}/p, \alpha_{max})/cwnd$, where ratio of the back-off and drop rates ($p^{'}/p$) is estimated as for TCP friendly rate control (TFRC) algorithm [29]. The TCP's window back-off factor is adjusted accordingly to reflect the ratio of current queueing delay ($\delta_{current}$) and maximum observed queue size ($\delta_{current} + \delta_{max}$), so the most conservative setting (that of standard TCP) is achieved when the queue is full.

While the aforementioned modifications do indeed improve the behaviour of PERT flows in the presence of loss-based TCP flows, a number of issues of concern do arise upon examination of the basic algorithm. This can be summarised as follows.

(i) **Fairness :** mPERT makes no attempt to ensure that PERT based flows compete fairly with loss-based flows.

(ii) **Loss rate :** The effect of increasing the aggressiveness with which PERT flows probe for bandwidth increases the network loss rate. This in turn leads to an increase in aggressiveness with which the flows probe for bandwidth and constitutes an unstable positive feedback loop which results in high network loss rate. As the loss rate increases the mPERT flows may get beyond the limit of the applicability of the square-root formula.

(iii) **Detection :** The presence of loss-based flows is inferred through an increase in the average queueing delay beyond some threshold. This is an unreliable indicator. In particular, it will be shown in Section II-A that in the presence of many PERT flows, the average queueing delay will not reduce below this threshold once loss-based flows leave the network.

TABLE I
SIMULATION PARAMETERS

PARAMETER NAME	VALUE / RANGE
Number of flows	30
Access link (each flow)	bandwidth: 100 Mbps one-way propagation delay: 5-30 ms (random)
Bottleneck link	bandwidth: 10-100 Mbps (for *fairness* and *loss rate* experiments) bandwidth: 40 Mbps (for *on/off switching* experiments) one-way propagation delay: 5 ms
Queue size at bottleneck	500·(bandwidth/40 Mbps) packets
T_{min}	5 ms
T_{max}	20 ms
δ_{max} (*initial value*)	100 ms
MTU size / Data payload	1000 / 960 Bytes
Intermittent loss-based flow (*on/off switching* experiments only)	on-time: 200 s off-time: 275 s
Simulation time	500 s

(iv) **Square root formula :** The design of mPERT is based on the square root formula. However, there are many situations in which this formula is not valid. These include situations in which new high speed TCP flows are present, or when drop-tail buffers are present.

Our objective in this paper is to propose changes to delay-based AIMD that address the coexistence issue, and that circumvent the issues mentioned above. Our key idea is to retain probabilistic back-off policies that are at the heart of PERT. However, rather than emulating a RED AQM, we carefully select a back-off policy to ensure both fair coexistence and appropriate on/off switching behaviour. As we shall see in Section III, this strategy is a better alternative than adapting AIMD parameters to the network environment in the sense that it avoids all of the objections raised above.

A. Side-effects of mPERT

Before proceeding, we now present a number of ns-2 [30] simulations to illustrate the side-effects (items (i)-(iii) mentioned above) of adaptive AIMD parameters.

1) **Basic experimental setup:** In the following simulations we consider a mix of 30 flows (20 mPERT flows and 10 standard TCP flows for the *fairness* and *loss rate* tests, and 29 mPERT flows and a single standard TCP flow for *on/off switching* tests). These flows compete for available bandwidth in a single-bottleneck dumb-bell topology. The bottleneck capacity varies from 10 to 100 Mbps in *fairness* and *loss rate*

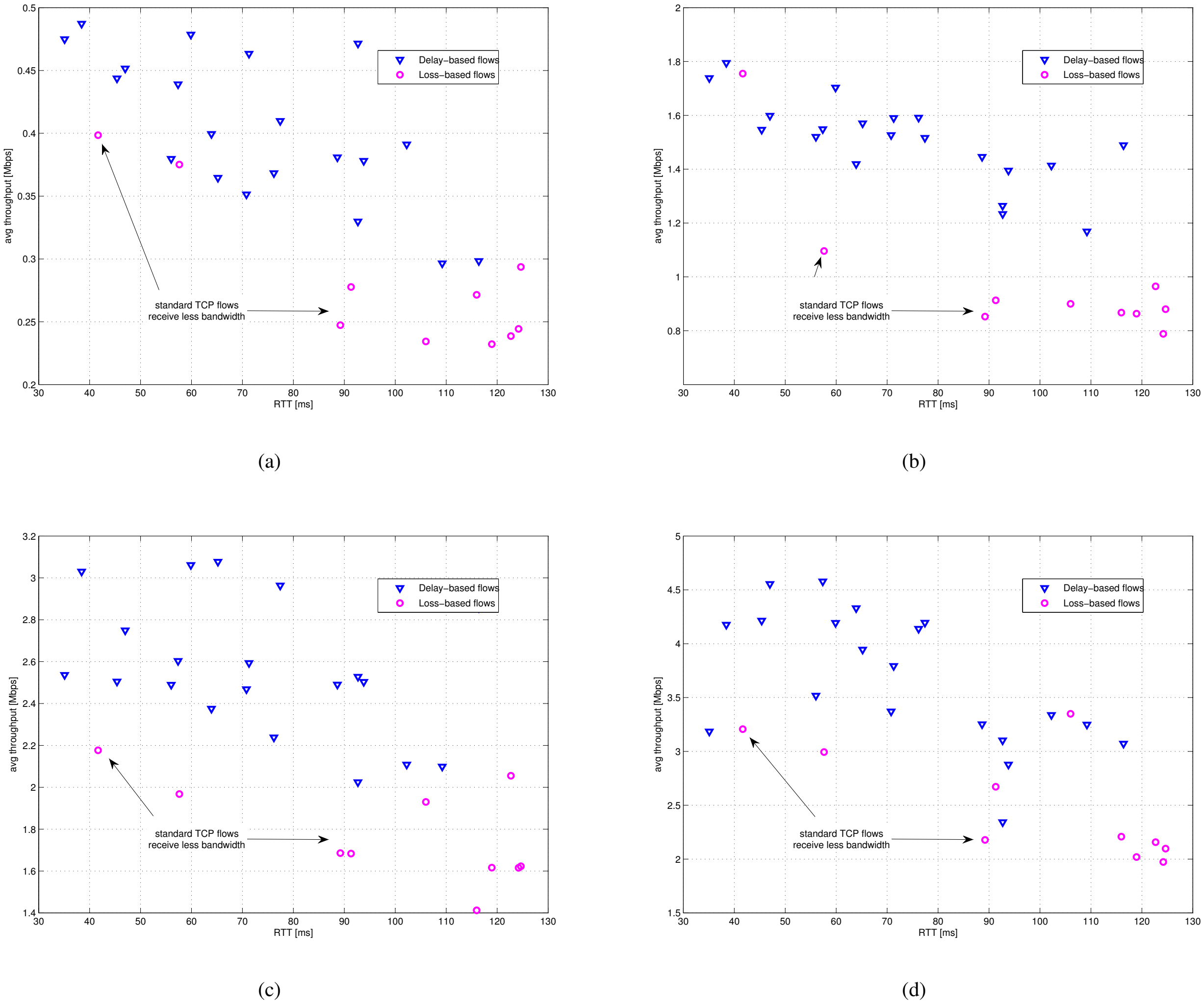


Fig. 3. Coexistence of 20 mPERT flows and 10 standard TCP flows in terms of average throughput, for following capacity of bottleneck: (a) 10 Mbps; (b) 40 Mbps; (c) 70 Mbps; and, (d) 100 Mbps .

experiments, whereas for *on/off switching* it is set to 40 Mbps. One-way propagation delay in the bottleneck is set to 5 ms for all experiments. All access links are 100 Mbps links with uniformly distributed one-way propagation delay (5-30 ms). Loss-based flows use standard TCP (conventional AIMD), while all mPERT flows operate according to the algorithm described in [27]. The most important simulation parameters are summarised in Table I.

2) **Fairness:** Our first set of experiments demonstrates that mPERT can be made to coexist with standard TCP, but not fairly. In Fig. 3 we present plots for the average throughput obtained by each of the flows in a scenario with a (20, 10) mix of delay/loss-based flows. As can be seen in all cases (a-d) the average value of the throughput for the loss based flows is noticeably lower than for theirs mPERT counterparts. This observed unfairness is caused by the increased aggressiveness of the mPERT flows in the presence of loss-based flows. Presented experiments demonstrate clearly that fairness aspects must be taken into consideration while addressing coexistence.

3) **Loss rate:** We now examine the effect on the network loss rate when mPERT operates in its loss-based mode. Specifically, we compare the behaviour of 20 mPERT flows coexisting with 10 standard TCP flows, with scenario in which all 30 flows are standard TCP (conventional AIMD). The respective loss rates are shown in Fig. 4. As can be seen, mPERT leads to loss rates that are an order of magnitude higher than would be the case with only loss-based flows. This observation makes incremental deployment of mPERT extremely problematic, since observed loss rates exceed the limit of the applicability of the square-root formula, on validity of which mPERT is based. Such an increased loss rate for mPERT flows is a consequence of a positive feedback loop that exists between the over-aggressive mechanism for increasing mPERT's *cwnd* and increased network loss rate it provokes.

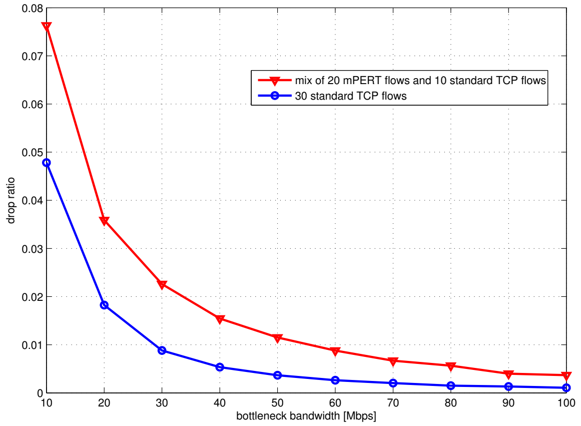
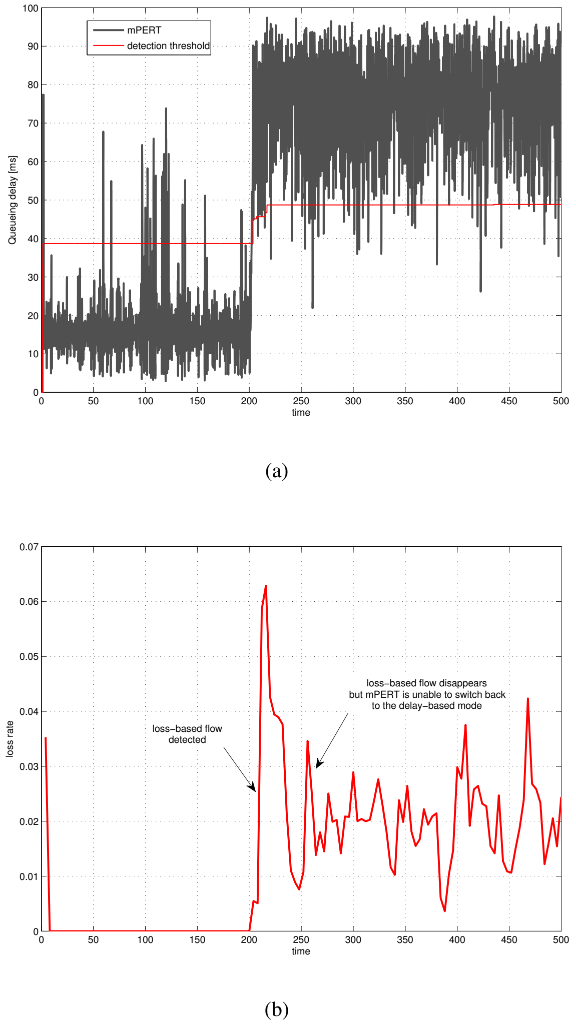


Fig. 4. Comparison in terms of loss rate: 20 mPERT flows coexisting with 10 standard TCP flows, and 30 standard TCP flows.



(a)

(b)

Fig. 5. Coexistence of mPERT flows with a single loss-based flow switching on and off: (a) queueing delay at the bottleneck; and, (b) loss rate.

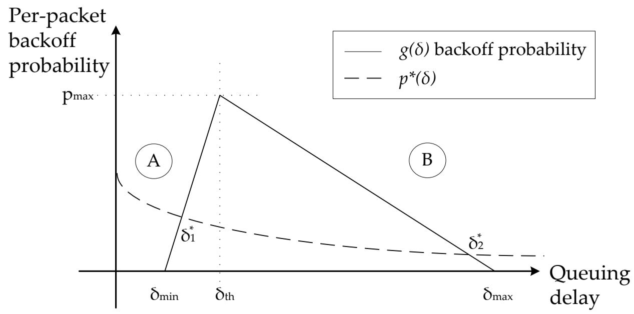


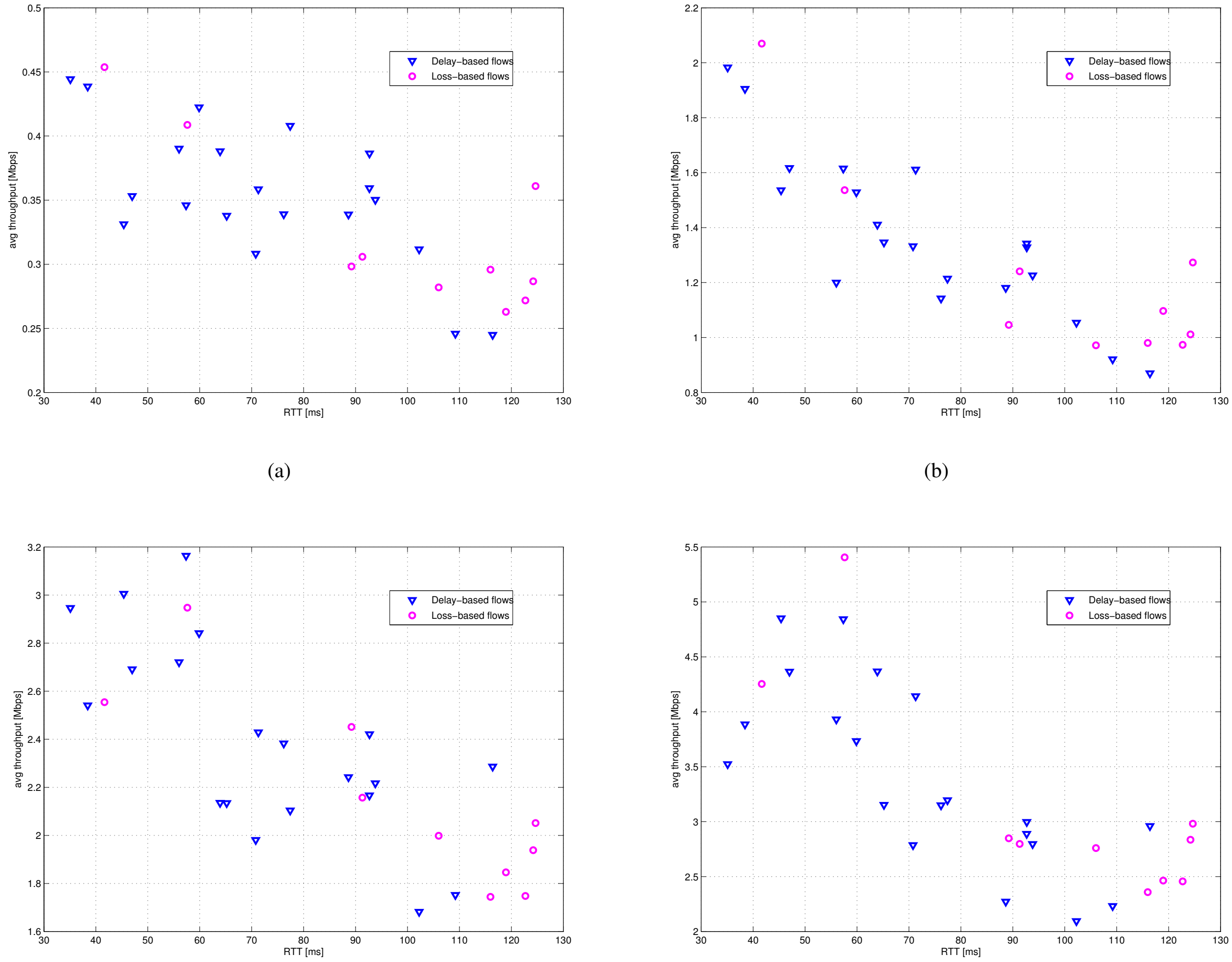
Fig. 6. Per-packet back-off probability as a function of observed delay

4) **On/Off switching:** The presence of loss-based flows is inferred by a mPERT flow by observing an increase in the average queueing delay beyond 50% of the δ_{max} value. Here we show that this detection strategy is not a reliable indicator of the presence of a loss-based flow. Specifically, we consider a scenario with 29 mPERT flows and one intermittent loss-based flow that is active between 200 s and 275 s. Notice that the average queueing delay does not reduce back below this 50% threshold once the loss-based flow leaves the network. Clearly, in this situation mPERT is unable to revert to the low queueing delay regime and stays in the loss-based mode (Fig. 5a), which is characterised by increased network loss rate (Fig. 5b) and high queueing delays, even though there are no loss-based AIMD flows present in the network.

III. AQM EMULATION TO ENSURE FAIR COEXISTENCE WITH LOSS-BASED FLOWS

We now present an alternative method, sketched initially in [31], to ensure coexistence of loss- and delay-based AIMD flows. As we shall see, this method avoids the problems of adjusting AIMD parameters, and ensures that the delay-based flows revert to delay-based operation when loss-based flows are no longer present in then network, even though these flows do not attempt to sense the presence of such flows directly. In what follows, we assume that δ, RTT_{min}, RTT_{max} can be estimated reliably by all delay-based flows in the network. Furthermore, we do not consider the issue of slow start for delay-based flows. These, and other issues are discussed in the original PERT publication and in other previous work: refer [1], [2], [23], [26].

Our basic idea, as briefly described in [31], is to carefully select the back-off policy to achieve fairness and on/off behaviour. Specifically, we select probabilistic back-off strategies of the form depicted in Fig. 6. As can be observed, the per-packet back-off probability function $p = g(\delta)$ has two parts; a part that increases monotonically with the delay δ (Region A), and a part that decreases monotonically with δ (Region B). This form of AQM emulation has the following properties:



(a)

(b)

(c)

(d)

Fig. 8. Coexistence of 20 delay-based flows and 10 standard TCP flows in terms of average throughput, for following capacity of bottleneck: (a) 10 Mbps; (b) 40 Mbps; (c) 70 Mbps; and, (d) 100 Mbps .

```
On receipt of each ACK:
Estimate the current queueing delay: $\delta_{current}$
Set $p = g(\delta_{current})$ (function shown in Fig. 6)
Pick a random number $rand$, uniformly from 0 to 1
**if** $rand < p$ **and** $cwnd \geq w_0$ **then**
    reduce $cwnd$ by $0.5 \cdot cwnd$
**else**
    increment $cwnd$ by $1/cwnd$
**end if**
```

Fig. 7. Pseudo-code for delay-based AIMD algorithm

(i) Assuming that p_{max} is large enough, the network stabilises in Region A when only delay-based flows are present.

(ii) When loss-based flows are present, the network is driven to Region B, and delay-based flows behave as loss-based flows due to the low per-packet back-off probability.

(iii) When loss-based flows switch off, the network cannot stabilise in this region due to a backward pressure exerted by the probability function. Namely, as the flows experience back-offs, the queueing delay reduces, thereby increasing the per-packet back-off probability, thus making further back-offs more likely. This process continues until the network stabilises in Region A.

As can be seen, this type of strategy should achieve co-existence of loss- and delay-based AIMD flows, without a discernible increase in network loss rate. Furthermore, the back-pressure described in (iii) should ensure on/off behaviour. Note also that the strategy makes no reference to the square-root formula. The basic algorithm is summarised in Fig. 7.

A. *Experimental results*

We now repeat the experiments presented in Section II.

1) **Basic experimental setup:** The aforementioned strategy is basically a modification of the well known RED AQM. Consequently, one method to select the parameter values for δ_{min}, δ_{th} and p_{max} is to use the rules for RED parameter settings [25]. δ_{max} is estimated for each flow separately, the default value is $100\,[ms]$. For purpose of comparative analysis with mPERT, we follow this strategy. The parameters δ_{min}, δ_{th} and p_{max} for proposed back-off policy correspond to the following parameters of mPERT: T_{min}, T_{max} and p_{max}, respectively and are given in Table I. Otherwise, the simulation scenarios are the same as in case of the mPERT analysis provided in Section II.

2) **Fairness:** We begin by repeating the fairness experiments that we used to evaluate the ability of mPERT to coexist with standard loss-based TCP flows. Fig. 8 depicts the results of a series of experiments; for a (20, 10) mix of delay/loss-based flows, and the bottleneck bandwidth changing from 10 to 100 Mbps respectively. Note that although there is a bias in favour of the loss-based flows (due to the fact that the delay-based flows experience a small number of non-loss induced back-offs in the high-queue regime), there is a reasonably fair coexistence of the loss-based and delay-based AIMD flows. Furthermore, in contrast to mPERT, the aforementioned bias in favour of loss-based flows can be controlled by carefully selecting the back-off policy. Notwithstanding this latter observation, the experiments nevertheless demonstrate very good coexistence of the delay-based and loss-based flows as measured by average throughput.

3) **Loss rate:** Next, we examine the effect of our algorithm on the network loss rate. To do this we repeat the analogous mPERT experiment where a mix of loss and delay-based flows coexist, and compare this to the corresponding situation of a network of loss-based flows only. The results are depicted in Fig. 9. Observe that the proposed algorithm does not increase significantly network loss rate in presence of loss-based flows.

4) **On/Off switching:** Our primary objective in this work was to develop a delay-based algorithm that behaves as a loss-based TCP when competing with loss-based TCP flows, but otherwise reverts to delay-based operation. This behaviour is depicted in Fig. 10. Here 30 flows (all delay-based except for a single loss-based flow) compete for the available bandwidth. From 200 to 275 second, when the loss-based flow appears in the network, the delay-based flows behave as standard TCP flows and compete fairly for bandwidth. Otherwise they strive in a cooperative manner to keep queueing delay below a certain threshold (δ_{th}). Note also that the mode switching occurs automatically (and swiftly) without any complicated sensing or signal processing to determine whether or not the loss-based flows have left the network. We believe that this mechanism is novel in the context of delay-based congestion control.

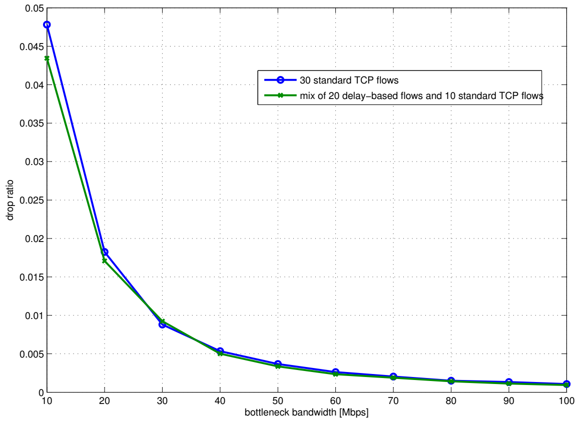
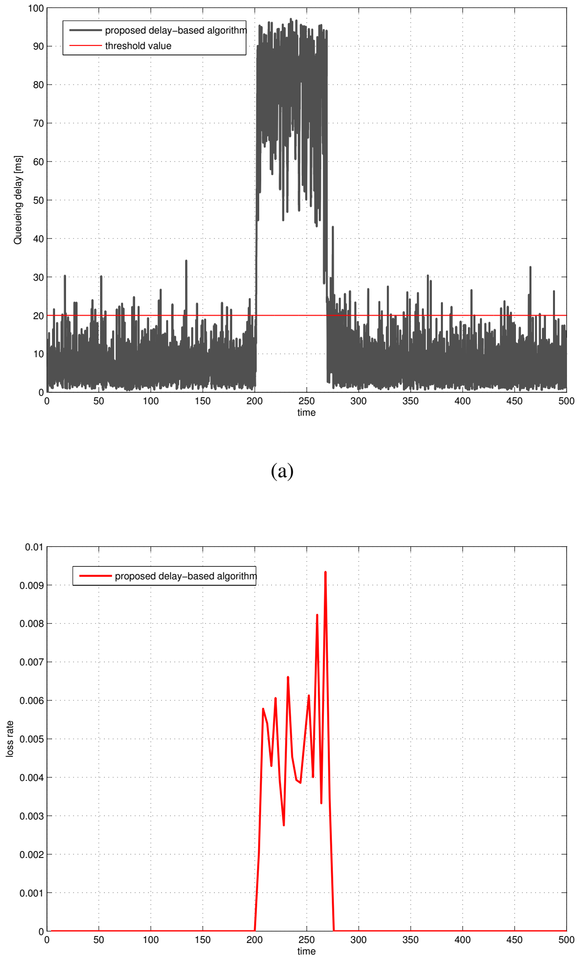


Fig. 9. Comparison in terms of loss rate: 20 delay-based flows coexisting with 10 standard TCP flows, and 30 standard TCP flows.



(a)

(b)

Fig. 10. Coexistence of delay-based flows with a single loss-based flow switching on and off: (a) queueing delay at the bottleneck; and, (b) loss rate.

B. Limitations

To conclude this section we note a number of potential limitations of proposed algorithm. **(A)** The maximum equilibrium loss rate is given by p_{max}. This means that the network will revert to a loss-based network if there is a very large number of network flows; namely, if the required equilibrium loss rate is greater than p_{max}. This property is very desirable as it is well known that estimation of queueing delay is difficult in networks with very large multiplexing of flows [26]. **(B)** Our algorithm works best in multiplexed environments with standing queues. In situations where this assumption is not valid, some unfairness may result. **(C)** A crucial part of the algorithm is the assumption that all flows use the same per-packet drop probability function and sense the same queueing delay. If this assumption is not valid, unfairness can be introduced. **(D)** The behaviour of networks (in the fluid limit) in which this algorithm is deployed is described by the Kelly framework [32].

IV. Stability of proposed solution

Next, an analytical description of the process described qualitatively in Section III is given. We demonstrate that N delay-based flows operating in the high-delay regime, will eventually return to low-delay regime (when loss-based flows are no longer present). More formally we show that this system has at most two equilibrium points, a low-delay and high-delay one. Furthermore, the low-delay equilibrium is stable, and the high-delay equilibrium is unstable. To demonstrate this we use a standard fluid model [32] to analyse the system of N flows with non-homogeneous RTT competing at a bottleneck link with capacity C.

By $\delta(t)$ and $W_i(t)$ $i = 1, \ldots, N$ we denote the queueing delay and $cwnd$'s at time t. Those quantities are related as $\sum_{i=1}^{N} \frac{W_i(t)}{RTT_i+\delta(t)} = C$. Recall in RED that in steady state the average window sizes do not depend on the round trip time; these are just a scaling for the speed of the evolution. Thus $W_1(t) = W_2(t) = \cdots = W_N(t) =: W(t)$ and together we get $W(t) = \frac{C}{\sum_{i=1}^{N} \frac{1}{RTT_i+\delta(t)}}$. Using the standard fluid model the evolution of $cwnd$ is given by:

$$\frac{\Delta W_i(t)}{\Delta t} = \frac{1}{RTT_i + \delta(t)} - \frac{q_0^i}{\Delta t} \cdot \frac{W_i(t)}{2}, \qquad (1)$$

where q_0^i is the probability that during the time interval $(t, t+\Delta t)$ a back-off of the i'th flow occurred. We denote by M_0^i the number of packets that belong to flow with $cwnd$ equal to $W_i(t)$ that are sent in the interval $(t, t+\Delta t)$. Then $M_0^i = \frac{\Delta t \cdot W_i(t)}{RTT_i+\delta(t)}$, and it follows that q_0^i can be approximated as: $q_0^i = 1 - (1-p)^{M_0^i} \approx pM_0^i = \frac{p\Delta t \cdot W_i(t)}{RTT_i+\delta(t)}$. It therefore follows that (1) can be written as:

$$\frac{\Delta W_i(t)}{\Delta t} = \frac{1}{RTT_i + \delta(t)} \left(1 - \frac{p}{2}\left(\frac{C}{\sum_{i=1}^{N} \frac{1}{(RTT_i+\delta(t))}}\right)^2\right) \qquad (2)$$

As the right hand side of this equation no longer depends on Δt we can now write $\dot{W}_i(t)$ instead of $\frac{\Delta W_i(t)}{\Delta t}$. The network equilibria are given by $\frac{p}{2}\left(\frac{C}{\sum_{i=1}^{N} \frac{1}{(RTT_i+\delta(t))}}\right)^2 = 1$. We denote by $p^*(\delta)$, the locus of equilibria: $p^*(\delta(t)) = \frac{2}{C^2}\left(\sum_{i=1}^{N} \frac{1}{RTT_i+\delta(t)}\right)^2$. Recall that the per-packet back-off rate p is a function of the delay δ: $p = g(\delta)$. Therefore, the system (1) is in equilibrium at the points of intersection of curves $p^*(\cdot)$ and $g(\cdot)$. Those two curves have zero or one, or two points of intersection $\delta_1^* < \delta_2^*$ (see Fig. 6). Our objective is to design the network so that there are two equilibria (the regular regime). Using the Lyapunov function $V(\delta(t)) = \delta(t)^2$, it is easily deduced that the right point of intersection (δ_2^*) is an unstable equilibrium and that the other left equilibrium (δ_1^*) is a stable one. It also follows that if δ becomes smaller than δ_2^* the system will be driven to the stable equilibrium δ_1^*. We formalise these statements in the following theorem.

Theorem 1: The system (1), has 0,1 or 2 equilibrium points: *(i)* If $p^*(\delta_{th}) < p_{max}$ there are two equilibrium points: $\delta_1^* < \delta_2^*$.The right equilibrium point δ_2^* is unstable and the left one, δ_1^* is stable. *(ii)* If $p^*(\delta_{th}) = p_{max}$ there is one, unstable, equilibrium and the $cwnd$ dynamics is mainly driven by the packet drops, when the queue is full. *(iii)* If $p^*(\delta_{th}) > p_{max}$ there is no equilibrium, and the $cwnd$ dynamics is mainly driven by the packet drops, when the queue is full.

Proof: For any given $\hat{\delta} \in \mathbb{R}$ we have that if $p(\hat{\delta}) < p^*(\hat{\delta})$ then $\dot{W}_i(t) > 0$, and if $p(\hat{\delta}) > p^*(\hat{\delta})$ then $\dot{W}_i(t) < 0$ for all $i = 1, \ldots, n$.
(i) Let W_i^1 be the steady state value of $W_i(t)$ at the equilibrium δ_1 for all $i = 1, \ldots, n$.
For the stability of the equilibrium at δ_1 we regard the following shifted system, where $\tilde{W}_i(t) = W_i(t) - W_i^1$ with $\tilde{\delta}(t) = \delta(t) - \delta_1$.
A Lyapunov function for this system is $V(t) = \tilde{\delta}^2(t)$. It is obviously proper and positive definite. And we have:

$$\dot{V}(t) = 2\tilde{\delta}(t)\dot{\tilde{\delta}}(t) = 2\tilde{\delta}(t)\frac{1}{C}(-C + \sum_{i=1}^{n} \frac{W_i(t)}{RTT_i + \delta(t)}) \qquad (3)$$

Here we have two distinct possibilities.

(a) if $\tilde{\delta}(t) < \delta_1$ then $\dot{\tilde{W}}_i(t) > 0$ for all $i = 1, \ldots, n$ and thus $-C + \sum_{i=1}^{n} \frac{W_i(t)}{RTT_i+\delta(t)} > 0$ and $\dot{V}(t)$ is negative

(b) if $\tilde{\delta}(t) > \delta_1$ then $\dot{\tilde{W}}_i(t) < 0$ for all $i = 1, \ldots, n$ and thus $-C + \sum_{i=1}^{n} \frac{W_i(t)}{RTT_i+\delta(t)} < 0$ and $\dot{V}(t)$ is again negative.

This proves that $V(t)$ is a Lyapunov function for our system and that the equilibrium at δ_1 is stable. The instability of the second equilibrium and part *(ii)* of the theorem follow when we consider the shifted system, where $\tilde{W}_i(t) = W_i(t) - W_i^2$

with $\tilde{\delta}(t) = \delta(t) - \delta_2$, where W_i^2 is the steady state value of $W_i(t)$ at the equilibrium δ_2 for all $i = 1, \ldots, n$. The same Lyapunov function as before ensures the instability. For part *(iii)* of the theorem for all possible $\delta \in \mathbb{R}$ we have that $p(\delta) < p^*(\delta)$. And hence $\dot{W}_i(t) > 0$ for all $t \geq 0$ and the claim follows. ■

Comment: To ensure that the system eventually returns to the low delay equilibrium we exploit the fact that the probability that $\delta < \delta_2^*$ at some time instant, is positive, due to the random effects in the network. This fact is sufficient to guarantee that the network converges to the stable equilibrium after loss-based flows leave the network.

V. Conclusion

In this paper we have presented a method that can be used to ensure that delay-based AIMD flows operate as loss-based flows when loss-based flows are present in the network, allowing fair coexistence with their loss-based counterparts, and otherwise revert to delay-based operation mode. This work was motivated by the PERT algorithm and its derivatives, which despite focusing on the coexistence issue failed to provide a fair solution capable of switching back to a low queueing delay regime once the loss-based flows are no longer present. Initial results provided for a single-bottleneck scenario indicate that this very simple idea is of merit, and requires more attention to address identified limitations as well as further evaluation in more complex scenarios.

References

[1] J. Martin, A. Nilsson, and I. Rhee, "Delay-based congestion avoidance for TCP," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 356–369, June 2003.

[2] R. Prasad, M. Jain, and C. Dovrolis, "On the effectiveness of delay-based congestion avoidance," in *Proc. of Int'l Workshop, PFLDnet*, 2004.

[3] A. Tang, J. Wang, S. Low, and M. Chiang, "Equilibrium of heterogeneous congestion control protocols," in *Proc. of the 24th IEEE INFOCOM Conference*, 2005.

[4] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. of SIGCOMM*, 1994, pp. 24–35.

[5] L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[6] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Comp. Commun. Rev.*, vol. 19, no. 5, pp. 56–71, 1989.

[7] Z. Wang and J. Crowcroft, "A new congestion control scheme: Slow Start and Search (Tri-S)," *ACM Comp. Commun. Rev.*, vol. 21, no. 1, pp. 32–43, 1991.

[8] ——, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," *ACM Comp. Commun. Rev.*, vol. 22, no. 2, pp. 9–16, 1992.

[9] C. Jin, D. X. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms, performance," Caltech, Tech. Rep. CSTR:2003.010, 2004.

[10] J. S. Ahn, P. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: emulation and experiment," *ACM Comp. Commun. Rev.*, pp. 185–195, 1995.

[11] G. Hasegawa, K. Kurata, and M. Murata, "Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet," in *Proc. of IEEE ICNP*, 2000.

[12] W. Feng and S. Vanichpun, "Enabling compatibility between TCP Reno and TCP Vegas," in *Proc. of the Symp. on Appl. and the Internet*, 2003.

[13] E. Weigle and W.-C. Feng, "A case for TCP Vegas in high-performance computational grids," in *Proc. of IEEE Int Symp High Perform Distrib Comput*, 2001, pp. 158–167.

[14] K. Srijith, L. Jacob, and A. L. Ananda, "TCP Vegas-A: Solving the fairness and rerouting issues of TCP Vegas," in *Proc. of IPCCC*, 2003.

[15] U. Hengartner, J. Bolliger, and T. Gross, "TCP Vegas revisited," in *Proc. of the 19th IEEE INFOCOM Conference*, 2000.

[16] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proc. of the 18th IEEE INFOCOM Conference*, 1999, pp. 1556–1563.

[17] C. Fu, L. C. Chung, and S. C. Liew, "Performance degradation of TCP Vegas in asymmetric networks and its remedies," in *Proc. of ICC*, 2001.

[18] H. Choe and S. Low, "Stabilized Vegas," in *Proc. of the 22th IEEE INFOCOM Conference*, 2003.

[19] S. Low, L. Peterson, and L. Wang, "Understanding Vegas: a duality model," *Journal of the ACM*, vol. 49, no. 2, pp. 207–235, March 2002.

[20] R. Jain and K. Ramakrishnan, "Congestion avoidance in computer networks with a connectionless network layer: concepts, goals and methodology," *Computer Networking Symposium*, pp. 134–143, 1988.

[21] J. Wang, D. X. Wei, and S. Low, "Modelling and stability of FAST TCP," in *Proc. of the 24th IEEE INFOCOM Conference*, 2005.

[22] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," 2005, Technical Report - Microsoft Research MSR-TR-2005-86.

[23] D. Leith, J. Heffner, R. Shorten, and G. McCullagh, "Delay-based AIMD congestion control," in *Proc. of Int'l Workshop PFLDnet*, 2007.

[24] D. Leith, R. Shorten, G. McCoullagh, L. Dunne, and F. Baker, "Making available base RTT for use in congestion control applications," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 429–431, June 2008.

[25] S. Bhandarkar, A. Reddy, Y. Zhang, and D. Loguinov, "Emulating AQM from end hosts," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 349–360, October 2007.

[26] G. McCullagh, "Exploring delay based TCP congestion control," Master's thesis, Hamilton Institute, NUI Maynooth, Ireland, 2008.

[27] K. Kotla and A. Reddy, "Making a delay-based protocol adaptive to heterogeneous environments," in *Proc. of 16th International Workshop on Quality of Service (IWQoS 2008)*, June 2008, pp. 100–109.

[28] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, April 2000.

[29] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. of the SIGCOMM Conference*, 2000, pp. 43–56.

[30] "The network simulator ns-2," http://www.isi.edu/nsnam/ns/.

[31] Ł. Budzisz, R. Stanojević, R. Shorten, and F. Baker, "A strategy for fair coexistence of loss and delay-based congestion control algorithms," *to appear in IEEE Commun. Lett.*, vol. 13, 2009.

[32] R. Srikant, *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. SpringerVerlag, 2004.