A Strategy for Fair Coexistence of Loss and Delay-Based Congestion Control Algorithms

Łukasz Budzisz, Rade Stanojević, Robert Shorten and Fred Baker

***Abstract*—Delay-based TCP variants have attracted a large amount of attention in the networking community because of their ability to efficiently use network resources, control queuing delays, exhibit virtually zero packet loss, etc. One major issue that discourages the wider deployment of delay-based TCP variants is their inability to co-exist fairly with standard loss-based TCP. In this note we propose a simple mechanism that allows delay- and loss-based (AIMD) TCP flows to compete fairly with each other. Further, our approach ensures that delay-based flows automatically (and swiftly) switch to a low-delay regime if no loss-based flows are present. We provide analytical and simulation results to validate presented algorithm.**

***Index Terms*—Delay-based AIMD congestion control.**

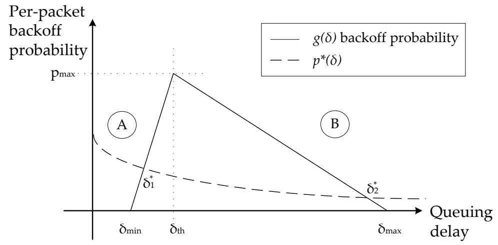


Fig. 1. Per-packet backoff probability as a function of observed delay

I. Introduction

A longstanding question in the networking community is whether queuing delay may be used as a reliable basis for network congestion control [1], [2]. Despite much work on this topic, 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 utilization of network links. Networks which exhibit this property are said to operate at the *knee of the throughput-delay* curve [3]. 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 [4], [5]; Microsoft Compound [6] (partially based on delay); more recent delay-based AIMD variants [7]–[9]; and this present work which is one 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] [10]; the inability of existing delay-based algorithms to maintain a low standing queue [10]; 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 which we do not repeat here [1], [2], [10]. Rather, we focus the specific issue of co-existence. We wish to develop delay-based algorithms that coexist fairly with loss-based counterparts. Note that the issue here is not just co-existence; after all, delay-based flows may simply switch to a loss-based mode once apacket loss is detected thereby solving the fairness problem. The issue that makes co-existence difficult is that delay-based flows must revert back to delay-based operation when loss-based flows are no longer present. Our principal contribution in this note is to propose a simple mechanism for achieving this in a robust and reliable manner.

II. Mode switching and coexistence

Assumptions: We assume that queueing delay 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 previous work: see [1], [2], [7], [10].

Basic idea: Our work is strongly motivated by the recent work of Reddy et al. [9]. In this paper the authors demonstrate that delay-based AIMD can be used to emulate AQM's by carefully selecting an end-user delay dependent back-off policy[1]. Our idea is to carefully select this policy to ensure that co-existence is achieved when loss-based flows are present in the network. Specifically, we select probabilistic backoff strategies of the form depicted in Fig. 1. As can be observed, the per-packet backoff probability function $p = g(\delta)$ has two parts; a part that increases monotonically with δ (Region A), and a part that decreases monotonically with δ (Region B). Assuming that p_{max} is large enough, the network stabilizes in Region A when only delay-based flows are present. 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 backoff probability. When loss-based flows switch off, the network cannot stabilize in this region due to a backward pressure exerted by the probability function. As the flows experience backoffs, the queueing delay reduces, thereby increasing the per-packet backoff probability. This process continues until the network stabilizes in Region A.

Analysis: Next, an analytical description of the process described qualitatively above is given. We demonstrate that N delay-based flows operating in the high-delay regime, will eventually return to low-delay regime. More formally we show

Ł. Budzisz, R. Stanojević, and R. Shorten, are with Hamilton Institute, NUI Maynooth, Ireland (e-mail: [Lukasz.Budzisz, Rade.Stanojevic, Robert.Shorten]@nuim.ie).

F. Baker is with Cisco Systems (e-mail: fred@cisco.com).

[1] In fact, in related work [11], Kotla proposes a strategy for co-existence with loss-based flows within this framework that is based on adjusting the aggressiveness of the delay-based flows. This strategy differs from our present strategy in which we use a carefully chosen back-off policy to achieve co-existence.

Algorithm 1 Pseudo-code for delay-based AIMD algorithm

```
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** $rand < p$ **then**
    reduce $cwnd$ to half
**else**
    increment $cwnd$ by $1/cwnd$
**end if**
```

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 standard fluid model [12] to analyze the system of N flows with a single RTT [2], competing at a bottleneck link with capacity C. Queuing delay and $cwnd$'s at time t, denoted here as $\delta(t)$ and $W(t)$, are related as:

$$\frac{N}{C}W(t) - RTT = \delta(t). \qquad (1)$$

Using standard fluid model the evolution of $cwnd$ is given by:

$$\frac{\Delta W(t)}{\Delta t} = \frac{1}{RTT+\delta(t)} - \frac{q_0}{\Delta t}\cdot\frac{W(t)}{2}, \qquad (2)$$

where q_0 is the probability that during the time interval $(t, t+\Delta t)$ a backoff occurred. We denote by M_0 the number of packets that belong to flow with $cwnd$ equal to $W(t)$ that are sent in the interval $(t, t+\Delta t)$. Then $M_0 = \frac{\Delta t \cdot W(t)}{RTT+\delta(t)}$, and it follows that q_0 can be approximated as: $q_0 = 1-(1-p)^{M_0} \approx pM_0 = \frac{p\Delta t \cdot W(t)}{RTT+\delta(t)}$. Therefore (2) can be written as:

$$\frac{\Delta W(t)}{\Delta t} = \frac{1}{RTT+\delta(t)}\left(1-\frac{p}{2}\left(\frac{C(RTT+\delta(t))}{N}\right)^2\right) \qquad (3)$$

The network equilibria are given by $\frac{p}{2}\left(\frac{C(RTT+\delta(t))}{N}\right)^2 = 1$. We denote by $p^*(\delta)$, the locus of equilibria: $p^*(\delta) = 2\left(\frac{N}{C(RTT+\delta)}\right)^2$. Recall that the per-packet backoff rate p is a function of the delay δ: $p = g(\delta)$. Therefore, the system (2) 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 $\delta_1^* < \delta_2^*$. Our objective is to design the network so that there are two equilibria (the regular regime). Given this basic setting we have 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: We prove item *(i)* here. Items *(ii)* and *(iii)* either follow directly or are proved analogously. Define a candidate Lyapunov function for the queue dynamics as $V(\delta(t)) = \delta(t)^2$. The assertions of the theorem follow from the fact that $\dot{V}(\delta(t)) = 2\delta(t)\dot{\delta}(t)$, and from the fact that $\dot{\delta}(t) > 0$ for all $\delta_{min} < \delta(t) < \delta_1^*$; $\dot{\delta}(t) < 0$ for all $\delta_1^* < \delta(t) < \delta_2^*$; and $\dot{\delta}(t) > 0$ for all $\delta(t) > \delta_2^*$. These facts follow directly from equation (1), (2) and (3). Namely, for queueing delays $\delta < \delta_1^*$, the dynamics of the congestion window "overcomes" the per-packet backoff rate and forces the network toward δ_1^*. Between δ_1^* and the apex of the per-packet backoff rate, the backoff rate is sufficiently large to overcome (2) and forces the network to δ_1^*. Between the apex and δ_2^*, this latter mechanism is reinforced by backoff rate that increases as δ decreases. Thus, using standard Lyapunov theory, one concludes that δ_1^* is a stable equilibrium, whereas δ_2^* is not. ■

Comment 1 (Loss based flows): Now suppose that at some time instant $\delta(t) > \delta_2^*$ due to the presence of a loss based flow. It follows that, once the loss based flow is no longer present, the probability that $\delta < \delta_2^*$ at some time instant, is positive, due to the random effects in the network. Given the above Lyapunov argument, this fact is sufficient to guarantee that the network converges to the stable equilibrium after loss-based flows leave the network.

Comment 2 (Choice of parameters): One method to select the values for δ_{min}, δ_{th} and p_{max} is to use the rules for RED parameter settings [9]. δ_{max} is estimated for each flow separately, the default value is $100\,[ms]$.

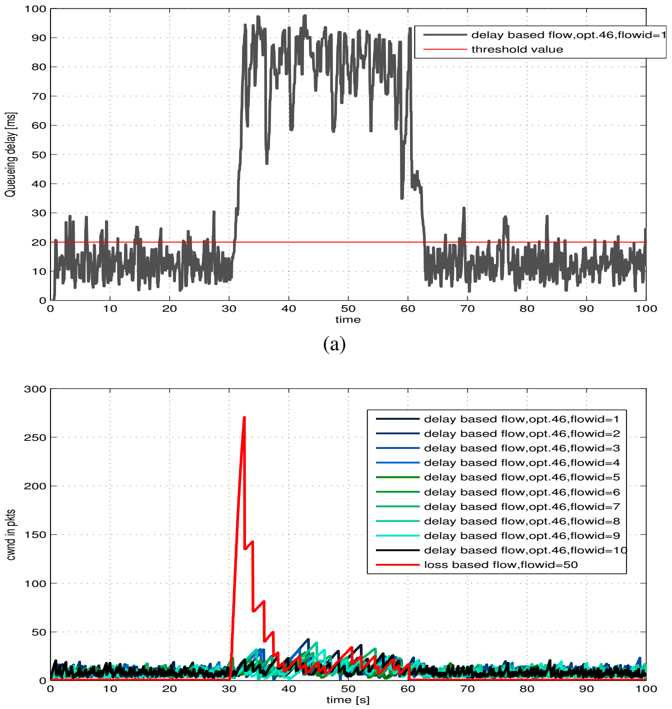
III. Simulation results

To illustrate the main features of our algorithm we present a number of $ns2$ simulations. We consider N flows competing at a bottleneck link of $40\,[Mbps]$. The maximum queuing delay is $100\,[ms]$ and RTTs are distributed randomly in the range $(30\,[ms], 130\,[ms])$. All delay-based flows operate conventional AIMD (TCP), but respond to a probabilistic backoff as well as loss. The backoff policy is linear as in Fig. 1 with: $\delta_{min} = 2\,[ms]$, $\delta_{th} = 20\,[ms]$, $p_{max} = 0.05$, and δ_{max} set to the maximum queueing delay.

Mode switching: Our primary objective 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. 2. Here 50 flows (all delay-based except for a single loss-based flow) compete for available bandwidth. From 30-60 seconds, when an intermittent loss-based flow appears, the delay-based flows behave as TCP flows and compete fairly for bandwidth. Note that in this region the queuing delay is high and the delay-based flows have a low probability of a probabilistic backoff. Otherwise they strive in a cooperative manner to keep queuing delay below a certain threshold. 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.

Throughput: Next, we investigate fairness properties in mixed environments. Fig. 3 illustrates the results of two experiments, for a $(80\%, 20\%)$ mix of delay/loss-based flows, and for a $(50\%, 50\%)$ mix, correspondingly. Both experiments are run over a 500 second period. Note that there is a slight bias in favor of the loss based flows. This is due to the fact that

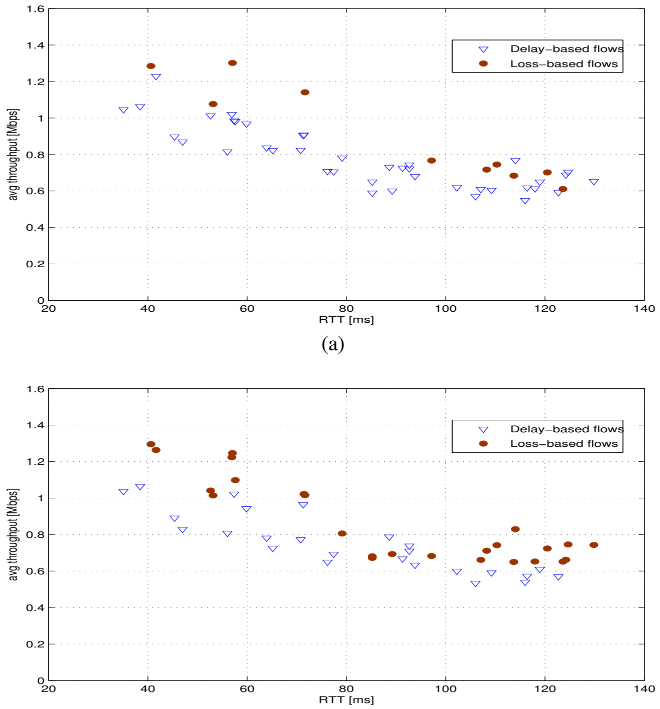
[2] Flows having homogeneous round-trip times is a technical assumption to simplify exposition. The argument can easily be extended to the heterogeneous round-trip time case.



(a)

(b)

Fig. 2. Coexistence of delay-based flows with a single loss-based flow switching on and off: (a) queueing delay at the bottleneck; and, (b) congestion window (to improve readability only first 10 delay-based flows are shown).



(a)

(b)

Fig. 3. Average throughput for network of 50 delay-based/loss-based flows. (a) $80\% - 20\%$ mix; (b) $50\% - 50\%$ mix.

the delay-based flows experience a small number of non-loss induced back-offs in the high-queue regime. However, this degree of unfairness can be controlled by carefully selecting the back-off policy. Notwithstanding this latter observation, the experiments nevertheless demonstrate very good co-existence of the delay-based and loss-based flows as measured by average throughput.

IV. Conclusion and Further Remarks

In this short 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, and otherwise revert to delay-based. Initial results indicate that this very simple idea is of merit.

To conclude the paper we note a number of potential limitations of our 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 [10]. **(B)** Our algorithm works best in multiplexed environments with standing queues. In situations where this assumption is not valid, some unfairness may result in mixed environments. **(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 [12].

Acknowledgement

This work was supported by the SFI grant 07/IN.1/I901.

References

[1] J. Martin, A. Nilsson, and I. Rhee, “Delay-based congestion avoidance for TCP,” *IEEE/ACM Trans. Networking*, 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. Int’l Workshop, PFLDnet*, 2004.

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

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

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

[6] 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.

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

[8] 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.

[9] S. Bhandarkar, A. L. Narasimha Reddy, Y. Zhang, and D. Loguinov, “Emulating AQM from end hosts,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 349–360, Oct. 2007.

[10] G. McCullagh, “Exploring delay based TCP congestion control,” Master’s thesis, Hamilton Institute, NUI Maynooth, Ireland, 2008.

[11] K. Kotla, “Delay-based protocol to heterogeneous environments,” Master’s thesis, Dept. of Comp. Science, Texas A&M University, 2008.

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