On the Effectiveness of Delay-Based Congestion Avoidance

Ravi S. Prasad
Georgia Tech
ravi@cc.gatech.edu

Manish Jain
Georgia Tech
jain@cc.gatech.edu

Constantinos Dovrolis
Georgia Tech
dovrolis@cc.gatech.edu

The debate between Loss-based Congestion Avoidance (LCA) and Delay-based Congestion Avoidance (DCA) is almost as old as TCP congestion control itself. Jacobson's original TCP congestion control [1] is the canonical example of LCA: packet losses indicate network congestion, and so a TCP transfer should decrease its window after a packet loss to reduce the load in the network. The obvious problem with LCA is that a TCP sender keeps increasing its window until it causes buffer overflows. These "self-induced" packet drops cause increased loss rate, decreased throughput, and significant delay variations (at least with Drop-Tail queues).

To deal with the previous issue, DCA schemes attempt to control the send-window of a TCP transfer based on Round-Trip Time (RTT) measurements. The basic idea is that if the send-window large enough to saturate the available bandwidth, the transfer will cause increasing queueing at the tight link of the network path,[1] and thus increasing RTTs. So, the sender should decrease the transfer's window when the RTTs start increasing.

There are several variations of DCA schemes. Starting with Jain's initial proposal in 1989 [3] and Mitra's fundamental work [4], the networking research community considered several ways to modify TCP, including TCP Tri-S [5], TCP Vegas [6], TCP BFA [7], and most recently TCP FAST [8]. Interest in DCA algorithms has been re-emerged in the last couple of years, as it becomes increasingly clear that Jacobson's LCA-based TCP cannot efficiently use high-bandwidth and long-distance network paths.

Recently, however, measurement studies have shown that there is little correlation between increased delays (or RTTs) and congestive losses [9], [10], [11]. This experimental observation raises major doubts on whether DCA algorithms would be effective in practice, as their main assumption is that RTT measurements can be used to predict and avoid network congestion [12].

Our objective in this short note is to suggest possible reasons for the weak correlations between delays and losses, and to identify conditions under which DCA schemes can fail to provide robust congestion control.

Maximum delay variation and network buffers. Suppose that the network path of a TCP transfer has a minimum RTT T_{min}. T_{min} is determined by the propagation and transmission delays along the path, and it does not include any queueing delays. Also, suppose that the tight link in the path has capacity C_t and buffer size B_t. The maximum queueing delay in that link is B_t/C_t. If queueing occurs only at the tight link, then the maximum RTT will be

$$T_{max} = T_{min} + \frac{B_t}{C_t} \tag{1}$$

DCA algorithms control the send-window based on the measured RTT variation $\Delta T \leq B_t/C_t$. A first issue here is that, if $B_t/C_t \ll T_{min}$, it may be hard to measure robustly increased RTTs due to tight link queueing. Random noise in the RTT measurements, which is unavoidable in practice, or errors due to limited RTT timestamping resolution, can be comparable or even larger than the queueing delays in the tight link. In that case, the sender may not be able to detect the increased queueing delays at the tight link and to avoid congestive losses.

RTT sampling rate. In DCA algorithms, the sender measures the RTT using the transfer's data packets. Even if every data packet generates an RTT measurement, the sender will effectively *sample* the path's RTT every $T_s = L/R$ seconds, where L is the data packet size and R is the transfer's send-rate over that time period. The RTT sampling rate will then be R/L. From Nyquist's theorem, however, it is known that if the sampling rate is less than twice the maximum frequency in the spectrum of the sampled signal, we cannot reconstruct that signal. Even if we do not aim in perfect signal reconstruction, the sampling rate should be at least comparable to the maximum signal frequency; otherwise, we are at risk of just sampling noise.

What determines the maximum frequency in the spectrum of the RTT signal, however? The RTT in a network path varies mostly due to queueing delays, and these delays are caused by variations in the incoming rate of the corresponding queues. To illustrate, consider the queue of the tight link at a time instant t_0. Suppose that just prior to t_0 the queue had some available bandwidth A and that the rate of our transfer was R. At t_0 a new cross traffic flow arrives at the tight link with rate $R_c > A$. This will cause a queue build-up at the tight link with rate $R_c - A$. If the tight link buffer was empty at t_0 and it has a maximum size of B_t bytes, it will take $\frac{B_t}{R_c - A}$ seconds for that buffer to overflow. In order for our transfer to detect this RTT increase on time, it should have a much lower sampling period, i.e.,

$$\frac{L}{R} \ll \frac{B_t}{R_c - A} \tag{2}$$

or, equivalently, its rate should be significantly higher than the rate with which the backlog increases at the tight link, normalized by the buffer size at that link,

$$R \gg \frac{R_c - A}{B_t/L} \tag{3}$$

[1] We prefer to use the term "tight link" rather than "bottleneck". The tight link of a network path is the link with the minimum available bandwidth [2].

The previous equation shows that if our transfer is a low-throughput flow in a high-bandwidth path (i.e., $R \ll A$), it may be unable to sample accurately the RTT variations in the path. High-bandwidth cross traffic flows will be causing significant RTT variations that our transfer will not be able to detect, and react to, on time. This conjecture agrees with the experimental and simulation results of [10] and [11], according to which DCA algorithms often react incorrectly to RTT variations, especially in high-bandwidth paths.

DCA in highly aggregated paths. DCA is often described and modeled considering a single transfer using the tight link of the path. Or, if cross traffic is part of the model, it is assumed to not cause any queueing delay variations. In that context, any variations in our transfer's RTT are caused by corresponding variations in that transfer's send-window. It is then relatively simple to design DCA algorithms that adjust the send-window based on RTT measurements, so that the TCP transfer fully utilizes the path but without causing congestion.

The situation is very different when we consider that the path carries multiple TCP transfers, and that they all dynamically adjust their windows based on the network state. Specifically, suppose that at some time instant our transfer's rate is R, while the capacity of the tight link is C_t, fully utilized by several cross traffic TCP connections with aggregate rate $R_c = C_t - R$. If $R \gg R_c$, our transfer will have a major impact on the tight link's queue, and so there will be a strong positive correlation between the window of our transfer and the measured RTTs. We can expect that a DCA algorithm would perform well in that case.

If however $R \ll R_c$, our transfer would be a minor contributor to the queueing delays at the tight link, and so the measured RTTs would be weakly correlated, or even uncorrelated, with that transfer's window. In that case, it is likely that our transfer will be reacting to RTT variations erroneously, mostly driven by the effect of cross traffic on the tight link, rather than by the effect of its own load. This is exactly what has been observed in [10] and [11]: DCA algrithms perform well in low-bandwidth tight links, such as dial-up models, because there is typically only one connection using those links at a time. In high-bandwidth links, on the other hand, where we typically have a large number of concurrent connections from different users, DCA schemes perform much worse as they cannot predict congestion robustly and they often vary their windows erroneously [10].

Dealing with losses. Even though a DCA-based transport protocol would be primarily reacting to RTT variations, it needs to also react, decreasing the send-window, to packet losses. Packet losses in network paths are unavoidable for two reasons. First, random packet drops occur independently of congestion due to bit errors, hardware/software errors, etc. Without global deployment of the Explicit Congestion Notification (ECN) bit, end-hosts are not able to distinguish between congestive and random losses and so they need to react to both. Second, congestive packet losses can be caused due to cross traffic, independent of our transfer's send-window. A TCP connection, however, has no way to infer whether an observed packet loss was caused because another flow needs to decrease its window; congestive losses can affect *any* active flow. So, DCA flows can be affected by packet losses and they need to react to them by decreasing their windows.

How should a DCA flow react to a packet loss? This is mostly a fairness issue. If DCA schemes do not decrease their windows by the same factor that LCA schemes do, significant unfairness in the bandwidth distribution can occur. We believe that it is necessary that any proposed DCA protocol specifies in detail how it reacts to packet drops, and how it maintains (or not maintains) fairness to existing TCP congestion control algorithms.

Another way to look at this issue is that, even though DCA algorithms are able to avoid self-induced packet losses, and thus to obtain higher throughput in cases where the latter are the only types of losses, DCA algorithms may not provide any performance benefit if they operate in network paths where random, or congestive losses due to cross traffic, are common. So, it is important that the evaluation of DCA schemes is also performed in lossy networks, as well as in networks where DCA flows compete with LCA flows.

Conluding remarks. The recent surge of interest around TCP FAST has brought new energy into the area of DCA-based transport protocols. Our objective in this short note was to identify some open issues regarding the effectiveness of DCA schemes. We believe that if the previous issues are not addressed, or resolved in some manner, it would be premature to consider replacing the existing TCP congestion control with much less understood DCA schemes.

REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of ACM SIGCOMM*, Sept. 1988, pp. 314–329.

[2] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 537–549, Aug. 2003.

[3] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Computer Communications Review*, vol. 19, no. 5, pp. 56–71, Oct. 1989.

[4] D. Mitra, "Asymptotically optimal design of congestion control for high speed data networks," *IEEE Transactions on Communications*, pp. 301–311, Feb. 1992.

[5] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-tahoe BSD TCP congestion control algorithm," *ACM Computer Communication Review*, vol. 22, no. 2, pp. 9–16, Apr. 1992.

[6] L. S. Brakmo and L.L.Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas of Communications*, vol. 13, no. 8, Oct. 1995.

[7] A. A. Awadallah and C. Rai, "TCP-BFA: Buffer Fill Avoidance," in *Proceedings of IFIP High Performance Networking Conference*, Sept. 1998, pp. 575–594.

[8] C. Jin, D. Wei, S. Low, G. Buhrmaster, J. Bunn, D. Choe, R. Cottrell, J. Doyle, H. Newman, F. Paganini, S. Ravot, and S. Singh, "FAST Kernel: Background Theory and Experimental Results," in *First International Workshop on Protocols for Fast Long-Distance Networks*, Feb. 2003.

[9] J. Andren, M. Hilding, and D. Veitch, "Understanding End-to-End Internet Traffi c Dynamics," in *Proceedings IEEE GLOBECOM*, Nov. 1998.

[10] J. Martin, A. Nilsson, and I. Rhee, "Delay-Based Congestion Avoidance for TCP," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 356–369, June 2003.

[11] S. Biaz and N. Vaidya, "Is the Round-Trip Time Correlated with the Number of Packets in Flight?," in *Proceedings Internet Measurement Conference (IMC)*, Oct. 2003.

[12] U. Hengartner, J. Bolliger, and Th. Gross, "TCP Vegas Revisited," in *Proceedings of IEEE INFOCOM*, Apr. 2000.