DFLOW: Low latency congestion control

Piers O'Hanlon
Oxford Internet Institute
University of Oxford
Email: piers.ohanlon@oii.ox.ac.uk

Ken Carlberg
G11
Email: carlberg@g11.org.uk

***Abstract*—This paper provides an overview of the DFlow congestion control algorithm, which aims to provide for lower delay and lower loss media transport. We provide an evaluation of the algorithm in a simulator which shows that it can provide for self-fairness and low delay operation. Furthermore we demonstrate that it can maintain reasonable throughput against TCP Vegas and LEDBAT.**

I. Introduction

This paper outlines DFlow, a congestion control algorithm that aims to minimise delay and loss by using delay-based techniques. The scheme is based upon TCP Friendly Rate Control (TFRC) [1], and adds delay-based congestion detection functionalities which feed into a 'congestion event history' mechanism based upon TFRC's loss history. This then provides for a 'congestion event rate' which drives the TCP equation.

Congestion control that aims to minimise the delay is important for real-time streams as high delay can render the communication unacceptable [2]. On today's Internet a number of paths have an excess of buffering which can lead to persistent high latencies, which has become known as Bufferbloat [3]. These problems are particularly apparent with loss-based congestion control schemes such as TCP, as they operate by filling the queues on a path till loss occurs, thus maximising the delay. The unfortunate consequence is that loss-based approaches not only lead to high delay for their own packets but also introduce delays and losses for all other flows that traverse those same filled queues.

Thus when competing with TCP, without the widespread deployment of suitable Active Queue Management (AQM), it is not possible to maintain low delay as TCP will do its best to keep the queues full and maximise the delay. Given the many different types of AQM mechanisms, to provide for low delay one would require an AQM mechanism that aims to minimise delay, such as Codel [4].

However there are many paths where the flows are not competing directly with TCP and where delay may be minimised. One example would be walled gardens, such as that realized in cellular network like Long Term Evolution (LTE), that have the ability to segment and differentiate traffic (e.g., toll quality VoIP versus best effort data). In these environments, types of data flows will be assigned different QoS Class Identifiers (QCI) that have minimal bounds corresponding to forwarding characteristics of a packet. Thus, backoff algorithms like DFlow may operate independently to other loss-based algorithms like those used in TCP.

The DFlow scheme can transport media with low delay and loss on paths where there is no direct competition with TCP in the same queue.

978-1-4799-1270-4/13/$31.00

II. Background

Whilst the existing standard for media transport, Real-time Transport Protocol (RTP) [5], suggests that congestion control should be employed, in practice many systems tend to use fixed or variable bit rate UDP and do very little or no adaptation to their network environment.

There are a number of commercial systems that now aim to provide for low latency operation such as Microsoft's Skype, Apple's Facetime/iChat, and Google's Hangouts. These systems are proprietary and closed so their behaviours are not well understood, though in broad terms they must provide reasonable service as they manage to maintain a significant user base. However there have been attempts [6] to characterise their algorithms, and some studies [7] have shown that they have their limits.

Most of the existing work on real-time congestion control algorithms has been rooted in TCP-friendly approaches but with smoother adaptation cycles. TCP congestion control is unsuitable for interactive media for a number of reasons including the fact that it is loss-based so it maximises the latency on a path, it changes its transmit rate to quickly for multimedia, and favours reliability over timeliness. Various TCP-friendly congestion control algorithms such as TFRC, Sisalem's LDA+ [8], and Choi's TCP Friendly Window Control (TFWC) [9] have been devised for media transport, that attempt to smooth the short- term variation in sending rate. More recently there have been some delay-based schemes developed which aim to provide for low delay.

A. Delay-Based schemes

In the last few years there has been a renewed interest in the use of delay-based congestion control for media, with the formation of the RTP Media Congestion Avoidance Techniques (RMCAT) Working Group within the Internet Engineering Task Force (IETF). Indeed the DFlow protocol is also specified as a draft [10] in the RMCAT working group. Currently there two other drafts; Google's RRTCC [11], and Cisco's NADA [12].

This latest movement has a slightly different emphasis to that of the history of TCP based approaches such as Jain's CARD, Wang and Crowcroft's Tri-S [13], Brakmo's Vegas [14], Tan et al's Compound TCP [15], CAIA Delay Gradient (CDG) [16], and more recently Budzisz's CxTCP [17]. Whereas the primary goal with media based transports is to actually minimise the latency of the flow, as opposed to just using delay as an early indication of loss. This is of particular relevance on paths with large queues, as is the case with a number of today's Internet paths. In 2007 Ghanbari et al [18] did some pioneering work on delay-based video congestion control using fuzzy logic based systems. Latterly there have been newer approaches from the RMCAT working group and others such as Sprout [7].

There has also been on going activity in the IETF as part of the Low Extra Delay Background Transport (LEDBAT) Working Group which aims to provide a less than best effort delay- based transport with lower delay. However the LEDBAT RFC [19] specifies a maximum one-way queuing delay target of 100ms which provides quite a high baseline for interactive media, considering the recommended total one-way delay limit for a VoIP call should be less than 150ms [2]. This effort is again not primarily motivated by the desire for low latency but more instead as provision of a background transport relative to TCP flows, where its most prominent deployment includes use as a BitTorrent transport.

B. TCP Friendly Rate Control (TFRC)

TFRC has, until recently, been seen as one of the most obvious choices for media transport. TFRC was primarily aimed at streaming media delivery where a smooth rate and TCP-friendliness are more important than low latency operation.

TFRC is a rate based receiver driven congestion control algorithm which utilises the Padhye TCP equation to provide a smoothed TCP- friendly rate. The sender explicitly sets the transmission rate, T, using the TCP equation driven by the loss event rate which is measured and fed back by the receiver, where a loss event consists of one or more packet losses within a single RTT. It utilises a weighted smoothed loss event rate, p, and EWMA smoothed RTT, as input to the TCP equation which enables it to achieve a smoother rate adaptation that provides for a more suitable transport for multimedia.

$$T = \frac{PacketSize}{RTT(\sqrt{2p/3} + 4 * (3\sqrt{3p/8})p(1 + 32p^2))} \quad (1)$$

However there are number of issues with TFRC as regards real-time media transport:

Loss-based operation: Firstly since it is a loss-based based scheme the latency is maximised which is a problem for real-time transport over heavily buffered paths. The other problem with loss-based protocols is that they rely on a certain level of packet loss which can be an issue for media traffic since lost media packet cannot usually be retransmitted in time. This problem becomes more of a concern at lower transmission rates since the TCP equation requires a corresponding increase in loss rates.

Bursty media flows: Many media flows exhibit bursty behaviour due to a number of factors. Firstly there may be negative bursts (i.e. gaps) due to silence or low motion which can lead oscillatory behaviours due to the data-limited and/or idle behaviours. Secondly there may be positive bursts (i.e. larger than normal) can also be due to the bursty nature of the media and codec (e.g. I-frames) which can lead to drops or increased latency. Whilst the current version of TFRC has attempted to address some of these issues, they are still a concern.

Small RTT environments: When operating in low RTT environments (<5ms), such as a LAN, systems implementing TFRC can have problems with scheduling packet transmissions as inter-packet timings can be lower than application level clock granularity. Whilst the current version of TFRC has attempted to address these issues, they can still be a concern in some low RTT environments.

Variable packet sizes: As originally designed TFRC will only operate correctly when packet sizes are close to MTU size, and when the packet sizes are much smaller fairness issues arise. Although there have been attempts to address this problem for small packets [20], it is not clear how to deal with flows that do vary their packet sizes substantially. However this issue is only really a marked problem with lower bit rate video flows or variable packet rate audio.

III. Objectives

The objectives of DFlow are to provide for low delay and low loss media transport when possible.

Lower Delay: By 'lower delay' we mean that the end-to-end delay of the flow is maintained close to the minimum RTT of the path, whilst achieving reasonable path utilisation. The one-way delay should be kept below 150ms to be deemed 'acceptable' [2], and should not exceed 400ms.

Lower Loss: For media transport it is important to minimise loss as it is usually not possible to retransmit within the delay budget for many connections. Whilst modern codecs can tolerate some loss it is beneficial to avoid it. The advantage of low delay congestion control is that since it aims to operate within the queuing boundaries it generally avoids loss.

Fairness: The system should aim to be reasonably fair with itself. Initially we aim for self fairness, and in future work we aim to tackle TCP fairness when we have a sufficiently robust loss-based competition mechanism. With a suitable loss-based competition mechanism we would aim to be obtain a fair share of the capacity though it is clear that the delay (and loss) would be largely beyond control. By 'fair' we roughly define it as each stream getting an approximately equal share of the capacity of the path given they operate under similar path parameters such as RTT and packet size.

Smoothness: The media rate should aim to be smooth within the constraints of the media, codec, and the network path. A smooth rate generally provides for more palatable media consumption.

IV. Design Outline

The DFlow scheme aims to primarily utilise delay measurements to drive the congestion control. It is currently based on some of the core aspects of TFRC, such as its rate based operation, utilisation of the TCP equation, and its rate smoothing. It also employs similar signalling mechanisms. In this section we outline the details of the delay-based mechanisms.

A. Delay Composition

The total end-to-end one-way delay (OWD) a packet incurs may be considered to consist of four elements; transmission (or serialisation), propagation, processing, and queuing delays. For our purposes the first three elements may be considered together as a largely static component, and termed the base delay. The base delay generally does not change significantly unless the node is mobile or the underlying link alters due to something like a route change. The main dynamic element of the delay, which DFlow aims to utilise, is the queuing delay. Taken together with the base delay, the queuing delay provides an indication of the actual path latency and also provides an insight as to the level of congestion on the path.

B. Delay Measurement

The notional one-way delay is measured for each packet by comparing the sender and receiver timestamps. Whilst the clocks on the sender and receiver are unlikely to be synchronised, it is assumed that their offset is relatively constant as the clock skew is generally quite small [21], also since the base delay measurement is regularly re-evaluated it's influence is limited. Thus the notional OWD may only be used in a relative context. The notional OWD is measured for each packet over two sampling periods; Firstly over the longer base_period (typically 10*RTT) from which the minima are stored as the base_delay. And secondly it is sampled over a shorter period current_period (typically 50ms), which is also filtered, usually also using a minima filter, and stored as current_delay. The minima of the OWDs are used to provide for some filtering of the measurement and reduce their noise, which can be beneficial in the case of variable link types such as wireless.

C. Congestion Detection

The delay-based detection algorithm, outlined in Fig.2, operates by comparing the current_delay to the base_delay, which gives an indication of the queuing delay. If it exceeds a set congestion detection threshold, cd_thresh, then the packet is considered for the next stage of detection. The cd_thresh sets the limit for the queuing delay incurred by the flow, and is typically set at 50ms (we have also investigated automated thresholds in Sec. V). Once a flow has exceeded its cd_thresh then it undergoes a second test which is based upon the gradient of the delay change over two current_period's, indicating that delay is on the increase, if it is positive then a 'congestion event' is flagged. We are still exploring the benefit of the second test as we have found it difficult to tune for differing rates, so we have omitted it in the simulations to simplify the situation.

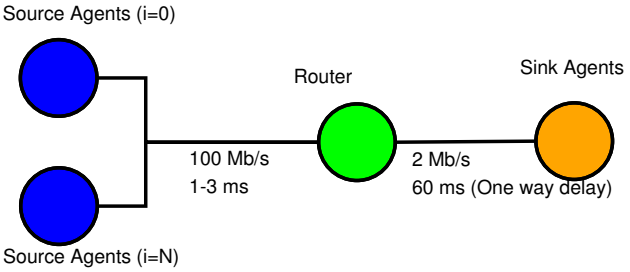

Fig. 1. Simulated Network Configuration

This algorithm then provides input to the 'congestion interval history' mechanism (based on TFRC's 'loss interval history'), which is combined with normal input from the TFRC packet loss detection mechanisms to form the 'congestion intervals'. These entries are then used to calculate the 'congestion event fraction' in a similar way to TFRC by using the Weighted Average Loss Interval mechanism to provide the 'loss rate', p, which is sent from the receiver to the sender, in a similar fashion to TFRC, where the new send rate is derived. If the resulting send rate calculated from the equation (1) is greater than the current send rate then it increases the rate by one packet per RTT, else it decreases the rate to at least the calculated rate.

Note that we currently disable TFRC's oscillation reduction mechanism from [22] (Section 4.5) as it adversely affects the delay-based operation.

D. Slow Start

The delay-based congestion detection is not only used during normal the congestion avoidance phase of the protocol but it also employed during slow start allowing for rapid, lower loss, attainment of the operating rate.

V. Evaluation

We implemented and tested DFlow in the ns2 simulator. Our implementation was based upon the TFRCSink Agent, which we modified to provide for DFlow. We added the delayed-based congestion detection mechanisms into the recv() function so that both delay and loss signals can feed into the congestion event history.

We ran 5 DFlows in the simulator across a bottleneck link of 2Mb/s, with a queue of 35 packets, an RTT of 120ms, and a delay target of 50ms. As may be seen from Fig.3 DFlow provides for low delay, whilst maintaining fairness between the flows, and ensuring good utilisation of the link. In this figure and the next we have also included a horizontal line marking the delay when the queue is full. Thus we can see that DFlow keeps the delay largely below its target threshold

If ((base_delay - current_delay) $>$ cd_thresh AND
(current_delay - prev_current_delay) $>$ 0)
DelayCongestionEvent = True

Fig. 2. Congestion Detection pseudo-code

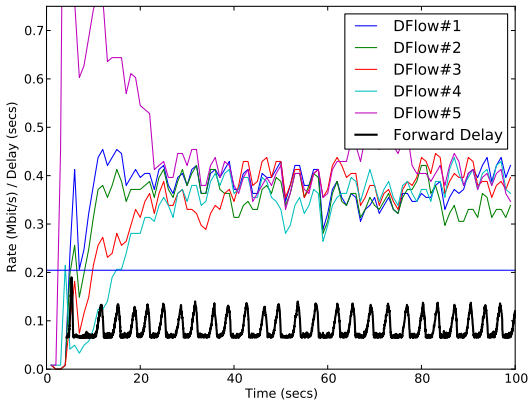

Fig. 3. DFlow flows

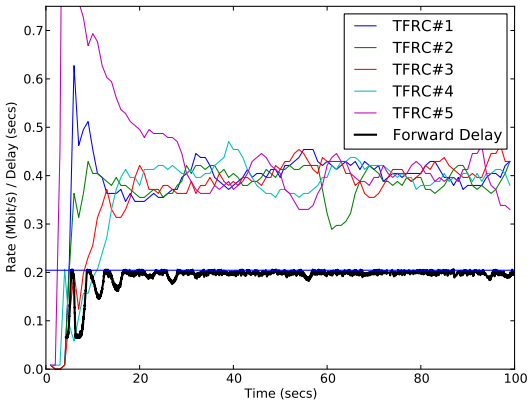

Fig. 5. TFRC Flows

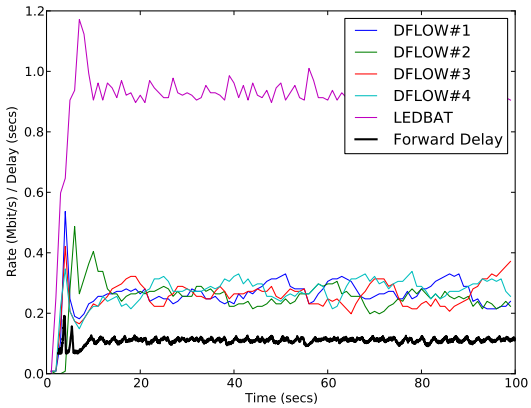

Fig. 4. DFlow vs. LEDBAT

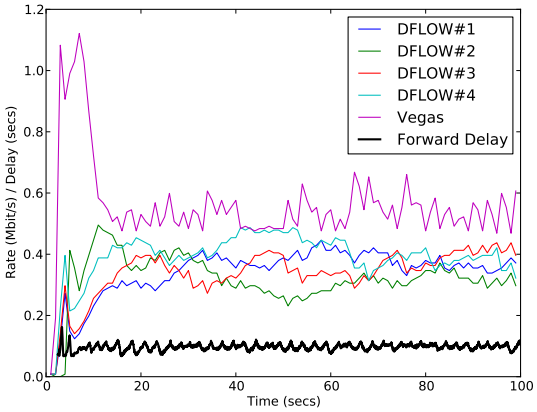

Fig. 6. DFlow vs. TCP Vegas

of 50ms. The actual loss rate incurred by DFlow is very low, with only 16 packets out of 22000 packets being lost.

It is clear that there are some oscillations occurring but they are not high enough to interfere with the low latency objective. The oscillation is related to DFlow's probing of the path's capacity, in a similar manner to TCP's sawtooth discovery cycles. All the flows are effectively synchronised as they flow through the same bottleneck and measure the same queue as it fluctuates, from which they derive their rate. In a more complex scenarios one would not see this synchronisation.

Analysis has shown that the period of these oscillations are inversely proportional to the RTT, as is the case with TFRC and TCP. It may be observed that DFlow does detect the rise in latency on the link but since it can only feedback congestion information once per RTT the rate at which these signals travel is limited by the RTT. Whilst the amplitude of the oscillations may be explained by the choice of the target threshold, so with a lower target the oscillation are lower. The problem with reducing the target too far is that it will destroy the throughput once the target gets within the jitter levels on the path.

We have also tried using the measured variance of the delay as the target delay which does work in some situations but in general it leads to a strong unfairness between the flows as faster flows have a higher variance so are self reinforcing. In theory other flows could measure the increased variance but their rate drops whilst they are pushed aside by the larger flows. The rate variance is generally larger with higher rate flows due to the nature of the TCP equation (1) which has a 'knee', which may be seen when plotted in Fig.7. On one side of the knee when the send rate is higher a small change in loss rate results in a large change in send rate, whilst on the other side of the knee, where the rate is generally slower a change in loss rate only results in small change in send rate.

A. Comparison with TFRC and TCP Vegas and LEDBAT

To understand how DFlow compares to TFRC we ran TFRC over a similar path, where it can be seen, in Fig.5, that the delay is maintained to full queue marker level, as expected. The loss rate incurred by TFRC it is much larger than DFlow, losing 300 packets out 23000 packets, although it does achieve slightly better overall throughput at the expense of delay.

Next we tested DFlow's behaviour against LEDBAT, a

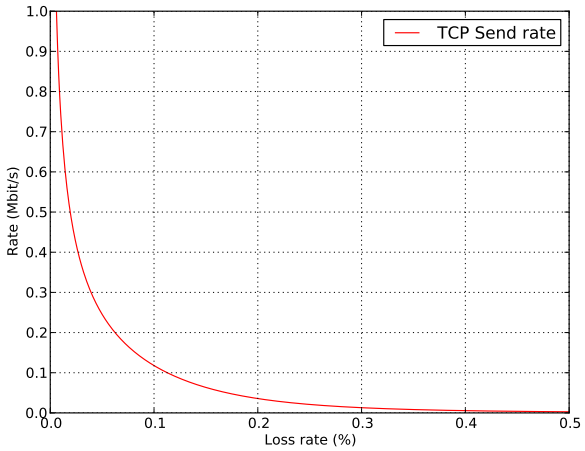

Fig. 7. TCP Equation rates over simulated path

delay-based version of TCP. From the simulation results in Fig.4 we can see that it competes comparatively well with it, achieving approximately a third the rate of LEDBAT. This is much better than when it attempts to compete with normal TCP, where it drops to near zero, due to the fact that TCP fills the queues leading to persist high latency. The reason that LEDBAT takes a larger share of the link is because it uses a more aggressive approach; Whilst we used comparative target of 50ms for LEDBAT, however it utilises a more aggressive back-off strategy where by it utilises additive decrease of the window [23].

Lastly we tested DFlow's behaviour against another delay-based version of TCP, namely TCP Vegas. From the simulation results in Fig.6 we can see that it competes comparatively well with it, achieving close to the same rate as TCP Vegas. The reason that TCP Vegas takes a larger share of the link is because it uses a more aggressive approach; Whilst the parameters of Vegas (α and β of 1 and 3 respectively) are lower than DFlow's, however it utilises a slightly more aggressive back-off strategy where by it also only performs additive decrease of the window.

VI. Conclusions

We have presented the DFlow congestion control algorithm that provides for self fairness, low delay, and good utilisation of path capacity. Whilst DFlow only provides for very limited throughput in the face of loss-based flows such as New Reno TCP, however we have shown that it does manage to compete with TCP Vegas and LEDBAT achieving a reasonable share of the capacity, given their more aggressive back-off strategy.

VII. Future Work

In future work we seek to address a number of areas. Firstly we are actively exploring the issue of better coexistence with loss-based protocols where we are investigating techniques to provide an 'implicit' loss rate based on the queue dynamics which aims to have some parity with TCP's state thus enabling adequate competition with TCP and other loss-based algorithms. It is noted that without additional network support it is clear that the delay (and loss) would be largely beyond control. Secondly we are working on mechanisms to manage the bursty nature of media allowing it maintain a smoother quality. Thirdly we are exploring the limits of the algorithm, in particular with respect to the use of the gradient measure.

Software Defined Networks (SDN) in walled gardens like LTE is a new and evolving research topic. Its potential in migrating cellular networks to a more flexible managed architecture, as outlined in [24], presents a new direction in dynamic traffic management that can leverage various capacity sharing algorithms like DFlow. The inclusion of SDNs in LTE has the potential of recognizing delay-based flows from loss-based flows and segmenting them with different aggregate queues and/or QCI values. This would require maintaining additional aggregate state at the edges of a walled garden, but the state can be expected to be minimal given the relative small number of participants in a cellular footprint (versus the aggregate ingress points between tier-2 and tier-3 providers). Future research can attempt to further develop congestion control algorithms like DFlow, and also potentially focus on indirect indicators such as the presence of Explicit Congestion Notification (ECN) support in RTP/UDP/IP flows.

We realise that there are issues with algorithms that utilise 'base delay' approaches such as Vegas, LEDBAT, and others. The majority of the approaches have utilised a measurement of the end-to-end delay a compare it to the current or filtered previous version of that delay. Unfortunately it is quite hard to make an reliable assessment of the actual path delay which has lead to a number of issues with some of these protocols. Such problems have been highlighted in the case of LEDBAT [25]. One of the problems is that the algorithms assume that a local minima or maxima is reliable in the face of constant background flows, but unfortunately it can lead to anomalous behaviour. However some of the original approaches and a number of the newer approaches, such as CAIA Delay Gradient (CDG), Google RRTCC and most recently Sprout, take a more relative approach and utilise delay gradient information which has less issues with constant background traffic levels providing for more a dynamic approach. Despite these new approaches there still seem to be some outstanding issues [26] as have been highlighted in Google's RRTCC.

It is apparent that there are still challenges in finding the right combination of latency and loss information to achieve suitable congestion control behaviour on the Internet today.

References

[1] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. New York, NY, USA: ACM, 2000, pp. 43–56.

[2] S. Abbas, M. Mosbah, A. Zemmari, and U. Bordeaux, "ITU-T Recommendation G.114 (05/2003), One-way transmission time," International Telecommunication Union (ITU), Tech. Rep., 2003.

[3] J. Gettys, "Bufferbloat: Dark buffers in the internet," *Internet Computing, IEEE*, vol. 15, no. 3, pp. 96–96, 2011.

[4] K. Nichols and V. Jacobson, "Controlling queue delay," *Commun. ACM*, vol. 55, no. 7, pp. 42–50, Jul. 2012.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550 (INTERNET STANDARD), Internet Engineering Task Force, Jul. 2003.

[6] L. De Cicco, S. Mascolo, and V. Palmisano, "A mathematical model of the skype voip congestion control algorithm," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, Dec. 2008, pp. 1410–1415.

[7] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2013, pp. 459–472.

[8] D. Sisalem and A. Wolisz, "LDA+: A TCP-Friendly Adaptation Scheme for Multimedia Communication," in *IEEE International Conference on Multimedia and Expo (III)*, 2000, pp. 1619–1622.

[9] S. H. Choi and M. Handley, "Fairer tcp-friendly congestion control protocol for multimedia streaming applications," in *Proceedings of the 2007 ACM CoNEXT conference*, ser. CoNEXT '07. New York, NY, USA: ACM, 2007, pp. 54:1–54:2.

[10] P. O'Hanlon and K. Carlberg, "Congestion control algorithm for lower latency and lower loss media transport," INTERNET-DRAFT draft-ohanlon-rmcat-dflow-02, March 2013.

[11] H. Lundin, S. Holmer, and H. Alvestrand, "A google congestion control algorithm for real-time communication on the world wide web," INTERNET-DRAFT draft-alvestrand-rmcat-congestion-00, August 2013.

[12] X. Zhu and R. Pan, "Nada: A unified congestion control scheme for real-time media," INTERNET-DRAFT draft-zhu-rmcat-nada-01, March 2013.

[13] Z. Wang and J. Crowcroft, "A new congestion control scheme: Slow start and search (tri-s)," *ACM Computer Communication Review*, vol. 21, pp. 32–43, 1991.

[14] 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, pp. 1465–1480, 1995.

[15] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound tcp approach for high-speed and long distance networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –12.

[16] D. A. Hayes and G. Armitage, "Revisiting tcp congestion control using delay gradients," in *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part II*, ser. NETWORKING'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 328–341.

[17] L. Budzisz, R. Stanojevic, A. Schlote, F. Baker, and R. Shorten, "On the fair coexistence of loss- and delay-based tcp," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1811–1824, 2011.

[18] E. Jammeh, M. Fleury, and M. Ghanbari, "Delay-based congestion avoidance for video communication with fuzzy logic control," in *Packet Video 2007*, Nov. 2007, pp. 8 –17.

[19] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," RFC 6817 (Experimental), Internet Engineering Task Force, Dec. 2012.

[20] S. Floyd and E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant," RFC 4828 (Experimental), Internet Engineering Task Force, Apr. 2007.

[21] J.-M. Combes, S. Krishnan, and G. Daley, "Securing Neighbor Discovery Proxy: Problem Statement," RFC 5909 (Informational), Internet Engineering Task Force, Jul. 2010.

[22] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 5348 (Proposed Standard), Internet Engineering Task Force, Sep. 2008.

[23] G. Carofiglio, L. Muscariello, D. Rossi, C. Testa, and S. Valenti, "Rethinking the low extra delay background transport (ledbat) protocol," in *Computer Networks*, 2013.

[24] L. E. Li, Z. M. Mao, and J. Rexford, "CellSDN: Software-Defined Cellular Networks," Princeton University, Tech. Rep., 2012.

[25] D. Ros and M. Welzl, "Assessing LEDBAT's Delay Impact," *Communications Letters, IEEE*, vol. 17, no. 5, pp. 1044–1047, 2013.

[26] L. D. Cicco, G. Carlucci, and S. Mascolo, "An experimental investigation of the google congestion control for real-time flows," in *ACM SIGCOMM 2013 Workshop on Future Human-Centric Multimedia Networking*, 2013.