A Framework for Less than Best Effort Congestion Control with Soft Deadlines

David A. Hayes, David Ros, Andreas Petlund and Iffat Ahmed
Simula Research Laboratory
Fornebu, Norway
Email: {davidh,dros,apetlund,iffat}@simula.no

***Abstract*—Applications like inter data-centre synchronisation or client-to-cloud backups require a reliable end-to-end data transfer, however, they typically do not have strong capacity or latency constraints, just a loose delivery deadline. Besides, their potential to disrupt more quality-constrained flows should be kept to a minimum. These applications could be well served by a transport protocol providing a *less-than-best-effort* (LBE) or *scavenger* service rather than TCP but, neither TCP nor standard LBE methods like LEDBAT consider any notion of deadline or completion time. TCP simply tries to maximise the use of available capacity, while LEDBAT tries to enforce an LBE behaviour regardless of any timeliness requirements.**

This paper introduces a framework for adding both LBE behaviour and awareness of "soft" delivery deadlines to *any* congestion control (CC) algorithm, whether loss-based, delay-based or explicit signaling-based. This effectively allows it to turn an arbitrary CC protocol into a scavenger protocol that dynamically adapts its sending rate to network conditions and remaining time before the deadline, to balance timeliness and transmission aggressiveness. Network utility maximization (NUM) theory provides a solid foundation for the proposal. The effectiveness of the approach is validated by numerical and simulation experiments, with TCP Cubic and Vegas used as examples.

I. Introduction

Many bulk data transfer applications do not need to consume as much capacity as the network can provide them. Instead they may be able to use a less-than-best-effort (LBE) service model [1], also called "scavenger", to send their data at a lower rate than that of a typical best effort (BE) service.

There are different ways in which an LBE service can be implemented; for instance, some form of priority scheduling can be used in routers to allocate residual capacity to LBE traffic flows. In this work we focus on *end-to-end LBE congestion control* (CC), that is provided by algorithms running in end-hosts as part of a transport protocol.

In particular we address data backup and replication applications that need to traverse networks that are not necessarily controlled by the application provider. Such a service should: (i) keep disruption of concurrent BE interactive services to a minimum; (ii) have a timeliness constraint, i.e., the transfer should be finished by a soft deadline to fit in with other network activity and to ensure a timely correctness of replicated data; (iii) be able to achieve Item ii, by dynamically adjusting its aggressiveness in competing with BE traffic from that of a scavenger-type service up to that of a BE-type service.

ISBN 978-3-901882-94-4

This paper introduces a framework for providing LBE congestion control that can trade (lack of) aggressiveness for meeting loose deadlines. To the best of our knowledge, no such *deadline-aware* LBE CC methods have ever been proposed in the literature. Such a deadline-aware LBE service would require API support to provide (i) the size of the data to transfer, and (ii) the soft completion time for the transfer. This may be done through a simple update to the transport API (such as an IOCTL) or by using a middleware framework that provides the necessary abstractions (like [2], [3]), however, the focus of this paper is on the CC mechanism itself.

We adapt and extend a network utility maximization (NUM) [4]–[6] based model where congestion control specific prices, like loss and delay, are mapped to a universal price measure [7]. This allows us to design a soft-deadline aware LBE service that can adapt *any* available end-to-end CC, independent of the type of congestion control or the congestion signal(s) used for making sending-rate decisions. As we will see, different CCs and congestion signals may provide different performance and tradeoffs. However, the framework can leverage whatever CC methods and congestion signals are available on a particular networking stack, and can support future, improved CC methods when they become available. This flexibility is one of the major contributions of this work.

The paper is organised as follows. Section II discusses the NUM theory our method builds on. Section III presents the design of the basic method and the mathematical model behind it, numerical validations and simulations, and pseudo-code for a loss-based TCP Cubic implementation. Section IV extends the basic design to the more realistic case of different CC methods coexisting in the network, and provides an implementation in pseudo-code for a delay-based TCP Vegas implementation. Section V evaluates, by means of simulations, how the proposed method performs, with both loss and delay-based CC flavors and under realistic cross-traffic patterns. Section VI positions our work and contributions with respect to previous work. Finally, Section VII concludes the paper.

II. Background: Network Utility Maximization

This work uses NUM, pioneered by Kelly [4], [5] and Low and Lapsley [6] as a basis for the design of a *deadline aware – less than best effort* (DA-LBE) service. With NUM the network congestion control problem is framed as an optimization problem where we seek to maximize the utility

traffic sources get from their send rates, U_s, subject to not exceeding the capacity limit of network links:

$$\max_{x \geq 0} \sum_{s=1}^{S} U_s(x^{(s)}) \tag{1}$$

$$\text{subject to} \sum_{s \in \mathbb{S}(l)} x^{(s)} \leq c_l \qquad \forall l = 1, \ldots, L \tag{2}$$

where $x^{(s)}$ is the send rate of source or flow s, c_l is the capacity of network link l, and $\mathbb{S}(l)$ is the set of flows that traverse link l. This optimization problem may be solved in a distributed manner through its Lagrangian dual:

$$\min_{p \geq 0} \left\{ \max_{x \geq 0} \left(\sum_{s=1}^{S} U_s(x^{(s)}) - \sum_{l=1}^{L} p_l \left(\sum_{s \in \mathbb{S}(l)} x^{(s)} - c_l \right) \right) \right\} \tag{3}$$

where the Lagrange multiplier, p_l, can be thought of as the "price" of congestion that the network assigns to each link. Each source, s, determines its send rate $x^{(s)}$ using the total measure of congestion along its end-to-end path:

$$x^{(s)} = \left((U_s)' \right)^{-1} \left(\sum_{l \in \mathbb{L}(s)} p_l \right) \tag{4}$$

where $\mathbb{L}(s)$ is the set of links that source s uses from end to end through the network. This framework fits our problem well. A congestion controller could "artificially" inflate or discount these congestion prices to change its relative share of network capacity.

Using NUM, Trichakis *et al.* [8] investigate a scenario where there may be a minimum rate requirement for particular flows in a particular time period. In order to meet the contracted rate, flows discount the measured congestion price, allowing them to achieve a higher relative send rate. In the case of a DA-LBE service we generally need to *inflate* the measured congestion price to allow a lower relative send rate; but reduce the amount of inflation, when necessary, to be able to deliver the desired amount of data before a deadline.

III. Deadline-Aware LBE Congestion Control

A DA-LBE traffic source should: (i) be no more aggressive than BE traffic, (ii) react appropriately to network congestion, (iii) take advantage of available network capacity when there is no congestion, and (iv) attempt to finish transmitting its data by the deadline. We model an LBE service in the NUM framework as a traffic source s that inflates its measured network price, $q^{(s)}$, by some weight $w^{(s)} \in [w_{\min}, w_{\max}]$:

$$x^{(s)} = \left((U_s)' \right)^{-1} \left(\frac{q^{(s)}}{w^{(s)}} \right) \tag{5}$$

where $q^{(s)} = \sum_{l \in \mathbb{L}(s)} p_l$. When $w^{(s)} = w_{\min}$, the send rate $x^{(s)}$ is at its lowest LBE rate, and when $w^{(s)} = 1$, the send rate $x^{(s)}$ is equivalent to a BE rate. We term the degree of LBE service *LBEness*. In this work, we choose $w_{\max} = 1$ to limit aggressiveness to be equivalent to that of BE traffic, however, choice of $w_{\max}$ can be a matter of policy.

Initially $w^{(s)} = w_{\min}$ for maximum LBEness (that is, maximum price inflation), but it is adjusted periodically with respect to the closeness of the "soft deadline" at intervals of duration T_w. This allows the traffic source to react to the dynamic network conditions over short time scales, but vary its LBEness over longer time scales in order to complete transmission of the data by the deadline. The manner of setting the weight w may be driven by policy which determines how optimistic one might be as to whether network congestion will improve before the deadline, knowledge of typical network conditions, or through prediction of future network conditions. For simplicity, in this paper we use the current network conditions as indicative of future conditions.

The lowest send rate for a source that will meet the deadline after the n^{th} interval of duration T_w, i.e., at $t_n = t_{n-1} + T_w$, is given by:

$$\zeta(t_n, t_D) = \frac{\text{data remaining}}{t_D - t_n} \tag{6}$$

where t_D is the soft deadline for completion of the data being sent by the source. This target rate ζ is used to determine an appropriate weight w for the source. We explore two methods of adapting w: a proportional-integral-differential (PID) controller and model-based-control (MBC).

A. PID-based update

PID controllers base their control on the error ϵ between the current state and the target state. The control signal from a PID controller is based on the weighted combination of the current error, past history, and the projected error. We use a normalized error signal mapped to w to enable easy scaling:

$$\epsilon_n = \frac{\zeta(t_n, t_D) - \bar{x}(t_{n-1}, t_n)}{\bar{x}(t_{n-1}, t_n)} w_{n-1} \tag{7}$$

where $\bar{x}(t_{n-1}, t_n)$ is the average send rate over the preceding T_w interval $(t_{n-1}, t_n]$, and w_{n-1} is the weight used by the source in the preceding interval.

The PID control signal u_n is calculated as follows

$$I_n = I_{n-1} + T_w \epsilon_n \tag{8}$$

$$u_n = K_p \epsilon_n + K_i I_n + K_d \frac{\epsilon_n - \epsilon_{n-1}}{T_w} \tag{9}$$

where K_p, K_i, and K_d are the weights (or gains) for proportional, integral, and differential parts respectively. We update w as follows:

$$w_n = [u_n]_{w_{\min}}^{1} \tag{10}$$

where $[y]_a^b \triangleq \min(b, \max(y, a))$.

The most difficult part of a PID controller is determining the values of K_p, K_i, and K_d. Using a normalized error signal helps to ensure the chosen gains are widely applicable.

B. MBC-based update

Model-based control relies on having a good model for protocol send rate with respect to the network price, in

TABLE I
TCP AND DA-LBE SOURCES

Source	Start	Finish	Deadline	Size
TCP 1	0 s	600 s	-	greedy
TCP 2	200 s	1000 s	-	greedy
TCP 3	800 s	1000 s	-	greedy
TCP 4	1010 s	1600 s	-	greedy
TCP 5	1200 s	2000 s	-	greedy
TCP 6	1400 s	1800 s	-	greedy
DA-LBE	400 s	-	1700 s	$\frac{\eta c_l}{(1700-400)}, \eta = \{0.1, 0.3\}$

particular what network price would achieve the desired LBE bit rate:

$$\hat{q}_n = M(\zeta(t_n, t_D), \text{RTT}_{\min}, \dots) \tag{11}$$

or in terms of network utility

$$\hat{q}_n = (U_s)'(\zeta(t_n, t_D)) \tag{12}$$

where $\hat{q}_n$ is the model estimated network price that will achieve the desired bit rate at interval n, and $\text{RTT}_{\min}$ is the RTT when there is no queuing.

The control estimates the model error as the relative difference between the target for the interval $(t_{n-1}, t_n]$ and the send rate measured for that target at the end of the interval[1]:

$$\epsilon_{n-1} = \frac{\zeta(t_{n-1}, t_D) - \bar{x}(t_{n-1}, t_n)}{\bar{x}(t_{n-1}, t_n)} . \tag{13}$$

The new weight is then based on the model's estimate of the price that will achieve the desired rate $\hat{q}_n$, with respect to the actual network price q_{n-1} measured in the previous interval, corrected by the error between target and measurement ϵ_{n-1}:

$$w_n = \left[\frac{q_{n-1}}{\hat{q}_n}(1 + \epsilon_{n-1}) \right]_{w_{\min}}^{1} . \tag{14}$$

C. Numerical validation

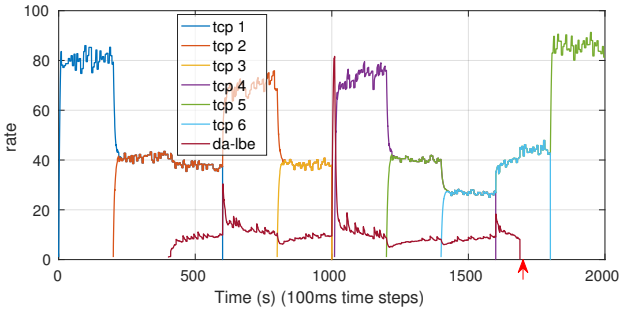
We conduct a simple numerical experiment to test the efficacy of the proposed scheme. We model a network with a maximum capacity of $c_l = 100$ that randomly varies around an average of 90 units, changing every 5 s. Over the experimental time of 2000 s six normal TCP sources start and complete at various times as well as a competing DA-LBE source (see Table I). The DA-LBE sources send a file of size equal to either 10% or 30% of the network capacity over the start to deadline target duration of the session. We use a w update period of $T_w = 10$ s in our experiments. In practice T_w needs to be long enough to obtain a good estimate of the average send rate and short enough to allow the DA-LBE sender to meet the target deadline if network conditions change.

We use the alpha-fair family of utility functions first proposed by Mo and Walrand [9]:

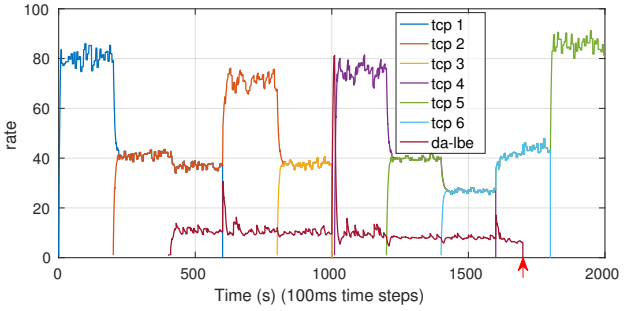
$$U(x) = \begin{cases} \log x & \alpha = 1 \\ (1-\alpha)^{-1} x^{(1-\alpha)} & \alpha > 1 \end{cases} \tag{15}$$

$$U'(x) = x^{-\alpha} . \tag{16}$$

[1]Note that the base term is slightly different to the error value used in the PID based control since it estimates the error in the model rather than the error between the control and the target.



(a) PID



(b) MBC

Fig. 1. Numerical experiments with $\alpha = 2$ in (15). Soft deadline is shown with an arrow.

All traffic sources, including the DA-LBE source have the same utility function with $\alpha = 2$ which reflects TCP fairness [10] and calculate their send rate as follows:

$$x_n^{(s)} = \left(\frac{q_n}{w_n^{(s)}} \right)^{-\frac{1}{2}} \tag{17}$$

where $w_n^{(s)} = 1, \forall n$ for all sources that are not DA-LBE sources. In this numerical experiment we model TCP adjusting its send rate incrementally, with the actual send rate:

$$a_n^{(s)} = a_{n-1}^{(s)} + \gamma \left(x_n^{(s)} - a_{n-1}^{(s)} \right) \tag{18}$$

where $\gamma = 0.02$ in these experiments.

For this experiment we set the price as the probability of a packet drop on a single bottleneck link modeled as a M/M/1/K ($K = 100$) queue of capacity c_l and load $\rho = \frac{\sum_s a^{(s)} - c_l}{c_l}$.

Figure 1 shows the results using both the PID and MBC based control. In this experiment the control model perfectly matches the system, so the resulting control resembles a critically damped system. The PID controller works on a normalized error signal and gains of $K_p = 0.5$, $K_i = 0.03$, and $K_d = 0.1$ for the proportional, integral and differential elements respectively. It can be difficult to tune PID controller gains, this being a little overdamped, but appropriate for a LBE service. The PID based control can be applied to any TCP CC mechanism, and normalizing the error signal helps to make the gains more widely applicable. However, many TCP congestion control mechanisms have good models, hence, the model-based approach may prove to be the more robust in the end. For this reason we explore MBC based control in the remainder of this paper.

In the numerical experiments the DA-LBE flow is able to use the spare capacity during the short 10 s interval at 1000 s

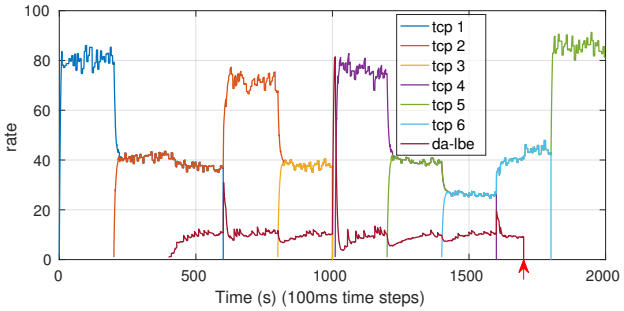


Fig. 2. MBC control with w increase rate limiter ($l_w = 0.05$), $\alpha = 2$ for all flows.

when there is no other traffic. Ideally we would also like DA-LBE to give way to other traffic during heavy loads. The PID controlled DA-LBE (Fig. 1a) gives way at first when there is an increase in the other traffic. The MBC based control (Fig. 1b) gives way in the same circumstances, but only for a short period, more strictly keeping to its target rate in order to meet the deadline. Since the deadline is soft and ensuring LBEness is important, we add a limit to the rate of increase of w, while not restricting the decrease in w, as follows[2]:

$$\hat{w}_n = \begin{cases} w_{n-1} + l_w w_n & (w_n - w_{n-1}) > l_w w_n \\ w_n & \text{otherwise .} \end{cases} \quad (19)$$

Figure 2 shows that limiting the w's rate of increase in this way allows the DA-LBE flow to give way more to other traffic. This has potential to cause it to overshoot its soft deadline, especially when there are heavy loads near the deadline. The gradient limit then becomes an additional parameter that determines how slowly the DA-LBE flow should adapt back to its target rate when it experiences heavy loads.

D. Applying the framework to the protocol stack

In this section we look at how this theory can be applied with minimal changes to the protocol stack and demonstrate its performance using NS2. To achieve LBEness the congestion price a congestion controller uses needs to be inflated. For a loss based CC mechanism, e.g., Cubic, this could be done by dropping packets to cause more loss, however, a better method would be to inflate the price by generating "phantom" ECN signals. Indirectly this could be done by changing the congestion controller's reaction to congestion, i.e. the multiplicative decrease factor commonly referred to as β. We investigate both of these methods.

Some CC mechanisms use delay as well as loss. This can enable earlier detection of congestion than relying on packet loss and may make it more suitable for an LBE mechanism. We investigate this in Sec. IV.

1) Phantom ECN: For a loss based CC mechanism the simplest way to inflate the congestion price is to generate additional *phantom* loss or ECN episodes. ECN is preferred since retransmissions are not part of the congestion response. The phantom ECN method increases the number of congestion indications, resulting in faster congestion window (cwnd) oscillations over a smaller cwnd range. This results in good short time scale LBE transmission characteristics. A draw back of this method of inflating the price is that generating phantom ECN signals can prevent the mechanism taking advantage of short periods of increased available capacity. Ideally it would be good to have a "no congestion" signal to trigger the cessation of phantom ECN generation. In loss based CC mechanisms this can only be inferred from the absence of packet loss. A delay based signal could supply this information.

2) Adjusting the decrease factor β: An indirect way of inflating the price is to inflate the response to congestion. In an AIMD CC, changing the β factor can achieve this. This allows the congestion controller to take advantage of short periods of increased available capacity, but it relies on a high enough congestion indication rate to be effective. This restricts the short term LBEness that can be achieved, especially in high bandwidth delay product (BDP) environments.

3) Adapting loss based TCP to be DA-LBE: Figure 3 shows the basic algorithm for adapting a loss based TCP to be DA-LBE. The mechanism has two parts: (i) the update of the price weight w every T_w, and (ii) either generate phantom ECN signals based on w or back off more aggressively. Function u_{PID} adapts w as a PID controller, and function u_{MBC} calculates w based on the steady state Cubic model [11] and the measured error. This algorithm is implemented outside the NS2 Linux congestion control module allowing the PID controller to work with any similar loss based CC mechanism.

The phantom ECN method measures the average time between real congestion indications, $\bar{\tau}_{\text{cong}}$, and stops sending phantom signals if more than $v\bar{\tau}_{\text{cong}}$ has elapsed since the last real congestion indication ($v = 3$ in these experiments).

We use NS 2.35 to run similar experiments to those in Sec. III-C. We use the dumbbell topology where access links have a capacity of 1 Gbps with a 1 ms propagation delay, and the bottleneck link has a capacity of 100 Mbps, BDP length buffer and a propagation delay of 10 ms. Exponentially distributed inter-packet time background traffic (1500 B packets) at an average of 10 Mbps is sent over the bottleneck in addition to the traffic described in Table I. We use the Cubic NS2 Linux congestion control [12] as the TCP sources and adapt this as outlined previously to be our DA-LBE traffic source.

Figure 4 illustrates Cubic based DA-LBE through phantom ECN generation with both the PID and MBC control. Both controls perform well. The w increase rate limiter allows the MBC control to give way more readily to other traffic where proximity to the deadline permits.

At $t = 1000\,s$ there is a 10 s period when there are no other TCP flows in the system. In this scenario this interval is too short for the mechanism to detect congestion abatement and cease phantom ECN signals. Thus the DA-LBE flow is unable to make use of this short period of available capacity.

Figure 5 shows the same scenario with LBEness achieved through manipulating the β-factor. The results are similar for both the PID and MBC controllers, the MBC again being

[2]Note that when the increase in w is limited, the model error, ϵ, is not updated to avoid artificially inflating the error due to the limiting.

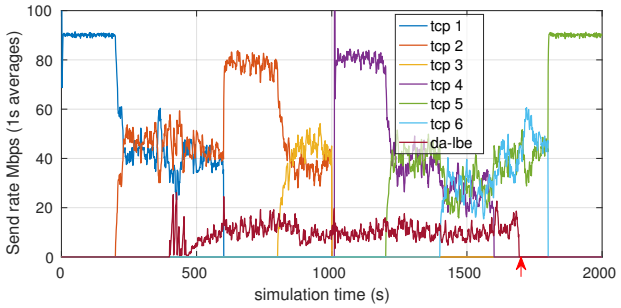
w **update procedure** *every* T_w
  Calculate $\bar{x}(t_{n-1}, t_n)$
  Calculate target rate $\zeta(t_n, t_D)$
  switch *control* **do**
    case *PID* **do**
      $w \leftarrow u_{\text{PID}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D))$
    case *MBC* **do**
      $w \leftarrow u_{\text{MBC}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D), \zeta(t_{n-1}, t_D))$

Modifications to congestion control
  switch *method* **do**
    case *Phantom ECN* **do**
      Augment ACK processing as follows:
      $t_{\text{cong}} \leftarrow$ time since last real congestion signal
      $\tau_{\text{cong}} \leftarrow$ time between the last two real congestion signals
      if $t_{cong} > v\bar{\tau}_{cong}$ **then**
        if *rand()* $< \mathbb{P}[loss](1/w - 1)$ **then**
          /* Initially rand() < 0.25 */
          Generate phantom ECN signal
    case *Adjusting* β **do**
      /* β_{orig}: original decrease factor */
      $\beta = \beta_{\text{orig}} w$ for congestion reaction

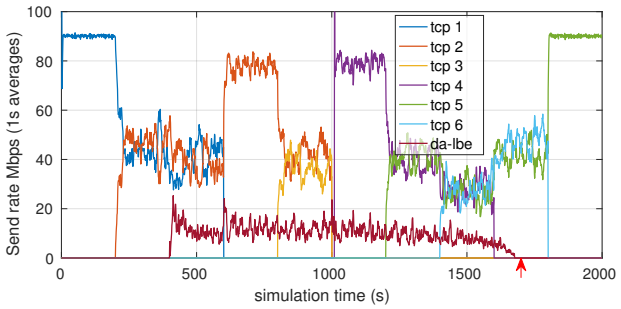
Function $u_{\text{PID}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D))$**:**
  Calculate w_n from (7) to (10)
  return w_n

Function $u_{\text{MBC}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D), \zeta(t_{n-1}, t_D))$**:**
  /* Cubic based model */
  $\widehat{\text{cwnd}} \leftarrow \zeta(t_n, t_D) \times \overline{\text{RTT}}$
  $\hat{q}_n \leftarrow \dfrac{\text{RTT}_{\min}}{\widehat{\text{cwnd}}^{\frac{4}{3}} \left(\frac{4(1-\beta)}{(0.4(4-(1-\beta)))} \right)^{\frac{1}{3}}}$
  Calculate $\hat{w}_n$ from (13), (14) and (19)
  return $\hat{w}_n$

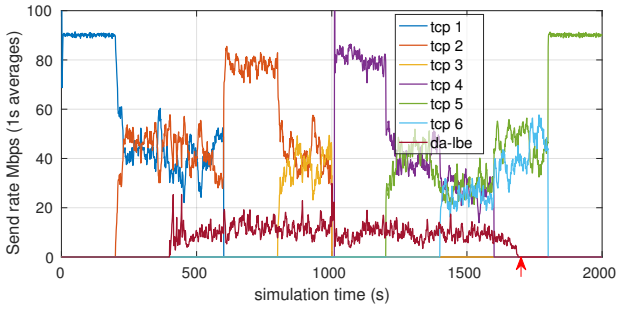
Fig. 3. Loss-based TCP DA-LBE algorithm. MBC based on Cubic.



(a) PID



(b) MBC



(c) MBC with w increase rate limiting

Fig. 4. NS2 simulation. Cubic TCP flows with a Cubic based DA-LBE phantom ECN flow. DA-LBE data size is based on 10% Capacity LBE rate.

a little smoother. In this scenario it is impossible for the DA-LBE flow to maintain a low send rate as probability of experiencing packet loss is too low. This is apparent in the Cubic model used in the algorithm from [11]. The β-factor mechanism is able to take advantage of that short 10 s period where there are no other TCP flows at $t = 1000\,s$.

4) PID versus MBC: Key to the PID controller's performance is tuning its gains. Key to the MBC controller's performance is a good model of the congestion controller. Normalizing the PID error input and controlling $w = (0, 1]$ enables the PID gains to be applicable across a wide range of conditions. Gains can be tuned to provide desirable characteristics for the DA-LBE source, however, wrongly configured gains could result in a wildly oscillating transmission rate; something not desirable for a DA-LBE traffic source. Given the availability of good models for commonly used congestion controllers, MBC control provides an alternative with similar performance but a minimum of configuration.

IV. Deadline-Aware LBE Congestion Control for Heterogeneous Traffic Sources

There are advantages in basing DA-LBE on a CC that reacts to more timely congestion indications than packet loss. However, mixing different CCs which react to different "prices" can make it difficult to ensure DA-LBE remains LBE and does not exceed our BE limit. We draw upon the heterogeneous congestion control work by Tang *et al.* [7], [13], [14], especially the price mapping and weighting, enhancing and extending the relative price adjustment to encompass DA-LBE.

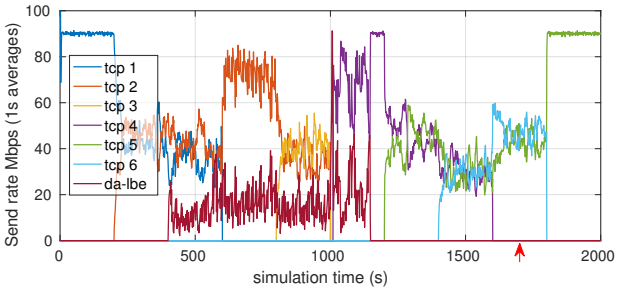
Tang *et al.* [7], [13], [14] show that the effective price a particular CC algorithm reacts to can be mapped to a common network price signal, such as queuing delay, loss or ECN marks. This mapping function depends on each type of CC as well as characteristics of each network element. Even though, the ratio of this effective price to a chosen common price can be used by the source to scale its effective price for fairer competition:

$$x_s = \left(\left(U_s^{(j)} \right)' \right)^{-1} \left(\frac{1}{\mu_s^{(j)}} \sum_{s \in S(l)} m_l^{(j)}(p_l) \right) \tag{20}$$

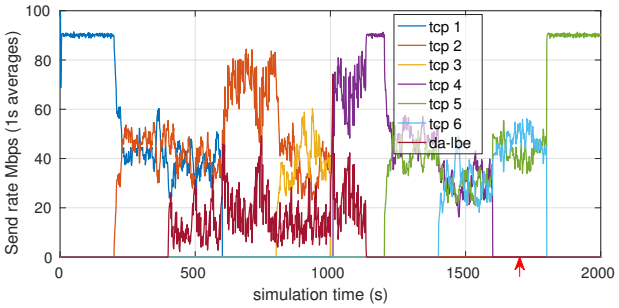
where

$$\mu_s^{(j)} = \frac{1}{w_s^{(\text{Tang})}} \frac{\sum_{s \in S(l)} m_l^{(j)}(p_l)}{\sum_{s \in S(l)} p_l} \tag{21}$$

where $j \in \{1, \ldots, J\}$ is the jth congestion controller of J operating in the network, $m_l^{(j)}(p_l)$ is the mapping function on



(a) PID



(b) MBC

Fig. 5. NS2 simulation. Cubic TCP flows with a Cubic based DA-LBE β tuned flow. DA-LBE data size is based on 10% Capacity LBE rate.

link l for congestion controller j operating on the common link price p_l, and $w_s^{(\text{Tang})}$ is a weight such that when $w_s^{(\text{Tang})} \neq 1$ an arbitrary share can be assigned to source s—given equivalent utility functions. In practice Tang *et al.* [7] use $w_s^{(\text{Tang})}$ as an adjustment parameter to aid fair competition between TCP FAST and TCP Reno since the price ratio alone is insignificant. This is necessary, because although the price ratio helps to balance the two different congestion controls, it does not take into account the different reactions to the congestion signals.

We adapt and extend the model in [7] to develop a model for Deadline-Aware LBE (DA-LBE) traffic sources which react to different network prices coexisting with BE traffic.

A. Heterogeneous Framework Design

We map prices to the probability of a congestion indication, weighting it by the relative effect the receipt of each congestion indication has:

$$\mathbb{P}_W^{(z,s)}[\text{cong\_ind}] = W^{(z,s)} \frac{I^{(z,s)}}{N^{(s)}} \tag{22}$$

where z is the type of congestion measure, e.g. delay, loss, ECN or some other network price measure; s the sender; $I^{(z,s)}$ is the number of congestion indications source s registers for congestion price type z; and N the total number of packets counted in the time interval. For Reno like CC where the cwnd is halved upon packet loss, $W^{(\text{reno},s)} = 0.5$. Similarly for Cubic where cwnd is reduced by 20 % for congestion indicated by packet loss, $W^{(\text{cubic},s)} = 0.2$. For Vegas reacting to delay-based congestion indications, cwnd is reduced by one. The relative effect depends on the size of cwnd when the congestion signal is received, the weight given by:

$$W^{(z,s)} = \frac{1}{I^{(z,s)}} \sum_{i=1}^{I^{(z,s)}} \kappa_i^{(z,s)} \tag{23}$$

where $\kappa^{(z,s)}$ is the corresponding proportion of cwnd that is reduced, different each time for delay based indications in Vegas. Vegas also reacts to packet loss, so the weighted probability of congestion for a multiple-price CC is given by:

$$\mathbb{P}_W^{(\text{delay-loss},s)}[\text{cong\_ind}] = \frac{W^{(\text{loss},s)}\mathbb{P}^{(\text{delay},s)}[\text{cong\_ind}] + W^{(\text{delay},s)}\mathbb{P}^{(\text{delay},s)}[\text{cong\_ind}]}{W^{(\text{loss},s)} + W^{(\text{delay},s)}} \tag{24}$$

We therefore redefine the price scaling of [7], terming it ϕ–the effective price ratio, to take into account the differences in cwnd adjustment as well as the different price measures:

$$\phi^{(s)} = \frac{\sum_{z \in Z_s} W^{(z,s)}\mathbb{P}^{(z,s)}[\text{cong\_ind}]}{\sum_{z \in Z_{\text{std}}} W^{(z,s)}\mathbb{P}^{(z,s)}[\text{cong\_ind}]} \tag{25}$$

where Z_s is the set of congestion price types observed by the CC operating at source s, and Z_{std} is the set of CC price types that the standard CC on the network uses. If both the standard and LBE CCs have a similar rate of increase in the absence of congestion, defining $\phi^{(s)}$ in this way allows us to use $w^{(s)}$ purely as an LBE scaling parameter rather than requiring it for fairness correction as in Tang *et al.* [7]. Note that in our work we define $w^{(s)} = (0, 1]$ which suits our control purposes more than the $w_s^{(\text{Tang})} = [1, \infty)$. Thus,

$$\mu^{(s)} = w^{(s)}\phi^{(s)} \tag{26}$$

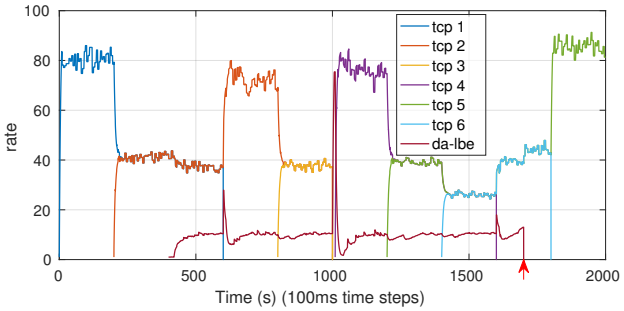
So that now:

$$x^{(s)} = \left(\left(U_s^{(j)}\right)'\right)^{-1}\left(\frac{1}{\mu^{(s)}}\mathbb{P}_W^{(z,s)}[\text{cong\_ind}]\right). \tag{27}$$

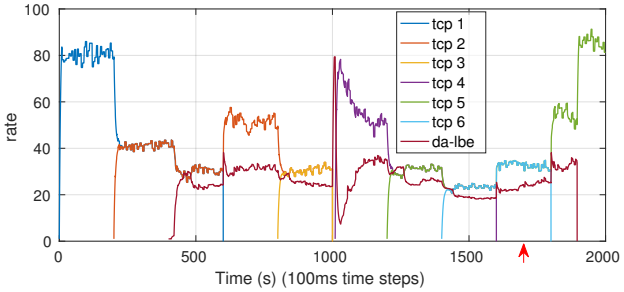
In the heterogeneous case it is insufficient to limit only the increase in $w^{(s)}$ as was done in Sec. III-C. Increases in network load also influence $\phi^{(s)}$. Limiting the increase, but not decrease of $\phi^{(s)}$ in the same manner as (19) (limiting factor l_ϕ) helps to maintain the LBE properties.

In situations where the standard TCP increase mechanism is significantly more aggressive than that used by the LBE mechanism, then the LBE mechanism will not quite receive its fair share when $w^{(s)} = 1$. This is not a significant shortcoming, since our aim is to be LBE. It could be adjusted for by using a model of the standard CC's increase mechanism, however, this is beyond the scope of this work.

1) Numerical validation: We conduct a simple numerical experiment, similar to that in Sec. III-C to test the idea. All TCP flows are modeled the same way as in Sec. III-C with (15) $\alpha = 2$ utility, with $\mathbb{P}[\text{loss}]$ as the price. The DA-LBE flow is modeled with (15) $\alpha = 1.2$ utility and uses the probability that



(a) 10% c_l DA-LBE target load



(b) 30% c_l DA-LBE target load

Fig. 6. MBC control. $\alpha = 1.2$ for DA-LBE and $\alpha = 2$ for the other flows

the queue (M/M/1/K=100 model) is larger than a target Q size as the price ($Q_T = 10$ in this experiment).

The DA-LBE flow working on a different price signal performs similarly to that in Sec. III-C. Importantly, even when there is a high target load for the DA-LBE flow, it competes effectively with the TCP modeled flows without taking more than its fair share. In general fair competition will depend on how different the utility functions are, or for real TCP congestion controllers it will depend predominantly on how different the cwnd increase functions are.

B. Applying the framework to TCP Vegas

As noted in Sec. III-D3, loss based DA-LBE has trouble either achieving LBE rates with β factor back-off control or making use of short periods of available capacity with phantom ECN control. Delay based CC can achieve LBE rates and make use of available capacity, with low latency [15]. In this section we illustrate the applicability of our heterogeneous DA-LBE NUM based work to TCP Vegas [16].

Vegas in congestion avoidance mode calculates the difference between the actual transmission rate and the expected rate (for $\text{RTT}_{\min}$) to increase or decrease cwnd—a measure of queueing delay in packets. The objective is to have a target $(\alpha - \beta)/2$ packets queued along the end-to-end path. If the difference is less than α, cwnd is increased; if more than β, cwnd is decreased. Since for a target cwnd $\alpha \propto \frac{1}{\mathbb{P}^{(s)}}$, we adjust α as follows:

$$\alpha^{(s)} = \mu^{(s)} \alpha_{\text{base}} \tag{28}$$

where $\alpha^{(s)} \in [1, \alpha_{\max}]$, $\alpha_{\text{base}} = 15$, and $\alpha_{\max}$ is set to half a BDP of packets in these experiments.

Figure 7 shows how Vegas can be modified to be DA-LBE. Vegas reacts to packet loss by halving cwnd but packet loss

ϕ **update procedure** *every* T_ϕ
- **if** *loss too low* **then** wait another T_ϕ
- Calculate $W^{(z)}$ for $z = \{\text{delay}, \text{loss}\}$ // (23)
- Calculate $\mathbb{P}_W^{(z)}$ for $z = \{\text{delay}, \text{loss}\}$, // (22)
- Calculate $\mathbb{P}_W^{(\text{delay-loss})}$ // (24)
- Calculation ϕ // (25)
- Limit ϕ increase similar to (19)

w **update procedure** *every* T_w
- Calculate $\bar{x}(t_{n-1}, t_n)$
- Calculate target rate $\zeta(t_n, t_D)$
- **switch** *control* **do**
  - **case** *PID* **do**
    - $w \leftarrow u_{\text{PID}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D))$
  - **case** *MBC* **do**
    - $w \leftarrow u_{\text{MBC}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D), \zeta(t_{n-1}, t_D))$
- $\mu \leftarrow w\phi$ // (26)
- $\alpha \leftarrow \mu\alpha_{\text{base}}$ // (28)

Adjust packet loss processing
- **if** $w == 1$ **then** `// maximum aggressiveness`
  - **if** *rand()* $< (1 - \frac{1}{\mu})$ **then**
    - `/* Competing with loss based CC */`
    - Skip cwnd reduction

Function $u_{\text{PID}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D))$**:**
- Calculate w from (7) to (9)
- **return** $\min(w, 1)$

Function $u_{\text{MBC}}(\bar{x}(t_{n-1}, t_n), \zeta(t_n, t_D), \zeta(t_{n-1}, t_D))$**:**
- `/* Vegas model */`
- $\tau \leftarrow \overline{\text{RTT}} - \text{RTT}_{\min}$
- $w_{\text{base}} = \frac{\tau\zeta(t_n, t_d)}{\phi\alpha_{\text{base}}}$
- $\epsilon \leftarrow \frac{\zeta(t_{n-1}, t_d) - \bar{x}(t_{n-1}, t_n)}{\bar{x}(t_{n-1}, t_n)}$
- $w \leftarrow w_{\text{base}} + \epsilon w_{\text{base}}$
- Limit w increase (19)
- **return** $\min(\hat{w}, 1)$

Fig. 7. Vegas DA-LBE algorithm

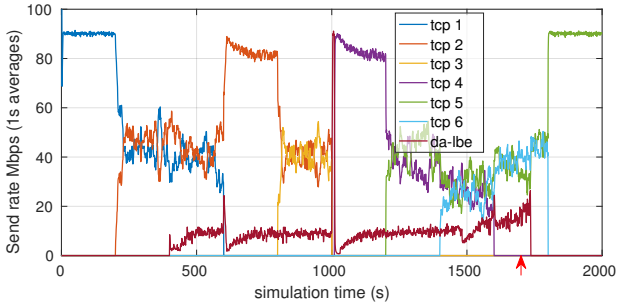


Fig. 8. Vegas MBC 10% Capacity LBE rate ($l_w = 0.05, l_\phi = 0.1$)

is not part of the Vegas congestion avoidance model. DA-LBE Vegas supplements the congestion avoidance model with a mechanism that probabilistically skips cwnd reduction on packet loss in proportion to $\phi^{(s)}$ when $w^{(s)} = 1$.

Figure 8 shows the Vegas based DA-LBE simulation results for the same scenerio. The Vegas based version has no problems taking advantage of the 10 s period from $t = 1000$ s where there is only background traffic. The lack of congestion in the 10 s period from $t = 1000$ s results in a queueing delay below Vegas' target during this period allowing Vegas to take advantage of available capacity. Unfortunately, reliance on

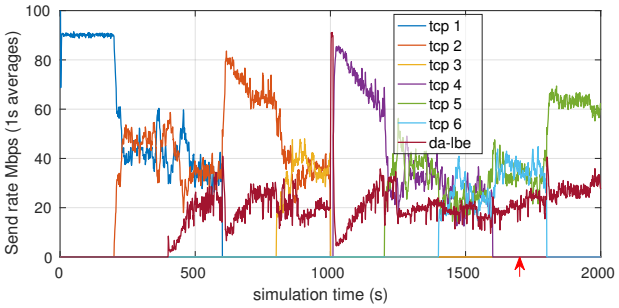


Fig. 9. Vegas MBC 30% Capacity LBE rate.

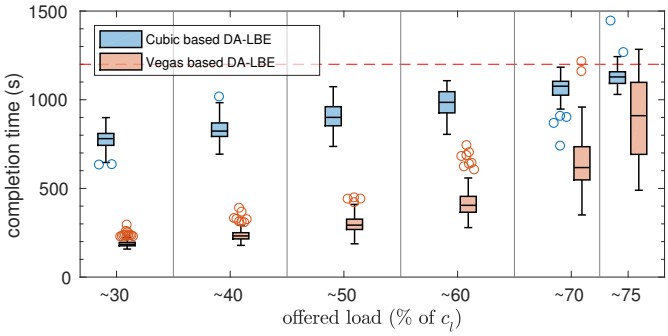


Fig. 10. Box-and-whisker plots of Cubic and Vegas based DA-LBE completion times. Boxes span the middle 50% with whiskers extending up to 1.5× the interquartile range. Red dashed line shows the deadline.

$\text{RTT}_{\min}$ to estimate path queuing causes Vegas to send at a higher rate than it should at the start, $t = [400, 600]$.

Figure 9 shows the same scenario in a situation where the deadline cannot be met. In these circumstances the DA-LBE mechanism should compete in a BE manner with the Cubic traffic. The ϕ parameter enables the Vegas based DA-LBE flow to compete more fairly than otherwise would be the case. When $w = 1$ the algorithm ignores packet loss triggered cwnd reductions in proportion to ϕ, reacting to delay scaled by ϕ as calculated by the mode.

V. Completion Time Evaluation

We evaluate the performance of our two DA-LBE NS2 Linux TCP implementations, Cubic based DA-LBE and Vegas based DA-LBE, over a bottleneck with realistic traffic based on real traffic measurements. We use Tmix [17] and the traffic traces used in [18]. Tmix preserves bidirectional application level interactions, using this high level data as input to the underlying TCP transport. This produces reactive background traffic, though many flows are short and do not leave slow start. This generates bursty realistic traffic.

We coarsely shuffle (50 s scaled bin size) the Tmix trace to reduce non-stationarity and scale the connection arrival time to achieve a target offered load. Note that this is the average load the network would experience if there were no losses and retransmissions, and as with real traffic at any point in time the actual load may be higher or lower than this value. The scaled traces have application session arrival rates of 99.5, 133.8, 167.1, 202.0, 235.7, and 255.8 sessions/s for offered loads of about 30%, 40%, 50%, 60%, 70%, and 75% respectively. The 75% load is close to the congestion knee.

With Tmix generating realistic traffic a simple dumbbell topology suffices for our experiments. After an accelerated Tmix traffic start up period and settling time a single DA-LBE source transmits data equivalent to 10% of the bottleneck link capacity with a soft deadline of 1200 s after start. When each flow completes, its completion time is recorded, the DA-LBE source reset, and after a 10 s delay the process repeated. We simulate under these conditions for ∼60 000 s and show the results using box-and-whisker plots.

The results in Fig. 10 show that under realistic traffic conditions both the Cubic and Vegas based mechanisms are able to complete their transmission before the deadline. The Cubic based mechanism has more consistent and tighter spread of completion times over the range of loads tested. In terms of a predictable behavior in meeting deadlines this is good, however, it shows that even in realistic traffic scenarios the pseudo-ECN method of controlling LBEness struggles to take advantage of the periods of available capacity. These results affirm our suggestion from the earlier experiments that the Cubic based mechanism may not be able to fully take advantage of the available capacity (cf. Fig. 4). The delay based congestion indications are an superior in this respect, though they do result in a larger spread of completion times as the load increases. Thus, though both the Cubic and Vegas based mechanisms are able to adjust their aggressiveness to meet a deadline (objective (iv) in Sec. III), only the latter mechanism seems to be able to take advantage of available capacity when there is no congestion (objective (iii) in Sec. III).

VI. Related work

LBE congestion control has been the subject of several proposals in the literature; see [1] for an in-depth survey and [19] for a more recent example. These proposals cover a range of approaches to detect congestion—losses, one-way delays, round-trip times, available bandwidth—and different reactions to congestion signals, but none of them include a notion of delivery deadlines. They all aim to achieve scavenger behavior irrespective of any flow-completion constraints. Some may approach normal TCP aggressiveness under some circumstances (e.g., LEDBAT, under loss), but this is done independently of application requirements. For most proposals, there's no simple way to tune the LBEness of the mechanism. Optimization work minimizing delay for short TCP transfers while sharing leftover capacity with background flows [20] provides some insight into this problem space, though it is not specifically related to deadlines. To the best of our knowledge, this paper is the first to both explicitly consider timeliness constraints, and try to balance such constraints with LBEness requirements—all adjustable as a matter of application / user policy.

There are quite a few proposals that add deadline awareness to the networking stacks of data centre (DC) hosts (see [21]–[24] and references therein). However, these proposals are not transferable to Internet CC as they rely on specialised network support, such as specific scheduling disciplines in

nodes or end-to-end support of ECN. Our approach does not impose any such requirements on the network, and can be used across an arbitrary Internet path[3]. Note also that these DC CC methods have been designed with typical DC applications (like web search) in mind. These usually have very short and strict deadlines, and severe user QoE degradation results if these are not met. Hence, their focus is on short deadline-constrained flows, with disruption of other flows being of secondary importance; our focus is on minimizing the disruption on other flows caused by long bulk transfers while maintaining a degree of timely completion. Our method allows for tuning the trade-off between timeliness and impact on other traffic.

VII. Conclusion

A deadline-aware–less-than-best-effort (DA-LBE) service provides a valuable transport for bulk data transfers such as backups. It allows transfers to be completed by a soft deadline while keeping disruption of other traffic sharing the network to a minimum. This paper develops a framework based on network utility maximization for transport protocols to *inflate* (or in certain circumstance discount) their network "prices" to achieve these goals.

We have demonstrated its applicability using Cubic and Vegas as base transports over networks where Cubic is the dominant transport. Our results show that both Cubic and Vegas achieve the desired LBE characteristics, although Vegas can better take advantage of brief instances of available network capacity. This is due to Vegas' faster detection of congestion abatement through its use of queueing delay signals. However, the Vegas mechanism can be too aggressive initially if it does not have a good estimate of $\mathrm{RTT}_{\min}$. As a part of future work we plan to explore applying this framework to a delay gradient based congestion control (CDG [25]) which does not have this shortcoming.

Our proposed framework has the flexibility to leverage whatever congestion control methods and congestion signals are available on a particular networking stack, even when they are different to the dominant network transport. This enables it to be applied to future more responsive congestion controls as they become available. Also as part of future work, we are planning to implement DA-LBE services in a real operating system network stack. This framework will also be integrated into the NEAT enhanced transport layer architecture [2].

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the authors.

References

[1] D. Ros and M. Welzl, "Less-than-Best-Effort Service: A Survey of End-to-end Approaches," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 898–908, May 2013.

[3]ECN signals can be leveraged by our method, but they are not mandatory for its proper operation.

[2] N. Khademi *et al.*, "NEAT: A Platform- and Protocol-Independent Internet Transport API," *IEEE Commun. Mag.*, Jun. 2017, to be published.

[3] K.-J. Grinnemo, T. Jones, G. Fairhurst, D. Ros, A. Brunstrom, and P. Hurtig, "Towards a flexible Internet transport layer architecture," in *Proc. of IEEE LANMAN*, Jun. 2016.

[4] F. P. Kelly, "Charging and rate control for elastic traffic," *European Trans. on Telecommunications*, vol. 8, pp. 33–37, 1997, corrected version: http://www.statslab.cam.ac.uk/~frank/elastic.pdf.

[5] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[6] S. H. Low and D. E. Lapsley, "Optimization flow control–i: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[7] A. Tang, X. Wei, S. H. Low, and M. Chiang, "Equilibrium of heterogeneous congestion control: Optimality and stability," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 844–857, Jun. 2010.

[8] N. Trichakis, A. Zymnis, and S. Boyd, "Dynamic network utility maximization with delivery contracts," in *Proc. of IFAC World Congress*, Jul. 2008, pp. 2907–2912.

[9] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 556–567, Oct. 2000.

[10] F. Paganini, A. Tang, A. Ferragut, and L. L. H. Andrew, "Network stability under alpha fair bandwidth allocation with general file size distribution," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 579–591, Mar. 2012.

[11] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[12] D. X. Wei and P. Cao, "NS-2 TCP-linux: An NS-2 TCP implementation with congestion control algorithms from linux," in *Proc. of Workshop on Ns-2: The IP Network Simulator*, 2006.

[13] A. Tang, J. Wang, S. Hegde, and S. H. Low, "Equilibrium and fairness of networks shared by TCP Reno and Vegas/FAST," *Telecommunication Systems*, vol. 30, no. 4, pp. 417–439, Dec. 2005.

[14] A. Tang, J. Wang, S. H. Low, and M. Chiang, "Equilibrium of heterogeneous congestion control: Existence and uniqueness," *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 824–837, Aug. 2007.

[15] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing Internet latency: A survey of techniques and their merits," *IEEE Commun. Surveys Tuts.*, Nov. 2014.

[16] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," *ACM SIGCOMM Comput. Comm. Rev. (CCR)*, vol. 24, no. 4, pp. 24–35, Oct. 1994.

[17] M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith, "Tmix: A tool for generating realistic TCP application workloads in ns-2," *ACM SIGCOMM Comput. Comm. Rev. (CCR)*, vol. 36, no. 3, pp. 65–76, Jul. 2006.

[18] D. Hayes, D. Ros, L. Andrew, and S. Floyd, "Common TCP evaluation suite," IRTF, Internet Draft draft-irtf-iccrg-tcpeval-01, Jul. 2014. [Online]. Available: http://tools.ietf.org/html/draft-irtf-iccrg-tcpeval.

[19] S. Q. V. Trang, E. Lochin, C. Baudoin, E. Dubois, and P. Gélard, "FLOWER – Fuzzy lower-than-best-effort transport protocol," in *Proc. of IEEE LCN*, Oct. 2015, pp. 279–286.

[20] C. Courcoubetis and A. Dimakis, "Fair background data transfers of minimal delay impact," in *Proc. of IEEE INFOCOM*, Mar. 2012, pp. 1053–1061.

[21] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-Aware Datacenter TCP (D^2TCP)," in *Proc. of ACM SIGCOMM*, Aug. 2012, pp. 115–126.

[22] G. Li, Y. Xu, and D. Cui, "A deadline and size aware TCP scheme for datacenter networks," in *Proc. of International Conf. on Comm. Tech.*, Nov. 2013, pp. 366–371.

[23] H. Zhang, X. Shi, X. Yin, F. Ren, and Z. Wang, "More load, more differentiation – a design principle for deadline-aware congestion control," in *Proc. of IEEE INFOCOM*, Apr. 2015, pp. 127–135.

[24] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with Karuna," in *Proc. of ACM SIGCOMM*, 2016, pp. 174–187.

[25] D. A. Hayes and G. Armitage, "Revisiting TCP congestion control using delay gradients," in *Proc. of IFIP Networking*, May 2011, pp. 328–341.