Improved Utility-based Congestion Control for Low-Delay Communication

Stefano D'Aronco, Laura Toni, Sergio Mena, Xiaoqing Zhu, and Pascal Frossard

***Abstract*—This paper proposes a novel congestion control algorithm for low-delay communication over best effort packet switched networks. Due to the presence of buffers in the internal network nodes, each congestion leads to buffer queueing and thus to an increasing delivery delay. It is therefore essential to properly control congestions in delay-sensitive applications. Delay-based congestion algorithms could offer a viable solution since they tend to minimize the queueing delay. Unfortunately they do not cohabit well with other types of congestion algorithms, such as loss-based algorithms, that are not regulated by delay constraints. Our target is to propose a congestion control algorithm able to both maintain a low queueing delay when the network conditions allows for it and to avoid starvation when competing against flows controlled by other types of policies. Our Low-Delay Congestion Control algorithm exactly achieves this double objective by using a non-linear mapping between the experienced delay and the penalty value used in rate update equation in the controller, and by combining delay and loss feedback information in a single term based on packet interarrival measurements. We provide a stability analysis of our new algorithm and show its performance in simulation results that are carried out in the NS3 framework. They show that our algorithm compares favorably to other congestion control algorithms that share similar objectives. In particular, the simulation results show good fairness properties of our controller in different scenarios, with relatively low self inflicted delay and good ability to work also in lossy environments.**

I. Introduction

NOWADAYS, many Internet applications aim to work not only at maximizing their throughput, but also at meeting crucial delay constraints in the transmission of data flows. Video conferencing applications are a good example of such delay sensitive services, where an excessive playback delay with the audio/video stream can drastically affect the quality of an internet call. On-line video gaming and desktop remote control are other examples of applications that require low latency in order for the user to have a valuable experience. When bandwidth resources are limited, these applications should ideally adapt their sending rate so that the experienced one-way delay is kept low and bounded, while preserving fairness with other flows. In particular, it is extremely important for delay sensitive algorithms to guarantee a good level of inter-protocol fairness when competing with congestion control algorithms such as loss-based methods that do not consider any delay constraint.

Congestion control can be seen as a constrained resource allocation problem that has to be solved in a distributed way due to scalability issues. Several solving methods of this optimization problem exist, but generally the resulting algorithms can be divided from a theoretical point of view into primal or dual algorithms [1]–[3]. From a more practical point of view, the primal and dual congestion control algorithms mentioned before roughly correspond, though not exactly, to loss-based and delay-based controllers. Loss-based controllers are widely deployed over the internet (e.g., TCP) and use congestion events triggered by packet losses to perform rate adaptation. However this class of controllers does not take into account any type of delay measurement, such as One-Way Delay (OWD) or Round Trip Time (RTT). Hence, there is no control on the latency that the packets might experience on their route. For example, in the presence of long buffers in the network, loss-based congestion mechanisms have no control over the increase of the queueing delay inside the network. As a result the playback delay, in the case of video conferencing applications, may grow very large. On the other hand, delay-based congestion control algorithms, can overcome the increasing delay issue by detecting congestion events from OWD measurements. They are able to keep a low communication delay, by adapting the sending rate to the evolution of the delay. However, they usually suffer when sharing the network with loss-based controllers. In the case where both algorithms are present the queueing delay could even grow so much that concurrent flows from delay-based controllers quickly reach starvation with an almost zero throughput. In these conditions, there is no chance that delay-based and loss-based controllers could nicely coexist while meeting the delay constraints. This calls for a congestion control that is robust against different working conditions and that could enable low delay communication when possible.

In addition to this challenge, deployment of new congestion control algorithms in the Internet has the following problems. First, new algorithms have to provide good inter-protocol performances when competing against existing controllers, such as TCP. Second, if the new protocols need a particular behavior from the inner nodes of the network the deployment in the Internet could also be impracticable. Third, the estimation of some network parameters may be problematic, even if many delay-based algorithms assume to know, or to be able to estimate, the one-way propagation delay of the route, this is actually not a trivial task. Many congestion algorithms have actually been proposed, but none of them is able to outperform the others in a large set of scenarios and, at the same time, easily be deployed in the Internet.

In this work, we propose a new distributed Low-Delay Congestion Control algorithm (LDCC) that is able to adapt the sending rate to both loss and delay based congestion events and overcome to the aforementioned issues. The ultimate objective is to keep low delays when the flows compete with other delay-

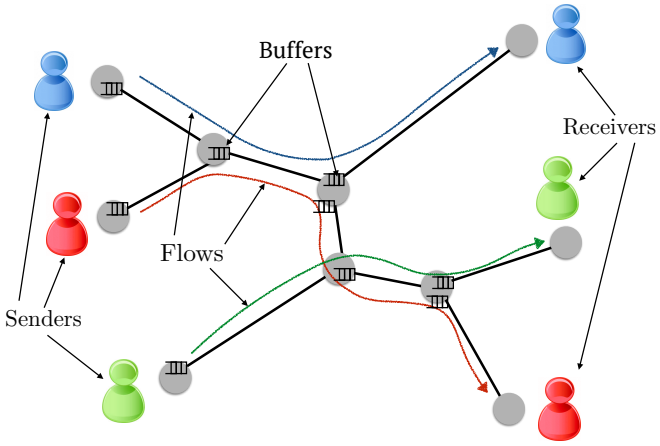

Fig. 1. Network system model.

based controlled flows, and at the same time not to starve when they compete with flows generated by loss-based congestion control algorithms. In particular, we target a scenario where users send delay-sensitive data over a packet switched network, as shown in Fig. 1, specifically, we have a network composed by links and nodes. The links are shared among different users who set up unicast communications between two end nodes of the network. The network topology as well as the network conditions (i.e., buffer lengths or presence of other flows) are a priori unknown to the users. The proposed controller measures the experienced OWD and the interarrival time of the received packets at the receiver node, and adjusts the rate accordingly in order to maximize the overall utility of the network flows. As we will see more specifically in the rest of this paper, the interarrival time of the packets is correlated to both losses and queueing delay variations. Hence, by using this quantity, the controller is able to work in both delay-based and lossy scenarios. Finally, we introduce a non-linear mapping between the experienced OWD and the penalty congestion signal used by the rate update equation. The non-linearity of the delay penalty is fundamental to preserve the delay-sensitive flows when operating in lossy environments and to avoid the starvation of the delay-based flows. The LDCC algorithm has been tested using different topologies and in different working conditions. It is shown to provide good intra-protocol fairness, namely good fairness when competing with other LDCC flows. It is also able to avoid starvation when competing with loss-based flows, such as TCP.

The remainder of this paper is organized as follows. In Section II, we provide a description of the system model and the NUM framework that is used in this paper. In Section III we describe our new congestion control algorithm in depth and we analyze its stability in Section IV. We present the simulation results in Section V. Section VI describes the related work and finally conclusions are provided in Section VII.

II. Network Utility Maximization

A. System Model

We now give a description of the system model used in this work. We consider a system with a set of R users represented by data flows and indexed by $r \in R$. The users share a set of network resources. Each of them transmits data at a sending rate of x_r, in a single-path unicast flow to one receiver node. All users transmit over a common set of links $l \in L$ and we indicate with y_l the rate at link l. We further refer to the $L \times R$ matrix $\mathcal{R}$ as the routing matrix, where $\mathcal{R}_{lr}$ is equal to 1 if link l is used by user r. Using the matrix $\mathcal{R}$, we can define the total rate passing through a link l as:

$$y_l = \sum_{r \in R} \mathcal{R}_{lr} x_r. \tag{1}$$

Every link has a fixed maximum channel capacity c_l, which is constant over time, and is preceded by a buffer. When the sum of the incoming flows of a link exceeds the link capacity c_l the data in surplus is accumulated inside the buffer of the link l. The queueing delay at the buffer of link l evolves over time:

$$\dot{q}_l = \frac{1}{c_l} (y_l - c_l)^+_{q_l}. \tag{2}$$

The notation $(z)^+_x$ is equal to z if x is positive and zero otherwise. The queueing delay can be zero if the buffer is empty. A data unit, or packet, that arrives at link l at time t takes a time equal to $q_l(t)$ before being actually sent over the link. We further define the propagation delay of the link l as the time required for the data to propagate through the link; we refer to this quantity with p_l and we assume that it is not varying over time. The total time taken by a unit data to traverse link l is given by the sum of the propagation and queueing delays $d_l = p_l + q_l$. The total one-way delay experienced by user r, e_r, is given by the sum of all the delays encountered on the path followed by its data. We can define this quantity using the routing matrix $\mathcal{R}$ defined previously:

$$e_r = \sum_{l \in L} \mathcal{R}_{lr} d_l. \tag{3}$$

The network buffers have a maximum size q_l^{MAX}. When the queue length reaches this size, the next incoming packets are discarded. Namely, the buffers implement a drop tail policy. The estimated loss ratio for link l when its buffer is full is:

$$\pi_l = \left(\frac{y_l - c_l}{y_l} \right)^+_{y_l - c_l} \tag{4}$$

and the total loss ratio for user r is:

$$\pi_r = 1 - \Pi_{l \in L} \mathcal{R}_{lr} (1 - \pi_l) \tag{5}$$

which, if the loss ratio of the links are small, can be approximated by:

$$\pi_r \simeq \sum_{l \in L} \mathcal{R}_{lr} \pi_l. \tag{6}$$

We further introduce another notion of communication delays between the network nodes. We refer to τ^f_{rl} as the time needed

by the data to travel from source node of user r to the link l, obviously only in the case where r uses link l. This time is composed by both propagation and queueing delay of the route from the source node of r to the link l. Similarly we define τ_{rl}^b as the time needed for the data to travel from the link l to the sink node and to be sent back to the source node r. Note that any kind of signal about congestion cannot be directly received by the source from an inner network node, but it must reach the end node, and only then it can be sent back to the source. For this reason, for a given user r the sum of the two values is equal to the RTT of the user independently of the link l:

$$RTT_r = \tau_{rl}^f + \tau_{rl}^b, \quad \forall l \in r \tag{7}$$

Due to the delays in the system, if we want to express the total rate at link l at time t as a function of the sending rates, we should delay them by an amount of time equal to τ_{rl}^f:

$$y_l(t) = \sum_{r \in R} \mathcal{R}_{lr} x_r(t - \tau_{rl}^f). \tag{8}$$

As final note we give the expression of the RTT_r for sender r as a function of the experienced user delay:

$$RTT_r = e_r + e_r^b \tag{9}$$

where e_r^b is the total experienced delay of the backward path. In the cases where the backward is as uncongested, it is reasonable to assume that e_r^b is equal to the propagation delay of the forward path. In any case we assume the backward delay e_r^b is not a function of the rate x_r.

B. Optimization Problem

We now define the NUM problem based on the framework of [1]. A key concept in the NUM framework is the utility of a flow. The utility of a flow with rate x_r associated to the user r is quantified by the value of the utility function $U_r(x_r)$; this function quantifies the benefits, or utility, for a user to have a data rate x_r. We assume that the utility function is a differentiable strictly concave function of the rate x_r. Then the NUM problem is typically defined as follows:

$$\begin{aligned} \text{NUM: } \underset{x}{\text{maximize}} & \sum_{r \in R} U_r(x_r) \\ \text{subject to} & \sum_{l \in L} \mathcal{R}_{lr} x_r \leq c_l, \ l = 1, \ldots, L. \end{aligned} \tag{10}$$

The problem consists in how to optimally allocate the rates of the users in such a way that the overall utility is maximized and that the capacity constraints on the network links are matched. An exact solution means that none of the capacity constraints is actually violated resulting in no queueing delay at the bottlenecks links. Since the utility functions are strictly concave, the problem is a concave maximization problem with a unique solution. The NUM problem can be solved exactly in a centralized way, but this requires a priori knowledge of the complete network state, which is not available in practical cases. A centralized solution is also impractical in real systems as it rapidly meets its strong limitations in terms of scalability and coordination. Thus, typically, the NUM problem is generally decoupled and then solved in a distributed manner.

C. Solution by Decomposition

We discuss now the most commonly used approaches to solve the problem, with solutions based on penalty decomposition and on dual decomposition, and we point out their main limitations. We refer to [4] for detailed tutorial on decomposition methods for NUM. Here we present the two methods that are mainly used and that are closely related to our study. The first method to solve the problem in Eq. (10) is by using penalty functions, called also prices, which map the violation level of the capacity constraints into a negative utility. A possible way to restate the problem using a penalty function is the following:

$$\underset{x}{\text{maximize}} \ \sum_{r \in R} U_r(x_r) - \sum_{l \in L} \int_0^{y_l} g_l(y_l) \tag{11}$$

where $g_l(y_l)$ can be thought of as the price to pay in order to use link l when the link rate is equal to y_l. It is important for $g_l(y_l)$ to be a positive and increasing function of the link rate y_l. The problem in Eq. (11) can be solved by a gradient-based algorithm following the rate update equation:

$$\dot{x}_r = \alpha_r \left(U_r'(x_r) - \sum_{l \in L} \mathcal{R}_{lr} g_l(y_l) \right) \tag{12}$$

When we use the loss ratio of link l as its price, $g_l(y_l) = \pi_l(y_l)$, and assuming $\mathcal{R}_{lr}\pi_l \simeq \pi_r$ for low loss ratios, we can deduce that, in order to converge to the equilibrium, the users need to know only their own utility function and their loss ratio. Note that every user can easily estimate its loss ratio by observing the number of dropped packets at the receiving node. In general, for a route, losses at equilibrium cannot be null and the bottleneck buffers are completely full. Thus the queue of the congested link always grows to its maximum value, q_l^{MAX}. The above congestion control solution does not take into account any sort of delay measurement, so it is not able to limit the experienced delays. A large family of congestion control algorithms, such as the classical loss-based version of TCP, can be modeled as systems governed by Eq. (12).

Another possible way to solve the NUM problem is by dual decomposition. The NUM problem is re-written as:

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \ \sup_{x} & \sum_{r \in R} U_r(x_r) - \sum_{l \in L} \lambda_l \left(y_l - c_l \right) \\ \text{subject to} \ \ & \lambda_l \geq 0 \ l = 1, \ldots, L. \end{aligned} \tag{13}$$

where λ_l are called dual variables. Eq. (13) corresponds to the lagrangian dual problem of Eq. (10). This optimization problem can be solved distributively using a primal-dual algorithm based on the following update equations:

$$\dot{x}_r = \alpha_r \left(U_r'(x_r) - \sum_{l \in L} \mathcal{R}_{lr} \lambda_l \right) \tag{14a}$$

$$\dot{\lambda}_l = \nu_l \left(y_l - c_l \right)_{\lambda_l}^+ . \tag{14b}$$

We notice that if we set $\nu_l = 1/c_l$, the dual variables have exactly the same form of the queueing delays defined in Eq. (2). Thus the value of a network variable that can be estimated at the end nodes, the total queueing delay, coincides with the value of the dual variables of the dual lagrangian formulation. As a result, users, by estimating the total queueing delay of their own route, and by adapting their rates accordingly, can achieve the maximization of the global utility. The value of the queueing delay at equilibrium depends on the shape of the utility functions. If we consider a fixed shape of the utility function, then the delay at equilibrium depends actually on the available capacity. Generally, the larger the bottleneck capacities c_l, the larger the user rates x_r at equilibrium, and the lower the delays at equilibrium. We also notice that a flow, which uses a single link with capacity c_l, in order to converge to an equilibrium rate equal to c_l needs to create some queueing delay along the route, which is called the self-inflicted delay.

The dual decomposition method has two main limitations: i) it assumes the knowledge of the total queueing delay, ii) it is not robust against loss-based flows. In order to reach a fair rate allocation, the users need to know the total queueing delay, while they can actually measure only the sum of the propagation and queueing delay. Estimating the queueing delay from the total delay is not a trivial task, as users are not able to measure the real propagation delay if the network is congested. An error in the propagation delay estimation may lead to unfairness among the delay-based flows. Furthermore, a delay-based congestion control should exhibit a nice behavior even when competing against the many internet flows that are driven by loss-based congestion control algorithms. A loss-based congestion control algorithm tends to fill the buffers, which results in an increasing queueing delay that can drastically lower the throughput of delay-based flows. Delay-based flows should therefore include in their algorithm a protection in the case where the queueing delay grows out of control and cannot be used as a valid congestion signal to reach the equilibrium.

In the next section, we describe our congestion control algorithm which is able to overcome to the aforementioned limitation of classical congestion control schemes.

III. Low-Delay Congestion Control Algorithm

A. Control Algorithm

In this section, we present our LDCC, and show how it overcomes the limitations of existing controllers. Our controller is defined by the following rate update equation:

$$\dot{x}_r = \; kx_r \left(U'(x_r) - v_r(e_r) - \dot{e}_r - \pi_r\right)^+_{x_r}. \qquad (15)$$

Eq. (15) shares some similarities with Eq. (12) and Eq. (14a) that have been discussed in detail in the previous section. The main differences are: i) the combined use of losses and delay as congestion signal, ii) the use of the experienced delay instead of the queueing delay and iii) the use of a non-linear function $v_r()$. Let us describe in details the different terms of Eq. (15). We first have the derivative of the utility function $U'_r(\cdot)$. Without loss of generality we will consider in this work logarithmic utility functions[1]. In order to simplify the development, we further assume that the utility is the same for all users and equal to:

$$U_r(x_r) = w \log(x_r). \qquad (16)$$

Note that the conclusions of this work hold even in the case where we assume different utility functions among the users. The second term, $v_r(\cdot)$, is the delay penalty function. Recall that we denote with e_r the total delay experienced by user r. The function $v_r(\cdot)$ simply maps the one-way delay into a penalty. This function is defined as:

$$v_r(e_r) = \beta \left(\frac{e_r - T_r}{RTT_r}\right)^+_{(e_r - T_r)}, \qquad (17)$$

where RTT_r is the round trip time of user r, β is a scaling factor and T_r is the delay threshold of user r. The delay threshold is related to delay that the system experiences at the equilibrium. The value of $v_r(e_r)$ is equal to zero if $e_r - T_r < 0$ and equal to $\beta(e_r - T_r)/RTT_r$ otherwise. The normalization of the price by RTT_r is motivated by rate fairness improvements and stability conditions. Note that this function is not a linear function of the experienced delay, e_r, but rather a monotonically non-decreasing function of it.

The third term in Eq. (15) depends on the derivative of the experienced delay, $\dot{e}_r$. This term does not modify the equilibrium of the system, since the time derivative at the equilibrium point will be zero by definition. However, it helps the controller during the transients, since it provides information about the variation rate of the feedback variable. The last term in Eq. (15) takes into account the experienced losses, following Eq. (4), so that the algorithm is able to operate in both delay-based and loss-based scenarios.

In more details, in the case of no loss ($\pi_r = 0$), our controller behaves as a delay-based controller. The experienced delay at equilibrium becomes:

$$\hat{e}_r = \frac{RTT_r}{\beta} U'_r(\hat{x}_r) + T_r = RTT_r \frac{w}{\beta \hat{x}_r} + T_r. \qquad (18)$$

This can be derived by setting the time derivatives to zero in Eq (15). By defining the RTT of the user r as in Eq. (9), Eq. (18) becomes:

$$\hat{e}_r = \frac{\hat{e}^b_r w/(\beta \hat{x}_r) + T_r}{1 - w/(\beta \hat{x}_r)}. \qquad (19)$$

From Eq. (19), we observe that the larger the equilibrium rate $\hat{x}_r$, the closer the experienced delay is to the threshold value T_r. Also, since the delay at equilibrium obviously has to be positive, we can see from the denominator of Eq. (19), that the rate at equilibrium has to verify the following inequality:

$$\hat{x}_r > w/\beta. \qquad (20)$$

If the above inequality is not verified, the delay-based part is not able to converge to the equilibrium. In this case the network queues grow until they hit the maximum buffer size,

[1]Our framework applies to any differentiable increasing concave utility function.

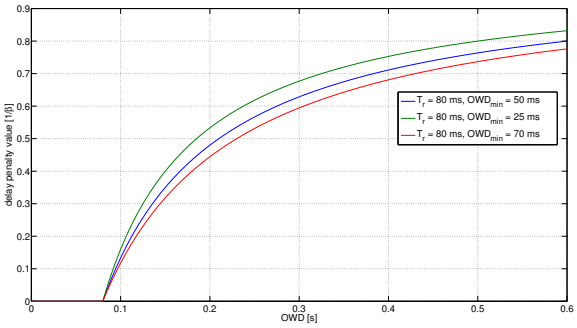

Fig. 2. Delay penalty as a function of the experienced delay for different values of propagation delays.

then the loss-based part of the algorithm becomes operative. We can finally observe that the controller is not able to achieve the equilibrium point for low values of $\hat{x}_r$, which is caused by the shape of the non-linear delay penalty mapping. However the penalty function has other nice properties that largely compensate this potential weakness. The main benefits of our penalty function can be summarized as:

- The non-linearity of the penalty function protects the flows from starvation when competing against loss-based algorithms. Fig. 2 depicts the shape of the penalty function of Eq. (17) for different values of the propagation delay, when the backwards delay, e_r^b is considered to be equal to the one-way propagation delay in the forward direction. The value of the penalty saturates to β for large values of the experienced delay. As a consequence the experienced delay can never force the sending rate to decrease to a value lower than w/β preventing starvation, this can simply be understood by looking at Eq. (20).
- The non-linearity of the penalty function alleviates unfairness problems caused by heterogenous propagations delays among the users. Since our control algorithm uses the total experienced one-way delay instead of the queueing delay, it may lead to unfairness when a bottleneck link is shared among users with different propagation delays. However the non-linear mapping of the delay helps to alleviate this problem when the available capacity is low. This can easily be understood by looking at the shape of the penalty function in Fig. 2. Since the penalty value tends to saturate for large delays, i.e., low available capacity, it means that in this scenario users with different propagation delays will have similar penalty values and as a consequence similar sending rates.

The loss-based part of the algorithm, π_r, simply takes into account the amount of lost packets in the flow r. It works as an additional penalty that lowers the value of the rate at equilibrium in the case where the delay part is not able to reduce it significantly. Losses could happen due to the presence of loss-based flows or to the presence of short buffers inside the network. Consider for example the case where the maximum queueing delay plus the propagation delay of a route is smaller than the delay threshold, i.e., $e_r^{MAX} < T_r$. In this case, the delay penalty is always zero and the flow behaves as a loss-based controlled flow. The flow is thus driven to its equilibrium rate exclusively by the term π_r.

B. Controller Implementation

We now briefly discuss the practical implementation of the LDCC described above. Even if our analysis relates to a continuous system, we obviously work in a discrete scenario in practice. The entities that travel through the links are data packets rather than continuous flows, and we describe below how the controller in Eq. (15) is modified in a discrete system.

We explain how, by using the interarrival time of the packets at the receiver node, we are able to build a term that incorporates both aspects of our controller, namely the delay-based part and the loss-based part. The advantage of merging these terms is that we do not need to discriminate between lossy environments, namely competition against loss-based flows (or in the presence of short buffers) or competition against other delay-based flows. More specifically, the two terms that will be unified are the queueing derivative term, $\dot{e}_r$, and the term that takes into account the losses of the flow π_r. We now start the derivation of the unified term by finding the connection between the interarrival time and the receiving rate, then we will give the expression of the merged term and show that it corresponds to approximates π_r or $\dot{e}_r$ for the loss-based and the delay-based case, respectively.

Ideally the interarrival time is inversely proportional to the received rate. If we measure the receiving rate in packets per second we can write:

$$x_r^{recv}(n) = \frac{1}{t^a(n) - t^a(n-1)} \tag{21}$$

where $t^a(n)$ is the instant of arrival of the n-th packet. Noting that $t^a(n) = t^s(n) + e(n)$ where $t^s(n)$ is the sending time of packet n, and that the time between two consecutive departures is equal to $1/x_r$, we can write:

$$x_r^{recv}(n) = \frac{1}{1/x_r + e_r(n+1) - e_r(n)} = \frac{1}{1/x_r + \Delta e_r(n)} \tag{22}$$

where $\Delta e_r(n)$ is the variation of the OWD between two consecutive packets. As the propagation delay is fixed, $\Delta e_r(n)$ corresponds to the variation of the queueing delay. The variation is evaluated between the sending times of two consecutive packets, so we can write $\dot{e}_r(n) \simeq \Delta e_r(n) x_r$. Recall that we are considering to measure the sending rate x_r in packets over seconds, thus $\Delta e_r(n) x_r$ is dimensionless, as $\dot{e}_r$. If we substitute this approximation in Eq. (22) and solve for $\dot{e}_r$ we obtain:

$$\dot{e}_r \simeq \frac{x_r - x_r^{recv}}{x_r^{recv}}. \tag{23}$$

The value of x_r^{recv} can be calculated at the receiver using Eq. (21) while the value of the true sending rate can be indicated in the header of the packets transmitted to the receiver, and then sent back to the sender to compute the value of $\dot{e}_r$.

Interestingly, Eq. (21) is also valid for losses: if the estimated received rate is lower than the sending rate, then it either means that the flow experiences some losses, or that the packets actually accumulate in the network nodes. In both cases, this is a signal of a congestion inside the network. Eq. (23) can thus be used indifferently for the delay-based or for the loss-based scenarios. In the case of losses, it is easy to prove that Eq. (23) is equal to:

$$\frac{x_r - x_r^{recv}}{x_r^{recv}} = \frac{\pi_r}{1-\pi_r} \tag{24}$$

where π_r is the loss ratio for user r. If the loss rate is low, then the denominator of the left hand side of Eq. (24) is close to one, $1-\pi_r \simeq 1$, hence Eq. (24) is a good approximation of π_r. We can finally write the rate update rule that governs the LDCC algorithm in the discrete system as

$$x_r^{new} = x_r + T_s k x_r \left(U_r'(x_r) - \beta \frac{e_r - T_r}{RTT_r} - \frac{x_r - x_r^{recv}}{x_r^{recv}} \right), \tag{25}$$

where T_s is the time interval at which the equation is updated. The gain of the controller, k, is chosen to be inversely proportional to the RTT for stability reasons (as discussed in Section IV). If the sampling period T_s (i.e., the feedback interval) is set equal to the RTT of the route, by multiplying both terms T_s and k the dependency on RTT disappears.

IV. Stability Analysis

In this section we discuss the stability of the LDCC algorithm based on Eq. (15). In particular, we concentrate on the case when the only congestion signal received by the users from the network is the experienced delay. The stability of similar delay based controllers has been studied in other works [3], [5]. However, due to the non-linear mapping of the experienced delay, those proofs do not apply directly to our algorithm. We first emulate a network where the only congestion signal is the experienced delay and set the length of network buffers to be infinite. In this case, the flows never experience any loss, (i.e., $\pi_l = 0$) and reach the equilibrium point using only the experienced delay as congestion feedback. We obviously assume that the equilibrium point exists in this case.

We first show the global stability of the non-linear LDCC for an ideal scenario when the communication delays are negligible, i.e., $\tau_{rl}^f \simeq 0$ $\tau_{rl}^b \simeq 0$, for a general case of R users and L links. The communication delays are negligible when they are remarkably smaller than the timing characteristics of the control system. Then we study the stability of the controller at the equilibrium when communication delays are non-negligible. We recall that both cases focus on the delay-based control regime, which means that no losses are experienced in any link, i.e., $\pi_l = 0,\ \forall l \in L$.

A. Negligible Delay Case

We first study the global stability of the undelayed system. We follow the analysis steps of [6], but we adapt the proof to our system and in particular to the non-linear function

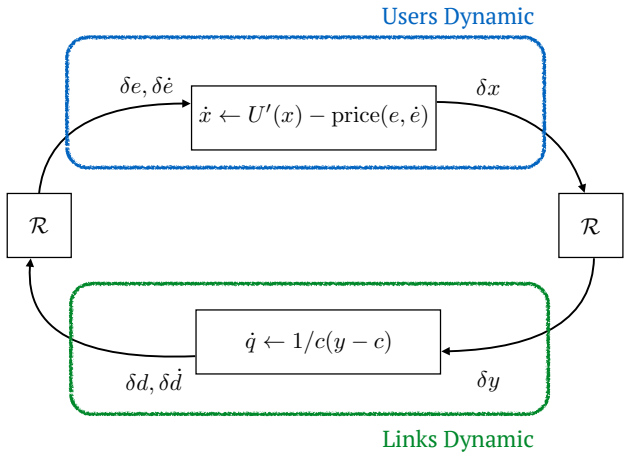

Fig. 3. Block scheme of the congestion control

$v_r(\cdot)$ in Eq. (17). The global stability proof is based on the passivity analysis of dynamic systems [6] [7]. A dynamic system characterized by an input u, state x, and output y is said to be passive if there exists a positive definite storage function V such that its derivative can be written as:

$$\dot{V} \leq -W(x) + u^T y, \tag{26}$$

where $W(x)$ is a positive semi-definite function. If two passive systems are interconnected in a negative feedback loop, then the sum of their respective storage functions is a Lyapunov function for the interconnected system. Indeed, the output of one system is the opposite of the input of the other one, so the two terms $u^T y$ vanish:

$$\dot{V} \leq -W_1(x_1) - W_2(x_2) + u_1^T y_1 + u_2^T y_2 \tag{27a}$$
$$= -W_1(x_1) - W_2(x_2) - y_2^T y_1 + y_1^T y_2 \tag{27b}$$
$$= -W_1(x_1) - W_2(x_2). \tag{27c}$$

Our dynamic system can actually be modeled as an interconnection of two systems (namely, the users and links dynamics in Fig. 3), which drive the behavior of our streaming framework. If we can prove that these two separate systems are both passive, then, from the above, we can prove the global stability in the sense of Lyapunov for the entire system.

Let us first introduce some notations that are used in our stability proof. The operator $\hat{\cdot}$ denotes the value of the variables at the equilibrium point, and δ denotes the deviation of a variable from its equilibrium point, e.g., $\delta x = x - \hat{x}$. We denote with V_{user} and V_{link} the storage function of the users and links sub-systems respectively. The input of the users system is the experienced delays e and $\dot{e}$, while its output corresponds to the sending rates x. Similarly, the links system has the link rates, y, as input variables and the links delays, d and $\dot{d}$, as output variables. The routing matrix $\mathcal{R}$ maps the user rates to the link rates and the link delays to the users' experienced delays. Finally, the equilibrium point of the system is reached when the dynamic equations (2) and (15) are set to zero. This

is equivalent to have $v_r(\hat{e}_r) = U'(\hat{x}_r)$ and $\hat{y}_l = c_l$ if $\hat{q}_l > 0$, or $\hat{y}_l \leq c_l$ if $\hat{q}_l = 0$. Since we are at equilibrium, $\dot{q}_l = 0$; we further have $\dot{e}_r = \sum_{l \in L} \mathcal{R}_{lr} \dot{d}_l = 0$, where d_l is the sum of the queueing delay q_l and the propagation delay p_l (which is constant over time).

We consider the following candidate storage functions for the user and link subpart respectively:

$$V_{user} = \frac{1}{k\alpha} \left(\sum_{r \in R} \int_{\hat{x}_r}^{x_r} \frac{z - \hat{x}_r}{z} dz \right) \tag{28}$$

$$V_{link} = \frac{1}{2} \sum_{l \in L} \delta d_l^2 c_l + \frac{1}{\alpha} \sum_{l \in L} (c_l - \hat{y}_l) d_l \tag{29}$$

Note that since $d_l \geq 0$ the two storage function are positive definite function for δx and δq respectively. It can be verified that by taking the time derivative of the two storage function we can verify the inequality of Eq. (27c) for two valid function W_{user} and W_{link}, proving the stability if the system. The full proof can be found in Appendix A.

B. Non-negligible Delay Case

We now study the stability of the system when the communication between network nodes is affected by non negligible delays, as it is typically the case in practice. Such delays affect the stability of the system. We first linearize the rate update equations of the non-linear delayed system around its equilibrium point and we study the local stability of the linearized delayed system, similarly to the analysis in [8] or [3], for example. When communication delays are non-negligible, Eq. (15) can be written as:

$$\dot{x}_r(t) = k_r x_r(t) \Big(U'(x_r(t)) - v_r(\sum_{l \in L} \mathcal{R}_{lr} d_l(t - \tau_{rl}^b)) - \sum_{l \in L} \mathcal{R}_{lr} \dot{d}_l(t - \tau_{rl}^b) \Big) \tag{30}$$

while the queueing delay equation, in Eq. (2), becomes:

$$\dot{q}_l(t) = \frac{1}{c_l} \left(\sum_{r \in R} \mathcal{R}_{lr} x_r(t - \tau_{rl}^f) - c_l \right). \tag{31}$$

These new equations (30) and (31) respectively represent the users and links dynamics taking into account the communication delays. Recall that τ_{rl}^f and τ_{rl}^b capture the time needed for the packets to go from source r to link l, and for the feedback information to be received by the source respectively. Recall also that the congestion information from link l needs to reach the end node of user r before being sent back to the source node. As a consequence, the sum $\tau_{rl}^f + \tau_{rl}^b$ does not depend on l and it is equal to the RTT of the route, i.e., $\tau_{rl}^f + \tau_{rl}^b = RTT_r$. When we linearize the equations (30) and (31) at the equilibrium point, we get the following equations to characterize the system in the Laplace domain:

$$\begin{aligned} s\delta x(s) &= \mathcal{K}\hat{\mathcal{X}} \left(\hat{\mathcal{U}}'' \delta x(s) - \hat{\mathcal{V}}' \mathcal{R}_b^T(s) \delta q(s) - s \mathcal{R}_b^T(s) \delta q(s) \right) \\ s\delta q(s) &= \mathcal{C}^{-1} \left(\mathcal{R}_f(s) \delta x(s) \right). \end{aligned} \tag{32}$$

Where δx indicates the deviation of the variable x from the equilibrium point, i.e., $\delta x = x - \hat{x}$, and $\delta x(s)$ refers to the Laplace transform of δx. The same notation applies to δq. The system equations (32) is written in a matrix form, where $\mathcal{C}$ is a L square diagonal matrix whose entries are equal to c_l. The routing matrices $\mathcal{R}_f(s)$ and $\mathcal{R}_b(s)$ embed the delay information and are defined as $\mathcal{R}_{f\ lr}(s) = e^{-s\tau_{rl}^f}$ if user r employs link l and 0 otherwise, and respectively $\mathcal{R}_{b\ lr}(s) = e^{-s\tau_{rl}^b}$ if user r employs link l and 0 otherwise. Then $\mathcal{K}$, $\hat{\mathcal{X}}$, are R squared diagonal matrices with respectively the values of gains k_r and the values at equilibrium of the rates x_r. $\hat{\mathcal{V}}'$ and $\hat{\mathcal{U}}''$ are R squared diagonal matrices whose entries correspond to the values at the equilibrium point of the first derivative of the penalty delay functions and the second order derivative of the utility functions.

Next, solving for $x(s)$ in Eq. (32) we obtain:

$$\begin{aligned} \delta q(s) = \mathcal{C}^{-1} \mathcal{R}_f(s) s^{-1} (Is - \mathcal{K}\hat{\mathcal{X}}\hat{\mathcal{U}}'')^{-1} \\ \mathcal{K}\hat{\mathcal{X}} \left(-\hat{\mathcal{V}}' - s \right) \mathcal{R}_b^T(s) \delta q(s). \end{aligned} \tag{33}$$

From Eq. (33) we can easily determine the return loop ratio $F(s)$ [9], [10], of our multivariable feedback system:

$$F(s) = \mathcal{C}^{-1} \mathcal{R}_f(s) s^{-1} (Is - \mathcal{K}\hat{\mathcal{X}}\hat{\mathcal{U}}'')^{-1} \mathcal{K}\hat{\mathcal{X}} \left(\hat{\mathcal{V}}' + s \right) \mathcal{R}_b^T(s). \tag{34}$$

Noting that $\mathcal{R}_b^T(s) = \text{diag}(e^{-sRTT}) \mathcal{R}_f^T(-s)$ and using the property that diagonal matrices commute, we can write:

$$\begin{aligned} F(s) = \mathcal{C}^{-1} \mathcal{R}_f(s) s^{-1} (Is - \mathcal{K}\hat{\mathcal{X}}\hat{\mathcal{U}}'')^{-1} \\ \mathcal{K} \left(\hat{\mathcal{V}}' + s \right) \text{diag}(e^{-sRTT}) \hat{\mathcal{X}} \mathcal{R}_f^T(-s). \end{aligned} \tag{35}$$

For the sake of clarity, we introduce the matrix $G(s)$:

$$G(s) = s^{-1} (Is - \mathcal{K}\hat{\mathcal{X}}\hat{\mathcal{U}}'')^{-1} \mathcal{K} \left(\hat{\mathcal{V}}' + s \right) \text{diag}(e^{-sRTT}), \tag{36}$$

this is an R by R diagonal matrix. Similarly to [5] we further introduce the matrix:

$$\tilde{\mathcal{R}}(s) = \text{diag}(1/\sqrt{c_l}) \mathcal{R}^f(s) \text{diag}(\sqrt{x_r}). \tag{37}$$

Since the eigenvalues of matrix product does not depend on the order of the matrices we can write:

$$\sigma(F(s)) = \sigma \left(\tilde{\mathcal{R}}(s) G(s) \tilde{\mathcal{R}}^T(-s) \right), \tag{38}$$

where $\sigma(\cdot)$ denotes the set of eigenvalues of a matrix. We can now verify the stability of the system using the Nyquist stability criterion. We set $s = j\omega$ and vary its value on the Nyquist path. The trajectories of the eigenvalues of the system as a function of $j\omega$ must not encircle the point $-1 + j0$ for the system to be stable. Using the main result of [11]:

$$\sigma(F(j\omega)) \in \rho(\tilde{\mathcal{R}}^T(j\omega) \tilde{\mathcal{R}}(-j\omega))\ \text{co}(0 \cup \sigma(G(j\omega))), \tag{39}$$

where $\rho(\cdot)$ denotes to the spectral radius of a matrix and $\text{co}(\cdot)$ denotes the convex hull. Eq. (39) simply tells that the eigenvalues of $F(j\omega)$ are located on the convex hull made by the eigenvalues of $G(j\omega)$ scaled by the spectral radius of $\tilde{\mathcal{R}}^T(j\omega) \tilde{\mathcal{R}}(-j\omega)$.

From [12], we further know that the spectral radius $\rho(\tilde{\mathcal{R}}^T(j\omega)\tilde{\mathcal{R}}(-j\omega)) \leq M$, with M being the maximum number of links used by any user r. Hence, to prove the stability of the system, we need to show that the eigenvalues of $G(j\omega)$ in Eq. (38) do not encircle the point $-1/M + j0$ in the complex plane. Since $G(j\omega)$ is a diagonal matrix, as can be seen from Eq. (36), we need to show that all the entries on its diagonal verify the Nyquist criterion of stability. Defining $G_r(j\omega)$ as the r element of the diagonal of matrix $G(j\omega)$ we have:

$$G_r(j\omega) = k_r \frac{e^{-j\omega RTT_r}}{j\omega} \frac{v'_r + j\omega}{j\omega - k_r \hat{x}_r U''_r}, \tag{40}$$

where U''_r correspond to the second derivative of the utility functions calculated at the equilibrium point, and v'_r is the derivative of the penalty function calculated at the equilibrium point. As the utility functions are concave, U''_r is negative, so both poles of Eq. (40) are non positive. With a utility function defined as $U_r(x_r) = w\log(x_r)$, the value of the second derivative of the utility and the derivative of the price at the equilibrium point are respectively equal to:

$$U''_r = -\frac{w}{\hat{x}_r^2}, \tag{41}$$

and

$$v'_r = \frac{\beta - w/\hat{x}_r}{RTT_r} \tag{42}$$

where in the last equation we used the property that $v_r(\hat{e}_r) = U'(\hat{x}_r)$ at equilibrium. Note that, in order to have a negative zero in the transfer function $G_r(j\omega)$ we need the inequality $\hat{x}_r > w/\beta$ to hold. This condition supports in fact that w/β is the minimum equilibrium rate for the purely delay-based subpart as described in Section III.

Finally, substituting Eq. (41) and (42) in Eq. (40), we obtain:

$$G_r(j\omega) = \eta_r \frac{e^{-j\omega RTT_r}}{j\omega RTT_r} \frac{\frac{\beta - w/\hat{x}_r}{RTT_r} + j\omega}{j\omega + \eta_r \frac{w}{\hat{x}_r RTT_r}}, \tag{43}$$

where we have introduced the normalized gain $\eta_r = k_r RTT_r$. The value of the normalized gain η_r should be divided by M according to Eq. (39). Eq. (43) is a simple transfer function with two poles and one zero. The location of the second pole and the zero are actually related since they depend on the same quantities. In particular, due to the previous considerations about the minimum value of the equilibrium rate, the pole cannot be located at an angular frequency larger than $\eta_r\beta/RTT_r$. The zero is located between 0 and β/RTT_r. By selecting appropriate values for the controller parameters, w, β and η_r, we can allocate the zero-pole couple to low frequencies. At the same time we can set a cross frequency, i.e., the frequency at which $|G_r(j\omega)|$ is equal to 1, to a sufficiently high frequency to ensure the stability of the system.

In summary, we have studied in this section the stability of our system in the case where the only congestion event received from the network was the varying experienced delay, for the case with negligible communication delays and for the case where user communication delays are not negligible. The main result is that the non-linear mapping of the experienced delay leads to a stable system. We finally note that the above study concerns the delay-based part of the controller. It can however be extended to the loss-based part of the algorithm. Since we consider the case of droptail queues, we observe that when a flow is experiencing losses at a bottleneck, then the queue has reached its maximum size: as long as losses are experienced, the queue delay cannot change (we do not consider the case of RED policy [13] or any other AQM mechanism). A constant delay plays no role in the placement of eigenvalues of the linear dynamic system at equilibrium. The delay-based terms of the rate update equation disappear in the linearization process of the system. Hence, for the loss-based part of our algorithm, the linearized update equation is consistent with the one used in [1], [14] and the stability proof can be carried out by following the steps described in these works.

V. Simulations

A. Simulation Setup

To evaluate the performance of our controller, we carried out experiments using the NS3 network simulator platform. We have tested the controller in different network topologies and scenarios in order to show that the algorithm is able to work in loss or delay-based control situations. In particular, we consider a single link topology, Topology 1 (see Fig. 4) and Topology 2 (see Fig. 9) in our simulations. The first one is the classic dumbbell topology, where several users share the same unique bottleneck link. The second topology is the so-called parking lot topology, with two bottleneck links, where two users employ only one of these links while a third user, with a longer data path employs both congested links. We evaluate the performance of the LDCC algorithm under different metrics such as throughput, self inflicted delay, and fairness.

We implement our congestion control algorithm on top of the UDP protocol. The controller parameters are set according to the stability analysis in Section IV. In particular we set $k = 1/(2.5RTT)$ and we neglect the dependency on M (the maximum number of links present in a route). The value of β is equal to 0.1. The utility functions are given by $w\log(x)$, with w equal to 20 kbps and the packets have a fixed size of 1KB. In our practical implementation the sender schedules the sending time of the packets according to the discrete rate update control algorithm of Eq. (25). We set a feedback interval of one RTT, i.e., $T_s = RTT$. In practice, more than one feedback is usually received during one RTT. The received values, such as the RTT, the sending rate and the one way delay, are averaged using a rectangular window and the update equation is computed every T_s with the average values of the feedback information.

B. Fairness Analysis

In this set of simulations, we evaluate the performance of the LDCC controller when it shares the bottleneck links with other flows controlled by the same algorithm. Fairness is an important metric for congestion control algorithms; it reflects the ability of flows to fairly share the available bandwidth

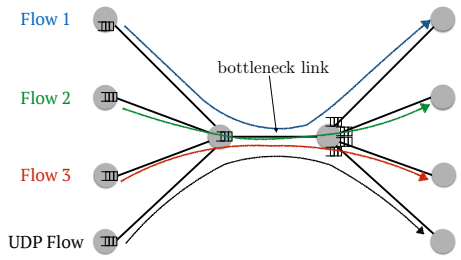

Fig. 4. Network Topology 1.

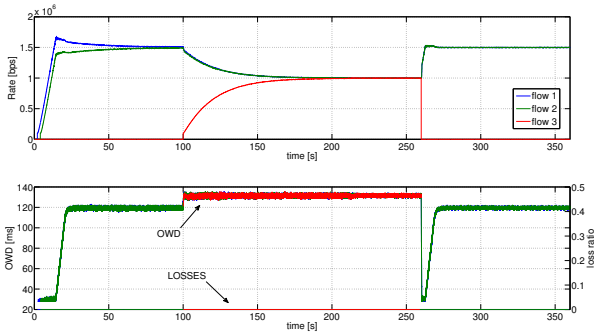

Fig. 5. Sending rate and OWD for the three LDCC flows sharing a common bottleneck link when the equilibrium is driven by the experienced delay.

without penalizing early or late starting flows. We consider a case with three LDCC flows that share a link with an unresponsive UDP flow with a constant rate of 500 kbps. The network topology is shown in Fig. 4. Two of the flows, flows 1 and 2, start respectively at 2 and 4 s and last until the end of the simulation. The third flow starts at 100 s and stops transmitting at 260 s. The unresponsive UDP flow is always present during the simulation. The bottleneck link has a one way propagation delay of 25 ms, and a total of capacity of 3.5 Mbps. The threshold delay imposed on the three LDCC flows is 50 ms.

In a first simulation we set the maximum length of the drop-tail buffers inside the network to 130 packets, equivalent to a maximum queueing delay 300 ms. Since the LDCC algorithm has a delay threshold parameter of 100 ms, the flows never fill the buffer and so no loss is experienced. The results for this simulation are provided in Fig. 5. It can be seen that the LDCC flows fairly share the bottleneck link without penalizing early or late starting flows. Next, we conduct a simulation with the same settings, but with a maximum buffer length of 25 packets, corresponding to a maximum experienced delay smaller than 100 ms. With these parameter values, the maximum delay allowed by the drop tail buffer is lower than the delay threshold. As a result, the LDCC flows are driven to equilibrium by experienced losses rather than

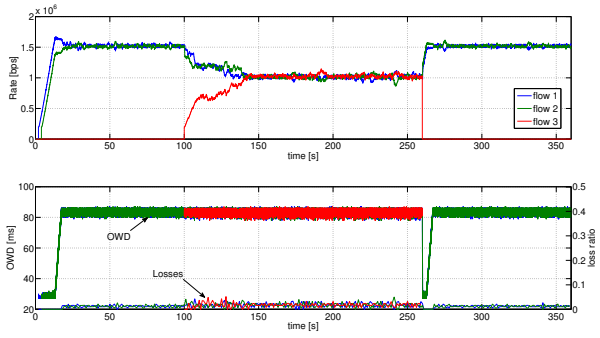

Fig. 6. Three flows sharing a common bottleneck, equilibrium is driven by losses. Drop tail bottleneck buffer.

by the experienced delay. The sending rates and OWD for this simulation are depicted in Fig. 6. We see that, when the system is mainly loss-based, the flows also fairly share the available bandwidth since each active flow gets the same sending rate.

Next, we perform a simulation where flows sharing the same bottleneck have different propagation delays. We used the same dumbbell topology of Fig. 4, with four LDCC flows and we set different values, $[p_1\ p_2\ p_3\ p_4]$, of the propagation delay for the different users. We varied the capacity of the bottleneck link from 1 Mbps to 6 Mbps with a max queueing delay of 0.5s and a delay threshold of 100 ms. We measured the average sending rate at equilibrium and we compute the Jain's fairness index [15], which is a well known index to measure the fairness of rate allocation. It is defined as:

$$J(x) = \frac{\left(\sum_{n=1}^{N} x_n\right)^2}{N \sum_{n=1}^{N} x_n^2}, \tag{44}$$

for the case of N flows. The Jain's Fairness index assumes values between 1, full fairness, and $1/N$, poor fairness. The results of the simulation are shown in Fig. 7. We plot the fairness index as a function of the normalized capacity, which is equal to the total capacity divided by the number of flows. It can be seen that, when the discrepancy between the propagation delay increases the Jain's score is lower. However, when the available bandwidth is lower, which is the most constrained and so the most critical scenario, the fairness level is higher. These results correspond to the observations made in the theoretical analysis of Section III. Finally, we also plot in Fig. 8 the value of the rates of the flow with the lowest and highest sending rate at equilibrium versus the normalized capacity. In the ideal case, this plot should feature a straight line with all the rates equal to the normalized capacity when full fairness is achieved. For high values of capacity the full fairness is however not achieved but flows do not starve and achieve, in absolute terms, a relatively good amount of bandwidth. In particular, in the worst case, where the maximum propagation delay is three times the minimum one, the flow is still able to send to a value that is approximately half of the normalized capacity. This confirms the ability of the LDCC algorithm to prevent flow starvation.

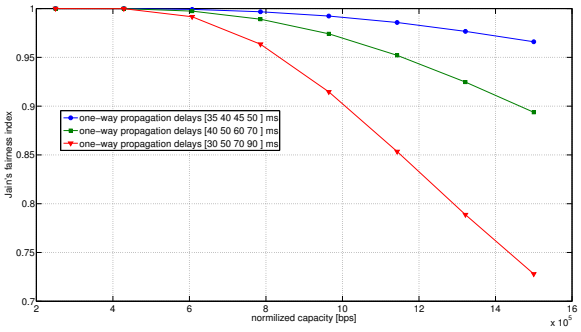

Fig. 7. Jain's fairness index of four LDCC flows with different propagation delays.

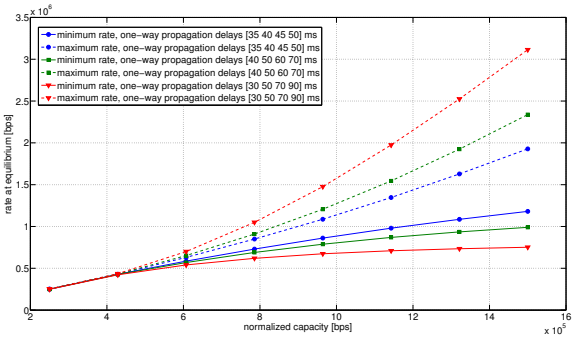

Fig. 8. Minimum and maximum rates at equilibrium versus normalized capacity for four LDCC flows sharing a bottleneck with different propagation delay.

Finally, we analyze the fairness of our controller for the parking lot network topology depicted in Fig. 9. Both bottleneck links have a channel capacity of 2.5 Mbps and a propagation delay of 25 ms. There is also an unresponsive UDP flow that passes through both links with a constant rate of 500 Kbps. In order for the equilibrium to be driven by the experienced delay, we first set a threshold value of 100 ms for all the users. Then we set the maximum length of the buffers inside the network to 100 packets, which ensures that no losses are experienced during this simulation. Flow 3 starts at 100 s and stops at 260 s while the other flows are always present. The simulation results are shown in Fig. 10. Flow 1 always gets a lower rate compared to the other flows. This is expected and due to the fact that its route is longer, hence it is penalized compared to the other flows. We finally conduct a last simulation for a mixed-based regime the maximum queuing delay is set to 50 ms. In this case we also obtain expected results with flow 1 getting a lower rate when it has to compete against the other two flows, as shown in Fig. 11. In this experiment flow 1 works in both loss and delay mode, since it is experiencing losses and having a total delay larger than its threshold.

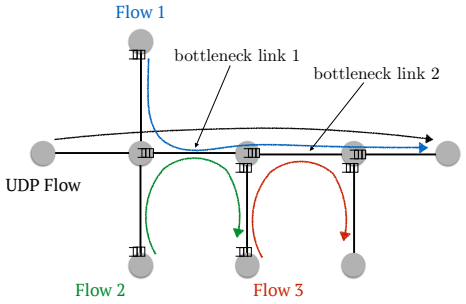

Fig. 9. Network Topology 2. The flow LDCC 1 passes through two bottlenecks links and competes with another LDCC flow on each of these links

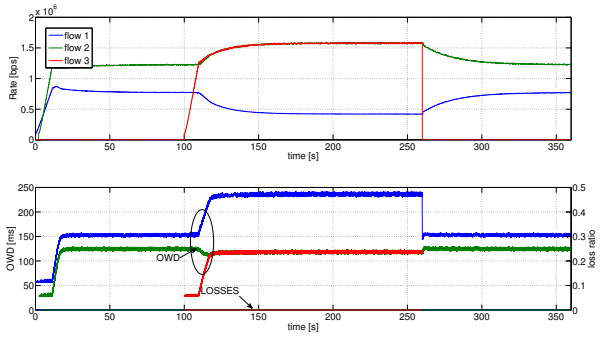

Fig. 10. Time evolution of the sending rates and delays for the parking lot topology when equilibrium is driven by the experienced delay.

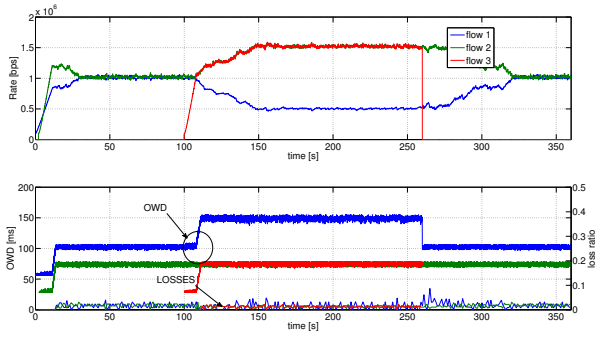

Fig. 11. Time evolution of the sending rates and delays for for the parking lot topology when equilibrium is driven by losses

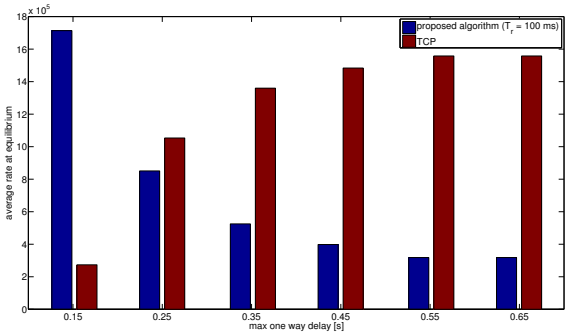

Fig. 12. Average rate at equilibrium of our algorithm competing and TCP when competing for a bottleneck for different drop tail buffer size.

C. TCP Coexistence

We now study the performance of the LDCC when it competes against TCP flows. We again use a single link topology, with a capacity of 2.5 Mbps and a propagation delay of 50 ms. Three flows are sharing the link: an unresponsive UDP flow with a constant sending rate of 500 kbps, a flow running the LDCC algorithm and a TCP New Reno flow. The delay threshold has been set to 100 ms in the LDCC algorithm.

We run simulations for different values of the size of the droptail buffer, and we vary it from 30 to 180 packets, which roughly correspond to 100 ms to 600 ms of maximum queueing delay respectively. The simulation results in Fig. 12 show the average rate at equilibrium for the LDCC and TCP algorithms. We can see that the level of fairness against TCP depends on the buffer size: more bandwidth is given to the LDCC flow for short buffers, and the reverse for long buffers. Ideally the ratio between the two flows should not depend on the buffer size. This dependency is caused by the delay-based part of the congestion algorithm: since with long buffers the experienced delay is higher, the equilibrium rate is smaller. However, due to the non-linearity of the LDCC penalty function, the rate does not reach the zero value but tends to w/β. By modifying the value of w, we can increase the minimum rate that is reached in large delay environments. On the other hand, with small buffers, our algorithm reaches an higher rate at equilibrium than TCP. This is caused by the fact that the loss-based part of LDCC is more aggressive than the TCP congestion control. We note that, even if the coexistence with TCP is not optimal, the new penalty function in LDCC prevents starvation of our flow even for large queueing delay values, which is not the case for other delay-based congestion control algorithms in the literature.

D. Comparison with Other Congestion Control Algorithms

We finally conduct some experiments to compare our algorithm with other congestion controllers for real time communication. We concentrate on the behavior of the algorithms when they operate in a delay-based mode. The algorithms that share most similarities with our proposal are those that have been proposed within the RTP media congestion control (RMCAT) IETF working group [16]. The first comparison is with the Network-Assisted Dynamic Adaptation (NADA) congestion control algorithm (as described in [17]), and the second one is with the Google congestion control (GCC) algorithm [18]. For a more detailed analysis of these congestion control algorithms we refer the reader to Section VI. Since GCC is already integrated and deployed in many web browsers. This controller comes together with the video encoding part, which renders the comparison difficult in its original version. Since there is no video encoding for the NADA and LDCC algorithms, we build a simplified Google congestion controller according to the description in [19], [20]. It permits to avoid non-idealities and issues possibly caused by live encoding in the original implementation, which may lead to an unfair comparison among controllers. The mathematical model of our simplified version of the Google algorithm is however the same as for the original controller version. In particular, GCC does not use absolute OWD measurements to adapt the rate, as for LDCC and NADA, but only the OWD variations. It keeps increasing the rate until the derivative of the queueing delay reaches a threshold; at that point, it decreases the rate until the queueing derivative is smaller than a negative threshold value. As a result, it is impossible for GCC to reach an equilibrium rate: it necessarily oscillates between overestimation and underestimation of the capacity, therefore the experienced delay also oscillates.

We consider a single link topology. With a varying channel capacity and latency of 25 ms. We impose a threshold delay of 100 ms for our controller. In Fig. 13, we can see a comparison of the average self-inflicted delay at equilibrium for NADA, our LDCC algorithm and the GCC. NADA tends to converge to a larger delay value for small capacities, while our algorithm is able to alleviate the delay variations across the different channel capacities. The main problem with NADA is the growth of the self-inflicted delay when the available capacity reduces. At equilibrium, the value of the OWD at which NADA converges is approximately given by $OWD_{eq} = R_{MAX}OWD_{min}/R_{eq}$ where OWD_{min} is the minimum OWD, R_{MAX} is a controller parameter indicating the maximum rate of the controller, and R_{eq} is the value of the rate at equilibrium. As can be noticed the value of the delay can become very high when the ratio R_{MAX}/R_{eq} gets large. Even if GCC, as can be seen in Fig. 13, achieves a lower average self inflicted delay for all the tested rates, the variability of the OWD of the GCC is pretty large, thus the maximum experienced delay is much higher than the average one (see Fig. 14).

In Fig. 14, we show a comparison of the proposed LDCC, the NADA algorithm and our simplified version of GCC when flows utilize a single link of 1.5 Mbps with a one way propagation delay of 25 ms. We focus on the qualitative behavior of the algorithms. Our algorithm uses an absolute measure of the OWD and achieves a constant rate and constant delay by avoiding variations of these two quantities. In contrast, the simplified GCC is not able to provide a stable rate at equilibrium. In the case where small variations of the rate and OWD is an important metric, our controller shows better performance than the simplified GCC algorithm. The NADA algorithm is also able to achieve a constant sending rate and

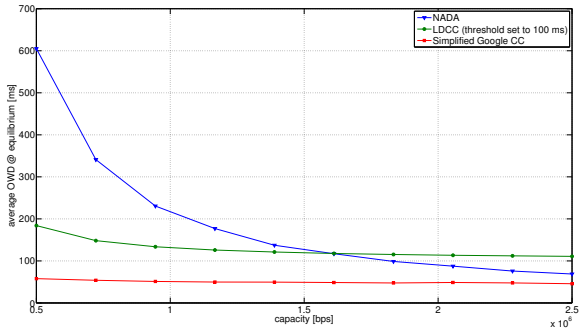

Fig. 13. Average OWD at equilibrium for the NADA, the simplified version of the GCC and the LDCC algorithms, in a single link topology.

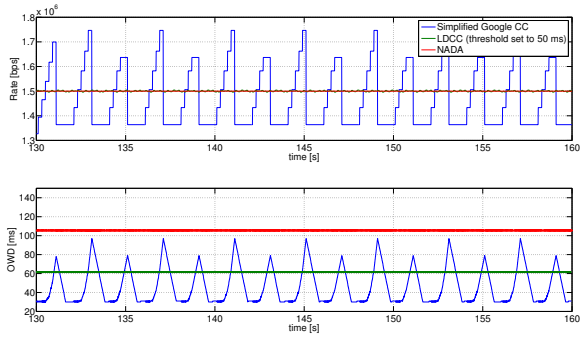

Fig. 14. Rate and OWD evolution at equilibrium for the NADA, the simplified version of the GCC and the LDCC algorithms, in a single link topology.

queueing delay at equilibrium. However, as seen in Fig. 13, the self inflicted delay for low capacity values is much higher compared to the LDCC algorithm.

VI. Related Work

We finally describe in this section some works closely related to our study, which have implemented congestion algorithms trying to overcome to the main limitations mentioned in the Section II-C. Namely the problem of avoiding starvation when competing against loss-based flows and the estimation of the true queueing delay of the user's route.

A congestion control that shares some similarities with our proposed controller is the Low Extra Delay Background Traffic (LEDBAT) protocol [21]. It is a quite recent delay-based congestion control that aims to keep a constant queueing delay at the buffer bottleneck in order not to harm other more important flows. In short this congestion controller decreases the sending rate when the estimated queueing delay is larger than a fixed threshold called target delay. It has been shown that this algorithm suffers from intra-protocol unfairness due to the additive-increase-additive-decrease (AIAD) mechanism used in the rate update equation [22]. A second problem is due to the usage of the queueing delay, with latecomer flows usually overestimating the propagation delay and achieving a higher and unfair rate at equilibrium. Since it is meant for less than best effort communication it is not a problem if it starves when competing against pure TCP flows (see [23] [24] for a detailed evaluation). However, in our case, we want to avoid flow starvation.

From a dynamic system perspective FAST TCP [25] is similar to our congestion control algorithm. In fact it is also a delay-based control algorithm however it uses the RTT instead of the OWD and since it is a modification of the TCP it is a reliable window-based congestion control which does not make it suitable to be used for real-time communication.

In [26], the authors present an interesting approach to solve the complex problem of loss-based and delay-based coexistence. The work proposes a modification of the TCP algorithm, called Cx-TCP, with a backoff probability that is a function of the experienced queueing delay. The backoff probability increases linearly with the queueing delay; after a fixed queueing threshold it then decreases to zero. If, due to the presence of purely loss based flows, the queueing delay grows above the threshold, the Cx-TCP flows should not backoff because of the delay; they should rather decrease their rate only after losses are experienced. Results in [26] show the very good performance of this congestion control algorithm. As for the LEDBAT algorithm however, Cx-TCP needs a correct estimation of the propagation delay in order to work properly. Besides, since it is a modification of the TCP protocol, its properties are not ideal for real time applications such as video conference applications, due to packet retransmission.

More recent solutions have been proposed in the framework of RMCAT, that is a working group of the IETF [16] which aims to standardize a congestion control algorithm for real time media applications. The proposals made in this group share similar objectives with our controller, since they both target the design of a congestion control algorithm that is able to work in different network environments. One proposal is NADA [27]. This congestion control algorithm makes use of a composite congestion signal to drive the flows at the equilibrium. The composed congestion signal is built by aggregating the measured one way queueing delay, the packet losses and an ECN [28] marking penalty. This algorithm needs to estimate the correct propagation delay of the route. As can be seen from [17], it seems to be able to maintain a consistent sending rate when competing against TCP flows. Unfortunately, the self-inflicted delay, which is the amount of queueing delay that the congestion control needs to reach equilibrium, attains remarkably high values when the available bandwidth is relatively low. However, NADA is still an ongoing work and according to the last version of the IETF draft the update rate equation has remarkably changed from the work [17]. In particular, the self-inflicted delay of the control algorithm in low capacity environments has been significantly reduced with the new design.

Another proposal in the RMCAT working group is the GCC algorithm [18]. This controller is mainly composed of two subparts. One subpart is the loss-based part, which uses the TFRC [29] throughput equation to achieve a fair rate when competing against TCP. The second subpart consists of a delay-based congestion control algorithm that estimates if the channel is underused or overused based on the measured OWD.

The delay-based algorithm in this case is pretty different from the other works. In fact, this algorithm uses the estimated variation of the queueing delay, rather than its absolute value. Hence it does not need to estimate the propagation delay. It has been shown in [19] that the original version of the GCC was suffering of starvation when competing against TCP flows, However in [30], the authors have proposed a modification of the delay-based algorithm that permits to alleviate this problem. The main issues of this delay-based algorithm is however the poor intra-protocol fairness [19] [20].

The controller proposed in this paper alleviates the problems of starvation when competing against loss-based flows. It also avoids the necessity of an estimation of the queueing delay, which may lead to latecomer unfairness if not done properly. Finally it achieves good intra-protocol fairness characteristics.

VII. Conclusion

In this work we have developed and analyzed a novel hybrid delay-based and loss-based congestion control algorithm that is able to maintain a low delay communication when network conditions allow for it, and to prevent starvation when competing against loss-based flows. We achieve a good flexibility between delay-based and loss-based modes by using in a single term, a congestion signal which depends on the interarrival time of the packets. We further prevent starvation of our LDCC algorithm when competing against loss-based flows by the use of a non-linear mapping between the experienced delay and the delay-based congestion signal. Moreover the non-linearity of the penalty function helps to mitigate unfairness problems when the data paths of the users have different propagation delays. Finally, by using the total experienced delay instead of the actual queueing delay, we avoid estimation problems and unfairness issues due to latecomer flows. We keep as further work some possible improvements of the LDCC algorithm, in particular with respect to TCP fairness and speed of convergence to the equilibrium point. The ability of achieving low delay at the equilibrium and the flexibility of being able to not starve against loss-based flows makes the LDCC algorithm a suitable congestion control algorithm to be used for delay sensitive network applications, e.g. video conferencing, over unmanaged networks such as Internet.

Appendix A
Non-linear Stability Proof

We start by analyzing the users subpart of the system. The time derivative of the storage function in Eq. (28) is given in Eq. (45a). It leads to $\dot{V}_{user} < -W_{user}(x) - 1/\alpha \delta y^T \delta \dot{d} - \delta y^T \delta d$ which has the form of a passive function similar to Eq. (26).

The result in Eq. (45e) comes from a first inequality in Eq. (45b), which holds due to the projection onto the positive orthant of Eq. (15), and from the second inequality in Eq. (45d), which holds if:

$$\frac{v(e) - v(\hat{e})}{\alpha} < \delta e. \tag{46}$$

This last inequality in Eq. (46) is true if the price function is chosen to be non decreasing with a maximum derivative equal to α, which is verified for our price function in Eq. (17). It remains to show that $W_{user}(x)$ is a positive definite function, which is needed to use the passivity theorem and eventually prove the stability of the system. The first term in $W_{user}(x)$ is: $-\delta x^T \left(U'(x) - U'(\hat{x})\right)$, which is always positive due to the concavity of the utility function. The two terms of Eq. (45e) will simplify when the derivative of the storage function of the user dynamics is summed with the links dynamics subpart.

For the second subpart of the system, namely the dynamics of the links, we now prove that it is also a passive system. We can easily calculate the time derivative of the storage function defined in Eq. (29):

$$\begin{aligned}
\dot{V}_{link} &= \delta q^T (y - c)_q^+ + \frac{1}{\alpha}(c - \hat{y})^T \dot{q} && \text{(47a)}\\
&\leq \delta q^T (y - c + \hat{y} - \hat{y}) + \frac{1}{\alpha}(c - \hat{y})^T \dot{q} && \text{(47b)}\\
&= \delta q^T (\hat{y} - c) + \delta q^T \delta y + \frac{1}{\alpha}(c - \hat{y})^T \dot{q} && \text{(47c)}\\
&\leq \underbrace{\delta q^T (\hat{y} - c)}_{-W_{link}(q)} + \delta d^T \delta y + \frac{1}{\alpha} \delta y^T \dot{d}. && \text{(47d)}
\end{aligned}$$

Where we used the fact that $\delta q = \delta d$ and $\dot{q} = \dot{d}$. Inequality (47b) is true due to the projection onto the positive orthant, while inequality (47d) holds if $\delta y^T \dot{q} \geq (c - \hat{y})^T \dot{q}$, this can easily be proved in the following way:

$$\begin{aligned}
\delta y^T \dot{q} - (c - \hat{y})^T \dot{q} &= (y - \hat{y})^T \delta \dot{q} - (c - \hat{y})^T \delta \dot{q} && \text{(48a)}\\
&= (y - c)^T \delta \dot{q} && \text{(48b)}\\
&= (y - c)^T (y - c)_q^+ > 0. && \text{(48c)}
\end{aligned}$$

The two right most terms in Eq. (47d) simplify with the two terms in (45e) when the two storage function are summed together. The missing part is to show that W_{link} is positive semidefinite, which means $(q - \hat{q})^T (\hat{y} - c) \leq 0$. In order to prove this consider the following deductions. If the link at the equilibrium is fully utilized, the value of the difference $\hat{y}_l - c_l$ is zero; otherwise it must be negative. At equilibrium a link rate cannot be greater than the link capacity, otherwise it would mean that the queue is growing contradicting the hypothesis of equilibrium. Alternatively, if the link is partially utilized the difference $q_l - \hat{q}_l$ is necessarily positive, since $\hat{q}_l = 0$; otherwise it can be either positive or negative. Combining these two deductions, we observe that the product of $\delta q^T (\hat{y} - c)$ is always non positive. Finally, the global stability of the undelayed control system for the case of no losses is guaranteed, as the system is the combination of two passive systems [7], and the sum of the two storage functions is a Lyapunov function for the entire dynamic system.

Acknowledgment

This work has been partly supported by the Swiss Commission for Technology and Innovation under grant CTI-13175.1 PFES-ES and by the Swiss National Science Foundation under grant CHISTERA FNS 20CH21_151569.

$$\dot{V}_{user} = \frac{1}{\alpha}\delta x^T \left(U'(x) - v(e) - \dot{e}\right)_x^+ \tag{45a}$$

$$\leq \frac{1}{\alpha}\delta x^T \left(U'(x) - v(\hat{e})\right) - \frac{1}{\alpha}\delta x^T \dot{e} - \frac{1}{\alpha}\delta x^T \left(v(e) - v(\hat{e})\right) \tag{45b}$$

$$= \frac{1}{\alpha}\delta x^T \left(U'(x) - U'(\hat{x})\right) - \frac{1}{\alpha}\delta x^T \dot{e} - \frac{1}{\alpha}\delta x^T \left(v(e) - v(\hat{e})\right) \tag{45c}$$

$$< \frac{1}{\alpha}\delta x^T \left(U'(x) - U'(\hat{x})\right) - \frac{1}{\alpha}\delta x^T \dot{e} - \delta x^T \delta e \tag{45d}$$

$$= \underbrace{\frac{1}{\alpha}\delta x^T \left(U'(x) - U'(\hat{x})\right)}_{-W_{user}(x)} + (-\frac{1}{\alpha}\delta y^T \dot{d}) + (-\delta y^T)\delta d \tag{45e}$$

REFERENCES

[1] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, pp. 237–252, 1998.

[2] S. Liu, T. Basar, and R. Srikant, "Controlling the internet: a survey and some new results," in *Proceedings IEEE Conference on Decision and Control*, vol. 3, 2003, pp. 3048–3057 Vol.3.

[3] F. Paganini, Z. Wang, J. Doyle, and S. Low, "Congestion control for high performance, stability, and fairness in general networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 43–56, 2005.

[4] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[5] J. Wang, D. X. Wei, J.-Y. Choi, and S. H. Low, "Modelling and stability of fast tcp," in *Wireless Communications*. Springer, 2007, vol. 143, pp. 331–356.

[6] J. Wen and M. Arcak, "A unifying passivity framework for network flow control," *IEEE Transactions on Automatic Control*, vol. 49, no. 2, pp. 162–174, 2004.

[7] J. Bao and P. L. Lee, *Process Control: The Passive Systems Approach*. Springer London, 2007.

[8] T. Voice, "A global stability result for primal-dual congestion control algorithms with routing," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 3, pp. 35–41, 2004.

[9] A. MacFarlane, "Return-difference and return-ratio matrices and their use in analysis and design of multivariable feedback control systems," *Proceedings of the Institution of Electrical Engineers*, vol. 117, no. 10, pp. 2037–2049, 1970.

[10] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, 2005.

[11] G. Vinnicombe, "On the stability of networks operating TCP-like congestion control," in *Proceedings of the IFAC World Congress*, 2002.

[12] H. Choe and S. H. Low, "Stabilized vegas," *In Proceedings of IEEE Infocom*, 2003.

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[14] L. Massoulie, "Stability of distributed congestion control with heterogeneous feedback delays," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 895–902, 2002.

[15] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984.

[16] "Rtp media congestion avoidance techniques (rmcat)," IETF Working Group, 2012. [Online]. Available: http://datatracker.ietf.org/wg/rmcat/charter/

[17] X. Zhu and R. Pan, "Nada: A unified congestion control scheme for low-latency interactive video," in *International Packet Video Workshop (PV)*, 2013, pp. 1–8.

[18] H. Lundin, S. Holmer, L. D. Cicco, S. Mascolo, and H. Alvestrand, "A google congestion control algorithm for real-time communication," Internet-Draft, 2013.

[19] L. De Cicco, G. Carlucci, and S. Mascolo, "Experimental investigation of the google congestion control for real-time flows," in *Proceedings of ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*. ACM, 2013, pp. 21–26.

[20] V. Singh, A. Lozano, and J. Ott, "Performance analysis of receive-side real-time congestion control for WebRTC," in *International Packet Video Workshop (PV)*, 2013, pp. 1–8.

[21] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low extra delay background transport (LEDBAT)," RFC-Experimental, 2012.

[22] G. Carofiglio, L. Muscariello, D. Rossi, C. Testa, and S. Valenti, "Rethinking the low extra delay background transport (LEDBAT) protocol," *Computer Networks*, vol. 57, no. 8, pp. 1838–1852, 2013.

[23] J. Schneider, J. Wagner, R. Winter, and H. Kolbe, "Out of my way-evaluating low extra delay background transport in an ADSL access network," in *International Teletraffic Congress (ITC)*. IEEE, 2010, pp. 1–8.

[24] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "Ledbat: the new BitTorrent congestion control protocol," in *Proceedings International Conference on Computer Communications and Networks*. IEEE, 2010, pp. 1–6.

[25] D. Wei, C. Jin, S. Low, and S. Hegde, "Fast tcp: Motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, pp. 1246–1259, 2006.

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

[27] X. Z. et al., "NADA: A unified congestion control scheme for real-time media," Internet-Draft, 2014.

[28] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," Proposed Standard, 2001.

[29] M. Handley, S. Floyd, J. Padhyea, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," 2003.

[30] L. De Cicco, G. Carlucci, and S. Mascolo, "Understanding the dynamic behaviour of the google congestion control for RTCWeb," in *International Packet Video Workshop (PV)*, 2013, pp. 1–8.