TCP-Puerto-Londero: A New Approach for End-to-End Queue Length Control

Luciano Mauro Arley Sup, Renato Mariz de Moraes, and Adolfo Bauchspiess

***Abstract*—This paper presents a new TCP protocol called TCP-Puerto-Londero, which makes dynamic tuning in the congestion window (*cwnd*) by means of adaptive control theory aiming to keep stable and small the queue length in the bottleneck link located on the path from source to destination. This adaptive control loop has the relative delay in the forward path measured as an input and the *cwnd* value as an output. Thus, unlike classic TCP protocols, TCP-Puerto-Londero does not use Round Trip Time (RTT) information. Also, unlike classic Active Queue Management (AQM) strategies and Explicit Congestion Notification (ECN) based protocols, TCP-Puerto-Londero employs end-to-end queue management without the need for ECN resources. Moreover, TCP-Puerto-Londero also aims to attend to the new challenges of the Industrial Internet of Things (IIoT). Its performance has been tested in a Dumbbell network topology shared by TCP and UDP-like Networked Control Systems (NCS) flows. Therefore, TCP-Puerto-Londero performance has been compared with ECN-based protocols like DCTCP, E-DCTCP, TCP-Jersey, and ENCN. Furthermore, this paper employs an approach for modeling, analysis, simulation, and verification of the communication network and NCS employing UPPAAL simulation software tool, where all network constituents (transmitters, channels, routers, receivers, controllers, and plants) were modeled employing timed automata, simplifying a formal verification of the complete studied system. Simulations and statistical verification indicate that even though utilizing fewer resources (as it does not require the AQM/ECN information) TCP-Puerto-Londero overcomes TCP-Jersey, DCTCP, and E-DCTCP with regard to throughput and fairness for TCP flows. Also, TCP-Puerto-Londero flows are capable to keep the queue length more stable and smaller than other protocols that were compared, and consequently, it reduces the impact in NCS-UDP-like flows sharing the same network, whose performance was measured employing the Integral Time Absolute Error (ITAE), which is a desirable feature for Industrial IoT.**

***Index Terms*—AQM, Internet of Things, NCS, TCP, Timed Automata Modeling.**

I. Introduction

Regularly the TCP protocol obtains congestion control using an end-to-end approach that calculates the proper sender congestion window according to the estimated traffic conditions. Furthermore, modern electronic devices enable to employ control algorithms at routers, such that besides the end-to-end window control strategies, other control schemes have been formulated called Active Queue Management (AQM) [1]–[4]. The fundamental AQM technique is the Drop Tail (DT) strategy in which packets arriving in a queue are dropped with probability one when the queue becomes full. Nonetheless, as a way to avoid problems like global synchronization [1] and bufferbloat inconveniences [2] various other AQM schemes have been considered such as Random Early Detection (RED) [1], Proportional Integral controller Enhanced (PIE) [3], Control Delay (CoDel) [2], and their variations [4]. Such algorithms, besides discarding packets in the queue, also allow marking packets, for example by setting a bit, if the Explicit Congestion Notification (ECN) resources are enabled [5], [6]. Other ECN-based protocols, such as TCP-Jersey [7], DCTCP [8], [9], E-DCTCP [10], and ENCN [11], [12] also match AQM schemes employed in routers with updates in the TCP window considering the ECN-bit information.

These notification dynamics can produce bufferempty phenomenon, that occurs if the AQM scheme overly restrains all TCP transmitters or if the transmitters overreact to congestion warnings to such a degree that the queues in routers can stay empty for various intervals of time. This bufferempty phenomenon can result in a serious decrease in TCP throughput.

Besides, the evolution of automation in industry, vehicles, and more lately, in building automation systems (BAS), and other applications, like smart cities and smart grids, have led to new challenges for the research and manufacturing of novel control systems. In order to effectively attain these goals, previous work has considered Networked Control Systems (NCS) [11]–[19].

For this type of control strategies, the controller and the plant can be situated in different places and they can exchange data through a communication network composing a control loop where the sensor, the controller, and the process can be physically distant. Furthermore, in some NCS architecture, the plant and controller can employ the Internet for data communication resulting in various important research issues compounding features of NCS and the Internet. At present, for instance, the Industrial Internet of Things (IIoT) is a solution in which many such novel scenarios are jointly studied. Therefore, new control techniques, packet drop and delay reduction, reliability, and development of novel data communication protocols are needed [18]–[23].

Accordingly, to prevent the undesired bufferempty phenomenon and to exploit the non-congestion state of the queue to additionally enhance the TCP performance without incurring bufferbloat issues, and yet attaining the NCS prerequisites,

L. M. A. Sup is with Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília-DF 70910-900, Brazil. Email: mauroarleysup@gmail.com

R. M. de Moraes is with Centro de Informática, Universidade Federal de Pernambuco, Recife-PE 50740-560, Brazil. Email: renatomdm@cin.ufpe.br. Orcid: 0000-0002-3437-7004.

A. Bauchspiess is with Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília-DF 70910-900, Brazil. Email: adolfobs@ene.unb.br. Orcid: 0000-0002-4512-4299.

This work was supported in part by Centro de Informática (CIn-UFPE), and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil.

Digital Object Identifier: 10.14209/jcis.2023.12

it is possible to employ ECN-based protocol called Explicit Non-Congestion Notification (ENCN) [11]. Results revealed that ENCN exceeds RED, CoDel, and PIE AQM schemes regarding throughput and fairness for TCP flows, and the Integral Time Absolute Error (ITAE) for NCS-UDP-like flow. Notwithstanding, ENCN uses ECN resources to make explicit non-congestion notifications, features that are not always available [5], [6], [24]. Thus, ENCN can be used only when ECN features are available. In order to overcome such ECN limitations, in this paper, a new protocol called TCP-Puerto-Londero is proposed, which aims to keep the queue length stable and small, based on the measurement of a relative delay in the forward path (*rdfp*) existing between the transmitter and receiver, instead to use the round trip time (RTT) and/or ECN/ENCN notifications.

TCP-Puerto-Londero implements adaptive control techniques with a dead band to act on the congestion window (*cwnd*) in order to keep the *rdfp* close to a reference forward delay (*rdfpTarget*). The adaptive control law is based on the dynamic computation of the communication network transfer function, whose input is the *cwnd*, and whose output is the *rdfp* value.

The rest of this paper is organized as follows. Section II discusses the relative delay in the forward path (*rdfp*). Section III extends the discussion for the TCP-Puerto-Londero. Section IV describes the new TCP-Puerto-Londero protocol, discusses the transfer function used in the control strategy, and presents the controller design of the new protocol and its algorithms. The results are presented and discussed in Section V. Finally, Section VII concludes the paper.

II. Relative delay in the forward path (RDFP)

Usually, classic TCP protocols measure the RTT based on the transmitter clock. So, every time that a transmitter sends a data packet, it will put in the Timestamp (*TSval*) field of the IP header a timestamp containing the time (measured on the transmitter clock) on which this packet is being sent (see RFC 1323 [25]).

When the receiver receives this packet, it will transfer this timestamp information to the timestamp echo reply field (*Tsecr*) of the corresponding acknowledgment (ACK) IP packet header to be sent back to the transmitter. Finally, when this ACK arrives at the transmitter, the RTT can be computed as

$$RTT = CurrentTime \quad Tsecr; \tag{1}$$

where both *Tsecr* and *CurrentTime* are measured on the transmitter clock. Hence, there is no participation of the receiver clock. So, it is not necessary to synchronize the receiver and transmitter clocks to measure the RTT. However, the RTT depends on both congestions in the forward and return paths. Therefore, RTT-based protocols may react to congestion in the return path and cause bufferempty on the forward path. This bufferempty phenomenon can affect the throughput performance.

The apparent solution to this problem would be to use the delay in the forward path (*dfp*) instead of using the RTT. However, it is not trivial to measure the forward delay, because synchronization of the transmitter and the receiver clocks could be necessary. Aiming to measure the *dfp* without synchronizations, some solutions have been proposed in the literature [26]. In our work, we propose to use a relative delay in the forward path (*rdfp*), which will be relative to the time differences between the transmitter and the receiver clocks. Accordingly, the *rdfp* is computed on the receiver as

$$rdfp = TSval \quad tar; \tag{2}$$

where *tar* (time of arrival in the receiver) is the time instant that the data packet arrived at the receiver, measured on the receiver clock, which does not need to be synchronized with the transmitter clock. Then, this information is sent back to the transmitter in the *Tsecr* field of the corresponding ACK. So, *rdfp* is the delay in the forward path relative to the time differences between the transmitter and the receiver clocks. Note that *rdfp* may be different from *dfp* when the transmitter and receiver clocks are not synchronized. However, *rdfp* and *dfp* will always follow the same dynamic regardless of whether the transmitter and receiver clocks are synchronized or not. Consequently, both *rdfp* and *dfp* reach their minimum values when the packet is routed through the shortest path and there is no congestion in the forward path. Thus, our TCP-Puerto-Londero proposal detects congestion based on variations of *rdfp* according to the method described next.

III. Relative delay in the forward path in TCP-Puerto-Londero

TCP-Puerto-Londero makes dynamic *cwnd* adjustments based on the *rdfp*. When the transmitter receives an ACK it uses the *rdfp* information (given in the *Tsecr* field) to make adjustments in *cwnd* in order to keep the *rdfp* close to a given reference value (*rdfpTarget*). The *rdfTarget* is computed as

$$rdfpTarget = rdfpmin + aqd; \tag{3}$$

where *rdfpmin* is the minimum *rdfp* observed from the beginning of the transmission or from a reset (carried out whenever *cwnd* reaches the value 1) until the current instant. The admissible queuing delay (*aqd*) is a constant whose recommended value is between 5 and 20 milliseconds (ms), analogous to the waiting time in the queue considered to be acceptable in algorithms such as CoDel [2] and PIE [3].

A. Dead band

The task of maintaining *rdfp* at a given reference value through actions on the congestion window can be complex since the segments to be transmitted are integers. Thus, in a situation where *rdfp* is smaller than *rdfpTarget*, an increment in the transmission window (*W*) of one segment could cause the *rdfp* to overcome the *rdfpTarget*, which would force a reduction on the transmission window of at least one segment. But, this reduction could make *rdfp* again smaller than *rdfpTarget*, and thus, the process would be repeated indefinitely. Moreover, several transient events that occur in the network would make it extremely difficult to keep the delay exactly at a reference value. In order to avoid these oscillations and to give better stability to the controller, TCP-Puerto-Londero uses a

dead band, that is, when the *rdfp* is between a region close to *rdfpTarget*, the controller will be turned off. Consequently, the value of *cwnd* will be kept constant, and the controller will only be turned on again if and when the *rdfp* leaves the dead band region, which is adjusted to be

$$rdfpTarget \qquad rdfp < rdfpTargget + \quad ; \tag{4}$$

where is a constant higher than zero and lower than *rdfpTargget*.

IV. TCP-Puerto-Londero

Aiming to keep the *rdfp* next to the *rdfpTarget*, TCP-Puerto-Londero uses a control loop, whose plant is the whole communication network that connects the transmitter and receiver. This plant has an input *cwnd* and generates one output *y(k)* given by the *rdfp*, which is used to compute the error *e(k)* (difference between *rdfpTarget* and *rdfp*). The error is the input for the adaptive controller whose output *u(k)* is the *cwnd* value. This control loop is illustrated in Fig. 1.

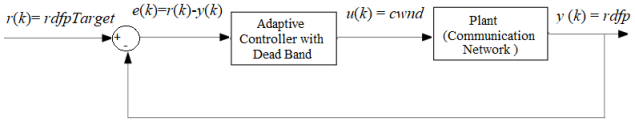

Fig. 1. TCP-Puerto-Londero adaptive control.

On the other hand, the communication network has time-dependent parameters. For example, the queue length in the routers is variable, the number of flows connected simultaneously in the communication network is variable, the drop and packet loss rate can vary, and the paths chosen by routers are dynamic.

Consequently, the plant transfer function parameters vary with time. Hence, it is necessary a kind of controller whose parameters are able to change according to the process dynamics. This kind of controller is called adaptive controller [27], and its design is based on a nominal controller whose parameters are set according to the nominal transfer function.

A. Nominal transfer function identification

In the first part of the adaptive control project that is used in TCP-Puerto-Londero, we employ a known (nominal) communication network topology (Plant), whose parameters of a nominal transfer function $G(z)$ were identified. Then, this nominal transfer function is utilized for the project of the nominal controller ($C(z)$). The nominal communication network used for this proposal is the Daisy-chain topology illustrated in Fig. 2 on which a TCP-Reno flow passes through two routers connected by a 1.5 Mbps bottleneck link, whose transfer function $G(z)$ was approximated by a first-order transfer function given by

$$G(z) = \frac{2}{z \quad 1}: \tag{5}$$

$_1$ and $_2$ are the corresponding communication network transfer function parameters estimated by means of a Least Squares Estimator with forgetting factor algorithm [27]. Thus, for every *hw* time unit (where *hw* is the interaction period), the transmitter computes the following recursive equations

$$K_k = \frac{P_{k-1} \quad k}{\frac{T}{k} P_{k-1} \quad k +}; \tag{6}$$

$$_k = \quad _{k-1} + K_k[y(k) \quad \frac{T}{k} \quad _{k-1}]; \tag{7}$$

$$P_k = \frac{1}{-}\left(P_{k-1} \quad \frac{P_{k-1} \quad k \quad \frac{T}{k} P_{k-1}}{\frac{T}{k} P_{k-1} \quad \frac{T}{k} +}\right); \tag{8}$$

where:

- is the forgetting factor, i.e., a measure of the speed in which the old data were forgotten, whose recommended values are between 0.975 and 1 [27];
- $y(k)$ is the process output, i.e., the *rdfp* value;
- $_k$ is the regressor vector with information until k-1;
- $\frac{T}{k}$ is the $_k$ transposed;
- K_k is the adaptation gain, also referred to as Kalman gain;
- P_k is the covariance matrix of the process;
- $_k$ is the estimated parameter vector.

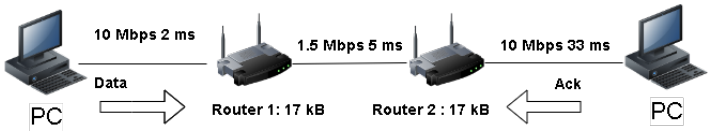

Fig. 2. Nominal communication network.

The dynamic estimation of $_1$ and $_2$ are illustrated in Figs. 3 and 4, respectively.

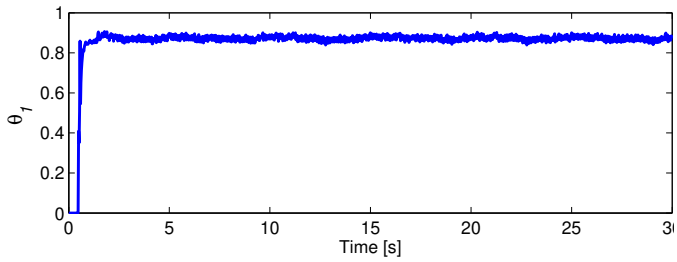

Fig. 3. Parameter $_1$ for the nominal communication network.

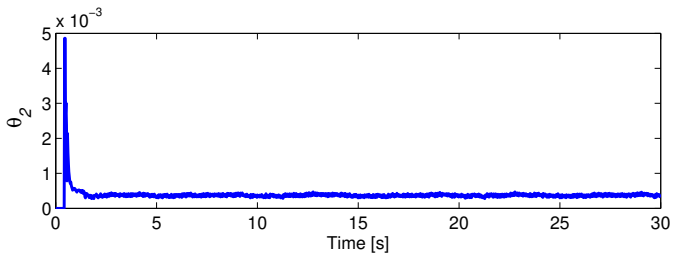

Fig. 4. Parameter $_2$ for the nominal communication network.

Accordingly, the transfer function of the nominal communication network illustrated in Fig. 2 can be written as

$$G(z) = \frac{0{:}0003678}{z \quad 0{:}8746}: \tag{9}$$

The TCP-Puerto-Londero congestion window (*cwnd*) dynamic for the nominal communication network is illustrated in Fig. 5, as can be seen, its value remains constant when the network transfer function parameters get stable.

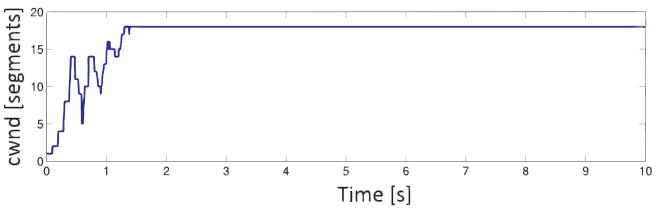

Fig. 5. Dynamic of *cwnd* for the nominal communication network.

B. TCP-Puerto-Londero algorithms

Basically, TCP-Puerto-Londero has two phases, the Slow-Start phase summarized in the pseudo-code of Algorithm 1 and an adaptive phase described in the pseudo-code of Algorithm 2. The TCP-Puerto-Londero parameters are summarized in Table I. Initially, the transmitter and receiver employ the traditional TCP-Reno protocol. When the transition starts (in the Slow-Start phase), the communication network parameters are unknown. So, the process parameters are initialized with the values obtained for the nominal network (Fig. 2), i.e., $_1 = 0.8746$ and $_2 = 0.0003678$. *rdfpmin* is also unknown, and it is initialized with a high value, which will be re-adjusted to the *rdfp* value obtained from the *Tsecr* field of the first ACK packet that arrives at the transmitter. Then, when subsequent ACKs arrive at the transmitter the *rdfpmin* will be always re-computed as the minimum value between the current *rdfpmin* and the *rdfp* value obtained from the *Tsecr* field of the corresponding ACK. However, the minimum delay between transmitter and receiver could increase by non-congestion events, such as a change in router path or an increase in the distance between transmitter and receiver (which is possible in wireless mobile communications). The transmitter might assume that this increase in the *rdfp* may be caused by congestion, and so, an unnecessary reduction in *cwnd* might occur. Thus, aiming to avoid this problem, when *cwnd* reaches the value one, the *rdfpmin* is adjusted to the *rdfp* value obtained from the *Tsecr* field of the last ACK received. Note that, since no new fields were added or removed to the TCP-header, the TCP-Puerto-Londero is still compatible with classic versions of TCP.

1) The slow-start phase: The TCP-Puerto-Londero initiates and remains in Slow-Start until the *rdfp* value overcomes the superior limit of the dead band, and this phase is analogous to the Slow-Start phase of the TCP-Reno protocol [26], [28]. However, unlike TCP-Reno, in this phase, TCP-Puerto-Londero also makes the communication network parameters estimations, by means of the recursive equations (6), (7), and (8). These estimations occur every *hw* time units by along all transitions lifetime (in both, Slow-Start and Adaptive phases).

The Slow-Start phase is outlined in the pseudo-code of Algorithm 1, on which, at every *hw* time units the Interaction() function is called. This function updates the communication network parameters according to equations (6), (7), and (8). Moreover, when ACK arrives the ComputeW() function is called, which computes the *W* value, analogous to the TCP-Reno Slow-Start phase (see [28]). When a timeout occurs, the retransmission() function is called, which retransmits the data packet sequence considered as lost, and the TCP-Puerto-Londero remains in the Slow Start phase. The retransmission due to timeout (RTO) is initialized in 100 ms, and then, when ACKs begin to arrive at the transmitter, the RTO is updated to *aqd* times the *rdfp*. When the *rdfp* value overcomes the superior limit of the dead band, the TCP-Puerto-Londero protocol goes to the Adaptive phase.

Algorithm 1: Slow-start phase of TCP-Puerto-Londero

```
if rdfp  rdfpTarget +  then
    At every hw time units {call Interaction()};
    if Ack received then
        Call ComputeW();
        RTO  aqd  rdfp;
        if if W = 1 then
            rdfpmin  rdfp;
        else
            rdfpmin  min(rdfpmin; rdfp);
        end
    else
        if Timeout occurs then
            cwnd  1;
            Call retransmission();
        end
    end
else
    Go to Adaptive-Phase();
end
```

2) Adaptive phase: The Adaptive phase is summarized in the pseudo-code of Algorithm 2, on which, upon an ACK reception the *W* value is computed as the integer closer to the minimum between *rwnd* and *cwnd*, and the RTO is set as *aqd* times the current *rdfp* value. Moreover, the *rdfpmin* is updated to the minimum value between the current *rdfpmin* and the *rdfp* value obtained from the *Tsecr* field of the corresponding ACK. If *W* reaches the value one, the *rdfpmin* is reset as the current *rdfp* value. At every *hw* time unit, the Interaction() function is called, which updates the communication network parameters accordingly the equations (6), (7), and (8). Furthermore, if and only if the *rdfp* value is away from the dead band (equation (4)) at every *hw* time unit also the Adaptive() function is called, which computes the *cwnd* value according to the control law *u(k)* of the adaptive controller with the dead band.

3) TCP-Puerto-Londero drawbacks: Among the main disadvantages of TCP-Puerto-Londero, we can mention: (i) Dependence on a large number of parameters, whose initial values can affect the transient regime. (ii) Possible loss of performance when a TCP-Puerto-Londero sender transmits to a receiver that uses another version of TCP. Because normally other versions of TCP measure the RTT instead of measuring the rdfp, TCP-Puerto-Londero would lose its particularity of

Algorithm 2: Adaptive phase of TCP-Puerto-Londero

```
if Ack received then
    W    min(rwnd; cwnd);
    RTO    aqd    rdfp;
    if W = 1 then
        rdfpmin    rdfp;
    else
        rdfpmin    min(rdfpmin; rdfp);
    end
    if rdfpTarget -    rdfp < rdfpTarget +    then
        At every hw time units fcall Interaction()g;
    else
        At every hw time units fcall Interaction() and
          Adaptive()g;
    end
else
    if Timeout occurs then
        Call retransmission();
        cwnd    1;
        Go to SlowStart-Phase();
    end
end
```

TABLE I
TCP-PUERTO-LONDERO PARAMETERS

Parameter	Description
aqd	Admissible queuing delay, a constant whose recommended value is between 5 and 20 milliseconds (ms), analogous to the waiting time in the queue considered to be acceptable in algorithms such as CoDel and PIE.
cwnd	Congestion window.
hw	Interaction period on which the TCP-Puerto-Londero updates the communication network parameters according to the adaptative control equations.
rdfp	Relative delay in the forward path.
rdfpmin	Minimum rdfp observed from the beginning of the transmission or from a reset (carried out whenever cwnd reaches the value 1) until the current instant.
rdfpTarget	Reference forward delay.
RTO	Retransmission due to timeout.
W	Transmission window.

acting on its cwnd based on forward path delay.

V. TCP-PUERTO-LONDERO PERFORMANCE IN A DUMBELL TOPOLOGY WITH NCS AND MULTIPLE CONCURRENT FLOWS

Control systems are commonly used over the Internet in various applications. Some examples include: Industrial Automation, Smart Home, Energy Management and Telemedicine [11]–[15], [17]–[19]. This type of control is sensitive to delays, packet loss, congestion, and delays caused by other data flows that share the same network topology [18]–[23]. Since different data transport protocols can introduce varying delays, and improving throughput may result in increased delay for some protocols, in this section, the TCP-Puerto-Londero performance has been compared with the follow ECN based protocols: TCP-Jersey, E-DCTCP, and ENCN protocols in a Dumbell topology (depicted in Fig. 6), where three generic TCP flows A, B, and C, and one NCS-UDP-like flow share the same network bottleneck with two routers. We chose these protocols to evaluate the performance of TCP-Puerto-Londero against protocols that use the ECN-Bit.

The whole network was modeled in the UPPAAL software tool [11], [12], [29], [30]. The performance has been measured in relation to throughput and fairness for TCP flows by using the fairness index of Jain [31].

Furthermore, depending on how TCP adjusts its transmission window, it can be more or less aggressive and generate more or less latency on the network. Delays are critical for NCS [13]–[17], so different TCP protocols have different effects on control flows that share the same network topology.

So, to weigh up the NCS performance, it was used the Integral Time Absolute Error (ITAE) measure (which is indirectly affected by queue delays introduced by the TCP flows that share the same network topology). Therefore, the best result for control systems is the one with the lowest ITAE value.

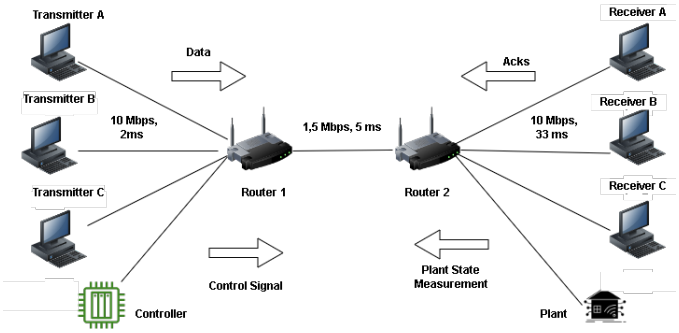

Fig. 6. Dumbbell topology with three TCP flows (A, B, and C Transmitter-Receiver pairs), and one NCS-UDP-like flow (Controller-Plant pair).

As an example, the network control system used here is a proportional-integral (PI) controller with K_p= 11.86 and K_i = 47.45 employed to control the position of a DC motor Maxon F2140 through the Internet. This control loop should not be confused with the control loop working inside of the TCP-Puerto-Londero transmitters depicted in Fig. 1. The motor was discretized with sampling period h = 0.014 seconds (s). In our simulations, the motor position follows a square wave reference $R(k)$ varying from 1 to 2 radians with a period of 2 seconds. In addition, a 3 volts disturbance was added at 7 seconds of simulation.

This type of controller could be used in applications such as smart city, smart grid, and smart home [11]–[19].

A. Results

The simulation was run for 20 seconds, enough for TCP-Puerto-Londero to enter in steady state, remaining within the dead band without significant oscillations in the congestion window (*cwnd*), and therefore in the transmission window (W), as can be seen in the Fig. 9, 10 and 11. The total throughput (sum of the throughput of flows A, B, and C) versus time for TCP-Jersey, E-DCTCP, ENCN, and TCP-Puerto-Londero are presented in Fig. 7.

Accordingly, even employing fewer resources as it does not need the AQM/ECN information, TCP-Puerto-Londero

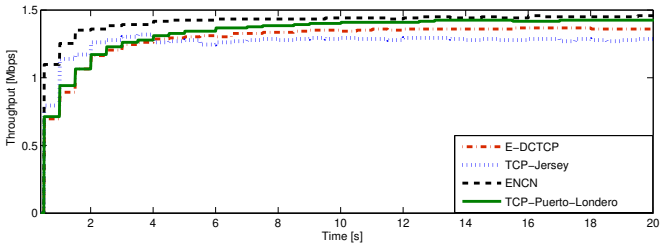

Fig. 7. Throughput for TCP flows.

presents a throughput very close to ENCN, overcoming TCP-Jersey and E-DCTCP as simulation evolves. This improvement in throughput is a consequence that after making the network topology transfer function estimation TCP-Puerto-Londero avoids bufferempty phenomenon as shown in Fig. 12. Thus, the router remains busy while there are packets to transmit. The ENCN also avoids bufferempty phenomenon, as shown in Fig. 13, and provides better throughput than TCP-Jersey and E-DCTCP. However, ENCN requires AQM/ECN resources implemented in Internet routers. Moreover, TCP-Puerto-Londero also presents better fairness regarding Jain's index for the three TCP flows (A, B, and C) as illustrated in Fig. 8.

This advantage with regard to fairness is a consequence of the TCP-Puerto-Londero adaptive controller, whereby each TCP flow estimates practically the same network transfer function, consequently, in steady state, the three flows tend to the same transmission window (W) as can be seen in Figs. 9, 10, and 11, thus, providing a fair sharing of the available band.

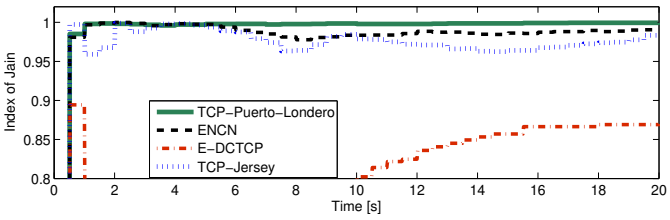

Fig. 8. Jain's Fairness Index.

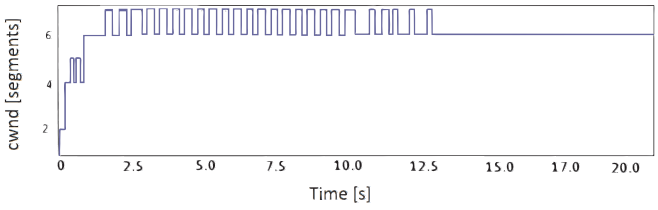

Fig. 9. Dynamic of *cwnd* for TCP-Puerto-Londero Flow A.

The queue length dynamics for TCP-Puerto-Londero, ENCN, TCP-Jersey, and E-DCTCP are illustrated in Figs. 12, 13, 14, and 15, respectively. Accordingly, TCP-Puerto-Londero presents some oscillations in queue length while

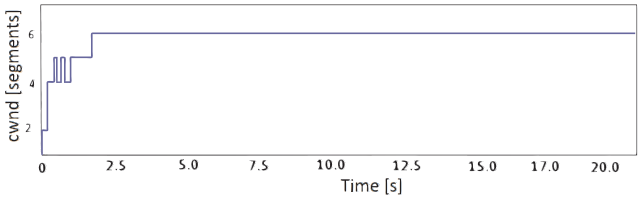

Fig. 10. Dynamic of *cwnd* for TCP-Puerto-Londero Flow B.

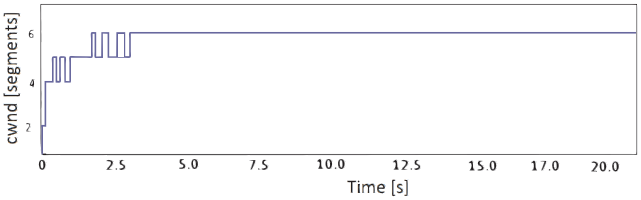

Fig. 11. Dynamic of *cwnd* for TCP-Puerto-Londero Flow C.

unaware of the network topology transfer function parameters. But, once these parameters are correctly estimated the queue remains stable and small, and its performance improves considerably.

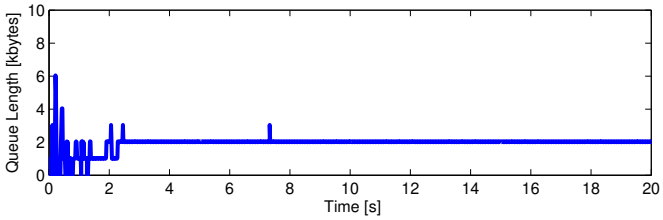

Fig. 12. Queue dynamic for TCP-Puerto-Londero.

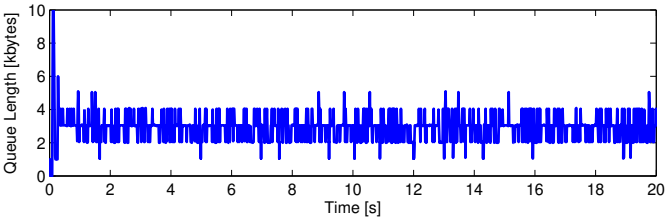

Fig. 13. Queue dynamic for ENCN.

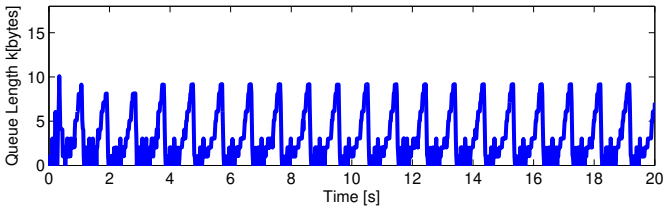

Fig. 14. Queue dynamic for TCP-Jersey.

This queue dynamics performance favors the NCS-UDP-like flow that shares the same network topology. Thus, as

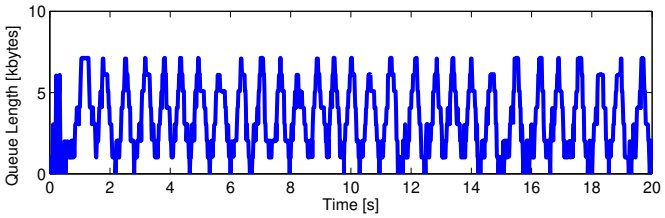

Fig. 15. Queue dynamic for E-DCTCP.

illustrated in Fig. 16, when the NCS-UDP-like flows share the network topology with TCP-Puerto-Londero flows, the ITAE for the NCS is better than when the network topology is shared with TCP-Jersey, E-DCTCP, or ENCN protocols.

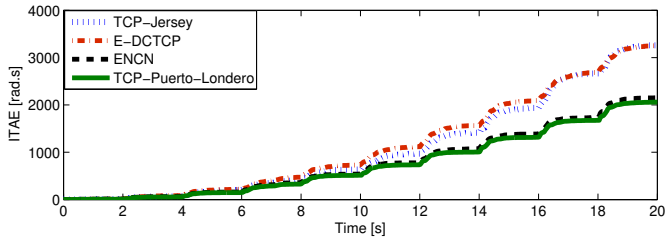

Fig. 16. ITAE for NCS-UDP-like flow.

Figs. 17, 18, 19, and 20 show the motor position following a square wave reference $R(k)$ between 1 to 2 radians with a period of 2 seconds when the NCS-UDP-like flow shares the network with Puerto-Londero, ENCN, TCP-Jersey, and E-DCTCP, respectively. Accordingly, TCP Puerto-Londero induces smaller amplitude oscillations.

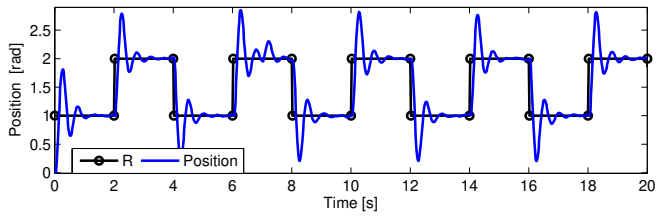

Fig. 17. Motor Position when NCS shares the network with TCP-Puerto-Londero flows.

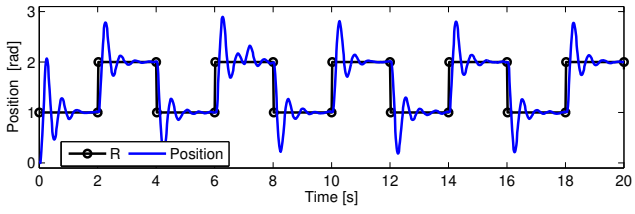

Fig. 18. Motor Position when NCS shares the network with ENCN flows.

Table II summarizes the final values reached within 20 s of simulation for total throughput, Jain's index, and ITAE for each of the simulated protocols. TCP-Puerto-Londero reduces the ITAE by 37% compared to TCP-Jersey and E-DCTCP, and

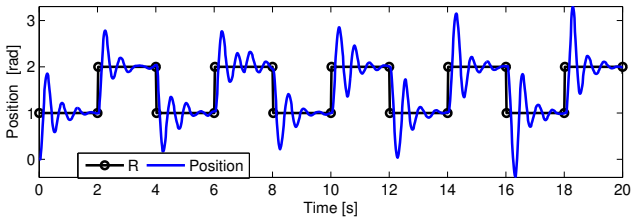

Fig. 19. Motor Position when NCS shares the network with TCP-Jersey flows.

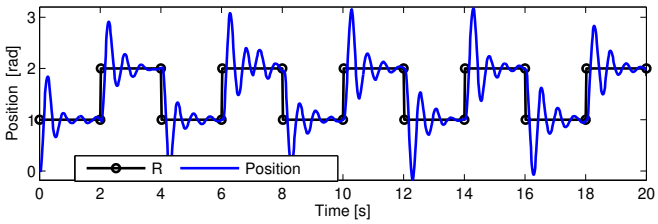

Fig. 20. Motor Position when NCS shares the network with E-DCTCP flows.

by 4% compared to ENCN. TCP-Puerto-Londero also had a throughput of 12% and 4% higher than TCP-Jersey and E-DCTCP, respectively.

TABLE II
FINAL VALUES REACHED AT 20 S OF SIMULATION FOR TOTAL THROUGHPUT, JAIN'S INDEX, AND ITAE FOR THE SIMULATED PROTOCOLS

Protocol	Throughput (Mbps)	Jain's Index	ITAE (rad.s)
TCP-Jersey	1.280	0.985	3261.2
E-DCTCP	1.376	0.952	3253
ENCN	1.449	0.998	2149.3
Puerto-Londero	1.434	1	2059.8

Note that the ITAE illustrated in Fig. 16 is the result of one simulation run. However, deterministic and random events happen on the Internet. Hence, one simulation run captures only one of the many possibilities. Therefore, we employ the UPPAAL SMC (Statistical Model Checking) [29], [30], in order to analyze the probability for several simulations (all of them with a 95% confidence level). Thus, Table III presents the probability of ITAE being greater than 3500 rad.s until 20 s (which is the worst case value shown in Fig. 16) when NCS-UDP-like flow shares the network with different protocols. Fig. 21 illustrates the corresponding Cumulative Probability Distribution. ITAE obtained with the implementation of the TCP-Jersey protocol is more likely to exceed 3500 rad.s, because TCP-Jersey presents a higher bufferbloat phenomenon than the other investigated protocols, and consequently, it introduces greater queue delays, which causes an indirect effect in the NCS-UDP-like flow that uses the same network. Besides, TCP-Puerto-Londero presents the lowest probability to exceed 3500 rad.s ITAE value, because the queue length remains small when this protocol is implemented (see Fig. 12).

We also calculated the probability of the total throughput, i.e. the aggregate TCP throughput of flows A, B, and C being greater than 1.45 Mbps in up to 20 s. Table IV presents the estimated probability interval with 95% confidence for TCP-Jersey, E-DCTCP, ENCN, and TCP-Puerto-Londero. Fig. 22

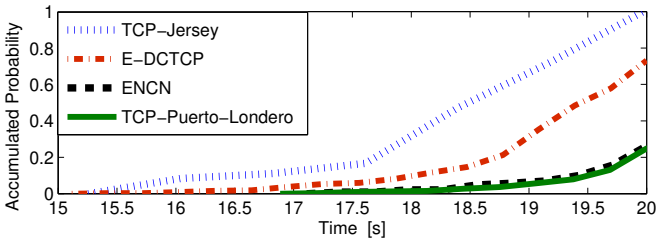

Fig. 21. Cumulative probability of ITAE being greater than 3500.00 rad.s.

TABLE III
PROBABILITY INTERVAL OF ITAE BEING GREATER THAN 3500 RAD.S UNTIL 20 S

Protocol	Probability Interval
TCP-Jersey	0.902 to 1
E-DCTCP	0.672 to 0.772
ENCN	0.238 to 0.338
TCP-Puerto-Londero	0.234 to 0.334

illustrates the corresponding cumulative probability distribution.

TABLE IV
PROBABILITY INTERVAL FOR THROUGHPUT BEING GREATER THAN 1.45 MBPS IN UP TO 20 S

Protocol	Probability Interval
TCP-Jersey	0 to 0.097
E-DCTCP	0.074 to 0.174
ENCN	0.637 to 0.737
TCP-Puerto-Londero	0.637 to 0.737

TCP-Puerto-Londero and ENCN are most likely to generate a total throughput greater than 1.45 Mbps in up to 20 s. On the other hand, there are no curves for TCP-Jersey because when these protocols were implemented in such network topology the total throughput does not overcome 1.45 Mbps in up to 20 s under any possible situation.

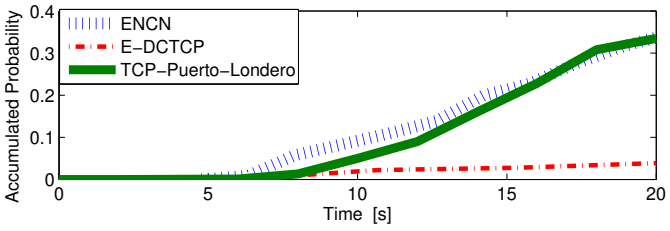

Fig. 22. Cumulative probability distribution for the throughput being greater than 1.45 Mbps in up to 20 s of simulation.

From the presented results, despite TCP-Puerto-Londero does not require AQM/ECN resources, it presents good throughput and fairness performance for the TCP flows while it provides good ITAE behavior for the NCS-UDP-like control flow compared to the other protocols. TCP-Puerto-Londero also introduces minor queue delay and keeps the queue length more stable and smaller than the other considered protocols.

VI. TCP-PUERTO-LONDERO PERFORMANCE WITH LATE FLOWS

In this section, we present TCP-Puerto-Londero's performance in terms of its ability to make adaptive control with late flows. For this purpose, the communication network topology illustrated in Fig. 6 is used, in which there are three TCP-Puerto-Londero flows (A, B, and C). At the beginning, only Flow A will be turned on. At 10 seconds (s), Flow B is turned on, and finally at 20 s flow C will be turned on. The simulation will be run for 30 s. The objective is to test if TCP-Puerto-Londero can readjust its control law due to changes that occur in the communication network and if the queue in the router remains stable without the unwanted bufferbloat and bufferempty phenomena.

The initial values of process parameters $_1$ and $_2$ on the three transmitters (A, B, and C) were 0.876 and 0.003678, respectively. These values are dynamically readjusted in each transmitter by means of the recursive equations (6), (7), and (8).

Figs. 23 and 24 illustrate the dynamics of $_1$ and $_2$ parameters estimation, respectively. As it can be seen, transmitter A keeps practically the same initial values of $_1$ and $_2$ parameters during the first 10 s of simulation. Then, when transmitter B starts in the Slow Start phase, transmitter A observes changes in communication response, that is, it realizes that the same input value (*cwnd*) generates a different output (*rdfp*), and then readjusts its $_1$ and $_2$ parameters. At the same time, transmitter B also observes that its initial $_1$ and $_2$ parameters need to be updated and make a quick readjustment of their values. During this transient in which transmitters A and B recompute the values of their $_1$ and $_2$ parameters, oscillations in the queue occur in the forward path of router 1, as shown in Fig. 25. However, this transient is short-lived and soon both transmitters A and B can again stabilize the queue size with small oscillations around 2 kBytes (2 packets). This value is maintained until 20 s when transmitter C begins to transmit packets, and a new transient effect is observed. Thus, undesired oscillations in queue length are observed only briefly after a new flow is turned on. Then, all TCP-Puerto-Londero transmitters recompute their parameters to the new traffic conditions and can again stabilize the queue length around small non-zero values. Thus, it avoids both the bufferbloat phenomenon (large queues) and the bufferempty phenomenon (absence of packets in attendance in the queues).

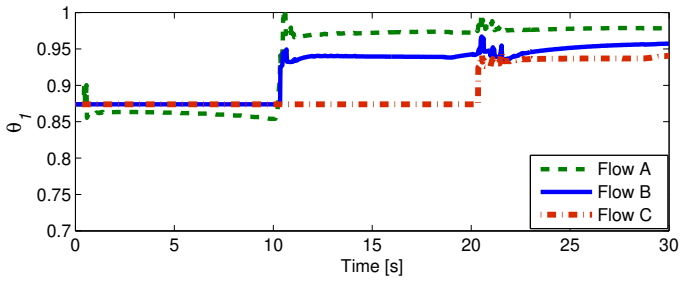

Fig. 23. Parameter $_1$ for the nominal communication network.

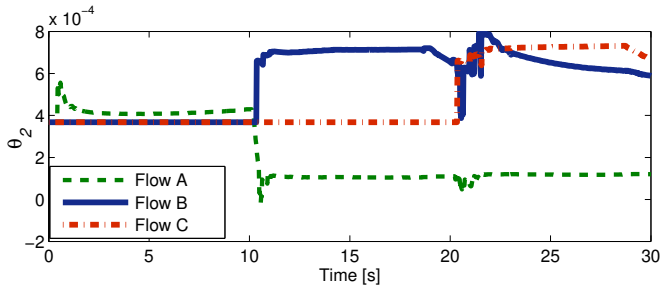

Fig. 24. $_2$ parameter estimation regards to late flows.

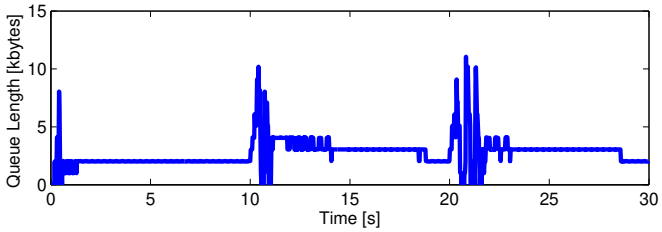

Fig. 25. Queue length dynamic in router 1 regarding to late flows.

VII. Conclusions

This work presents a new protocol called TCP-Puerto-Londero, that makes dynamic tuning in the congestion windows (cwnd) of transmitters by means of adaptive control theory and dead band control, aiming to keep stable and small the queue length in the bottleneck link. The new protocol avoids both bufferbloat and bufferempty undesired phenomena observed in TCP networks.

For this purpose, each TCP-Puerto-Londero transmitter uses a least squares estimator with a forgetting factor algorithm in order to dynamically estimate the transfer function of the communication network existent between transmitter and receiver. Then, based on these estimated parameters, each TCP-Puerto-Londero transmitter computes a control law in order to keep the delay on the forward path close to a reference delay. Unlike ECN-based protocols such as TCP-Jersey, DCTCP, E-DCTCP, and ENCN, TCP-Puerto-Londero does not need to work together with AQM techniques and does not need technological resources to do explicit congestion notification, since these events are detected implicitly based on a relative delay between the transmitter and receiver. The TCP-Puerto-Londero protocol was detailed and compared through simulation and statistical verification with TCP-Jersey, E-DCTCP, and ENCN schemes, where three generic TCP flows share the same network bottleneck of two routers with an NCS-UDP-like flow. Accordingly, TCP-Puerto-Londero improved throughput performance, along with better fairness regarding Jain's index for the generic TCP flows. The new protocol provides a more stable and small queue length dynamics and consequently introduces smaller network delays than other compared protocols, improving the NCS-UDP-like flow that shares the same network. Modeling and simulations employ the UPPAAL software tool for timed automata systems employing a language based on CTL [29], [30]. Timed automata modeling simplifies the study of network communication, along with UDP networked control systems (NCS) jointly implemented with other TCP flows, making easy a formal verification of the whole modeled system. Future works can expand the analysis to other scenarios and traffic conditions (like more complex Dumbbell topologies).

References

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993, doi: 10.1109/90.251892.

[2] K. Nichols and V. Jacobson, "Controlling queue delay: a modern AQM is just one piece of the solution to bufferbloat," *ACM Queue - Networks*, vol. 10, no. 5, pp. 1–15, 2012, doi: 10.1145/2208917.2209336.

[3] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the bufferbloat problem," in *Proc. of IEEE International Conference on High Performance Switching and Routing (HPSR)*, Taipei-Taiwan, July 2013, doi: 10.1109/HPSR.2013.6602305.

[4] J. Gomez, E. Kfoury, J. Crichigno, and S. Gautam, "A survey on tcp enhancements using p4-programmable devices," *Computer Networks*, vol. 212, no. 109030, pp. 1–35, 2022, doi: 10.1016/j.comnet.2022.109030.

[5] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communications Review*, vol. 24, no. 5, pp. 8–23, 1994, doi: 10.1145/205511.205512.

[6] K. D. Schepper and B. Briscoe, "The explicit congestion notification (ECN) protocol for low latency, low loss, and scalable throughput (L4S)," Requests for Comments, RFC 9331, November 2022.

[7] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 4, pp. 747–756, 2004, doi: 10.1109/JSAC.2004.825989.

[8] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proc. of ACM SIGCOMM*, New York, NY, USA, Aug. 2010, doi: 10.1145/1851182.1851192.

[9] S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, and G. Judd, "Data center TCP (DCTCP): TCP congestion control for data centers," Requests for Comments, RFC 8257, October 2017.

[10] Y. Huang and B. Hu, "Enhanced DCTCP to explicitly inform of packet loss," in *Proc. of IEEE International Conference on Communications (ICC)*, London, UK, June 2015, doi: 10.1109/ICC.2015.7249200.

[11] L. M. A. Sup, R. M. Moraes, and A. Bauchspiess, "Simultaneous TCP and NCS flows in a UPPAAL framework with a new AQM technique," in *Proc. of IEEE International Conference on Industrial Informatics (INDIN)*, Poitiers, France, July 2016, doi: 10.1109/INDIN.2016.7819138.

[12] ——, "Explicit non-congestion notification: A new AQM approach for TCP networks," in *Proc. of IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*, Valencia, Spain, June 2017, doi: 10.1109/IWCMC.2017.7986462.

[13] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[14] W. Zhang, M. S. Brannicky, and S. M. Philips, "Stability of networked control systems," *IEEE Control Syst. Mag.*, vol. 21, no. 1, pp. 84–99, 2001, doi: 10.1109/37.898794.

[15] Z. Taferra, "Process control over wireless sensor networks," Master's thesis, Kungliga Tekniska Hogskolan (KTH), Stockholm, Sweden, 2013.

[16] F. Janabi-Sharifi and I. Hassanzadeh, "Experimental analysis of mobile-robot teleoperation via shared impedance control," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 591 – 606, 2011, doi: 10.1109/TSMCB.2010.2073702.

[17] X. Ge, F. Yang, and Q.-L. Han, "Distributed networked control systems: A brief overview," *Information Sciences*, vol. 380, no. 1, pp. 117 – 131, 2017, doi: 10.1016/j.ins.2015.07.047.

[18] C. F. O. C. Neves, U. F. Moreno, and A. Boava, "Iot-based distributed networked control systems architecture," in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Turin, Italy, Sept. 2018, doi: 10.1109/ETFA.2018.8502500.

[19] W. Liu, G. Nair, Y. Li, D. Nesic, B. Vucetic, and H. V. Poor, "On the latency, rate, and reliability tradeoff in wireless networked control systems for IIoT," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 723–733, 2021, doi: 10.1109/JIOT.2020.3007070.

[20] H. Lin, S. Hongye, and S. Zhan, "Optimal estimation in UDP-Like networked control systems with intermittent inputs: Stability analysis and suboptimal filter design," *IEEE Trans. Autom. Control*, vol. 61, no. 7, pp. 1794 – 1809, 2016, doi: 10.1109/TAC.2015.2479195.