IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 30, 2020, accepted January 1, 2021, date of publication January 8, 2021, date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050174

Federated Intelligence for Active Queue Management in Inter-Domain Congestion

CESAR A. GOMEZ, (Member, IEEE), XIANBIN WANG, (Fellow, IEEE), AND ABDALLAH SHAMI, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Western University, London, ON N6A 3K7, Canada
Corresponding author: Cesar A. Gomez (cgomezsu@uwo.ca)

ABSTRACT Active Queue Management (AQM) has been considered as a paradigm for the complicated network management task of mitigating congestion by controlling buffer of network link queues. However, finding the right parameters for an AQM scheme is very challenging due to the dynamics of the IP networks. In addition, this problem becomes even more complex in inter-domain scenarios where several organizations interconnect each other with the limitation of not sharing raw and private data. As a result, existing AQM schemes have not been widely employed despite their advantages. Therefore, we present a solution that tackles the challenges of tuning the AQM parameters for inter-domain congestion control scenarios where the network management goes beyond an organization's domain. We then introduce the Federated Intelligence for AQM (FIAQM) architecture, which enhances the existing AQM schemes by leveraging the Federated Learning approach. The proposed FIAQM framework is capable of dynamically adjusting the AQM parameters in a multi-domain setting, which is hard to achieve with the conventional AQM solutions working alone. To this end, FIAQM uses an artificial neural network, trained in a federated manner, to predict beyond-own-domain congestion and an intelligent AQM parameter tuner. The evaluation results show that FIAQM can effectively improve the performance of the inter-domain connections by reducing the congestion on their links while preserving the network data private within each participating domain.

INDEX TERMS Active queue management (AQM), AQM tuning, congestion control, congestion prediction, federated learning, inter-domain communication, machine learning.

I. INTRODUCTION

Communication over the Internet relies on data packet transmission across a selected network path, while involved over the complex interconnected network elements. To achieve this, different network elements of the Internet, *e.g.* routers, usually first place the received data packets in queues, where they wait their turn to be transmitted over the next determined link. When there are too many queued packets awaiting transmission, the buffers of the network element's interface may overflow and the involved link is said to be congested. Therefore, determining the proper buffer size is deemed as a key component to evade packet losses along network paths when congestion appears. While a large buffer could reduce packet losses, excessive buffering could lead to increased latency, as packets have to wait longer in the queues. This phenomenon is known as bufferbloat and causes poor performance at bottleneck links of today's Internet [1]. This effect can be tackled by the network elements through Active Queue Management (AQM) methods, which are designed to control the flow of the arriving packets and avoid network congestion. To achieve so, AQM schemes determine whether there is incipient congestion on the involved link and choose either dropping specific packets or marking them with ''experienced congestion'' labels. The main advantage of dropping packets with AQM rather than with tail-drop queues, *i.e.* non-AQM buffers, is to eliminate the unnecessary global synchronization of flows when a queue overflows. In this way, an AQM scheme can decide to drop packets when the network experiences incipient congestion in a controlled fashion. As a result, packets experience shorter delays, as their flows are regulated by the AQM mechanism in use, and the throughput is improved. Despite the advantages of AQM, it is not widely adopted on the network elements of the Internet Service Providers (ISPs), since the AQM mechanisms have parameters that might be difficult to tune in dynamic environments. Also, network elements with more memory available in the market have created the misconception that the larger the buffers, the better.

Accordingly, we proposed an intelligent method for implementing AQM in our previous work [2] by exploiting the standardized Explicit Congestion Notification (ECN): a process

The associate editor coordinating the review of this manuscript and approving it for publication was Long Wang.

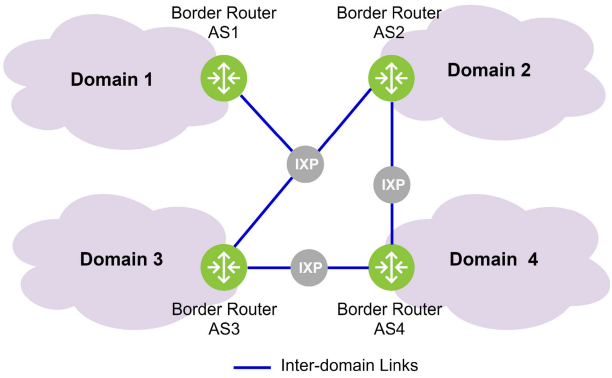

FIGURE 1. Example of an inter-domain communication scenario.

of making incipient congestion visible by exposing the presence of congestion on a path to network and transport layers through codepoints and flags in both IP and TCP headers. Our goal was to boost the alleviation performance that AQM techniques provide at bottlenecks by dynamically adjusting the AQM parameters and considering the specific network conditions. Therefore, we introduced a Machine Learning-based solution that comprises a Recurrent Neural Network to predict congestion and an AQM parameter tuner based on the Q-learning algorithm. The proposed scheme, however, was delimited to scenarios where only one router performs the intelligent AQM process (IAQM). For instance, a setting where an edge router predicts the congestion ahead, based on the ECN feedback that it receives from the core network, and then tunes its AQM parameters. As a result, the IAQM scheme dynamically reduces the Round-Trip Time (RTT) and increases the throughput of the connections being handled by the edge router.

In this work, we address the problem of congestion control by significantly enhancing existing AQM methods and taking into account the routers involved in inter-domain communications. This problem turns out to be even more challenging than a single-domain communication scenario, as each border router may not be able to receive ECN feedback in order to predict the congestion ahead. Additionally, a kind of cooperative mechanism is needed to achieve an effective Machine Learning solution where the privacy is paramount: an inter-domain link involves routers at several organizations or geographical regions, which means the possibility of having one or more domains not willing to share their data. That is why these domains are also known as Autonomous Systems (ASes), which consist of ISPs or Content Providers (CPs) communicating each other through an Internet Exchange Point (IXP), as depicted in Figure 1.

Managing congestion is an essential factor for an IXP and its connecting ASes. However, despite experiencing significant and persistent congestion at multiple peering links, both ASes and IXPs have no primary means of controlling congestion. That is, as the traffic sources and destinations are beyond its domain, a border router or an IXP cannot rely on the traditional congestion notification mechanisms such as ECN [3]. On the other hand, understanding the performance of the network elements requires measuring several parameters, such as utilization, loss rates, and variation in latency. Operators that control IXPs could measure such parameters for their links, although accurate assessment of these parameters may require cooperation of the operator at the other end of the links [4]. Moreover, the operators do not usually share this kind of information with their counterparts. For these reasons, we propose to apply the Federated Learning (FL) paradigm to intelligently address the inter-domain congestion problem.

FL is an approach where multiple entities collaborate in solving a Machine Learning problem, under the coordination of a central server or service provider [5]. To achieve the learning objective, each entity participates without exchanging private raw data, which are stored locally. The original emphasis of FL was on cross-device settings, *i.e.* mobile and edge devices applications [6], but FL has been applied to an increasing number of scenarios where a few and relatively reliable entities, such as the data centers of several organizations, collaborate to train a model. These kinds of scenarios are known as cross-silo settings. The main difference between the cross-device and cross-silo settings is that, in the former, a very large number of devices participate in the learning and their participation is likely to occur once in a task. On the other hand, in cross-silo settings only a small number of elements (typically, 2 to 100) contribute to the learning process by training a model on siloed data. In both cases, the data are generated locally and remain decentralized. At the same time, a central entity orchestrates the training process and receives the contributions of all entities. These characteristics make FL conceptually different from the decentralized and distributed learning approaches. A more detailed comparison of the FL settings versus the distributed and peer-to-peer learning can be found in [5]. It is also important to highlight that, different from many Machine Learning approaches, in FL the data are usually considered as unbalanced and not independently or identically distributed (non-i.i.d.) because each entity has different amount of local data to train on and these data rely on particular entities' behaviours [6]. Furthermore, depending on the distribution characteristics of the data, FL can be categorized as horizontal or vertical. In horizontal FL scenarios, the local datasets have the same feature space, but may have different sample ID space. In contrast, vertical FL refers to those cases where the datasets have the same sample ID space, but dissimilar feature space [7], [8].

Accordingly, in this work we propose an intelligent scheme for AQM where the inter-domain congestion is predicted based on the horizontal FL approach. That is why we introduce our solution as the Federated Intelligence for AQM (FIAQM), whose key contributions are summarized as follows:

- **A proof-of-concept study on non-static AQM**. We demonstrate how the idea of dynamically tuning AQM parameters may boost the adoption of AQM mechanisms to mitigate the Internet's bufferbloat effect.
- **An intelligent congestion control framework that is compatible with other solutions**. Our proposed

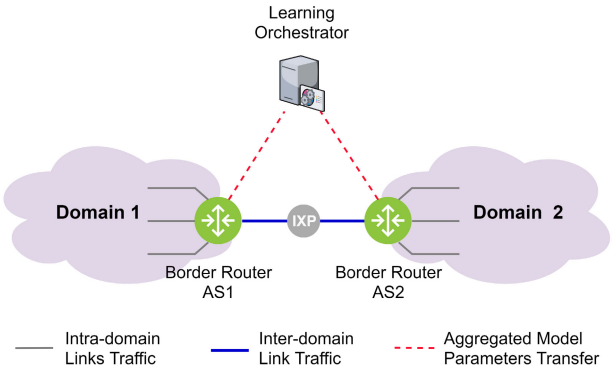

FIGURE 2. Typical scenario for the proposed FIAQM scheme.

FIAQM leverages the benefits of using existing AQM mechanisms to control congestion over inter-domain links, which are not managed by a single party.

- **A multi-domain learning approach in which local network data remains private**. As in inter-domain scenarios privacy is a major concern, FIAQM allows the cooperation of two or more ASes to achieve common goals in terms of congestion by avoiding the sharing of raw data.
- **A practical application of Deep Learning and FL in networking**. We propose an adaptation of the FL algorithm that, along with a tailored neural network, effectively learns congestion levels of the link queues involved in cross-domain connections.
- **An open source environment for real-time evaluation**. Finally, we evaluate the performance and feasibility of the FIAQM scheme in a setting that emulates a realistic inter-domain network communication, whose code is publicly available for further research and development.

Overall, a typical scenario for FIAQM comprises two border routers, which belong to different ASes, and an IXP. Each border router has intra-domain link buffers corresponding to the interfaces that connect them to other network elements within their own domains, as depicted in Figure 2. Both border routers exchange the aggregated parameters of the model to be trained with a central server, known as the Learning Orchestrator in our solution. We propose to place the Learning Orchestrator at the IXP premises, since it is supposed to be a neutral player. In this way, FIAQM applies FL to predict the IXP congestion based on the buffer statistics of the intra-domain links of the border routers involved (denominated as the Local Learners). The predicted IXP congestion is then used for the AQM parameter tuning of the inter-domain link buffers, similar to the tuning process introduced in [2].

The remainder of this paper is, then, organized as follows. We review the related work on inter-domain congestion in Section II. In Section III, we provide further details about the FIAQM architecture, whose evaluation performance results are discussed in Section V. Conversely, we explain the details of our experimentation design in Section IV and, finally, the conclusions and future work are presented in Section VI.

II. RELATED WORK

The inter-domain congestion control problem has been addressed from different perspectives. One common approach is to tackle the routing bottlenecks. These bottlenecks are inevitably caused by the Border Gateway Protocol (BGP), since the border routers tend to forward packets along the path with minimal routing cost. As a result, routing bottlenecks concentrate on a few links and happen to be asymmetrical, *i.e.* the inbound congestion does not correspond to the outbound one on the same link [9]. Therefore, the solutions for routing bottlenecks proposed in the literature mainly rely on dynamic load balancing, which can operate either on inter-domain or intra-domain links.

To this end, authors in [10] present a system to improve the ISPs network throughput by jointly optimizing intra-domain routes and inter-domain routes. Their solution provides an ISP and its neighbor CPs with a network abstraction on a virtual switch that allows to program requirements in a collaborative way. Conversely, an architecture for an efficient inbound traffic control based on the Software Defined Networking (SDN) paradigm is proposed in [11]. This architecture exploits the features of the OpenFlow protocol for network traffic engineering tasks in inter-domain routing. Similarly, Chiesa *et al.* describe the benefits of using the SDN approach for traffic engineering at IXPs. The authors explain how SDN enables such a network programmability that permits the members of an IXP to optimize their traffic load balancing and overcome the limitations of BGP [3]. Considering the privacy preservation in SDN-enabled scenarios for inter-domain traffic, authors in [12] propose a solution to avoid incorrect forwarding behaviours without exposing private routing information among domains. Likewise, [13] presents a mechanism for a dynamic end-to-end Quality of Service (QoS) coordination in multi-domain scenarios. This mechanism processes information in a distributed manner at the domain level and optimizes the routing by adaptively learning the results of past QoS requests.

It is important to highlight that a routing bottleneck is essentially different from a bandwidth bottleneck. The latter refers to the link with the smallest available bandwidth on a route, while the former is related to the number of routes carried by a link regardless the provisioned link capacity [14]. Even though they do not necessarily imply each other, routing bottlenecks can derive in bandwidth bottlenecks, which are the ones that ultimately cause the congestion that affects the networks' communication performance. For this reason, we address the inter-domain congestion problem with a focus on the bandwidth bottlenecks. This does not mean that our method cannot be used along with some of the described solutions for routing bottlenecks. Nevertheless, how to combine both approaches is beyond the scope of this paper.

With regards to our learning setting based on buffer statistics, there is some literature about the use of queue measurements for congestion control improvement. For instance, authors in [15] propose a fine-grained queue measurement solution in the data plane for immediate control actions,

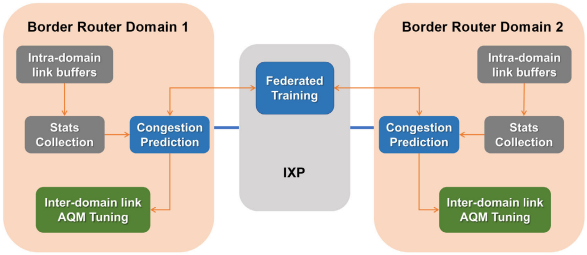

FIGURE 3. The FIAQM architecture for inter-domain congestion control. Main modules are replicated within each border router.

which can support the deployment of new and more sophisticated AQM schemes. Using In-band Network Telemetry (INT) and traffic snapshots (fixed-sized time windows of traffic on a queue), their solution can determine the flows that consume large portions of a queue. Similarly, Li *et al.* propose a High Precision Congestion Control mechanism, which leverages the INT metadata reported by the routers along the path [16]. The metadata includes egress port metrics such as timestamp, queue length, transmitted bytes, and link bandwidth capacity to avoid congestion in high-speed networks. Although we acknowledge the value of the INT framework and its metadata, we consider not using INT in this work because it aims to monitor the performance of a core network within a single domain. However, we believe that the application of the INT metrics for the solution of an inter-domain problem, like the one presented in this paper, could be a promising direction for a future work.

III. ARCHITECTURE OF FIAQM

In this section, we describe our solution in detail. Primarily, FIAQM consists of two principal modules: a congestion predictor and an AQM parameter tuner, like the IAQM solution presented in [2]. In FIAQM, however, the congestion ahead is predicted by means of the FL approach. This prediction is then utilized for the AQM parameter tuning of the inter-domain link buffers in both directions. Figure 3 depicts the overall architecture of FIAQM and the following subsections explain each component, respectively.

A. FEDERATED CONGESTION PREDICTOR

The first of the main components of the FIAQM architecture is a congestion predictor based on a Long Short-Term Memory (LSTM). An LSTM is a type of Recurrent Neural Network and deemed as an effective tool for time-series forecast. Its inputs include both the current sample and the previous observed sample, such that output at time step t − 1 affects the output at time step t. Each neuron of the LSTM has a feedback loop that returns the current output as an input for the next step [17]. For these reasons, FIAQM employs an LSTM to predict congestion in a federated manner by considering drop rates at each queue per time interval as inputs. Hence, the drop rate x in a time interval i is calculated as follows:

$$x_{t_i} = \frac{D_{t_i}}{P_{t_i}} \tag{1}$$

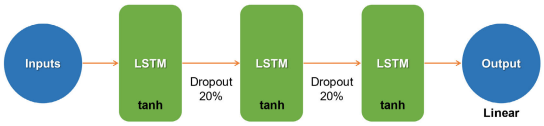

FIGURE 4. LSTM network structure for the FIAQM's congestion predictor.

where D is the number or dropped packets and P the total packets arriving at the queue within each time interval. Additionally, we rearrange the vector of drop rates as an input matrix $\mathbf{X}$ corresponding to ten time steps and an output vector $\mathbf{y}$ of one time step, such that:

$$\mathbf{X} = \begin{bmatrix} x_{t_0} & x_{t_1} & \cdots & x_{t_9} \\ x_{t_1} & x_{t_2} & \cdots & x_{t_{10}} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t_{N-10}} & x_{t_{N-9}} & \cdots & x_{t_{N-1}} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} x_{t_{10}} \\ x_{t_{11}} \\ \vdots \\ x_{t_N} \end{bmatrix} \tag{2}$$

where N is the total number of local samples. The rationale behind rearranging the samples in ten time steps is to improve the performance of the predictive model by having additional context. In this way, the estimation of drop rates contemplates more prior observations. Note that this data rearrangement is performed with the available samples of each queue participating in the FL training.

The structure of the LSTM is similar to the one described in [2] and encompasses $L = 3$ hidden layers with 30 neurons each. The output layer employs a linear activation function while the hyperbolic tangent (tanh) is used as the non-linear activation function at the hidden layers, since it provides a three-state decision making (negative/neutral/positive) on what information to add or remove to/from the hidden cells [18]. Also, a dropout regularization of 20% is included at the output of each hidden layer, except the last one, in order to avoid model's overfitting, as shown in Figure 4. More specifically, each hidden layer $l \in [0, L)$ of the LSTM network computes the following function for each element in the input sequence [19]:

$$h_t^{(l)} = \tanh\left(W_{\text{ih}}^{(l)} x_t^{(l)} + b_{\text{ih}}^{(l)} + W_{\text{hh}}^{(l)} h_{t-1}^{(l)} + b_{\text{hh}}^{(l)}\right) \tag{3}$$

where $h_t^{(l)}$ is the hidden state at time t, $W_{ih}^{(l)}$ and $b_{ih}^{(l)}$ represent the weight and bias of the block input at layer l, and $W_{hh}^{(l)}$ and $b_{hh}^{(l)}$ are the weight and bias values of the hidden cells. Correspondingly, $h_{t-1}^{(l)}$ is the hidden state of the layer at time $t-1$ and the input of the l-th layer, $x_t^{(l)}$, is the hidden state of the previous layer $h_t^{(l-1)}$ multiplied by the dropout of the previous layer, $\delta_t^{(l-1)} = 0.2$. Conversely, each output in the sequence is computed at the output layer through a linear function, as follows:

$$y_t = W_o\left(h_t^{(L-1)} + b_o\right) \tag{4}$$

where W_o and b_o are the weights and bias of the output layer, respectively, and $h_t^{(L-1)}$ is the state of the last hidden layer.

The Learning Orchestrator performs the global training of the LSTM model, which is used for the congestion prediction of the inter-domain link in each direction. In this way, the proposed LSTM-aided Federated Congestion Predictor (FCP)

functions as follows: each router has a fixed local dataset that differs from the other router's dataset, since they might have different number of intra-domain links with dissimilar levels of queue drop rates. At the beginning of each learning round, the Learning Orchestrator sends the current global model state to the routers, also known as the Local Learners in our solution. Next, each router performs a local computation based on the global state and its local dataset and, afterwards, sends an update to the orchestrator. Finally, the Learning Orchestrator applies the updates received from the Local Learners to its global state and the learning process repeats.

Due to the nature of our problem, we employ a cross-silo FL since individual routers or group of routers might belong to different proprietary networks. Our learning model is intended to be trained across these silos without exchanging raw data, which may represent ASes private information or a single organization's data that cannot be centralized between different geographical regions. Additionally, we consider the routers data as unbalanced and non-i.i.d., as well as the synchronous model updates that proceed in rounds of communication, as presented in [6]. The canonical FL problem involves learning a single, global statistical model from data stored on remote entities. For our problem, we aim to learn this model under the constraint that border routers data are stored and processed locally, with only intermediate updates being periodically communicated to the Learning Orchestrator. In particular, the goal is to minimize the objective function for the global learning [8], as follows:

$$\min_{w} F\ (w) := \sum_{k=1}^{M} p_k F_k\ (w) \tag{5}$$

where w represents the model parameters, *i.e.* the weight and bias values of the hidden and output layers of the LSTM network. In our scenario, M is total number of queues involved in the congestion prediction process and p_k is the relative impact of each queue. On the other hand, F_k is the local objective function for the learning on the k queue, as follows:

$$F_k = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w; x_{j_k}, y_{j_k}) \tag{6}$$

where n_k is the number of samples available locally. To solve this federated optimization problem, we adapt the Federated Averaging (FedAvg) algorithm presented in [6]. Accordingly, the algorithm combines a local stochastic gradient descent computed with the data of each queue at each border router and a model averaging performed by the Learning Orchestrator. The adaptation of the FedAvg algorithm for our proposed FCP is detailed in Algorithm 1, where η is the learning rate, which is assumed to be the same for all the Local Learners. It is important to highlight that d_k contains the number of data samples with non-zero values. The rationale behind this idea is that queues with higher drop rates affect the parameter averaging with higher values of relative impact p_k. In this way, the federated LSTM model learns more from those queues with non-zero drop rates for the congestion prediction.

Algorithm 1 Federated Congestion Predictor (FCP)

$q \leftarrow$ set of queues with non-zero drop rate data
for each round $r = 1, 2, 3, \ldots, \Gamma$ **do**
  $u \leftarrow$ random subset, $u \in q$
  for each queue $k = 1, 2, \ldots, M \in u$ **in parallel do**
    get w from Learning Orchestrator
    $w_k \leftarrow w$
    $d_k \leftarrow$ count $n_k \forall x_{j_k} \neq 0$
    for each local training iteration $z = 1, 2, 3, \ldots, Z$ **do**
      $w_k \leftarrow w_k - \eta \nabla F_k$
    return w_k and d_k to Learning Orchestrator
  $p_k \leftarrow d_k \Big/ \sum_{k=1}^{M} d_k, \forall k$
  $w_{t+1} \leftarrow \sum_{k=1}^{M} p_k w_k$

On the contrary, the queues with a few or zero samples of congestion data make a little or no contribution to the learning process.

B. AQM PARAMETER TUNER

In general, the parameters of the AQM algorithms are set to values that yield a reasonable performance for the typical network conditions. However, AQM mechanisms are expected to allow parameters adjustment depending on the specific characteristics of a network and their interactions with other network tasks over time [20]. Consequently, we embrace the idea of adjusting the AQM parameters according to the network's changing circumstances, so that the performance is dynamically improved, as well. Nevertheless, the achievement of this goal can end up in a very complex job and that is the main reason why network managers prefer not to use AQM at all. Another point to consider is the right metric to evaluate the effectiveness of a resource allocation/configuration in a network. The key metrics to be considered for queue management are, usually, throughput and delay. Accordingly, the objective is to minimize the delay and maximize the throughput. It turns out that, trying to increase the throughput by allowing as many packets into the links as possible, results in a rising length of the queues and, therefore, longer delays. As an alternative, a separate metric that combines throughput and delay can be taken into account. That is why the ratio of throughput, T_{put}, to measured RTT, m_{RTT}, has been proposed by network designers as a metric to evaluate the effectiveness of a resource configuration, such as the AQM parameters. This throughput-to-delay ratio is also known as the power of the connection, $P_c = T_{put} \big/ m_{\mathrm{RTT}}$, and, even though this metric has some limitations, it is widely accepted for evaluating the network resource configuration effectiveness [21], especially the queue management for congestion control [22]. Maximizing P_c is, however, a non-trivial task considering the network dynamics.

For the reasons explained above, we model the AQM parameter-tuning problem as a Markov Decision Process (MDP). In the FIAQM scheme, the decision process is based on the inferred congestion ahead, *i.e.* the output of the FCP

Algorithm 2 AQM Tuner

$S \leftarrow$ set of discretized values of predicted congestion
$A \leftarrow$ set of AQM target parameter values
$Q(S, A) \leftarrow$ Q-table initialization
$\varepsilon \leftarrow$ exploration/exploitation rate, $\varepsilon \in [0, 1]$
$s \leftarrow$ get state from FCP, $s \in S$
for each period $T = 1, 2, 3, \ldots$ **do**
  if random number $< \varepsilon$
    then $a \leftarrow$ select a random action, $a \in A$
    else $a \leftarrow \text{argmax}_a Q(s, A)$
  change parameters according to a
  $m_{\text{RTT}} \leftarrow$ measure delay
  $T_{put} \leftarrow$ measure throughput
  $R \leftarrow P_c$
  $\overline{s} \leftarrow$ get state from FCP, $\overline{s} \in S$
  update $Q(S, A)$
  $s \leftarrow \overline{s}$

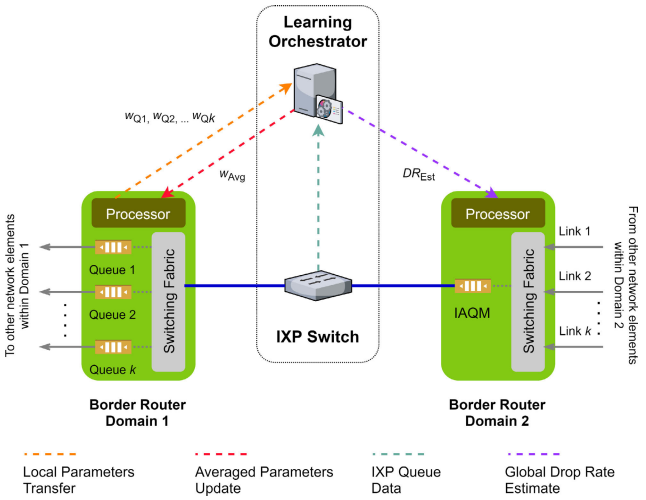

FIGURE 5. Implementation of the FIAQM for experimentation.

described in Section III-A. In this way, we define the states S as a set of discrete levels of congestion that the inter-domain link will be likely to experience, the set of actions A comprises specific values of the target parameter of the AQM algorithm in use, and the reward R depends on P_c. In our scenario, each border router acts as the agent that makes the decisions. This way, our method can adjust the target parameter so that more packets are dropped proactively and in a controlled manner at the sending border router, as they will be likely dropped ahead in the other domain. In other words, the AQM parameter tuner is modelled as an MDP with the objective of finding an optimal behavior that maximizes P_c. To do so, we utilize the Q-learning algorithm [23], which defines the function $Q(S, A)$, representing the quality of a certain action in a given state, and that is defined by:

$$Q(S, A) := Q(s, a) + \alpha [R + \gamma \max_{\overline{a}} Q(\overline{s}, \overline{a}) - Q(s, a)] \tag{7}$$

where $\alpha \in [0, 1]$ is the learning rate and the discount factor $\gamma \in [0, 1]$ describes the preference of the agent for current rewards over future rewards. This equation characterizes the maximum future reward of present state s and action a in terms of immediate reward and maximum future reward for the next state $\overline{s}$ and action $\overline{a}$. In this manner, the Q-learning algorithm iteratively approximates the function $Q(S, A)$, as shown in Algorithm 2. More specifically, our AQM parameter tuner observes current and next states as levels of congestion, *i.e.* the predicted drop rates of the link buffer at the router in the destination domain. Additionally, both current and next states are discretized to delimit the complexity of the environment. Finally, the actions are a set of predefined values for the target parameter of the specific AQM in use. As the agent does not know what action to take at the beginning, there is an initial stage of exploration, which depends on the parameter ε. The value of this parameter determines if the Q-learning algorithm prefers to explore random actions rather than exploit the historical data to take an action.

IV. EXPERIMENTATION DESIGN

In order to evaluate our FIAQM scheme, we set up a network emulation environment on Mininet to run experiments and obtain more realistic results. We chose Mininet as the tool to validate our prototype since it allows a flexible SDN environment with high degree of confidence for real-time tests [24]. Moreover, Mininet eases the sharing of our solution, which could be deployed into a real production network using our code and test scripts, publicly available at [25]. Accordingly, our emulation network consists of two border routers and 20 hosts connected to each one, forming a dumbbell topology. In this way, there are 20 pairs of hosts generating traffic from one domain to the other (hosts of each pair are in different domains). Figure 5 depicts the implementation of our experimentation setting. Note that for simplicity, only one direction of the learning process for the congestion prediction is depicted, that is, considering traffic from Domain 2 to Domain 1. Therefore, the IAQM tuning happens at the egress buffer of the Border Router Domain 2 in this setting.

With respect to the FCP implementation, our environment involves three Mininet hosts acting as the Learning Orchestrator and two Local Learners, the latter being represented by the processor block at each border router. Additionally, PyTorch is employed on these hosts for the execution of the learning process as described in Algorithm 1. We chose PyTorch as the framework for the implementation of our FCP algorithm because it provides a high level of control and flexibility, which we weigh as a key feature for our network emulation. Moreover, PyTorch's usability and developer-centric design facilitates the implementation of new Deep Learning architectures, using the familiar concepts developed for general purpose programming languages such as Python [26]. This is particularly relevant for the application of the FL approach, since it needs to be deployed in a distributed manner when implementing real-world setups. We see this fact as

a significant advantage of PyTorch over other Deep Learning frameworks like TensorFlow. For example, we employed TensorFlow Federated (TFF) for the fulfilment of the FL version of the LSTM model proposed in [2]. We were able to confirm that TFF only enables the simulation of FL models with decentralized datasets, as stated in [27], but not an actual distributed deployment. For these reasons, we decided to utilize PyTorch as the Deep Learning framework for the validation of our FIAQM scheme.

In relation to the traffic generation between the host pairs for the queue metrics, we use the NetPerf tool [28], which allows to stress the network under a combination of several types of IP traffic. Furthermore, we perform tests according to the Real-time Response Under Load (RRUL) Specification to emulate a more core-network-like IP traffic. In fact, RRUL-based tests reliably saturate the measured link and, therefore, exposes any presence of the bufferbloat effect. To this end, the RRUL specification contemplates simultaneous bidirectional TCP and UDP streams, VoIP-like streams, multiple up/down TCP streams to shorten the ramp-up-to-saturation period, running traffic long enough to defeat bursty bandwidth optimizations, and test server(s) within 80 ms of testing client(s) [29]. Next, the emulator collects the buffer statistics in intervals of 100 ms using the Linux Traffic Control (TC), since this utility lets monitor the queue events generated by the kernel [30]. The value of the time interval corresponds to the typical assumption for a single RTT interval in IP networks.

Subsequently, we use set both the intra-domain and the inter-domain link buffers to a relatively small hard limit of 1000 packets. This assumption is based on the fact that small buffer sizes in backbone routers are sufficient for many networks and recommended for overall scalability [31], [32]. Additionally, all the intra-domain link buffers are configured with AQM. More specifically, we consider the Flow Queue - Controlling Queue Delay (FQ-CoDel) whose target parameter to configure is the acceptable minimum standing/persistent queue delay [33]. As this parameter decreases, more packets are dropped in a controlled manner, since they are supposed to stay for shorter times in the queue. Consequently, there are less packets in the queue and the link delay decreases. On the other hand, when the FQ-CoDel target parameter is high, the scheme does not drop packets and there is a higher delay due to longer queues. Also, packets start to be dropped uncontrollably as the queue overflows and, therefore, the throughput is deteriorated.

As a preliminary experiment, we show that the drop rate data of the queues at the Border Router Domain 1 describe dissimilar patterns, as depicted in Figure 6a. Therefore, the traffic data generated by the RRUL test and gathered with the TC utility exhibits the kind of non-i.i.d. behaviour necessary for the FL model of the FCP. For the sake of clarity, we depict the drop rate data corresponding to ten queues only, but similar graphs are obtained when more queues are considered. On the other hand, to show the influence of tuning FQ-CoDel, we set up a simple test that consists of modifying

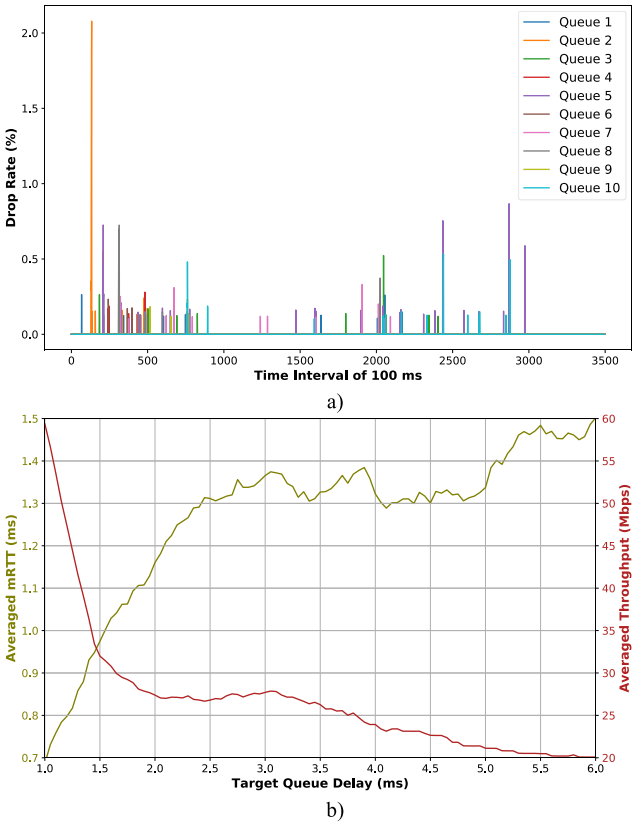

FIGURE 6. Preliminary tests for the Experimentation Design. a) Queues data at Border Router Domain 1. b) Effects of tuning FQ-CoDel target parameter on m_{RTT} and T_{put}.

its target and interval parameters at the egress buffer of the Border Router Domain 2 while data are constantly transferred between two hosts, each one in a different domain. The interval parameter ensures that the measured minimum delay does not become too old and, typically, the target delay is 5% of this interval [33]. Therefore, we set FQ-CoDel with target values from 1 ms to 6 ms and intervals from 20 ms to 120 ms, respectively. As can be seen in Figure 6b, although an AQM scheme such as FQ-CoDel is meant to operate unchangeably, there is a noticeable effect when its target parameter varies: both m_{RTT} and T_{put} are affected by the target delay configuration. This is consistent with our solution formulation explained in Section III-B.

For the parameters exchange between the Learning Orchestrator and the Local Learners, we use the Secure File Transfer Protocol (SFTP), which runs over the Secure Shell (SSH) protocol, to avoid sending the parameters in the clear. SFTP protects the data integrity through cryptographic hash functions and provides authentication for both the server and the client [34]. In this way, we also consider security concerns in a real inter-domain scenario by adding encryption functionality to the communication between the parties involved in the FCP. Additionally, we assume that the pair of private and public keys have been shared prior to the execution of the Algorithm 1 and that a different port from the default SSH port, *i.e.* port 22, is agreed for the transfer.

TABLE 1. Model parameters to be transferred for the FCP.

Parameter	Description	Tensor Dimension
$W_{ih}^{(0)}$	Weights of block input, hidden layer 0	120×1
$b_{ih}^{(0)}$	Bias of block input, hidden layer 0	120
$W_{hh}^{(0)}$	Weights of hidden cells, hidden layer 0	120×30
$b_{hh}^{(0)}$	Bias of hidden cells, hidden layer 0	120
$W_{ih}^{(1)}$	Weights of block input, hidden layer 1	120×30
$b_{ih}^{(1)}$	Bias of block input, hidden layer 1	120
$W_{hh}^{(1)}$	Weights of hidden cells, hidden layer 1	120×30
$b_{hh}^{(1)}$	Bias of hidden cells, hidden layer 1	120
$W_{ih}^{(2)}$	Weights of block input, hidden layer 2	120×30
$b_{ih}^{(2)}$	Bias of block input, hidden layer 2	120
$W_{hh}^{(2)}$	Weights of hidden cells, hidden layer 2	120×30
$b_{hh}^{(2)}$	Bias of hidden cells, hidden layer 2	120
W_o	Weights of output layer	1×30
b_o	Bias of output layer	1

The use of SFTP is sufficient for the needs of our experimentation, since w_k are transferred as a Python dictionary with the parameters of the FCP model. The size of this dictionary is 77.8 kB and it contains the PyTorch tensors with the weight and bias values, whose dimensions are specified in Table 1. On the other hand, d_k is a Python list with u elements. For private and secure transfer of high-dimensional parameter vectors in a FL setting, which is not the case of this work, we point the reader to other research papers such as [35], [36]. It is also important to highlight that, although the FIAQM is tested in a distributed setting, the FCP algorithm is synchronously executed between the Learning Orchestrator and the Local Learners. This means that our experimentation design considers the coordination of the learning algorithm execution along with the transfer of the parameter files.

V. FIAQM PERFORMANCE EVALUATION

To evaluate our FIAQM scheme, we first demonstrate how the FCP algorithm predicts congestion accurately as a standalone entity. Next, we illustrate how the FCP integrates with the AQM parameter tuner to attain the objective of reducing congestion and improving the performance of an inter-domain connection.

A. FCP ALGORITHM PREDICTIONS ACCURACY

The experiments of this subsection are conducted in an offline setting with data previously gathered during the preliminary tests described in Section IV. Hence, we count on 21 datasets: one from the IXP queue, corresponding to the link between the IXP switch and Border Router Domain 1, and 20 from the queues of the intra-domain links of the aforementioned router. Subsequently, we train the FPC with $\eta = 0.001$ and $u = 2$, which means that two queues of the router are randomly selected to average the model parameters in each round. Also, the number of training rounds are set to $\Gamma = 10$ and the local iterations to $Z = 1000$. To make predictions, we utilize the data from the IXP queue as the set of test samples. Figure 7 shows how the predicted congestion of the FPC model, trained with the queue data of Border Router Domain 1, resembles the actual congestion of the IXP's queue. Note that, rather than predicting the exact value of drop rate in a particular time interval, we are more interested in capturing the tendency of that value. Hence, the predictions are accurate enough for our goal. In terms of the loss metric, we chose the Mean Square Error (MSE), which yields a value of 0.002 over the test subset.

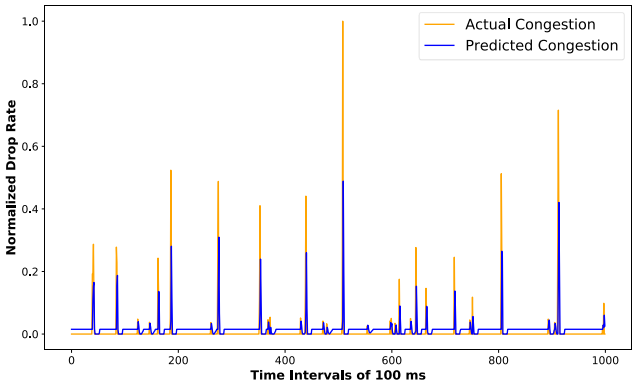

FIGURE 7. Actual congestion of the IXP queue and predicted congestion by the FCP.

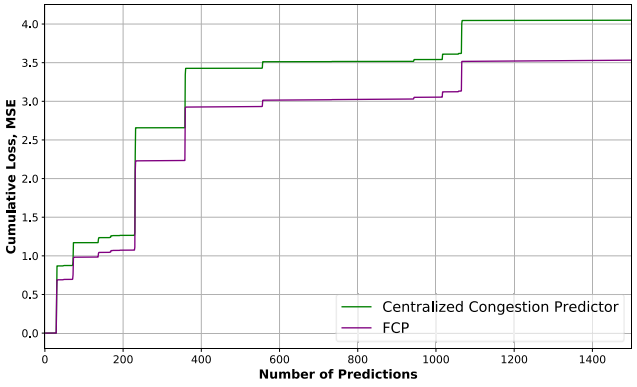

FIGURE 8. Evaluation loss comparison between a centralized congestion predictor and the FCP algorithm.

On the other hand, we compare the loss obtained when the congestion predictor is trained in a federated fashion and in a centralized manner. As this comparison requires more exhaustive tests, we change the emulation parameters Γ and Z to 50 and 2000, respectively. We also run a separate centralized model that is trained with data from the IXP's queue. As can be seen in Figure 8, FCP gets lower cumulative loss than the LSTM model of the centralized congestion predictor. What is interesting about this result is that both federated and centralized models are evaluated by making predictions over a test subset from the IXP's queue. That is, the FCP outperforms the centralized congestion predictor, even though the test data is a subset of the dataset used for the centralized model training. This result is consistent with those presented in [6].

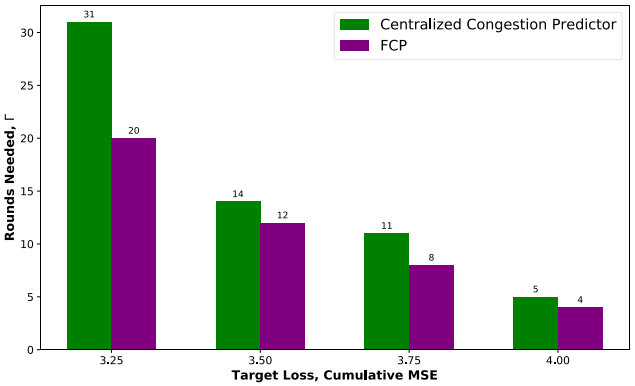

FIGURE 9. Number of training rounds needed to reach the target loss by a centralized congestion predictor and the FCP algorithm.

In contrast, the time complexity of the FedAvg algorithm can be expressed in terms of the total training rounds, Γ, local epochs, E, and the number of local samples, n_k, as $\mathcal{O}(\Gamma \times E \times max_k(n_k))$. This means that the time taken for the FL training depends on the slowest participant in each round, also known as stragglers, because of the number of local updates that those participants need to execute [37]. In our proposed FCP algorithm, we reduce this complexity by considering that all the participants have the same number of local samples, that is $N = n_k$. It is important to highlight that this is a realistic consideration, since the traffic in core networks is very high and the routers' queues are likely to expose congestion frequently. In this way, the local epochs and local batches of the FedAvg algorithm are converted into Z local training iterations in FCP (step 8, Algorithm 1), which correspond to n_k. In other words, different from the FedAvg algorithm, in the FCP algorithm every participant happens to have the same number of local updates (or local training iterations, Z), which yields a time complexity of $\mathcal{O}(\Gamma \times N)$. Nevertheless, we show that $\Gamma \ll N$ is generally the case for our problem scenario.

To this end, we set various target loss values in order to determine how many rounds of training the FCP needs to reach those targets. Thus, four benchmarks are defined based on the cumulative loss over 2000 predictions as targets. In this experiment, the number of local iterations is $Z = 2000$, as well. Similar to the evaluation test explained previously, the predictions are made considering a test subset from the IXP's queue. We then compare the number of training rounds needed by the FPC algorithm against an LSTM trained in a centralized host, Figure 9. As can be seen, the FCP algorithm requires less rounds during the training process to attain the desired loss on the test data. This result shows that, although there is an overhead in the congestion predictor training of the FIAQM, the proposed algorithm compensates this overhead by enabling a lighter training process in terms of the rounds needed. Moreover, this outcome evidences that the complexity of the FCP algorithm is heavily influenced by the number of samples used for the training process, N, rather than Γ.

TABLE 2. Emulation parameters for the evaluation of the FIAQM scheme in realtime.

Network Emulation Parameter	Value
Border Router Domain 1 - IXP link bandwidth	1 Gbps
Border Router Domain 2 - IXP link bandwidth	1 Gbps
Intra-domain links bandwidth	Random integer, [250, 500) Mbps
Border Router Domain 1 - IXP link delay	2 ms
Border Router Domain 2 - IXP link delay	2 ms
Intra-domain links delay	Random integer, [2, 10) ms
Number of hosts per domain	20
Buffers hard limit (all queues)	1000 packets
AQM mechanism (all queues)	FQ-CoDel
Intra-domain links AQM target (static)	2 ms
Intra-domain links AQM interval (static)	40 ms
Period of AQM parameters tuning, T	2 s
Emulation time	600 s

B. REAL-TIME AQM TUNING WITH FIAQM

In this subsection, we elaborate about the experiments that we conducted in real time to show the performance of our proposed method as a whole, that is, the FIAQM's main components working together. To this end, we carry out several experiments in the emulation setting described in Section IV. The network emulation parameters for this evaluation are summarized in Table 2. We assess the MDP for the AQM tuning problem by considering 100 levels of congestion as current or next states. To determine their levels, we keep the maximum observed and predicted values as reference for the discretization. We also delimit the actions to 100 values, which are the target delay of FQ-CoDel. In this way, the possible actions to take are a set of values from 1.1 ms to 11 ms in steps of 100 μs. As we explained in Section III-B, we modify two parameters at the same time: the target delay and the interval. Thus, the experiments are more consistent as these two parameters are tightly related. For this assessment, the Border Router Domain 2 performs the IAQM while the Border Router Domain 1 is configured with the default target and the interval parameters in the Linux kernel: 5 ms and 100 ms, respectively.

In terms of the FIAQM execution, the FCP runs in the background while the AQM tuner performs its job in an online manner. To achieve so, the Q-values are updated iteratively every 2 seconds based on both the predicted level of congestion ahead and P_c, which is calculated from the T_{put} and m_{RTT} values that two monitoring hosts, one in each domain, measure with active probes. Once the reward based on P_c is known, the algorithm updates the Q-values by applying (7).

On the other hand, the FCP utilizes pre-trained model parameters while the first training round is completed. Thus, the FCP predictions during this time are accurate enough for the AQM tuner. Additionally, 100 samples of the IXP's queue data are considered for the predictions, which means the

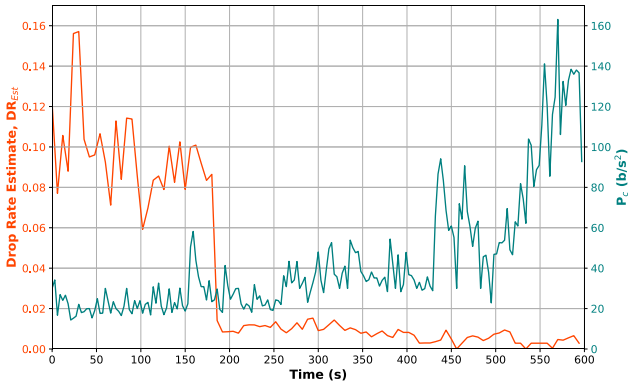

FIGURE 10. Improvement over time provided by FIAQM in terms of congestion reduction and P_c growth. AQM tuning starts at 150 s.

historical levels of congestion in the past 10 seconds. Those predictions are transferred from the Learning Orchestrator to the Local Learner asynchronously, in form of a NumPy array of dimension 100×1 and 928 B in size. This array corresponds to the global drop rate estimate of the other domain, $\mathrm{DR}_{\mathrm{Est}}$, as depicted in Figure 5. In this way, the AQM tuner takes into account the most recent available values of $\mathrm{DR}_{\mathrm{Est}}$, even if the FCP is still processing a new training round.

Accordingly, Figure 10 shows the results of the real-time network emulation in 600 s. Note that, for the comparison sake, we set the FIAQM's tuner to start operating at 150 s of the emulation. Then, the AQM parameters of Border Router Domain 2 are fixed to the default values during the first 150 s and, from this time on, the IAQM tunes these parameters according to Algorithm 2. As can be seen, the drop rate ahead at the Border Router Domain 1, which corresponds to the $\mathrm{DR}_{\mathrm{Est}}$ values forecasted by the FCP, decreases significantly once the FIAQM starts the tuning process. Conversely, P_c tends to get higher values as the AQM tuner improves over time. As a result, the tuning process populates the Q-table with the values of P_c in the respective (s, a) coordinates at every iteration of Algorithm 2. We highlight that, thanks to the way that we design the AQM tuner, the resulting Q-table is a light NumPy array of 100×100 elements and 39.1 kB in size.

Finally, Table 3 summarizes the hyperparameters of both modules of the FIAQM scheme utilized for its evaluation in the real-time emulation. It is also important to point out that, although we designed our experimentation setting to make it as realistic as possible, Mininet has some limitations regarding the links bandwidth of the emulated network elements. In actual backbone networks, link data rates are of the order of tens or hundreds of Gbps. However, Mininet emulations are constrained by the data rate of the computer's network interface where Mininet is running and the number of emulated network interfaces. This means that, in order to achieve results that resemble real-world networks, this data rate capacity must be considered for all the links in the emulation environment. Nevertheless, the emulation parameters can be easily scaled when running our setting on other computers, actual SDNs, or even Linux-based bare metal

TABLE 3. Hyperparameters of the FIAQM's learning modules.

Module	Hyperparameter	Value
FCP	LSTM hidden layers, L	3
	Cells per LSTM hidden layer	30
	LSTM dropout regularization, δ	0.2
	Learning rate, η	0.001
	Subsets of non-zero queues, u	2
	Local training iterations, Z	1000
	Training rounds (maximum), Γ	10
AQM Tuner	Learning rate, α	0.5
	Discount factor, γ	0.8
	Exploration/exploitation rate, ε	0.5

routers [38]. Last but not least, we would like to remind the reader that the code of the experiments described in this subsection is publicly available at [25]. We intent to make our contribution accessible to researchers and developers who are actively working on congestion-related problems of the Internet. Please cite this paper if you use any posted script for your works.

VI. CONCLUSION

In this work, we showed how the appropriate tuning of AQM parameters can improve the RTT and throughput of inter-domain connections. We presented our FIAQM solution, which leverages the characteristics of existing AQM schemes in such scenarios where several parties are involved in a communication process and privacy is a major consideration. The main components of the FIAQM architecture effectively applies the fundamentals of the FL approach to attain congestion control between ASes managed by different organizations and whose network data cannot be shared. We described in detail the main components of FIAQM: an LSTM trained in a federated fashion to predict the beyond-own-domain congestion and an AQM parameter tuner based on the Q-learning algorithm. We also explained how these two components integrate to make possible for a border router to dynamically tune the AQM scheme of its link queue that connects to the border router in another domain.

On the other hand, we evaluated the performance of FIAQM in a realistic environment by means of network emulations. Despite the limitations of the software tool used to this end, our solution can be easily adapted to other environments. Additionally, the performance of future FIAQM implementations may be further improved by considering other design aspects for the neural network of the FCP. For instance, different activation functions could yield more accurate and faster predictions of congestion in situations where shorter time intervals for the measurements are required. Finally, we point out that, although our experiments included only FQ-CoDel as the AQM scheme, the proposed FIAQM method could be straightforwardly implemented with other schemes. In those cases, the only necessary changes would

be the redefinition of the set of actions for the AQM tuner module and the inclusion of the specific instructions for the desired AQM scheme configuration in Linux.

Lastly, although in this work we proposed the use of metrics directly taken from the queues as the income data for FIAQM, other kinds of data may easily feed our proposed method. For example, as we mentioned in Section II, the metadata reported by routers employing the INT standard can be adapted to be used in FIAQM. However, how to incorporate INT metrics in multi-domain settings and Machine Learning-based solutions such as FIQAM requires further research.

REFERENCES

[1] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *Queue*, vol. 9, no. 11, p. 40, Nov. 2011, doi: 10.1145/2063166.2071893.

[2] C. A. Gomez, X. Wang, and A. Shami, "Intelligent active queue management using explicit congestion notification," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6, doi: 10.1109/GLOBECOM38437.2019.9013475.

[3] M. Chiesa, C. Dietzel, G. Antichi, M. Bruyere, I. Castro, M. Gusat, T. King, A. W. Moore, T. D. Nguyen, P. Owezarski, S. Uhlig, and M. Canini, "Inter-domain networking innovation on steroids: Empowering ixps with SDN capabilities," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 102–108, Oct. 2016, doi: 10.1109/MCOM.2016.7588277.

[4] K. C. C. Claffy, D. D. Clark, S. Bauer, and A. D. Dhamdhere, "Policy challenges in mapping Internet interdomain congestion," in *Proc. 44th Res. Conf. Commun., Inf. Internet Policy (TPRC 44)*, Aug. 2016. [Online]. Available: https://ssrn.com/abstract=2756868

[5] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*. [Online]. Available: http://arxiv.org/abs/1912.04977

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, Apr. 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019, doi: 10.1145/3298981.

[8] T. Li, A. Kumar Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," 2019, *arXiv:1908.07873*. [Online]. Available: http://arxiv.org/abs/1908.07873

[9] Y. Yang, X. Yin, X. Shi, Z. Wang, J. He, T. Z. Fu, and M. Winslett, "Inter-domain routing bottlenecks and their aggravation," *Comput. Netw.*, vol. 162, Oct. 2019, Art. no. 106839, doi: 10.1016/j.comnet.2019.06.017.

[10] Y. Zhao, A. Saeed, M. Ammar, and E. Zegura, "Unison: Enabling content provider/ISP collaboration using a switch abstraction," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–11, doi: 10.1109/ICNP.2019.8888032.

[11] W. Silva, "An architecture to manage incoming traffic of inter-domain routing using OpenFlow networks," *Information*, vol. 9, no. 4, p. 92, Apr. 2018, doi: 10.3390/info9040092.

[12] A. Dethise, M. Chiesa, and M. Canini, "Privacy-preserving detection of inter-domain SDN rules overlaps," in *Proc. SIGCOMM Posters Demos*, Los Angeles, CA, USA, Aug. 2017, pp. 6–8, doi: 10.1145/3123878.3131967.

[13] K. D. Joshi and K. Kataoka, "PRIME-Q: Privacy aware end-to-end QoS framework in multi-domain SDN," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2019, pp. 169–177, doi: 10.1109/NETSOFT.2019.8806645.

[14] M. S. Kang and V. D. Gligor, "Routing bottlenecks in the Internet: Causes, exploits, and countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Scottsdale, AZ, USA, Nov. 2014, pp. 321–333, doi: 10.1145/2660267.2660299.

[15] X. Chen, S. L. Feibish, Y. Koral, J. Rexford, O. Rottenstreich, S. A. Monetti, and T.-Y. Wang, "Fine-grained queue measurement in the data plane," in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol.*, Orlando, FL, USA, Dec. 2019, pp. 15–29, doi: 10.1145/3359989.3365408.

[16] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "HPCC: High precision congestion control," in *Proc. ACM Special Interest Group Data Commun.*, Beijing, China, Aug. 2019, pp. 44–58, doi: 10.1145/3341302.3342085.

[17] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling with Recurrent Neural Networking*, A. Graves, Ed. Berlin, Germany: Springer, 2012, pp. 37–45.

[18] S. Skansi, "Recurrent Neural Networks," in *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*, S. Skansi, Ed. Cham, Switzerland: Springer, 2018, pp. 135–152.

[19] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "Recurrent Neural Network Architectures," in *Recurrent Neural Netw. for Short-Term Load Forecasting: An Overview Comparative Analysis*, F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, R. Jenssen, Eds. Cham, Switzerland: Springer, 2017, pp. 23–29.

[20] F. Baker and G. Fairhurst, *IETF Recommendations Regarding Active Queue Management*, document RFC 7567, 2015.

[21] *6.1 Issues in Resource Allocation—Computer Networks: A Systems Approach Version 6.2-dev Documentation*. Accessed: Oct. 9, 2020. [Online]. Available: https://book.systems.approach.org/congestion/issues.html

[22] S. Floyd, *Metrics for the Evaluation of Congestion Control Mechanisms*. document RFC 5166, 2008.

[23] R. S. Sutton and A. G. Barto, "Temporal-difference learning," in *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018, pp. 131–132.

[24] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, New York, NY, USA, Oct. 2010, pp. 1–6, doi: 10.1145/1868447.1868466.

[25] C. A. Gomez. *FIAQM*. Accessed: Oct. 5, 2020. [Online]. Available: https://github.com/cgomezsu/FIAQM

[26] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 8026–8037.

[27] *TensorFlow Federated*. Accessed: May 6, 2020. [Online]. Available: https://www.tensorflow.org/federated

[28] *Care and Feeding of Netperf 2.7.X*. Accessed: May 13, 2020. [Online]. Available: https://hewlettpackard.github.io/netperf/doc/netperf.html

[29] *Realtime Response Under Load (RRUL) Specification—Bufferbloat.net*. Accessed: Jul. 13, 2020. [Online]. Available: https://www.bufferbloat.net/projects/bloat/wiki/RRUL_Spec/

[30] *TC(8)—Linux Traffic Control Manual Page*. Accessed: May 5, 2020. [Online]. Available: https://man7.org/linux/man-pages/man8/tc.8.html

[31] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. Conf. Appl., Technol., architectures, protocols for Comput. Commun.*, Portland, OR, USA, 2004, pp. 281–292, doi: 10.1145/1015467.1015499.

[32] D. Wischik and N. McKeown, "Part I: Buffer sizes for core routers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 75–78, Jul. 2005, doi: 10.1145/1070873.1070884.

[33] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, *The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm*, document RFC 8290, 2018.

[34] J. Galbraith and O. Saarenmaa. (Jul. 2006). *SSH File Transfer Protocol*. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-secsh-filexfer-13

[35] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. Mcmahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.

[36] M. Ion, B. Kreuter, A. E. Nergiz, and S. Patel, "On deploying secure computing: Private intersection-sum-with-cardinality," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Aug. 2020, pp. 370–389.

[37] S. Feng and H. Yu, "Multi-participant multi-class vertical federated learning," 2020, *arXiv:2001.11154*. [Online]. Available: http://arxiv.org/abs/2001.11154

[38] *Mininet Overview—Mininet*. Accessed: May 6, 2020. [Online]. Available: http://mininet.org/overview/

CESAR A. GOMEZ (Member, IEEE) received the B.E. degree in electronics engineering from St. Thomas Aquinas University at Bogota, in 2005, and the M.E.Sc. degree in telecommunications engineering from the National University of Colombia, in 2010. He is currently pursuing the Ph.D. in electrical and computer engineering with Western University, Canada. His background includes industrial experience for over ten years in several network engineering roles at companies, such as Siemens, ZTE, and Nortel. His current research interest includes application of machine learning techniques towards the realization of the intelligent networking automation paradigm. He is also the Vice-Chair of the IEEE Computer Chapter, London Section, Region 7.

XIANBIN WANG (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the National University of Singapore, in 2001. From January 2001 to July 2002, he was a System Designer with STMicroelectronics. From July 2002 to December 2007, he was a Research Scientist/Senior Research Scientist with the Communications Research Centre Canada (CRC). He is currently a Professor and a Tier 1 Canada Research Chair with Western University, Canada. He has over 400 peer-reviewed journal and conference papers, in addition to 30 granted and pending patents and several standard contributions. His current research interests include 5G and beyond, the Internet-of-Things, communications security, machine learning, and intelligent communications. He is a Fellow of the Canadian Academy of Engineering and the Engineering Institute of Canada. He has received many awards and recognitions, including the Canada Research Chair, the CRC President's Excellence Award, the Canadian Federal Government Public Service Award, the Ontario Early Researcher Award, and six IEEE Best Paper Awards. He was involved in many IEEE conferences, including GLOBECOM, ICC, VTC, PIMRC, WCNC, and CWIT, in different roles, such as symposium chair, tutorial instructor, track chair, session chair, and TPC co-chair. He is also serving as the Chair of the IEEE London Section and the Chair of ComSoc Signal Processing and Computing for Communications Technical Committee. He was also an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, from 2007 to 2011, and the IEEE WIRELESS COMMUNICATIONS LETTERS, from 2011 to 2016. He currently serves as an Editor/Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON BROADCASTING, and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He is an IEEE Distinguished Lecturer.

ABDALLAH SHAMI (Senior Member, IEEE) received the B.E. degree in electrical and computer engineering from Lebanese University, Beirut, Lebanon, in 1997, and the Ph.D. degree in electrical engineering from the City University of New York, New York, NY, USA, in 2002. He is currently a Professor with the ECE Department, Western University. He is also the Director of the Optimized Computing and Communications Laboratory. His research interests include performance and optimization modeling, machine learning and data analytics, the IoT, virtualization, cloud computing, and software-defined networks. He has chaired key symposia for IEEE GLOBECOM, IEEE ICC, IEEE ICNC, and ICCIT. He was the elected Chair of the IEEE Communications Society Technical Committee on Communications Software, from 2016 to 2017, and the IEEE London Section Chair, from 2016 to 2018. He is also an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE NETWORK, and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.