Machine Learning Controller for Data Rate Management in Science DMZ Networks

Christian Vega[a,c], Elie F. Kfoury[b] , Jose Gomez[b], Jorge E. Pezoa[a,†], Miguel Figueroa[a,d], Jorge Crichigno[b]
[a]Departamento de Ingeniería Eléctrica, Universidad de Concepción, Concepción, Chile
[b]College of Engineering and Computing, University of South Carolina, Columbia, U.S.A
[c]Facultad de Ingeniería, Universidad CESMAG, Pasto, Colombia
[d]Advanced Center for Electrical and Electronic Engineering (AC3E), Valparaíso, Chile.

Abstract

This article presents a Machine Learning Controller (MLC) supported by a P4 switch for improving rate control in non-dedicated Science Demilitarized Zone (Science DMZ) cyberinfrastructures. The proposed scheme utilizes passive data plane measurements such as Round Trip Time (RTT), throughput, queuing delay, and active flow count to regulate campus network output and achieve a desired Data Transfer Node (DTN) target rate. We evaluated our solution through a testbed using a bare-metal data plane switch, legacy router, and emulated hosts. Results show that including a rate controller based on data plane programmable devices on a non-dedicated Science DMZ cyberinfrastructure can effectively improve the completion time of scientific big data flows, while having a low impact on the campus network traffic and bottleneck link utilization. Specifically, the proposed controller achieved an average improvement of 21.72% in Flow Completion Time (FCT) compared to a trivial fixed-rate solution when the DTN uses BBR2 as a Congestion Control Algorithm (CCA). The results highlight the potential of machine learning techniques in conjunction with data plane measurements for optimizing the performance of non-dedicated networks.

Keywords: Science DMZ, Machine Learning Control, rate control, programmable data planes, P4 language, passive measurement.

1. Introduction

Cooperative research experiments, such as astronomical data collection, weather prediction, and molecular simulations, generate massive amounts of data. This Scientific Big Data (SBD) is transferred to processing and storage sites commonly located at long distances using dedicated links and cyberinfrastructures. However, SBD flows are also exchanged over non-dedicated networks owing to economic or technical constraints, thereby sharing links with general-purpose traffic, such as web browsing, Voice over Internet Protocol (VoIP), and streaming. Under these conditions, the high throughput demanded by SBD flows exchanged over high-latency links degrades significantly, resulting in longer completion times [1]. Nevertheless, because non-dedicated networks are designed and optimized for exchanging bursty and real-time traffic, the transmission of SBD flows over such networks is an open research field.

Rate control in cyberinfrastructures often relies on the default behavior of congestion control algorithms, Active Queue Management (AQM) mechanisms, and shaper filters. Our solution, instead, addresses three key innovations to improve the performance of SBD flows over non-dedicated networks. First, we employ a Science Demilitarized Zone (DMZ) composed of a Data Transfer Node (DTN) and a high-performance switch dedicated to sending and receiving SBD traffic [1]. Second, we exploit the capabilities of Programming Protocol-Independent Packet Processors (P4) switches [2] to collect data with nanosecond resolution and compute network metrics in the data plane at line rate [3]. Lastly, we devise a data-driven rate controller using a Machine Learning Controller (MLC) [4]. The MLC technique is a strategy for identifying effective model-free control laws from data-driven models in complex non-linear systems [4] such as building energy systems [5], robotic manipulators [6] and fluid mechanics [7]. In MLC, control laws can be supported by machine learning algorithms such as decision trees, support vector machines, and Artificial Neural Network (ANN), among others. Given the complexity of flow behavior in networks due, among other causes, to the operation of congestion control algorithms, the presence of short flows and long flows, packet loss, and retransmissions, we claim that MLC is an alternative to legacy end-to-end and network-assisted mechanisms to improve rate control in non-dedicated networks.

The MLC keeps the DTN transmission rate around the network administrator's desired target value while maintaining the Round Trip Time (RTT) and queuing delay within an appropriate operating regime. We developed an MLC law for SBD traffic over non-dedicated networks, using passive measurements supported on a P4 programmable switch. Because we perform passive measurements in the data plane, the proposed solution does not generate network traffic overhead or performance degradation. Such variables are fed to the MLC based

Email addresses: christianvega@udec.cl (Christian Vega*), ekfoury@email.sc.edu (Elie F. Kfoury), gomezgaj@email.sc.edu (Jose Gomez), miguel.figueroa@udec.cl (Miguel Figueroa), jcrichigno@cec.sc.edu (Jorge Crichigno)
* Corresponding author
†*Deceased*

on an ANN that implements a data rate driving law at the control plane. The control signal is refined using an Anti-windup filter, which acts as a limiter or modifier of the control signal to avoid accumulating the integral error that could induce large oscillations in the control loop [8]. The solution was deployed in an experimental testbed using a bare-metal data plane switch and a legacy router. The solution was evaluated for Cubic [9] and Bottleneck Bandwidth and Round-trip version 2 (BBR2) [10] Congestion Control Algorithms (CCAs) at the DTN. The proposed rate controller outperforms the fixed Token Bucket Filter (TBF) [11, 12] rate control strategy.

The main motivations for carrying out this work are summarized as follows:

- To improve the transmission of SBD traffic in non-dedicated infrastructures, where short and long flows coexist.
- To validate the applicability of data plane programmable devices as accurate telemetry elements to determine the current state of the network and perform control actions.
- To provide the network administrator with a solution based on intelligent control to optimize traffic control in academic cyberinfrastructures.

The most significant contributions of the present work can be framed as follows:

- A framework for network state variables abstraction leveraging the data plane capabilities to passively collect variables at the network bottleneck: rate, RTT, queuing delay, and the number of active flows. The proposed data plane telemetry system can be helpful for future applications in various areas, such as congestion control, security, network management, and routing.
- Data-driven models to control SBD flows over a non-dedicated network built from network state traces collected through extensive experiments on a real testbed. One of the models was trained when the DTN operates with Cubic as the congestion control algorithm, whereas another model was developed using BBR2 . These models use the foundations of direct inverse control supported by an ANNs.
- The evaluation of the proposed solution in a testbed with hardware-based routing and data plane processing devices. We used synthetic traffic generated by emulated hosts that recreated long and short flows for the tests. We employed controller performance and network-related metrics to evaluate the proposed solution.

2. Background and related work

2.1. Software-defined networks

Software-Defined Networks (SDN) is a modern network architecture that separates the control and data planes in the network. The control plane makes decisions about how network traffic should be managed and routed, while the data plane is responsible for physically forwarding the packets based on the instructions received from the control plane. Decoupling the planes enables flexible, programmable, and agile network management, enhancing scalability, efficiency, and adaptability [13].

SDN involves a centralized controller, operated by software, to manage network traffic dynamically by communicating with network devices through standardized protocols. The control plane of SDN provides an extension for managing the network's data plane components, such as switches and interfaces, through OpenFlow [14]. OpenFlow is the most prevalent application programming interface utilized to manage the control and data planes in SDN.

2.2. Programmable data plane

In the last decade, advances in the development of Application-Specific Integrated Circuits (ASICs) have shown that it is feasible to achieve terabit per second forwarding speeds with a set of packet-processing capabilities. These enhancements enable the creation of network devices capable of monitoring and controlling network traffic at line rates in the data plane. P4 [2] emerged as a de-facto data plane programming language that tackles three main goals: reconfigurability, protocol independence, and target independence because it is extensible for ASIC, Field Programmable Gate Array (FPGA), and virtual-based switches, among others [15]. The P4 ecosystem is growing with a wide range of products, projects, and services [16].

P4 leverages Protocol-Independent Switch Architecture (PISA) [17], a pipeline forwarding architecture for programmable data planes. Figure 1 depicts PISA, which is composed of three main programmable elements: (i) a parser that extracts packet headers based on a defined policy; (ii) a programmable match-action pipeline, which executes operation over the packet headers such as ordering or filtering using Arithmetic Logic Units (ALUs) and stateful memories; and (iii) a deparser that reassembles the packets and serializes them for transmission.

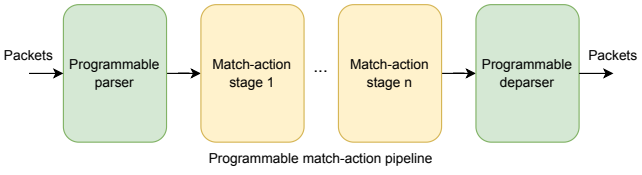


Figure 1: PISA architecture.

2.3. Science DMZ

The term Science Demilitarized Zone (Science DMZ) was first introduced in the Energy Science Network (ESnet) by Dart et al. as a design pattern that is suitable for optimizing the interactions between Wide Area Networks (WANs), campus networks, and computing systems [18]. The so-called "Science DMZ paper" also addresses use cases where the Science DMZ pattern has been implemented at the University of Colorado, Pennsylvania State University, Virginia Tech Transportation Institute, National Oceanic and Atmospheric Administration, and the National Energy Research Scientific

Computing Center. The authors also claim that there is a need to improve the transport protocols, the extension of virtual circuits, the adoption of high throughput standards, such as 100-Gigabit Ethernet, and the deployment of SDN solutions to enhance the performance of the SBD exchange. Several institutions and academic collaboration networks worldwide [19–21] have joined this initiative, favoring the technological platform for exchanging large volumes of scientific information. The main motivations for adopting the Science DMZ design pattern are collaboration, optimized network performance, scalability and security.

From the state-of-the-art in Science DMZ, we claim that two issues have been prioritized: performance and security. Regarding performance, the scope of the present work, ESnet proposes a set of good practices for network device selection and DTN tuning. The use of pacing, parallel flows, and novel congestion control protocols such as Bottleneck Bandwidth and Round-trip (BBR) are recommended strategies to maximize the performance of SBD transmissions [22, 23]. Network architects also have exploited SDN technology to deploy virtual circuits, thereby dealing with SBD traffic in actual network implementations. The authors in [24] examined architectural models for leveraging OpenFlow switches and the SDN model within the science-networking context. Then, they presented designs for SC12 SCinet Research Sandbox [25]. In Thailand, the National Research and Education Network was boosted by the adoption of SDN to control the operation on six research DMZ nodes [26]. The results showed that the throughput among these nodes increased from 100 Mbps to 900 Mbps. The AmoebaNet, an SDN-enabled service for SBD [27], was proposed to address three significant insights in Science DMZ: the last mile problem, the scalability problem, and the programmability problem. The solution considered traffic control based on quality of service policies, obtaining acceptable performances in differentiated traffic. However, AmoebaNet required the addition of functions specified in the SDN controller, including quality of service based routing path computation, flow manager, resource monitoring, topology manager, and programming interfaces. Additional software in DTN is required to connect with programming interfaces.

2.4. Rate control

Rate control in networks is a crucial area of study that continuously evolves with the growth of network technologies. The goal of rate control and bandwidth allocation is to manage network resources efficiently, ensure fair utilization, and provide high-quality services to end-users. Rate control strategies have been developed into end-to-end and network-assisted approaches. In the first, the end devices autonomously regulate the output rate according to indirect measurements that can be made on the state of the network. In contrast, network-assisted rate control is achieved by having a central entity, such as a controller, that monitors and regulates the data flows on the network.

2.4.1. Legacy network-assisted rate control

Legacy network-assisted rate control mechanisms rely on router's AQM algorithms and congestion notification mechanisms. One of the first AQM algorithms was Drop-Tail, in which packets from all hosts are accepted until the queue size is reached. Floyd and Jacobson in [28] proposed Random Early Detection (RED), which performs probabilistic packet dropping to inform the hosts about network congestion. The greater the queue utilization in the router, the higher the probability of a dropping. In the test performed, RED performed better than Drop-Tail. However, RED has some drawbacks, including relying only on queue occupancy, lack of knowledge of the number of flows that share the bottleneck, complex parameter tuning, and lack of suitability for small buffers [29, 30]. Controlled Delay (CODEL) was proposed in [31] focusing on router queuing delay. CODEL tracks the sojourn time and the time the packet takes between router ingress and egress. If the sojourn time overreaches a predefined threshold, the packet is discarded during the de-queue process [29]. Performance tests, such as those conducted in [32], showed that CODEL overcomes RED regarding queuing delay and link utilization. Despite the mentioned advantages, CODEL has issues with the router's memory usage and scalability owing to the additional computational load involved in the queuing delay computation. Subsequently, CODEL and Proportional Integral Controller Enhanced (PIE) [33] were proposed by Internet Engineering Task Force (IETF), with the premise of offering the best RED and CODEL. PIE relies on control theory to estimate the dropping probability based on queuing latency computation. The tests performed in [34] demonstrated that PIE outperformed CODEL in terms of queuing delay under high congestion conditions.

Regarding congestion notification mechanisms, Explicit Congestion Notification (ECN) [35] is a network feature that allows routers to mark packets as they pass through the network to notify congestion. Instead of dropping packets, ECN-capable devices set a specific bit in the IP packet header to indicate congestion. Therefore, it is possible to mitigate congestion proactively. Experiments carried out in [36–38] highlight the benefits of ECN, namely: avoid the global synchronization of retransmissions, reduce packet drops, promote a fairer environment, increase throughput, and reduce the delay.

2.4.2. Legacy end-to-end rate control

The default end-to-end rate control used on the Internet is based on Transmission Control Protocol (TCP) [39], whose algorithms have been refined over time. One of the most popular flavors of TCP CCA is Cubic, which is the default algorithm for Linux-based systems. Cubic is a loss-based CCA that relies on packet loss to detect congestion because routers discard packets when buffers fill up. However, when short flows compete with long flows, even a few of the latter, bandwidth starvation occurs in the network [40], affecting overall performance. In 2016, Google released BBR [41], a disruptive model-based CCA that estimates the bandwidth and RTT to infer Bandwidth Delay Product (BDP) and compute the host output rate. The authors of [42] concluded that BBR works well for a single long flow at a bottleneck. However, it presents coexistence issues with Cubic, as evidenced by the fairness index in tests performed in [43] and [44]. Subsequently, BBR2 [45] was released to mitigate

the drawbacks of the first version using ECN and packet-loss rate estimation. The tests conducted in [44] and [46] confirmed that BBR2 outperformed BBR in terms of fairness and packet loss.

2.4.3. Learning-based rate control

Due to the heterogeneous and complex traffic in today's networks, it is not feasible for a rate control mechanism to work correctly under different network conditions. Therefore, in recent years, initiatives have emerged that involve learning as an element to be considered in the modeling and operation of rate control algorithms. There is a set of end-to-end performance-oriented solutions involving upgrades at the end devices of the network. Remy [47], PCC [48], PCC Vivace [49], PCC Proteus [50], GCC [51], Copa [52], Indigo [53], and ZiXia [54] are based on calculating an objective function that leverages measurements of the network state. The objective function parameters in the above algorithms were tuned using recurrent neural networks or reinforcement learning. These solutions have shown significant improvements over legacy rate control algorithms. Other learning-based solutions train machine-learning-based models from datasets using supervised, unsupervised, and reinforcement learning techniques. A comprehensive survey of previous end-to-end rate control solutions based on machine learning was conducted by [55].

Although research has focused on end-to-end machine-learning-based rate controllers, there is a set of solutions for the network-assisted approach to enhance the AQM algorithms. Neural networks [56, 57], Q-learning [58–60] and reinforcement learning [61] are the preferred ML algorithms. The aforementioned network-assisted rate control mechanisms operate with limited functions restricted by vendors or software routers with performance impairments that make them impractical for high-performance applications [62].

2.4.4. Rate control supported by programmable data planes

P4 has enabled the deployment of rate control mechanisms from diverse approaches, including host-centric, enhanced feedback, traffic isolation, and fast rerouting [63]. Some solutions [64–69] require software modifications to the end hosts, and would therefore be applicable when the network administrator has complete control of the devices. Legacy AQM algorithms have been adapted and implemented for P4 switches with certain limitations [70–75]. P4 also enabled the fast development of custom AQM algorithms such as P4-SRPT [76], P4-ABC [77], SP-FIO [78], FDPA [79], P4QoS [80], CoDel++ [81] and QoSTCP [82]. These schemes benefit from the advantage of accurate measurements on the data plane, which allows for more timely control actions in the network.

Passive measurement is an emerging field in which data planes are used to support the operations performed at the control plane. Due to their line rate processing capability, these devices can perform accurate measurements without disturbing the network behavior. Devices such as passive taps are commonly used to create an exact copy of the traffic at monitoring points. The authors of [83] proposed a system that tracks RTTs online using a P4 switch as a passive measurement device. They then applied a meter to support interception attack detection. In [84], passive measurements of the number of flows and RTT supported by a P4 switch were used to control the queue of a legacy router and improve the Flow Completion Time (FCT) of short flows. Chen et al. used a similar approach by developing ConQuest, a tool that measures the queuing delay to identify flows that blow up the legacy router's queue.

3. Materials and methods

3.1. Experimental testbed

Figure 2 illustrates the network topology used to test the proposed solution. The testbed developed is a controlled environment that allows the reproduction of various aspects of a non-dedicated network that supports SBD flows in conjunction with commodity traffic. Therefore, some simplifications were considered in its design. The campus network consists of 100 sender nodes (s1, s2, ..., s100) connected by an Open vSwitch (OvS) (S1), generating commodity traffic. The Science DMZ is co-located with the campus network and consists of a DTN sender (DTN-s) and an OvS switch (S3). The testbed was implemented on a Lenovo ThinkSystem SR630 in which each network segment was linked to a physical interface to connect to the legacy router. The campus network and Science DMZ are connected to a Wide Area Network (WAN) through a Juniper MX204 border router (BR1). The remote network consists of a Linux border router (BR2), an OvS switch (S2), 100 receiver nodes (r1, r2, ..., r100), and a receiver DTN (DTN-r). The receivers and DTN-r are on the same logical network segment to represent remote locations over which there is no control by the network administrator. Hence, the testbed is configured as a non-dedicated cyberinfrastructure, allowing us to evaluate the coexistence between SBD and commodity traffic. All senders and receivers, including DNT-s and DTN-r, were deployed on isolated namespaces through Mininet [85] on the server. Mininet offers a lightweight platform to emulate networks and namespaces. Mininet has been employed in several works [44, 63, 84, 86, 87] enabling researchers to test protocols, algorithms, and applications in a controlled environment, facilitating the development of network solutions.

Router BR1 is connected through an optical fiber link to a physical interface of the server that is tied to Linux Virtual Router BR2 at a preconfigured link rate of 1Gbps to promote congestion. Thus, the link between BR1 and BR2 represents the network bottleneck. Optical passive taps are connected at the links of router BR1 and exactly replicate the observed traffic to an Edgecore 100BF-32 P4 switch (S4), which computes the network variables at line rate. The position of the passive taps enables the collection of the metrics of interest without generating traffic overhead in the network. The P4 switch is included as an element of innovation to pre-compute network metrics, and it is not inherently part of a Science DMZ architecture. The P4 switch uses the management link to communicate with the Controller, which receives the pre-computed metrics and outputs control actions to the S1-eth1 interface.

Table 1 shows the main technical specifications of the server, and Figure 3 presents the Cumulative Distribution

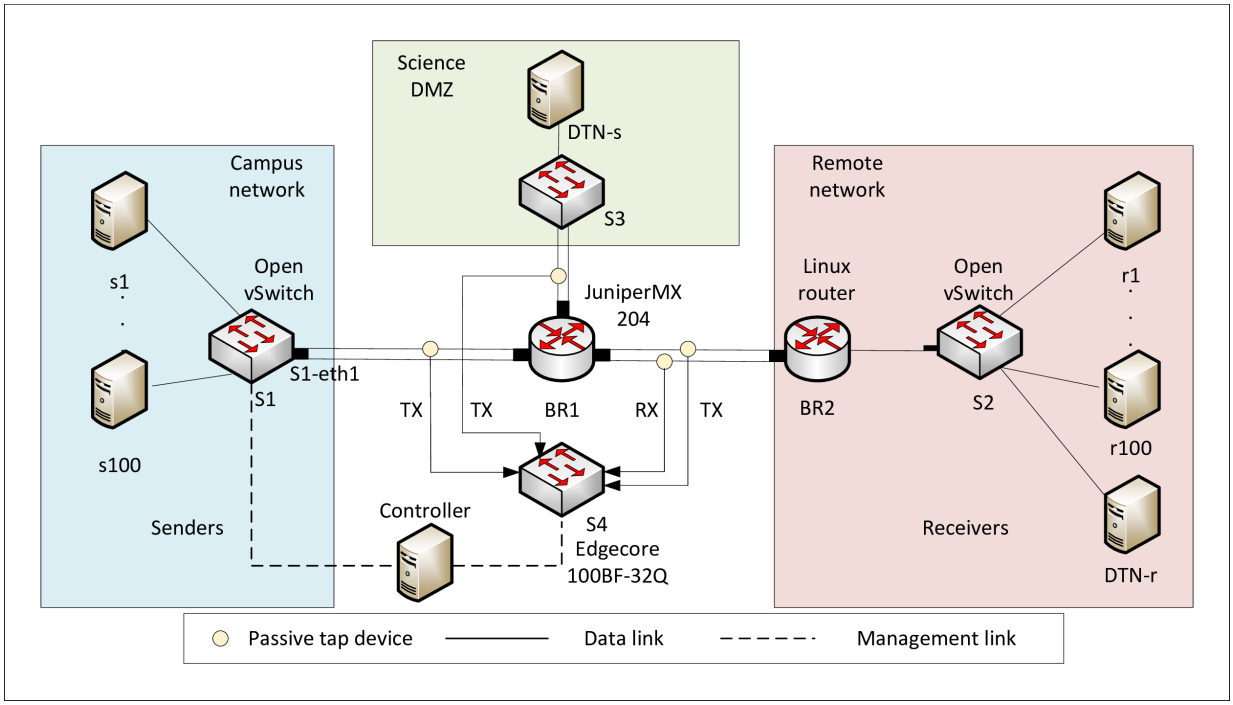


Figure 2: Testbed topology used for emulating scientific and general-purpose traffic in a shared cyberinfrastructure.

Function (CDF) of normalized CPU and memory usage traces in a test with traffic generated from all sources on the network during 600 seconds. The CPU usage remains below 11.2% while RAM usage remains below 2.3%. It is worth concluding that the computational resources are sufficient, and therefore, the bottleneck is conditioned only by the congestion on the network.

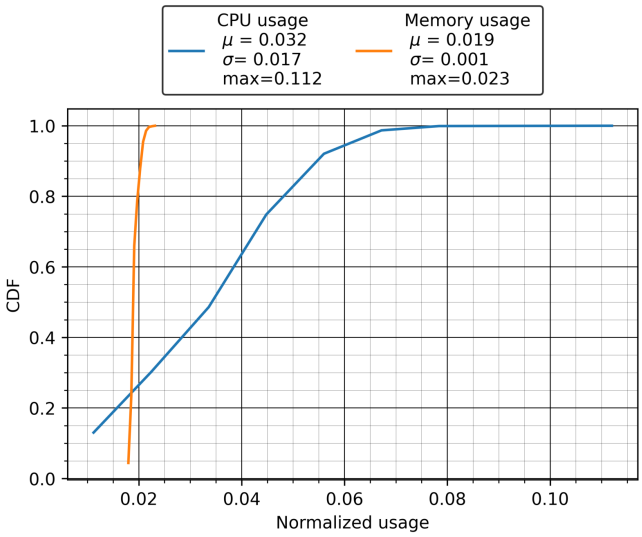


Figure 3: CPU and memory usage of the testbed.

Table 1: Technical specifications of physical server.

Model	Lenovo ThinkSystem SR630
Form factor	1U rack-mount
Total RAM	62.6GB
Processor	Intel Xeon 4114 @ 2.2GHz
Number of cores	40
Disk	ATA 480GB

Using the Netem tool [88], we artificially introduced a 100ms delay at the link connecting the DTN-s and DTN-r nodes to emulate a high-latency Science DMZ link. Netem was also used to introduce, at the switch labeled as S1, a 10ms propagation delay for the campus network traffic. Every sender node at the campus network sets up a single TCP connection with one receiver in the remote network. DTN-s establishes eight TCP connections to the DTN-r node, emulating the behavior of commonly used applications for scientific data transmissions such as Globus and GridFTP [89]. The number of flows generated with the former configuration is enough to saturate the network bottleneck link. The proposed solution allows the configuration of the CCA algorithm for both campus network senders and DTN-s.

3.2. Machine learning control system architecture

Control systems are a combination of components working together to maintain a desired output or response of a specific dynamical system by adjusting the input. The primary goal is to ensure that the actual output of a system follows the desired or reference output, compensating for disturbances and variations, by manipulating the system's input variables using a control strategy [90]. Often the actual response is considered as input information to the controller, thus resulting in a closed-loop or feedback control system.

In the present work, we abstract the network as a dynamical system with the network state $y(t)$ and an input $u(t)$ at time t as shown in Figure 4. The network state $y(t)$ is defined in terms of seven network variables: rate of DTN-s, rate of campus network, RTT of the connection between the DTNs, RTT of campus network, queuing delay of DTN-s, queuing delay of campus network, and the number of active flows. The signal $u(t)$ is the output rate at the S1-eth1 interface, which connects the campus network with the router BR1, as show in Figure 2.

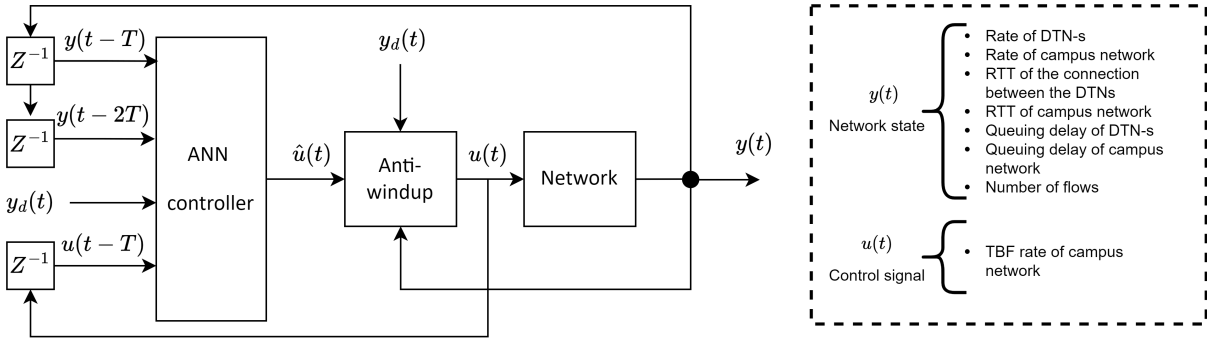


Figure 4: Machine learning control system architecture.

We claim that the variation in $u(t)$ affects the network state $y(t)$ and can be used as a control variable for the system. For instance, suppose that $u(t)$ is set to zero, and a transfer from DTN-s to DTN-r is then established. This connection has all the bandwidth available. In the BR1 queue, there will be only packets coming from DTN-s. Therefore, the queuing delay and the RTT will tend to be uniform and conditioned by the operation of the congestion control algorithm in DTN-s. At the opposite extreme, if $u(t)$ is set to the maximum possible value in S1-eth1, traffic from the campus network coexists with traffic from DTN-s, causing congestion and generating changes in the network state. By fine-grained regulating the signal $u(t)$, we claim that it is conceivable to restrain the traffic from the campus network when congestion is generated, attempting to favor the traffic coming from DTN-s.

The widely used direct inverse control strategy [91–94] seeks to lead the response of the system to a desired state or set-point $y_d(t)$ by modifying the input signal $u(t)$ obtained by the inverse model. The inverse model aims to cancel or nullify the system dynamics to equate $y_d(t)$ with $y(t)$. Past states of $u(t)$ and $y(t)$ can be considered to improve the estimation of the inverse model. Therefore, memory elements are required to feed the controller with delayed network states. The inverse model can be obtained from training an ANN with the input and output data of the system. An ANN is a computational model inspired by the human brain. It comprises interconnected nodes named neurons. These neurons collectively process and learn from input data, adjusting the weights between neurons through training to produce desired outputs [95].

We chose ANN model because of its straightforward implementation, the fact that we can obtain a data-driven model based on passive measurements in the data plane, and the capability to include past states to enhance the prediction. The ANN is fed with the desired network state $y_d(t)$, two previous network states $y(t-T)$ y $y(t-2T)$, and one past value of the system's input $u(t-T)$. In Figure 4, the blocks denoted as Z^{-1} are unitary delay blocks with a sample time T. The control law $\hat{u}(t)$ implemented with the ANN controller is defined as follows:

$$\hat{u}(t) = f(y_d(t), y(t-T), y(t-2T), u(t-T), \phi) \quad (1)$$

Where f is a non-linear function with parameters ϕ and T is a constant sample time. In the proposed solution, the set of parameters ϕ are the weights between the connections of neurons in the ANN layers. The set ϕ is obtained by probing random values of $u(t)$ and collecting the corresponding network state $y(t)$ every sample time T.

The controller is followed by an Anti-windup filter to reduce the error and avoid saturation of the control signal in the system loop. Since the output rate of S1 is between established limits, it is possible that the signal $\hat{u}$ overflows the maximum or minimum admissible value. A problem associated with this situation is the accumulation or windup of the error signal when the control signal is in saturation constraints, which can generate a degradation in the response time of the system while the error signal is being discharged [96]. An alternative solution to this problem is using an Anti-windup filter [8], which has also been applied to problems related to congestion control in computer networks [97]. The principle of the Anti-windup filter is to leverage the difference between the saturated control signal, i.e., between the admissible limits and the signal a controller computes. This component is added to the signal calculated by the controller to accelerate the discharge of the integral action. Figure 5 shows the block diagram of the Anti-windup filter, where the inputs are the error signal $e(t)$ computed by $y(t) - y_d(t)$ and the output of the ANN controller $\hat{u}(t)$. The signal $\overline{u}$ adds the contributions of signal $\hat{u}$, integral error, and Anti-windup action $w(t)$. Signal $w(t)$ is the difference between $u(t)$ and $\overline{u}$ and indicates the saturation of the control variable. The $\overline{u}$ value is coerced by a saturator in a $\pm\Delta u(t)$ interval. Finally, the $u(t)$ signal is set at the S1-eth1 output rate. The Anti-windup filter has three tuning parameters, namely, K_p, K_i, and K_w, which handle proportional, integral, and Anti-windup actions. In summary, the proportional action accelerates the system response to reach the set point, in this case, the desired output rate for the campus network; the integral action aims to reduce the bias of the system response; and the Anti-windup action accelerates the drain of the accumulated error when there is saturation in the control signal.

3.3. Tuning of controller parameters

To tune the controller parameters, we propose an objective function to minimize the Integral Absolute Error (IAE) that accumulates the absolute error over time. The absolute error is computed as the difference between the set-point rate $R^*_{DTN}(t)$ and the measured throughput $R_{DTN}(t)$ of DTN-s. Hence, IAE is defined as follows:

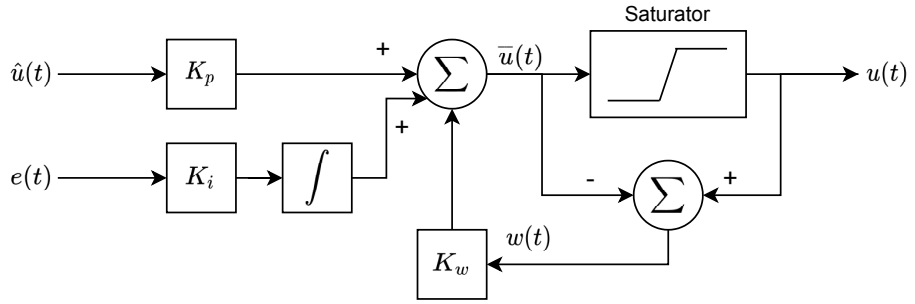


Figure 5: Anti-windup filter.

$$\min_{K_p,K_i,K_w} IAE = \int_0^t \left| R^*_{DTN}(\tau) - R_{DTN}(\tau) \right| d\tau \qquad (2)$$
$$K_p, K_i, K_w \geq 0$$

Due to the complexity of the system dynamics resulting from the concurrence of heterogeneous flows and their collateral effects, such as delay, packet loss, and retransmissions, it is not viable to find a theoretical model to obtain the optimal operating values of the parameters. However, the fact that different combinations of the parameters can be attempted on the testbed makes it possible to explore the solution space to determine suitable operating values. Since the three parameters can adopt positive continuous values, the solution space is ample, and therefore, an effective technique is required to find the appropriate values of the parameters. Evolutionary algorithms provide an effective alternative search strategy for identifying near-optimal solutions in a high-dimensional search space [4]. A Genetic Algorithm (GA) is an optimization algorithm inspired by natural selection. It starts with a population of potential solutions represented as individuals. These individuals undergo processes like mutation, crossover, and selection to produce a new generation of solutions. Over multiple generations, the algorithm refines and improves the solutions based on their fitness to a defined objective or fitness function, mimicking evolution process [98]. As shown in [99–101] GA has proven to be a practical method for controller tuning, particularly proportional integral derivative controllers. Hence, a GA was used to find the controller parameters K_p, K_i and K_w.

Selecting the parameters of the GA is worthy of considering the computational limitations of the testbed. The maximum time allowed for an individual iperf3 test is 86400s. In every computation of the fitness function or IAE inside the GA, we generate traffic from the sources to their corresponding destinations for 120 seconds using a pseudo-random $R^*_{DTN}(t)$ signal. Regarding the traffic generation time of 120 seconds, this was chosen because it is longer than TCP time needed to reach a stable behavior once the slow-start phase has been overcome, and also it allows collecting a representative sample of data to estimate the state of the network. The mating pool size and the number of generations must be enough to offer diversity and the possibility to explore the search space and fit that maximum time restriction. The GA parameters are summarized in Table 2, and match with the requirements and computational limitations. For crossover and mutation probability, we use typical values. To build and implement the GA, we used the python library PyGAD [102].

Algorithm 1 shows the parameter tuning process by using the GA.

Algorithm 1 GA for MLC parameter tuning problem.

Require: Generation G; Max generations, G_{max}; number of parents mating, m_p; mating pool size, m_s; crossover probability, p_c; mutation probability, p_m

Ensure: Best controller parameters: Kp, Ki, Kw

$G = 0$; $P(G)$: Initial population

Compute IAE of population ($P(G)$)

repeat

$M \leftarrow$ Select parents ($P(G), m_p$)

$P_c(G) \leftarrow$ Crossover($P(G)$, p_m, M)

$P_m(G) \leftarrow$ Mutation($P_c(G)$, p_m)

Compute IAE of ($P_m(G)$)

$P(G+1) \leftarrow$ Selection ($P_m(G)$, steady state selection)

$G = G + 1$

Compute IAE ($P(G)$)

$K_p, K_i, K_w \leftarrow$ Get best solution ($P_m(G)$)

until $G \geq G_{max}$

return K_p, K_i, K_w

Table 2: Genetic algorithm parameters.

Parameter	Value
Number of generations	100
Number of genes	3
Number of parents mating	15
Mating pool size	30
Crossover probability	0.90
Mutation probability	0.1

3.4. Data collection and computation of network variables

The control system architecture shown in Figure 4 considers a data-driven network model. This model is defined from seven network variables that jointly define the network state. We exploited the capabilities supported by the P4 switch to collect data and compute such network variables online at the data plane, as shown in Figure 6.

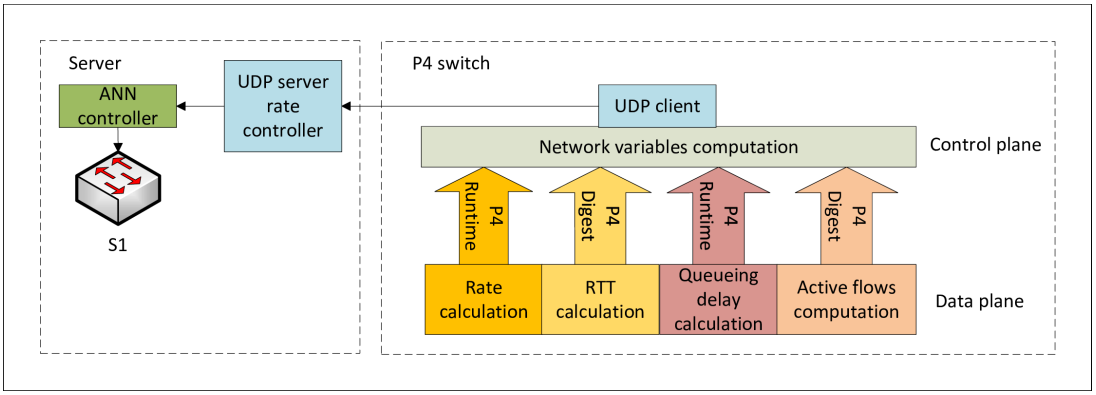


Figure 6: Functional block diagram of the solution.

One of the advantages of data plane processing is the low computational complexity required to enable fast-packet processing. The $P4_{16}$ specification indicates that P4 programs execute a constant number of operations for each byte of an input packet received and analyzed [103]. Therefore, the computational complexity of a P4 program is linear in terms of input bytes or $O(n)$ using the Big O notation. In the programs developed in the present study, IP and TCP fixed-size headers are used, hence the processing time of a packet is constant, and we claim that all data plane algorithms have a $O(1)$ computational complexity. However, in control plane, the complexity of the algorithms increases in proportion to the number of flows handled in the data structures. The computation of each variable is explained as follows.

3.4.1. Rate calculation

Rate Calculation relies on codes running concurrently at the data and control planes. At the data plane, counters for campus network traffic and DTN-s traffic are permanently updated. We used *Direct Counters* elements available in the P4 switch architecture. As shown in Algorithm 2, when a packet enters the switch, a match-action table first separates the incoming traffic between the campus network and the DTN-s using a Longest-Prefix Match (LPM) rule based on their destination IP. Subsequently, a counter is assigned and incremented with the packet size in bytes for each traffic source.

Simultaneously, as shown in Algorithm 3, the control plane polls the counters every T seconds through the P4-Runtime interface and computes the difference between the current and previous byte count per traffic flow, ΔB_{Campus} and ΔB_{DTN-s}. Additionally, the control plane program computes the time difference between the timestamps, namely Δt. Thus, the data rates are obtained from the ratio between ΔB and Δt. The computational complexity of this algorithm is $O(1)$ because it only performs assignment and mathematical operations from fixed-size data.

3.4.2. RTT calculation

We leverage the method proposed in [104] for the RTT computation in which the switch tracks TCP connections through the TCP sequence (SEQ) and acknowledgment (ACK) flags of outgoing and incoming packets using hash functions available in the P4 Tofino switch

Algorithm 2 Update counters at data plane.

Require: Packet size $pkt.s$; packet headers hdr; DTN-r IP address $IP_{DTN\text{-}r}$; remote network IP address segment IP_{RN}; DTN-s counter, $C_{DTN\text{-}s}$; campus network counter, C_{Campus}
Ensure: $C_{DTN\text{-}s}, C_{Campus}$
  if $hdr.ipv4.dst\_addr = \text{LPM}(DTN\text{-}r)$ **then**
    $C_{DTN\text{-}s} \leftarrow C_{DTN\text{-}s} + pkt.s$
  else if $hdr.ipv4.dst\_addr = \text{LPM}(IP_{RN})$ **then**
    $C_{Campus} \leftarrow C_{Campus} + pkt.s$
  end if

Algorithm 3 Rate calculation at control plane.

Require: DTN-r IP address $IP_{DTN\text{-}r}$; remote network IP address Segment IP_{RN}; DTN-s counter, $C_{DTN\text{-}s}$; campus network counter, C_{Campus}; sample time, T
Ensure: $R_{DTN\text{-}s}, R_{Campus}$
  while True **do**
    $Cur\_Bytes_{Campus} \leftarrow$ Read Counter (C_{Campus})
    $Cur\_Bytes_{DTN\text{-}s} \leftarrow$ Read Counter $(C_{DTN\text{-}s})$
    $\Delta B_{Campus} \leftarrow Cur\_Bytes_{Campus} - Prev\_Bytes_{Campus}$
    $\Delta B_{DTN\text{-}s} \leftarrow Cur\_Bytes_{DTN\text{-}s} - Prev\_Bytes_{DTN\text{-}s}$
    $Cur\_ts \leftarrow$ Get Timestamp(now)
    $\Delta t \leftarrow Cur\_ts - Prev\_ts$
    $R_{Campus} \leftarrow (\Delta B_{Campus} \times 8)/\Delta t$
    $R_{DTN-s} \leftarrow (\Delta B_{DTN\text{-}s} \times 8)/\Delta t$
    Wait (T seconds)
  end while

architecture. The RTT computation is obtained by subtracting the timestamps of the outgoing packet and the corresponding ACK incoming packet. We modified the code in the control plane and the data plane to distinguish the RTT from the flows coming from the campus network and those coming from the DTN. In our solution, we seek outgoing and incoming flows using a passive tap at the bottleneck interface of router $R1$. Next, the RTT samples are pushed to the control plane by using the P4 Digest interface.

Algorithm 4 describes the operations required at the control plane to obtain the RTT of the flows between the campus network and the remote network and also between DTN-s and DTN-r. The *flow_vector* structure tracks the active flows from the campus network that are

Algorithm 4 RTT calculation at control plane.

Require: campus network flow, f; DTN Flow, f_{DTN}; sample time, T; vector of active flows, $flow\_vector$; smooth factor, α
Ensure: RTT_{DTN}, RTT_{Campus}

```
while True do
    flow_vector ← Update(flow_vector)
    S ← 0
    for f in flow_vector do
        S ← S + f.RTT
    end for
    Avg_RTT ← S/length(flow_vector)
    RTT_Campus ← α × RTT_Campus + (1 − α) ∗ Avg_RTT
    RTT_DTN ← Retrieve from Digest (f_DTN.RTT)
    Wait (T seconds)
end while
```

transiting to the destination network and the respective estimated RTT values from the data plane. In each cycle, the *flow_vector* is updated using the database of flows saved in the data plane, including new flows and deleting inactive flows using a timeout. The representative RTT value of the campus network, RTT_{Campus} is obtained by averaging the values of the vector. Thereafter, a smoothing function driven by an α parameter is applied to reduce fluctuations. The RTT of the connection between DTN-s and DTN-r, RTT_{DTN}, is retrieved directly from the Digest interface. The complexity is $O(N)$ because of the *for loop* needed, which is the most computationally complex operation of the algorithm, and it is conditioned by the size of $flow\_vector$.

3.4.3. Queuing delay calculation

According to [105], the queuing delay is a crucial metric for early congestion detection. As shown in Algorithm 5, we compute the queuing delay as the difference between every packet's timestamps at ingress and egress and store such value in a register. Registers are stateful elements of switch architecture that can be stored over time. We also used a match-action table driven by a LPM rule to separate the DTN-s and campus network traffic measurements.

Algorithm 6 describes the tasks performed at the control plane to support queuing delay polling. Every sample time T, the registers are polled from the data plane to obtain queuing delay samples from campus network traffic and DTN-s traffic. The complexity of this algorithm is just O(1), because it implies only variable assignment operations with fixed-size registers.

3.4.4. Active flows computation

Using the approach addressed in [87], we consider a flow to be *active* if the packets associated with this, exceed a predefined threshold C_TH in a time less than T_TH. Algorithm 7 shows the process of updating the Active Flows register AF_Reg. The addition or deletion of flows in that structure is notified to the control plane using Digest interfaces, and the capability of P4 switch to read protocol-specific fields is exploited to remove flows from the registers using the FIN flag present when a TCP connection is closing. This update process is necessary

Algorithm 5 Update queuing delay at data plane.

Require: Packet, pkt; DTN-r's IP address $IP_{DTN\text{-}r}$; remote network IP address segment IP_{RN}; DTN's switch port, DTN_port; campus network switch port, $Campus\_Network\_port$; Switch timestamp, SW_{ts}; Register, Reg
Ensure: $QD\_Reg_{DTN}, QD\_Reg_{Campus}$

```
flow_id ← Hash function (pkt.hdr)
in_port ← pkt.intr_metadata.in_port
if in_port = Campus_Network_port or in_port = DTN_port then
    Reg(flow_id) ← Sw_ts
else if in_port = Bottleneck_port then
    if hdr.ipv4.dst_addr = LPM(DTN-r) then
        QD_Reg_DTN ← SW_ts − Reg(flow_id)
        Push to Digest(QD_Reg_DTN)
    else if hdr.ipv4.dst_addr = LPM(IP_RN) then
        QD_Reg_Campus ← SW_ts − Reg(flow_id)
    end if
end if
```

Algorithm 6 Queuing delay polling at control plane.

Require: Queuing delay register for DTN's traffic QD_Reg_{DTN}; queuing delay register for campus network traffic, QD_Reg_{Campus}, sample time, T
Ensure: QD_{DTN}, QD_{Campus}

```
while True do
    QD_Campus ← Read register (QD_Reg_Campus)
    QD_DTN ← Read register (QD_Reg_DTN)
    Wait (T seconds)
end while
```

not only for the accurate computation of the number of flows but also to optimize the device's memory resources.

At the control plane, as shown in Algorithm 8, the Digest interfaces are periodically polled at time T, allowing the $flow\_vector$ to be updated in the control plane. Finally, the number of active flows is calculated as the length of the flow vector. Here the algorithm's complexity is O(1), because the algorithm supports variable assignment, attachment, removal, and length computation operations.

3.5. ANN model training

To obtain the inverse model, we first collected input and output data from the network by applying the principles of systems identification. To do so, we injected a pseudo-random step signal at $u(t)$ into the network for 28.800 s, to recreate different output rates of the campus network and simultaneously collect the network state variables $y(t)$. Figure 7 shows a portion of 600 seconds of the pseudo-random signal used to perform the experiment. To put the above reasoning into practice, we developed a Bash script that generates a random number between zero and 1000 every time T and set it as the output rate in Mbps of the S1-eth1 interface. We used the P4 switch as a network state sensing device to measure the network state, as shown on the right side of Figure 6. To obtain the inverse model, the role of $u(t)$ is changed by

Algorithm 7 Update active flows at data plane.

Require: packet headers, hdr; switch timestamp, SW_{ts}; flow identifier, $flow\_id$; counter register, C_Reg; timestamps register, TS_Reg; active flow register, AF_Reg; new flow Digest interface, new_flow; timeout Digest interface, $timeout$.
Ensure: Active flows register, AF_Reg

```
flow_id ← Hash function (pkt.hdr)
if flow_id in C_Reg then
    prev_ts ← TS_Reg[flow_id]
    TS_Reg[flow_id] ← SW_ts
    if TS_Reg[flow_id] − prev_ts < T_TH then
        if C_Reg[flow_id] = C_TH then
            Attach(flow_id) to AF_Reg
            Push (flow_id) to new_flow Digest
        else
            C_Reg[flow_id] ← C_Reg[flow_id] + 1
        end if
    else
        C_Reg[flow_id] ← 0
    end if
    if hdr.tcp.flags = FIN then
        if C_Reg[flow_id] = C_TH then
            Remove(flow_id) from AF_Reg
            Push (flow_id) to timeout Digest
        end if
    end if
else
    Attach (flow_id) to C_Reg
    Attach (flow_id) to TS_Reg
end if
```

considering it now as an output and the variable predicted by the ANN model. To predict $u(t)$ we exploited the data collected from the identification process taking as predictors the set $I = \{y(t), y(t-T), y(t-2T), u(t-T)\}$. The set I was employed to train a fully connected neural network using the Keras and Tensorflow ANN libraries [106, 107].

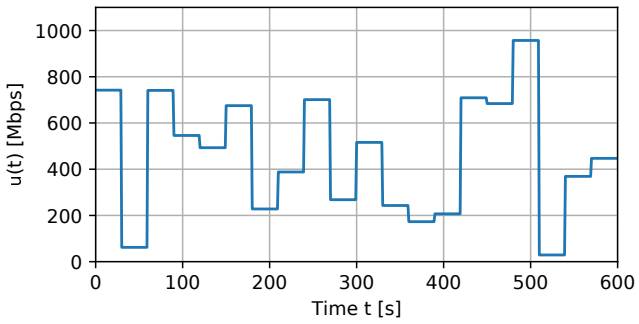


Figure 7: Portion of a signal used to train the MLC.

The chosen architecture of the ANN is a fully connected neural network, which is a fundamental type of ANN where each neuron in one layer is connected to every neuron in the subsequent layer. In this architecture, the neurons are organized into input, hidden and output layers as shown in Figure 8. In the present work, the input layer is composed by the the set I. The second layer

Algorithm 8 Active Flows computation at control plane.

Require: Digest for new flows report, new_flow; Digest for timeout report, $timeout\_flow$; sample time, T; vector of active flows, $flow\_vector$; flow identifier, $flow\_id$
Ensure: Number of active flows AF

```
while True do
    new_flow_id ← Retrieve from Digest(new_flow[flow_id])
    if new_flow_id then
        Attach(new_flow_id) to flow_vector
    end if
    timeout_flow_id ← Retrieve from Digest (timeout_flow[flow_id])
    if timeout_flow_id then
        Remove(timeout_flow_id) from timeout_vector
    end if
    AF = length(flow_vector)
    Wait (T seconds)
end while
```

is a hidden layer that is 64 times the dimension of the set I. The third layer is a pre output layer with the same dimensions as the set I. The output layer has a size of one and represents the predicted signal $u(t)$. In the experiments conducted, we achieved a relative validation error of 1% using the cross-validation technique. Two inverse models were obtained according to the CCA configured at DTN-s, either Cubic or BBR2. Training was performed for 12,000 s to provide sufficient information to build the model.

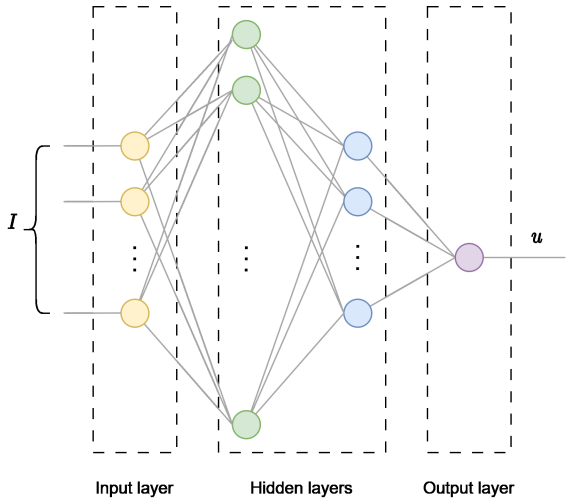


Figure 8: Artificial neural network used for the MLC.

3.6. Control law implementation

The MLC control signals are carried using User Datagram Protocol (UDP) sockets on the management interface between the server and the switch S1. The control plane generates a JavaScript Object Notation (JSON) structure containing the computed network variables $y(t)$, $y(t-T)$, $y(t-2T)$, and $u(t-T)$ every T time interval. Then, it sends such a structure to the server, which computes the output rate $u(t)$ by feeding the trained ANN controller with the abovementioned

structure. The administrator sets the desired network state $y_d(t)$ by considering the target DTN-s rate and network constraints.

Table 3 lists the values that fill the vector y_d with the desired network state. The expected value of R_{campus} is the remaining capacity that is not required by DTN-s. The RTT values are set considering the restrictions imposed by the testbed, namely $100ms$ for SBD flows and $10ms$ for general-purpose traffic. The expected queuing delays are set to 0 to minimize bottleneck congestion. Finally, the number of Active Flows AF is set using the last active flow computation.

Table 3: Parameters of y_d.

k	Variable name	Value
1	$R_{DTN\text{-}s}$	Target Rate $R^*_{DTN\text{-}s}(t)$
2	R_{campus}	$B_{max} - R^*_{DTN\text{-}s}(t)$
3	RTT_{DTN}	100 ms
4	RTT_{Campus}	10 ms
5	$QD_{DTN\text{-}s}$	0 ms
6	QD_{Campus}	0 ms
7	AF	Last AF

3.7. Comparison scenario

A common solution for controlling campus network traffic is to set a target output rate leveraging the configuration capabilities of the network device. The network manager often coarsely changes this target rate according to the network operating conditions but without making fine-tuning as proposed in this article. Open vSwitch (OVS) switches admit TBF for rate control. Therefore, if the available bandwidth at the bottleneck is B_{max} the campus output rate is set to $B_{max} - R^*_{DTN\text{-}s}(t)$. We call this method the trivial solution; it is the reference point for analyzing the performance of the proposed solution.

4. Results

In this section, we present the results of the tests conducted to evaluate the performance of the proposed solution under diverse scenarios. The experiments were performed in a data center using real network devices. The tests were replicated a significant number of times in order to obtain statistically reliable results. Performance evaluation focuses on FCT and link utilization as metrics that quantify the performance from a network perspective. However, Mean Relative Absolute Error (MRAE) is also analyzed as a metric to evaluate the ability of the algorithm to track the target DTN-s rate.

4.1. Controller parameters

Using the models obtained in Section 3.5, we conduct parameter tuning of the controller presented in Section 3.2, supported by the genetic algorithm setup shown in Table 2. Figure 9 shows the convergence curve of IAE in the genetic algorithm used to tune the controller parameters when either Cubic or BBR2 is configured at the DTN-s. For the case of BBR2, the cumulative error difference between the actual rate and the set point is smaller than when we use Cubic. Table 4 summarizes the results of the parameters' tuning procedure.

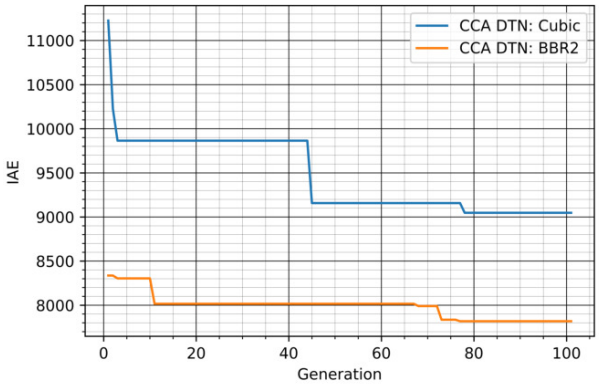


Figure 9: Convergence curve of IAE in controller parameters tuning. As the number of AG generations increases, the IAE decreases. Although the greatest reduction in IAE occurs when Cubic is used as CCA in DTN-s, better error tracking is obtained with BBR2.

Table 4: Selected controller parameters.

DTN-s CCA	Generation	K_p	K_i	K_w
Cubic	77	0.285	1.486	1.565
BBR2	76	0.388	1.918	1.340

4.2. Controller performance oriented test with multiple long flows in campus network

This experiment evaluated the proposed solution from the DTN-s performance perspective. Therefore, the senders in the campus network generate long flows to their respective receivers with an induced delay of 10ms. However, the DTN-s generates long flow traffic with an induced delay of 100ms.

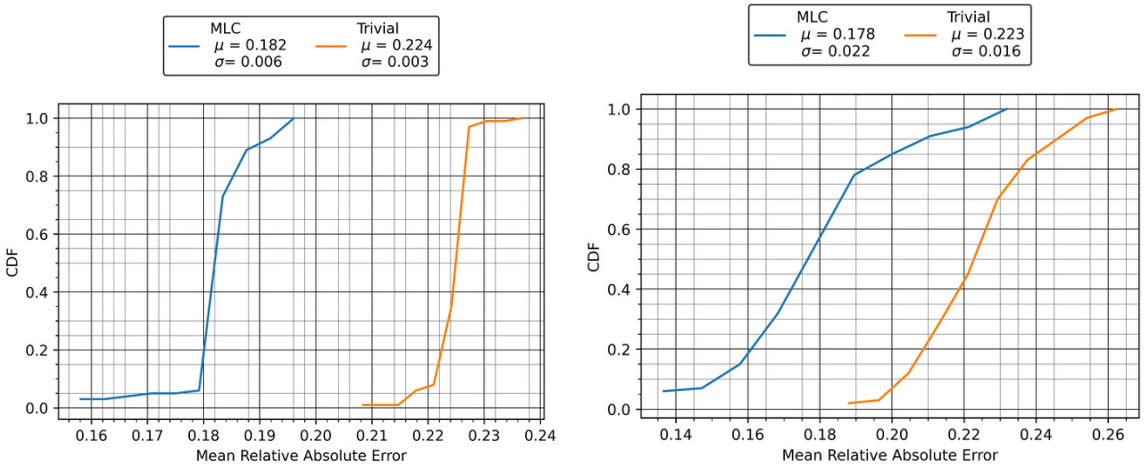
In order to test the MLC accuracy, we compute MRAE, every t_N samples as follows:

$$MRAE = \frac{1}{t_N}\sum_{t=1}^{t_N}\left|\frac{R^*_{DTN\text{-}s}(t) - R_{DTN\text{-}s}(t)}{R^*_{DTN\text{-}s}(t)}\right| \quad (3)$$

MRAE is a measure of set-point tracking which is the ability to maintain or follow the desired reference, in this case R^*_{DTN-s}. In the experiment, $t_N = 120$, and because $\Delta t = 1s$ each trial takes 120s. In total, we perform 100 trials for each MLC. This number of trials guarantees an expected error of less than 0.43% and a confidence level of 95%, assuming a normal distribution of the traces. CDFs of MRAE are shown in Figure 10. In general, the implementation of MLC outperforms the trivial solution in terms of MRAE, with an average reduction of 4.1% and 4.5% for the MLCs of Cubic and BBR2, respectively.

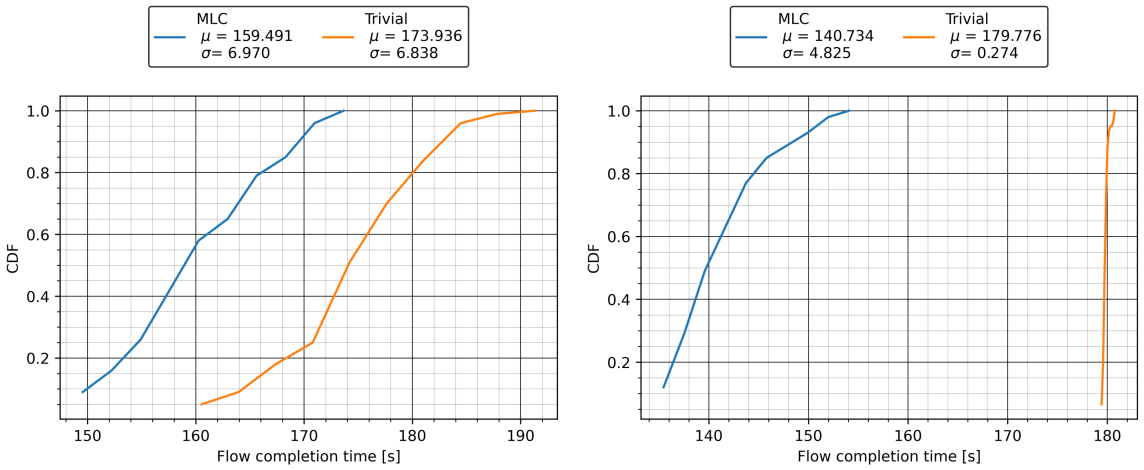
4.3. DTN's performance oriented test with long flows in campus network

Although the proposed controllers adequately track the reference, evaluating the solution's performance in terms of network metrics is essential. Flow Completion Time (FCT) is the time elapsed between sending the first packet and receiving the last packet for a given TCP connection. This experiment evaluates FCT for 10GB data transmission from DTN-s to DTN-r. The experiment was repeated 100 times for each MLC to obtain the CDFs



(a) (b)

Figure 10: Mean Relative Absolute Error (a) DTN-s CCA: Cubic (b) DTN-s CCA: BBR2. The proposed solution provides better set-point tracking in both scenarios.



(a) (b)

Figure 11: DTN-s Flow Completion Time (a) DTN-s CCA: Cubic (b) DTN-s CCA: BBR2. The proposed solution improves in both scenarios the FCT of the SBD flows, having a more significant impact when BBR2 is configured as CCA in DTN-s.

shown in Figure 11. This number of trials guarantees an expected error of less than 1.47s and a confidence level of 95%, assuming a normal distribution of the traces. An average reduction against the trivial solution of 12.932s and 39.042s of FCT for Cubic and BBR2 controllers, respectively, was found. The previous means an FCT reduction of 7.4% in the case of Cubic's MLC and an average reduction of 21.72% using BBR2's MLC. We also note that 100% of the attempts with the proposed controller obtained a lower FCT than the trivial solution for the BBR2 case. The results imply that it is possible to transmit more scientific data in a given time interval with the proposed MLCs.

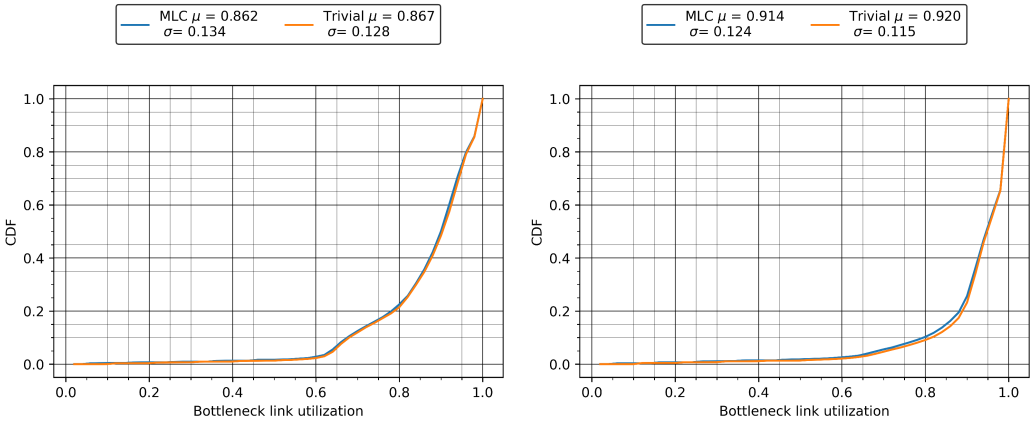
We also compute the bottleneck Link Utilization ρ using data plane measurements. This value is obtained at time t through the sum of the flow rates, including traffic generated by the DTN-s $R_{DTN\text{-}s}$ and the campus network hosts R_{s_h}, divided by the maximum link bandwidth B_{max}, as follows:

$$\rho(t) = \frac{R_{DTN\text{-}s}(t) + \sum_{h=1}^{100} R_{s_h}(t)}{B_{max}} \tag{4}$$

For link utilization, we also use 100 trials, and the expected error is 0.0255, with a confidence level of 95%, assuming a normal distribution of data. Figure 12 presents the CDFs of bottleneck link utilization for the MLCs and their respective trivial comparison scenario; we found similar behavior in the two situations, with an average negligible decrease of 0.57% and 0.65% in the ρ of the proposed solutions.

4.4. DTN's performance with multiple streams in campus network

The aim of these experiments is to assess how the proposed solution behaves when the number of streams in the campus network varies. To achieve this, we configured the testbed such that the campus network establishes a specific number of streams N_s given by TCP connections with the remote network, and then measured the FCT of a 10GB data transmission between DTN-s and DTN-r.



(a) (b)

Figure 12: Bottleneck link utilization. (a) DTN-s CCA: Cubic (b) DTN-s CCA: BBR2. The proposed solution does not impact the bottleneck link utilization.

We ran the test for Cubic and BBR2 algorithms configured in DTN-s, and for 1, 10, 100, 1000, and 10000 streams. We tested the P4 switch during compilation and execution for overflow problems and did not find any issues.

The results of the experiments are shown in Figure 13. The percentage reduction of the FCT of the proposed MLC, with respect to the trivial solution, is presented in Table 5. The proposed solution is capable of reducing the FCT of the SBD traffic by an average of 0.07% to 21.72%, depending on the CCA and the number of streams. The MLC implemented is particularly useful for reducing the FCT above 10 streams. This is because TCP inherently tends to utilize all available bandwidth, regardless of the number of streams established between the campus network and the remote network. Additionally, the proposed MLC calculates the output rate of the campus network from aggregated network state metrics, allowing the system to be scalable to the maximum number of admissible inputs in the active flows table.

Table 5: Flow completion time reduction of the proposed solution varying the number of streams.

CCA	N_s	Mean FCT [s]		FCT reduction [%]
		MLC	Trivial	
Cubic	1	108,51	110,46	1,77
	10	154,37	167,83	8,02
	100	159,49	173,94	8,30
	1000	162,47	169,17	3,96
	10000	156,27	169,10	7,58
BBR2	1	91,85	91,91	0,07
	10	105,99	114,86	7,72
	100	140,73	179,78	21,72
	1000	140,58	150,62	6,66
	10000	140,67	151,78	7,32

4.5. Performance of short flows coexisting with long flows in the bottleneck

The present experiment evaluates the FCT of short flows when sharing the bottleneck link with SBD flows for the proposed MLCs. The Weibull heavy-tailed distribution has been widely used to model the behavior of short flows on the Internet [108]. The probability distribution function of a Weibull random variable is defined as:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} & \text{if} \quad x \geq 0 \\ 0 & \text{if} \quad x < 0. \end{cases} \tag{5}$$

where k and λ are the *shape* and *scale* parameters, respectively. We used two variables to model short flows. First, inter-departure time measures the difference between the departure times of one packet and the next. Second, flow size denotes the number of bytes occupied by each flow. We used the SourcesOnOff tool [109], which allows us to generate flows with given distributions for inter-departure time and flow size. Table 6 summarizes the parameters that describe the short flows from Weibull distributions in the experiments.

Table 6: Selected controller parameters.

Variable	k	λ	min	max
Inter-departure time	0.5	20ms	1ms	100ms
Flow size	0.5	100kB	10kB	10MB

In this experiment, we set up a similar configuration to the one described in Section 4.3, with the exception that one of the senders generated short flows. We captured the packets for 600 seconds and analyzed them using the *tcptrace* tool. The resulting CDFs are presented in Figure 14. Comparing the CDFs of the proposed MLCs with the trivial solution, we found that the curves are very similar. However, there were improvements of 18.7% and 14.4% in terms of the median η of ρ for the Cubic and BBR2 controllers, respectively. The absolute variations in η were 25ms and 18ms for the MLCs trained with Cubic and BBR2, respectively, which are negligible for most campus network applications.

5. Conclusion

This study demonstrated the advantages of integrating data plane devices and machine learning algorithms

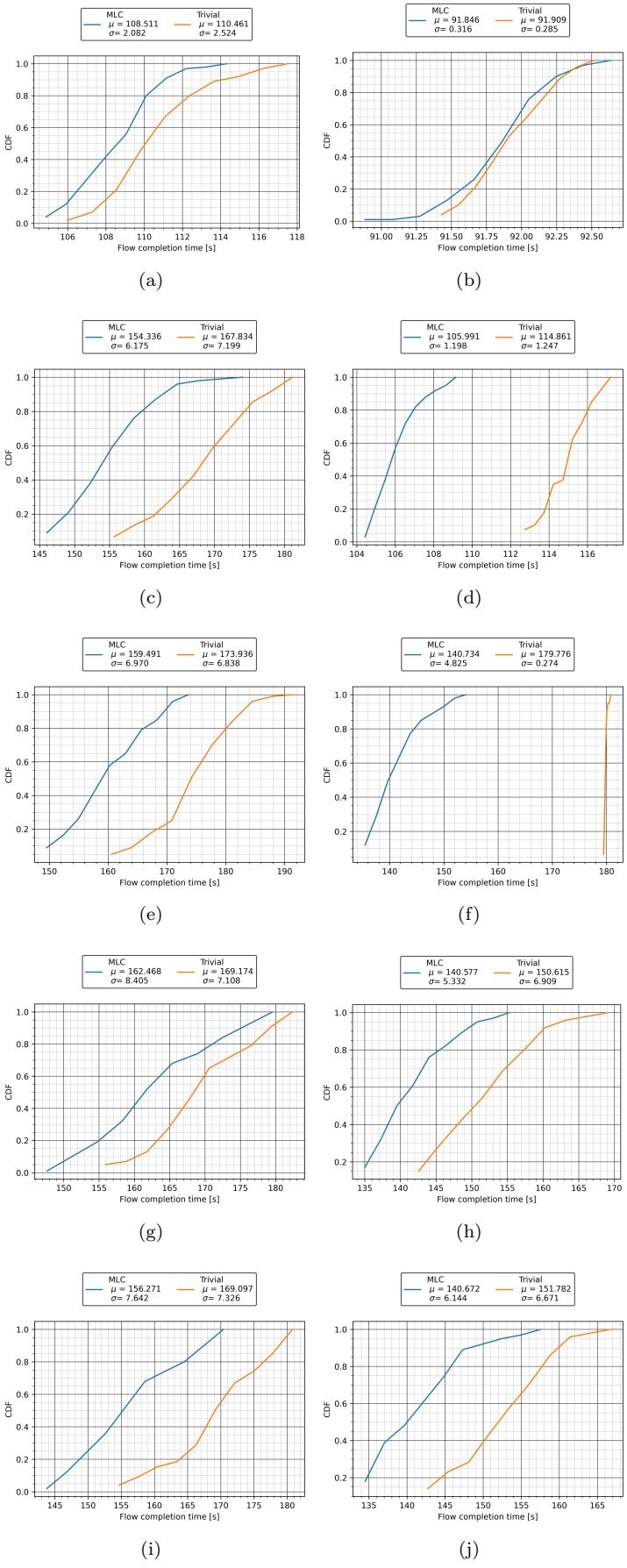


Figure 13: Flow completion time for a different number of streams on the network. (a) N_s=1, DTN-s CCA:Cubic (b) N_s=1, DTN-s CCA:BBR2 (c) N_s=10, DTN-s CCA:Cubic (d) N_s=10, DTN-s CCA:BBR2 (e) N_s=100, DTN-s CCA:Cubic (f) N_s=100, DTN-s CCA:BBR2 (g) N_s=1000, DTN-s CCA:Cubic (h) N_s=1000, DTN-s CCA:BBR2 (i) N_s=10000, DTN-s CCA:Cubic (j) N_s=10000, DTN-s CCA:BBR2.

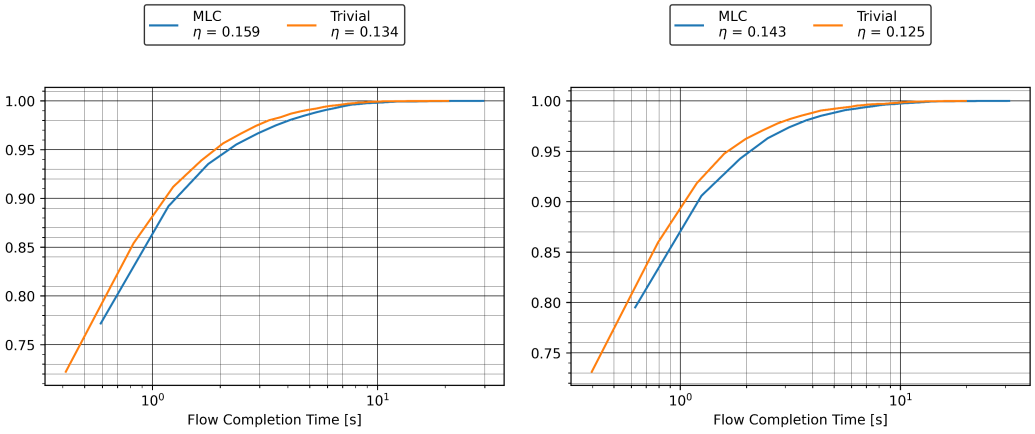


Figure 14: Flow Completion Time for short flows. (a) DTN-s CCA: Cubic (b) DTN-s CCA: BBR2. Although the proposed solution prioritizes SBD flows, the impact on the FCT of short flows is minimal and does not represent a significant degradation in performance.

to improve the rate control of Science DMZ networks. By employing an MLC to adjust traffic flows dynamically, we can enhance the FCT of data-intensive scientific flows of the Science DMZ coexisting with the short flows generated in the campus network.

The results suggest that this approach has significant potential for optimizing the efficiency of transferring SBD over non-dedicated networks. The proposed solution minimizes the FCT of long flows, with a negligible impact on the bottleneck link utilization and short flows FCT. In particular, our study highlights the importance of adaptive rate control, which can respond online to changing network conditions and traffic demand.

There are several potential future directions for this work. One avenue for further research is the development of more sophisticated machine learning algorithms at the control plane that can better adapt to changing network conditions and traffic demands. Another exploration area is deploying machine-learning-based controllers directly on the data plane, considering architecture restrictions. Furthermore, there is a need for an additional evaluation of this approach under different traffic patterns and network conditions. The above will help identify the optimal configurations and tradeoffs between performance, scalability, and overhead.

As data plane processing continues to advance and be applied in real-world scenarios, solutions such as those presented in this paper will become increasingly essential for optimizing network performance and ensuring reliable data transfer in a wide range of scientific applications.

6. Acknowledgement

In memory of Doctor Jorge E. Pezoa, who significantly contributed to the research and writing of this paper. His dedication, expertise, and passion for his field of study will forever be remembered and cherished. This work has been supported by Chilean National Agency for Research and Development (ANID) Fondecyt Regular 1220960, PCHA/Doctorado Nacional Folio 2018-21180418, Basal Fund FB0008, and U.S. National Science Foundation (NSF) (grant number 2118311).

7. Appendix

Table 7 provides a list of abbreviations while table 8 presents the notations and symbols used in this article.

Table 7: Abbreviations used in this article.

Abbreviation	Term
ACK	Acknowledgement
ALU	Arithmetic Logic Unit
AQM	Active Queue Management
ASIC	Application-specific Integrated Circuit
BBR	Bottleneck Bandwidth and Round-trip Time
BBR2	Bottleneck Bandwidth and Round-trip version 2
BDP	Bandwidth-Delay Product
CCA	Congestion Control Algorithm
CDF	Cumulative Distribution Function
CODEL	Controlled Delay
DTN	Data Transfer Node
FCT	Flow-Completion Time
FPGA	Field Programmable Gate Array
IAE	Integral Absolute Error
IETF	Internet Engineering Task Force
JSON	JavaScript Object Notation
LPM	Longest-Prefix Match
MLC	Machine Learning Controller
MRAE	Mean Relative Absolute Error
PIE	Proportional Integral Controller Enhanced
PISA	Protocol-Independent Switch Architecture
RED	Random Early Detection
RTT	Round Trip Time
SBD	Scientific Big Data
Science DMZ	Science Demilitarized Zone
SDN	Software-Defined Networks
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
VoIP	Voice over Internet Protocol
WAN	Wide Area Network

Table 8: Notations and symbols used in this article.

Symbol	Usage or Signification
η	Median value
λ	Scale parameter Weibull distribution
μ	Mean value
ρ	Link utilization
σ	Standard deviation
ϕ	Model parameters
k	Shape parameter Weibull distribution
K	Tunning parameter
N_s	Number of streams
$O(\cdot)$	Complexity of Algorithm
t	Instant time
u	Control signal
w	Windup signal
T	Sample time
R	Rate (bps)
y	Network state
Z^{-1}	Unitary delay

References

[1] J. Crichigno, E. Bou-Harb, and N. Ghani, "A Comprehensive Tutorial on Science DMZ," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 2041–2078, 2019.

[2] P. B. et. al, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[3] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends," *IEEE Access*, vol. 9, pp. 87094–87155, 2021.

[4] T. Duriez, S. L. Brunton, and B. R. Noack, *Machine learning control-taming nonlinear dynamics and turbulence*. Springer, 2017.

[5] L. Zhang, Z. Chen, X. Zhang, A. Pertzborn, and X. Jin, "Challenges and opportunities of machine learning control in building operations," *Building Simulation*, vol. 16, no. 6, p. 831 – 852, 2023.

[6] D. Rawat, M. K. Gupta, and A. Sharma, "Intelligent control of robotic manipulators: a comprehensive review," *Spatial Information Research*, vol. 31, no. 3, p. 345 – 357, 2023.

[7] J. Velino, S. Kang, and M. B. Kane, "Machine learning control for floating offshore wind turbine individual blade pitch control," *Journal of Computing in Civil Engineering*, vol. 36, no. 6, 2022.

[8] S. Galeani, S. Tarbouriech, M. Turner, and L. Zaccarian, "A tutorial on modern anti-windup design," *European Journal of Control*, vol. 15, no. 3-4, pp. 418–440, 2009.

[9] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, p. 64–74, jul 2008.

[10] N. Cardwell, Y. Cheng, S. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, and V. Jacobson, "Bbrv2: A model-based congestion control," *Presentation in the Internet Congestion Control Research Group (Iccrg) at Ietf 105 Update, Montreal, Canada*, 2019.

[11] S. Heckmuller and B. E. Wolfinger, "Analytical modeling of token bucket based load transformations," in *2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pp. 15–23, IEEE, 2008.

[12] B. Kovács, "Mathematical remarks on token bucket," in *SoftCOM 2009-17th International Conference on Software, Telecommunications & Computer Networks*, pp. 151–155, IEEE, 2009.

[13] S. Singh and R. K. Jha, "A survey on software defined networking: Architecture for next generation network," *Journal of Network and Systems Management*, vol. 25, pp. 321–374, 2017.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[15] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, p. 87–95, July 2014.

[16] Open Networking Foundation, "P4 Contributors." [Online]. Available: `https://p4.org/ecosystem/`, Accessed on 2023-12-24.

[17] N. McKeown, "Pisa: Protocol independent switch architecture, 2015," in *P4 Workshop*.

[18] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science dmz: A network design pattern for data-intensive science," in *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–10, Nov 2013.

[19] L. Smarr, C. Crittenden, T. DeFanti, J. Graham, D. Mishin, R. Moore, P. Papadopoulos, and F. Würthwein, "The pacific research platform: Making high-speed networking a reality for the scientist," in *Proceedings of the Practice and Experience on Advanced Research Computing*, pp. 1–8, 2018.

[20] W. E. Allcock, B. S. Allen, R. Ananthakrishnan, B. Blaiszik, K. Chard, R. Chard, I. Foster, L. Lacinski, M. E. Papka, and R. Wagner, "Petrel: A programmatically accessible research data service," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, pp. 1–7, 2019.

[21] W. Ahmad, B. Alam, S. Sharma, and A. Kushwaha, "Epigenomics scientific big data workflow scheduling for cancer diagnosis in health care using heterogeneous computing environment," *Brazilian Archives of Biology and Technology*, vol. 66, 2022.

[22] E. S. N. (Esnet), "Esnet fasterdata knowledge base." [Online]. Available: `https://www.es.net/`, Accessed on 2023-12-24.

[23] B. Tierney, E. Dart, E. Kissel, and E. Adhikarla, "Exploring the bbrv2 congestion control algorithm for use on data transfer nodes," 2021.

[24] I. Monga, E. Pouyoul, and C. Guok, "Software-defined networking for big-data science-architectural models from campus to the wan," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pp. 1629–1635, IEEE, 2012.

[25] SC12, "SCinet Research Sandbox." [Online]. Available: `https://sc12.supercomputing.org/content/scinet-research-sandbox.html`, Accessed on 2023-12-24.

[26] K. Jutawongcharoen, V. Varavithya, K. Lekdee, A. Chaichit, and T. Sribuddee, "The implementation of the uninet's research dmz," in *2016 International Computer Science and Engineering Conference (ICSEC)*, pp. 1–5, IEEE, 2016.

[27] S. Shah, W. Wu, Q. Lu, L. Zhang, S. Sasidharan, P. DeMar, C. Guok, J. Macauley, E. Pouyoul, J. Kim, and S.-Y. Noh, "Amoebanet: An sdn-enabled network service for big data science," *Journal of Network and Computer Applications*, vol. 119, pp. 70 – 82, 2018.

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

[29] D. A. Alwahab and S. Laki, "A simulation-based survey of active queue management algorithms," in *Proceedings of the 6th International Conference on Communications and Broadband Networking*, pp. 71–77, 2018.

[30] G. Patil, S. I. McClean, and G. Raina, "Drop tail and red queue management with small buffers: Stability and hopf bifurcation," *ICTACT Journal on Communication Technology*, vol. 02, pp. 339–344, 2011.

[31] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Rfc 8289: Controlled delay active queue management," *IETF, Jan*, 2018.

[32] S. Muhammad, T. J. Chaudhery, and Y. Noh, "Study on performance of aqm schemes over tcp variants in different network environments," *IET Communications*, vol. 15, pp. 93–111, 1 2021.

[33] 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 *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, pp. 148–155, 2013.

[34] 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 *2013 IEEE 14th international conference on high performance switching and routing (HPSR)*, pp. 148–155, IEEE, 2013.

[35] K. Ramakrishnan, S. Floyd, and D. Black, *RFC3168: The Addition of Explicit Congestion Notification (ECN) to IP*. USA: RFC Editor, 2001.

[36] M. M. Kadhum and S. Hassan, "The effect of ecn on short tcp sessions," in *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*, pp. 708–712, 2007.

[37] K. Pentikousis, H. Badr, and B. Kharmah, "On the performance gains of tcp with ecn," in *2nd European Conference on Universal Multiservice Networks. ECUMN'2001 (Cat. No.02EX563)*, pp. 82–91, 2002.

[38] N. L. Ewald, C. Kulatunga, and G. Fairhurst, "Performance impact of ecn on multimedia traffic with satellite delay," in *2009 International Workshop on Satellite and Space Communications*, pp. 120–124, 2009.

[39] J. Zhang, Z. Yao, Y. Tu, and Y. Chen, "A survey of tcp congestion control algorithm," in *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*, pp. 828–832, 2020.

[40] V. Arun, M. Alizadeh, and H. Balakrishnan, "Starvation in end-to-end congestion control," in *Proceedings of the ACM*

SIGCOMM 2022 Conference, pp. 177–192, 2022.

[41] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016.

[42] J. Crichigno, Z. Csibi, E. Bou-Harb, and N. Ghani, "Impact of segment size and parallel streams on tcp bbr," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pp. 1–5, 2018.

[43] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pp. 1–10, 2017.

[44] J. Gomez, E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "A performance evaluation of tcp bbrv2 alpha," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 309–312, 2020.

[45] N. Cardwell, Y. Cheng, S. H. Yeganeh, P. Jha, Y. Seung, K. Yang, I. Swett, V. Vasiliev, B. Wu, L. Hsiao, *et al.*, "Bbrv2: A model-based congestion control performance optimization," in *Proc. IETF 106th Meeting*, pp. 1–32, 2019.

[46] S. Zhang, W. Lei, W. Zhang, and H. Li, "An evaluation of bottleneck bandwidth and round trip time and its variants," *International Journal of Communication Systems*, vol. 34, 6 2021.

[47] K. Winstein and H. Balakrishnan, "Tcp ex machina: Computer-generated congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 123–134, 2013.

[48] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "{PCC}: Re-architecting congestion control for consistent high performance," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pp. 395–408, 2015.

[49] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "{PCC} vivace: Online-learning congestion control," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pp. 343–356, 2018.

[50] T. Meng, N. R. Schiff, P. B. Godfrey, and M. Schapira, "Pcc proteus: Scavenger transport and beyond," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pp. 615–631, 2020.

[51] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *Proceedings of the 7th International Conference on Multimedia Systems*, pp. 1–12, 2016.

[52] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pp. 329–342, 2018.

[53] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pp. 731–743, 2018.

[54] L. Jia, T. Huang, and L. Sun, "Zixia: A reinforcement learning approach via adjusted ranking reward for internet congestion control," in *ICC 2022-IEEE International Conference on Communications*, pp. 365–370, IEEE, 2022.

[55] H. Jiang, Q. Li, Y. Jiang, G. Shen, R. Sinnott, C. Tian, and M. Xu, "When machine learning meets congestion control: A survey and comparison," *Computer Networks*, vol. 192, p. 108033, 2021.

[56] J. Sun and M. Zukerman, "An adaptive neuron aqm for a stable internet," in *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet: 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007. Proceedings 6*, pp. 844–854, Springer, 2007.

[57] Q. Yan and Q. Lei, "A new active queue management algorithm based on self-adaptive fuzzy neural-network pid controller," in *2011 International Conference on Internet Technology and Applications*, pp. 1–4, IEEE, 2011.

[58] A. P. Silva, K. Obraczka, S. Burleigh, and C. M. Hirata, "Smart congestion control for delay-and disruption tolerant networks," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, IEEE, 2016.

[59] S. Masoumzadeh, G. Taghizadeh, K. Meshgi, and S. Shiry, "Deep blue: A fuzzy q-learning enhanced active queue management scheme," in *2009 International Conference on Adaptive and Intelligent Systems*, pp. 43–48, IEEE, 2009.

[60] A. P. Silva, K. Obraczka, S. Burleigh, and C. M. Hirata, "Smart congestion control for delay-and disruption tolerant networks," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, IEEE, 2016.

[61] C. Zhou, D. Di, Q. Chen, and J. Guo, "An adaptive aqm algorithm based on neuron reinforcement learning," in *2009 IEEE International Conference on Control and Automation*, pp. 1342–1346, IEEE, 2009.

[62] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford, "Pisces: A programmable, protocol-independent software switch," in *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, (New York, NY, USA), p. 525–538, Association for Computing Machinery, 2016.

[63] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on tcp enhancements using p4-programmable devices," *Computer Networks*, vol. 212, p. 109030, 2022.

[64] A. Feldmann, B. Chandrasekaran, S. Fathalli, and E. N. Weyulu, "P4-enabled network-assisted congestion feedback: A case for nacks," in *Proceedings of the 2019 Workshop on Buffer Sizing*, pp. 1–7, 2019.

[65] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 29–42, 2017.

[66] M. Kang, G. Yang, Y. Yoo, and C. Yoo, "Proactive congestion avoidance for distributed deep learning," *Sensors*, vol. 21, no. 1, p. 174, 2020.

[67] S. Shahzad, E.-S. Jung, J. Chung, and R. Kettimuthu, "Enhanced explicit congestion notification (eecn) in tcp with p4 programming," in *2020 International Conference on Green and Human Information Technology (ICGHIT)*, pp. 35–40, 2020.

[68] A. Laraba, J. François, S. R. Chowdhury, I. Chrisment, and R. Boutaba, "Mitigating tcp protocol misuse with programmable data planes," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 760–774, 2021.

[69] A. Sacco, A. Angi, F. Esposito, and G. Marchetto, "Hint: Supporting congestion control decisions with p4-driven in-band network telemetry," vol. 2023-June, p. 83 – 88, 2023.

[70] R. Kundel, J. Blendin, T. Viernickel, B. Koldehofe, and R. Steinmetz, "P4-codel: Active queue management in programmable data planes," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–4, IEEE, 2018.

[71] I. Kunze, M. Gunz, D. Saam, K. Wehrle, and J. Rüth, "Tofino+ p4: A strong compound for aqm on high-speed networks?," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 72–80, IEEE, 2021.

[72] R. Kundel, A. Rizk, J. Blendin, B. Koldehofe, R. Hark, and R. Steinmetz, "P4-codel: Experiences on programmable data plane hardware," in *ICC 2021-IEEE International Conference on Communications*, pp. 1–6, IEEE, 2021.

[73] C. Papagianni and K. De Schepper, "Pi2 for p4: An active queue management scheme for programmable data planes," in *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*, pp. 84–86, 2019.

[74] L. Toresson, "Making a packet-value based aqm on a programmable switch for resource-sharing and low latency," 2021.

[75] N. K. Sharma, A. Kaufmann, T. E. Anderson, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the power of flexible packet processing for network resource allocation.," in *NSDI*, pp. 67–82, 2017.

[76] A. Mushtaq, R. Mittal, J. McCauley, M. Alizadeh, S. Ratnasamy, and S. Shenker, "Datacenter congestion control: Identifying what is essential and making it practical," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 3, pp. 32–38, 2019.

[77] M. Menth, H. Mostafaei, D. Merling, and M. Häberle, "Implementation and evaluation of activity-based congestion management using p4 (p4-abc)," *Future Internet*, vol. 11, no. 7, p. 159, 2019.

[78] A. G. Alcoz, A. Dietmüller, and L. Vanbever, "Sp-pifo: Approximating push-in first-out behaviors using strict-priority queues.," in *NSDI*, pp. 59–76, 2020.

[79] C. Cascone, N. Bonelli, L. Bianchi, A. Capone, and B. Sansò, "Towards approximate fair bandwidth sharing via dynamic priority queuing," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 1–6, IEEE, 2017.

[80] B. Turkovic, S. Biswal, A. Vijay, A. Hüfner, and F. Kuipers, "P4qos: Qos-based packet processing with p4," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pp. 216–220, IEEE, 2021.

[81] T. V. Doan, T. Scheinert, O. Lhamo, J. A. Cabrera, F. H. P. Fitzek, and G. T. Nguyen, "Interplay between priority queues and controlled delay in programmable data planes," p. 64 – 71, 2023.

[82] C. Chen, H.-C. Fang, and M. S. Iqbal, "Qostcp: Provide consistent rate guarantees to tcp flows in software defined networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[83] S. Sengupta, H. Kim, and J. Rexford, "Continuous in-network round-trip time monitoring," in *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, (New York, NY, USA), p. 473–485, Association for Computing Machinery, 2022.

[84] E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "Dynamic router's buffer sizing using passive measurements and p4 programmable switches," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 01–06, 2021.

[85] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 253–264, 2012.

[86] J. Ali, R. H. Jhaveri, M. Alswailim, and B.-h. Roh, "Escalb: An effective slave controller allocation-based load balancing scheme for multi-domain sdn-enabled-iot networks," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 6, p. 101566, 2023.

[87] E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "Dynamic Router's Buffer Sizing using Passive Measurements and P4 Programmable Switches," in *IEEE Global Comm. Conf. GLOBECOM*, 2021.

[88] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, vol. 5, p. 2005, Citeseer, 2005.

[89] K. Chard, S. Tuecke, and I. Foster, "Globus: Recent enhancements and future plans," in *Proc. XSEDE16 Conf. Diversity, Big Data, and Science at Scale*, pp. 1–8, 2016.

[90] N. S. Nise, *Control systems engineering*. John Wiley & Sons, 2020.

[91] M. T. Hagan and H. B. Demuth, "Neural networks for control," in *Proc. ACC 1999*, vol. 3, pp. 1642–1656, IEEE, 1999.

[92] B. Kamanditya and B. Kusumoputro, "Elman recurrent neural networks based direct inverse control for quadrotor attitude and altitude control," in *2020 Int. Conf. Intelligent Eng. and Management*, pp. 39–43, 2020.

[93] B. Y. Suprapto and B. Kusumoputro, "A comparison of back propagation neural network and elman recurrent neural network algorithms on altitude control of heavy-lift hexacopter based on direct inverse control," in *2018 Int. Conf. ICECOS*, pp. 79–84, 2018.

[94] H. Alshareefi, C. Lupu, S. Olteanu, and L. Ismail, "Design and simulation of adaptive neuro-fuzzy inference system inverse controller for a coupled tank system," in *2021 10th International Conference on ENERGY and ENVIRONMENT (CIEM)*, pp. 1–5, 2021.

[95] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[96] S. Tarbouriech and M. Turner, "Anti-windup design: an overview of some recent advances and open problems," *IET control theory & applications*, vol. 3, no. 1, pp. 1–19, 2009.

[97] O. Lamrabet, N. E. Fezazi, F. E. Haoussi, and E. H. Tissir, "Using input delay approach for synthesizing an anti-windup compensator to aqm in tcp/ip networks," in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pp. 1–6, 2017.

[98] K. M. Passino, *Biomimicry for optimization, control, and automation*. Springer Science & Business Media, 2005.

[99] A. Jayachitra and R. Vinodha, "Genetic algorithm based pid controller tuning approach for continuous stirred tank reactor," *Advances in Artificial Intelligence*, vol. 2014, pp. 9–9, 2015.

[100] A. Mirzal, S. Yoshii, and M. Furukawa, "Pid parameters optimization by using genetic algorithm," *arXiv preprint arXiv:1204.0885*, 2012.

[101] J. Zhao and M. Xi, "Self-tuning of pid parameters based on adaptive genetic algorithm," in *IOP conference series: materials science and engineering*, vol. 782, p. 042028, IOP Publishing, 2020.

[102] A. F. Gad, "Pygad: An intuitive genetic algorithm python library," 2021.

[103] P. L. Consortium *et al.*, "P416 language specification," *Version*, vol. 1, no. 3, p. 8, 2018.

[104] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring TCP round-trip time in the data plane," in *Proc. Workshop Secure Programmable Net Infrastructure*, pp. 35–41, 2020.

[105] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pp. 403–415, 2015.

[106] F. Chollet *et al.*, "Keras," 2015.

[107] M. A. et. al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[108] A. Arfeen, K. Pawlikowski, D. McNickle, and A. Willig, "The role of the weibull distribution in modelling traffic in internet access and backbone core networks," *Journal of network and computer applications*, vol. 141, pp. 1–22, 2019.

[109] A. Varet and N. Larrieu, "Realistic network traffic profile generation: Theory and practice," *Computer and Information Science*, vol. 7, no. 2, pp. pp–1, 2014.