Flow-level Bandwidth Allocation on P4 Tofino Switch with In-Network DRL Inference

Muhammad Irfan[1,2], Hang Hu[1], Myung J. Lee[1], Arslan Qadeer[1], Yang G. Kim[3], Kazi Ahmed[4], and Daiki Nobayashi[5]
[1]Department of Electrical Engineering, City College of New York, New York, NY, USA, 10031
[2]School of Computing & Data Science, Wentworth Institute of Technology, Boston, MA, USA, 02115
[3]Department of Computer Engineering Technology, NYCCT, Brooklyn, NY, USA, 11201
[4]Department of Electrical & Computer Engineering Technology, NYIT, New York, NY, USA, 10023
[5]Department of Electrical and Electronic Engineering, Kyushu Institute of Technology, Fukuoka, Japan, 804-8550
Emails: {mirfan001@citymail, hhu002@citymail, mlee@ccny, aqadeer000@citymail}.cuny.edu, irfanm@wit.edu, yakim@citytech.cuny.edu, kahmed17@nyit.edu, nova@ecs.kyutech.ac.jp

***Abstract*—The Software-Defined Networking (SDN) paradigm has significantly improved network efficiency by integrating machine learning (ML) capabilities into the control plane (CP). This integration allows adaptive management of network resources in response to dynamic traffic conditions. However, the significant geographical distance between the CP and data plane (DP) causes considerable round-trip latency, measured in milliseconds, which poses a major challenge to time-sensitive traffic. To address this challenge, our research proposes a novel in-network Reinforcement Learning (RL) inference framework. This framework extends programmability from the CP to the DP, enabling timely and precise control of network resources to meet the stringent Quality of Service (QoS) requirements of mission-critical applications. The in-network RL inference is implemented using match-action tables within the Protocol Independent Switch Architecture (PISA) framework in the DP and validated through hardware deployment of protocol-independent packet processors (P4). To allocate bandwidth precisely based on QoS requirements, a P4 meter extern is used to differentiate the unique demands of individual traffic flows. Our enhanced deep-deterministic policy gradient (eDDPG)-based RL inference achieves superior performance with minimal processing overhead, reducing latency by 88.7% and jitter by 89.1% compared to systems without RL inference.**

Keywords— Programmable Data Plane, P4, Tofino Native Architecture, In-Network RL Inference, Network Resource Management

I. Introduction

The importance of a programmable data plane (DP) in the context of Ultra-Reliable Low-Latency Communication (URLLC) cannot be overstated. URLLC services demand a network architecture that can adapt dynamically to varying traffic patterns and application requirements [1]. The traditional approach of Software-Defined Networking (SDN), which primarily emphasizes control plane (CP) programmability, struggles to meet the stringent requirements of emerging URLLC services. The centralized decision-making inherent in the CP, while effective for general network management, introduces delays that are incompatible with the demands of mission-critical applications, where rapid response times and minimal latency are critical [2]. The data plane programming (DPP), on the other hand, plays a pivotal role in achieving these objectives by enabling the customization of packet processing within the network devices, enabling optimizations that are not achievable with CP-based approach. With the ability to locally define rules and logic at the device level, DPP allows for fine-grained control and quick decision-making. This distributed intelligence significantly reduces communication overhead and delays, ensuring that URLLC services receive ultra-low latency and high reliability.

A. P4 Language and PISA Architecture

Currently, programmable protocol-independent packet processors (P4) [3] have become the primary language for programming the DP, with P416 [4] as the most widely used version. P4 includes core library terms and introduces supplementary constructs called externs. Externs, defined by architecture-dependent libraries, specify elements like counters, meters, registers, and checksum functions. Systems supporting DPP and P4 can be implemented on targets like Behavioral Model version 2 (BMv2) and Application Specific Integrated Circuits (ASICs) such as Tofino. The Tofino Native Architecture (TNA) [5] abstracts the pipeline, linking P4 to the target, including the widely known Portable Switch Architecture (PSA). Fig. 1 illustrates the TNA ingress pipeline, which includes the programmable parser, match-action units (MAUs), and deparser blocks. The parser establishes a state machine that controls header extraction. It matches packet bits to typed representations. MAUs define packet processing logic with match-action tables (MATs) and other constructs. The deparser then assembles the packet structure. In the Protocol Independent Switch Architecture (PISA) framework, after parsing, the packet enters a pipeline of control flows and MATs. Actions are executed by Arithmetic Logic Units (ALUs) and must operate at line rate. MATs mappings support diverse forwarding behaviors, which the CP can quickly adapt to non-stationary environments. The forwarding process also enables shared instructions on MATs, improving memory efficiency in the DP. Thus, PISA becomes a pivotal solution, leveraging programmable P4 for efficient implementation of advanced DP functions.

B. In-Network Inference

A range of approaches, such as the use of binary neural networks (BNNs), customized hardware designs, and methods based on MATs mapping [6-9], can be employed to implement in-network reinforcement learning (RL) inference. The implementation based on BNNs requires at least 12 MATs to represent a single layer in the DP, which can grow exponentially to maintain the desired accuracy level. On the other hand, methods relying on modified hardware design can only be validated using FPGAs, representing a complex and costly approach that necessitates the redesign of existing hardware. Models like Net2Net [10] and N3IC [11] demonstrate innovative approaches to adapt NN architectures for hardware with limited computational capabilities, such as the absence of floating-point operations. These methods leverage quantization to enable deployment within switches and NICs, addressing the critical need for low-latency and

efficient data processing at the network edge. Techniques like pForest [12] and Planter [13] highlight the effective adaptation of decision trees and ensemble methods to programmable data planes. These approaches optimize memory usage and computational efficiency, facilitating the implementation of complex ML models like Random Forest and XGBoost within the constrained environment of network devices. Diverse models such as IIsy [14] and OPaL [15] indicate the field's

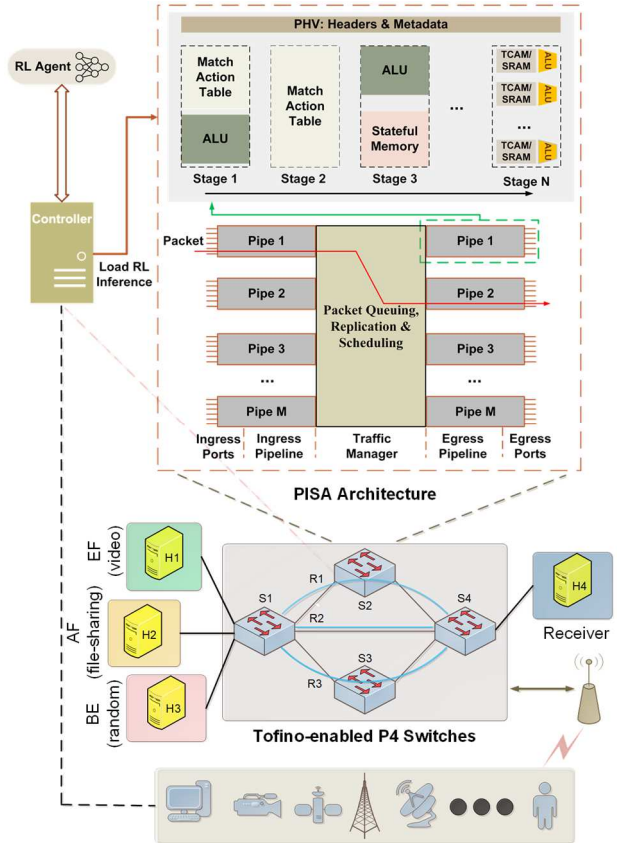

Fig. 1. The structure of in-network RL inference and PISA architecture framework.

exploration beyond conventional paradigms, embracing a variety of ML algorithms, including SVM, Naïve Bayes, and RL. These efforts illustrate the potential for multifaceted applications of ML within networks, from traffic classification to adaptive learning in response to network dynamics.

In this paper, we implement an in-network resource allocation framework using RL inference on a Tofino-enabled P4 switch. We focus on executing RL inference in the DP to efficiently allocate flow-level bandwidth at line rate, meeting user contract and QoS requirements. Our previous work [16] integrated in-network inference by the eDDPG agent with minimal processing overhead, jitter, and operational costs for bandwidth allocation on the BMv2 model. This work extends our previously proposed in-network RL inference for flow-level bandwidth allocation, now using P4 switches, advancing research in this domain. The main contributions of this paper are summarized as follows:

- Introduce a novel flow-level bandwidth allocation model that considers individual QoS requirements, enhancing resource allocation granularity.
- Deploy the in-network RL inference model on an Intel Tofino-based P4 switch, demonstrating real-world applicability and alignment with modern networking technologies.
- Enable in-network RL inference for dynamic bandwidth management by adjusting meter control parameters to meet QoS requirements in response to varying traffic patterns, ensuring continual optimization in real-world networks.
- Conduct comprehensive performance analyses by comparing RL inference with a traditional strict priority scheduler, evaluating the model's efficacy across metrics such as latency, jitter, and packet loss, providing a holistic understanding of its impact on network performance.

II. Problem Formulation

A. Traffic Model

The adoption of flow-level bandwidth allocation in SDN represents a transformative shift, providing finer control and responsiveness to the evolving demands of modern communication flows compared to traditional networking architectures. This approach enables the CP to dynamically allocate bandwidth based on individual flow requirements according to the integrated services model, ensuring QoS and optimizing network performance. Dynamic prioritization ensures time-sensitive applications receive sufficient bandwidth for proper functionality, while lower-priority traffic is deprioritized. To address scalability challenges in large networks, the differentiated services (DiffServ) model was introduced. DiffServ enables traffic classification into multiple classes, each potentially consisting of multiple flows with similar QoS requirements.

In our traffic model, bandwidth allocation is managed to meet diverse traffic requirements. We consider three types of QoS traffic flows in the DiffServ model: bandwidth-intensive, delay-sensitive expedited forwarding (EF) for video flows; bandwidth-intensive, delay-tolerant assured forwarding (AF) for file transfers; and default forwarding (DF), or best-effort (BE), for random traffic to introduce link congestion under excessive traffic conditions. To support QoS for each class, the SDN controller uses forwarding rules to distinguish traffic based on TCP/UDP port numbers or differentiated services code point (DSCP) codes, assigning a weight w_f to each flow f, reflecting its QoS requirements. This traffic management strategy ensures effective handling of diverse traffic types, offering adaptability to the varying demands of applications and network conditions.

B. Bandwidth Model and P4 Meter Extern

In pursuit of a comprehensive model for bandwidth allocation applicable to diverse SDN-based systems, we build upon the previously proposed framework [17]. Our network encompasses N core network nodes, denoted as programmable network switches {1, 2, 3, ..., N}. Each node, represented by m hosts, p ports {1, 2, 3, ..., P}, with f flows {1, 2, 3, ..., F} per switch port, reflecting the dynamic nature of traffic. The capacity of a switch port is denoted as B_p (e.g., 10 Mbps). The traffic flow f represents a TCP/UDP port from a host. Multiple traffic flows with the same traffic characteristics can be grouped into a traffic class cl = {1, 2, 3, …, CL}. The practicality of the switch is evaluated on a per-switch port basis, ensuring a fine-grained and adaptable approach to bandwidth allocation. At time t, the total usage of the switch port capacity is the sum of the occupied resources by the traffics of all classes in Eq. (1).

$$U_p^n(t) = \frac{\sum_{cl=1}^{CL}\sum_{f=1}^{F} b_f^p(t)}{B_p} \quad (1)$$

where the total usage of the port capacity U_p^n at time t , b_f^p represents the number of bandwidth units that are allocated to the flow f on port p at time t.

Our approach uses RL inference to dynamically assign a P4 meter extern for bandwidth allocation to traffic flows. The meter applies the two-rate, three-color marker (trTCM) algorithm [18] as shown in Fig. 2, categorizing packets into green, yellow, or red based on Peak Information Rate (PIR) and Committed Information Rate (CIR), representing the maximum and guaranteed bit rates, respectively. The algorithm uses two token buckets, P and C, with rates PIR and CIR, and capacities Peak Burst Size (PBS) and Committed Burst Size (CBS). Packets are classified by token depletion and assigned colors based on three conditions: (1) both P and C have enough tokens, (2) only P has enough tokens, and (3) neither has enough tokens. Based on these classifications, specific action policies such as transmit or drop can be enforced to packet processing.

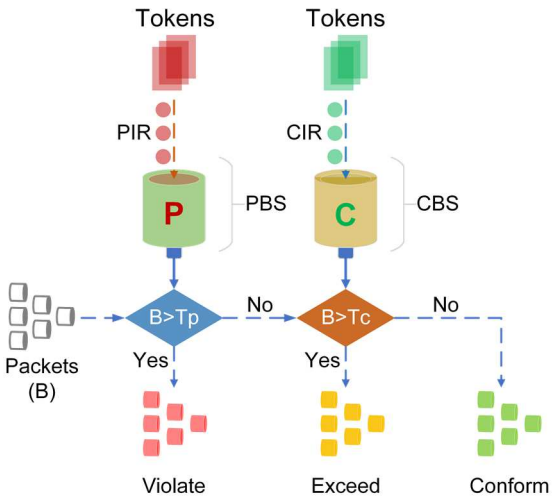

Fig. 2. Meter as a 'rate-limiter' following trTCM

III. Proposed Framework for RL-based In-Network Traffic Management

This framework addresses RL inference and bandwidth allocation in the data plane (DP). Fig. 1 illustrates the SDN controller acting as an intermediary between the eDDPG agent and the DP. The controller adjusts meters according to the eDDPG strategy and converts RL decisions into forwarding rules for the MATs with hardcoded configuration. This control mechanism enables the eDDPG agent to interact with the network environment and derive an optimal bandwidth allocation scheme.

A. eDDPG Agent

Temporal feature learning attentional network (TFLAN) [16] is introduced, which accounts for temporal variations to enhance state representation and improve function approximation in resource allocation systems. Our TFLAN-based eDDPG agent consists of three parts: (1) A Conv1D residual block structure to learn the correlations among local features of each input state; (2) A GRU layer to learn temporal dependencies; and (3) An attention mechanism to capture meaningful information at certain moments. Additionally, we proposed a heuristic-based priority experience replay (hPER) to reduce variance and break undesired temporal correlations in the training samples. The hPER stores all experiences (s_t, a_t, r_t, s_{t+1}) and ranks them using an efficient heuristic function. This approach guides the agent towards useful regions, improving the convergence rate and stability during training.

*1) **States**:* In the DP, the system's state is characterized at the flow level, which is articulated as: $s_t = \{w_f(t), R_f(t), D_f(t)\}$, where w_f denotes the time-variable weight of the flow, R_f stands for the currently assigned data rate, and D_f signifies the demand. Initially, flows are allocated a default data rate based on their weights, subject to subsequent adjustments in response to their demand. The demand $D_f(t)$ for a flow f at time t is calculating the average demand rate over the interval $(t-\tau)$. In operational terms, the DP determines this demand rate using Tofino P4 registers, incrementing the byte count for each packet received (excluding the header) in the corresponding counter. This process involves maintaining stateful data on counters and timestamps through register utilization.

*2) **Actions**:* From the sequence of observations, the devised eDDPG agent is trained to identify an optimal bandwidth allocation strategy. At every state s_t, the system determines the bandwidth allocation necessary to minimize system expenses while adhering to the QoS requirements. Within the DP, the action space, denoted as $a_t = \{b_f\}, (f \in [1, F])$, where parameter b represents bandwidth allocation at time t. This allocated bandwidth b_f corresponds to a meter with the appropriate CIR/PIR value.

*3) **Rewards**:* The objective of the eDDPG agent is to lower the overall system cost through a sequence of actions across all states. In the model we propose, this reward is determined by taking the inverse of the aggregate of the costs and penalties associated with resource acquisition in the system. The cost is expressed as: $C_f = b_f \times C$, where C is the system cost per unit of bandwidth, which is adjustable based on the environment. The penalty for a flow refers to the additional cost caused by inaccurate resource allocation and is defined as: $P_f = w_f \times \frac{D_f}{b_f}$. The penalty equation helps the eDDPG agent learn a precise resource allocation policy by minimizing the penalty ratio. Therefore, the overall reward is given as follows:

$$r_t(s_t, a_t) = \frac{1}{C_f(s_t, a_t) + P_f(s_t, a_t)} \quad (2)$$

Rewards are calculated individually for each flow. This approach is advantageous as it acknowledges the distinct QoS requirements across different flows, enabling the agent to formulate a more effective resource allocation strategy. The system's goal is to enhance the cumulative reward over the long term by reducing the overall system cost during bandwidth allocation, considering the diverse QoS needs and limited resources. This optimization challenge can be articulated as follows:

$$maximize \sum_{t=1}^{T} r_t(s_t, a_t) \quad (3)$$

s.t.

$$U_p^n(t) \le 1, \forall t \in T, \forall p \in P, \forall n \in N \quad (3a)$$

$$1 \le b_f \le b_{f_{max}}, \forall f \in F \quad (3b)$$

$$\delta_f(t) = \left|D_f(t) - R_f(t)\right| < \theta_f, \forall f \in F \quad (3c)$$

where constraint (3a) identifies the total usage of the port capacity at time t, which is the sum of the occupied resources by all the classes carrying multiple flows and ensures that the utilization of ports on all switches does not surpass the maximum available capacity (B_p) at any moment. Constraint (3b) ensures that the bandwidth allocated to flow f stays between 1 and its maximum allowable units. Constraint (3c) serves three objectives: first, it ensures the difference between the demand rate and the assigned rate remains within the threshold θ_f, aligning with QoS requirements. Second, it facilitates resource recovery when demand falls below the current rate. Lastly, it helps reduce per-packet processing time in the DP by removing unnecessary inferences.

B. RL Inference through MATs

Fig. 3 presents our proposed framework leveraging the PISA pipeline. Upon packet arrival, the programmable parser extracts relevant flow information. A hash computation on the 5-tuple details generates an index value for the registers, storing the flow information. For a new flow, the packet is duplicated and sent to the controller for rule installation to facilitate bandwidth allocation. For an existing flow, the demand is calculated. Following pruning, if the δ_f value exceeds the predetermined threshold θ_f, the system formulates the state (i.e., $\{w_f, R_f, D_f\}$), resulting in an action estimate. Based on the estimated bandwidth b_f from the eDDPG agent's hardcoded configuration, the process applies a MAT in the pipeline to match the 3-component entry $\{w_f, R_f, D_f\}$, selecting a meter with the appropriate CIR/PIR value, to assign bandwidth at the line rate. For example, if the b_f value is 3 then CIR/PIR will setup as 3Mbps on Tofino switch to control the traffic rate. This MAT mapping strategy enables RL inference in programmable network devices, avoiding computationally intensive calculations. Grouping multiple related flows with the same weight, based on their significance, conserves memory and table entries, enhancing overall operational efficiency.

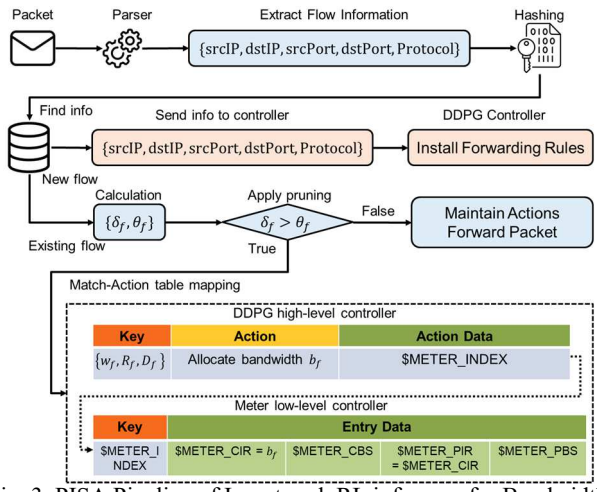

Fig. 3. PISA Pipeline of In-network RL inference for Bandwidth Allocation.

IV. Experimental Results

A. TFLAN-based eDDPG Convergence

To compare the performance of convergence, two alternative agents are considered alongside our TFLAN-based DDPG agent:

TFLAN-Uniform: This variant employs the proposed TFLAN-based architecture for both the actor and critic networks, with the distinction that it samples transitions randomly from the replay buffer for training purposes.

DDPG-NN-Uniform: An established DDPG agent [19] that incorporates two fully connected layers within both actor and critic networks, alongside a uniform sampling strategy for the replay buffer.

In all examined agents, the learning rates for the actor and critic networks are established at 0.0001 and 0.001, respectively, with a discount factor γ set to 0.99. The Adam optimizer is employed to minimize the loss function throughout the training phase, which encompasses 1000 episodes, each containing 1000 learning steps. Fig. 4 demonstrates the convergence trajectories for the three DDPG-based agents, where the TFLAN-enhanced methodologies exhibit fewer fluctuations and attain superior episodic rewards in comparison to the conventional DDPG-NN. Our methodology surpasses the alternative agents regarding convergence speed, higher reward, and training stability, attributable to our innovative TFLAN architecture and the hPER-enhanced replay buffer. For more details, refer to our earlier work [16].

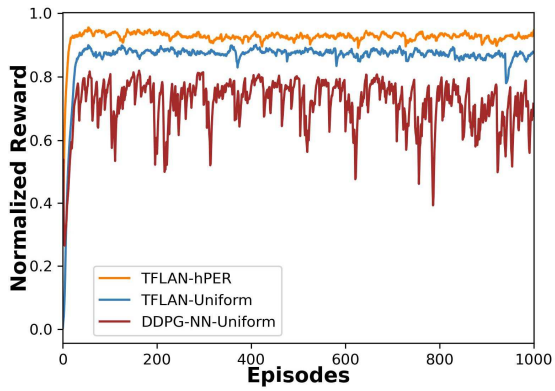

Fig. 4. Convergence comparisons of three agents.

B. Tofino Pipeline Processing Time

To assess the processing time associated with in-network RL inference, we measure the per-packet processing time in the PISA pipeline of the Tofino chip by applying Packet Test Framework (PTF). PTF is automated testing of the data plane under the control of a selected API. Tofino provides the per-packet timestamp information needed to create seven timestamps. The timestamp information can be accessed by the P4 program via intrinsic metadata fields. As shown in Fig. 5 (a), Packets are transmitted and received through packet processing either via Pipe 0 through CPU Ethernet Ports 64 and 65 or via Pipe 1 through CPU PCIe Port 192. The introduction of RL inference in the DP across various packet sizes (e.g., 100, 500, 1000, 1500 bytes) is depicted in Fig. 5 (b), compared to scenarios without in-network RL inference. The observation shows that when packet size is small, the additional overhead of RL inference through processing P4 header is obvious. As the packet size increases, this additional overhead becomes progressively less noticeable.

C. Latency Analysis

As shown in Fig. 1, the experimental network topology comprises four interconnected P4 switches utilizing IP forwarding logic. By marking all flows conformant based on the metering strategy, the QoS of each class can be maintained. Bandwidth resource allocation is enhanced with the integration of eDDPG, dynamically optimizing resource utilization based on varying importance levels of traffic sources and adapting to evolving network conditions. Switch S1 accommodates hosts H1 and H2 for generating EF based video flow, AF based file-sharing flow by setting the respective meter extern. On P4 switch, 6 Mbps is allocated to EF with PIR and CIR both set to 6000 Kbits/s, allowing only green packets and ensuring low latency, minimal jitter, and low packet loss rates. Similarly, 3 Mbps with PIR and CIR both set to 3000 Kbits/s is assigned to the AF based traffic flow. Hosts H3 is also connected to S1, generating a BE random traffic at variable rates between 1Mbps to 5Mbps to induce different levels of congestion in the network links, thereby emulating real-world network congestion scenarios. The destination H4, connected to switch S4, serves as the receiver for evaluating traffic dynamics. In the experimental analysis, the end-to-end (E2E) latency varies by route, with the direct path (R2) from S1 to S4 having the lowest latency (0.42ms) due to fewer hops, while indirect routes (R1, R3) increase delay (0.55ms). The round-trip time (RTT) to the SDN controller is 0.176ms, and frequent communication can further raise latency, especially if the controller is remote. The data plane (DP) reduces latency, ensuring faster packet delivery, which is crucial for critical applications.

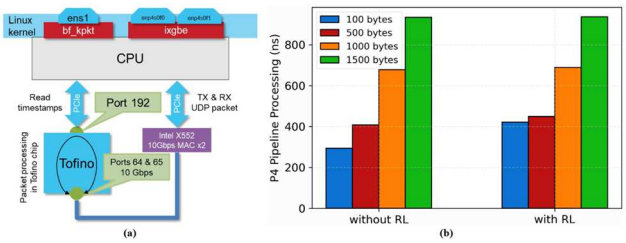

Fig. 5. Internal processing. (a) Internal pipeline processing architecture. (b) Processing time.

D. System Performance Evaluation

The system performance evaluation tests the performance of using in-network RL inference and without RL inference, i.e., traditional strict priority queue scheduling, for bandwidth allocation based on real-time network conditions. In our evaluation, a network is set up with P4 switches with a link capacity limited to 10 Mbps as proof-of-concept experiment, while the maximum full link capacity of P4 switches can go up to 100 Gbps. The above configuration is intentionally chosen to evaluate performance under constrained bandwidth scenarios.

Fig. 6 shows a without RL strict-priority scheduling that categorizes packets into three queues: EF (high priority), AF (medium priority), and BE (low priority). Packets are queued and forwarded based on their priority, with EF packets given the highest priority pre-emptively. Though this approach ensures prompt delivery of critical data but lacks adaptability, as it relies on static priorities that do not adjust to real-time network conditions. RL control is implemented by linking high-level eDDPG inference to low-level meter control for data-rate adjustment to dynamically allocate bandwidth, giving the arriving packet two options: transmit or drop. Referring to the experimental network topology, the EF, AF, BE traffic applied to input queue of the P4 switch (S1). Performance metrics, such as latency, jitter, and packet loss, are evaluated for the proposed RL inference approach.

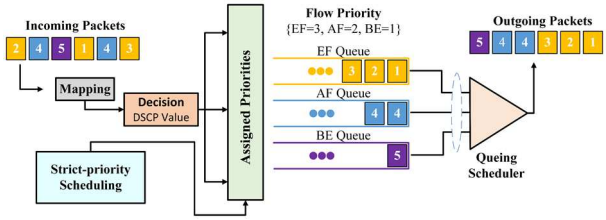

Fig. 6. Strict-priority scheduling implementation on P4 Tofino switch without RL inference.

Latency: First, we observe the latency performance of EF and AF traffic in the presence of BE random traffic. Fig. 7 (a) demonstrates that with RL inference, both EF and AF flows achieve the target average delay of less than 1ms. Considering a maximum BE traffic of 5 Mbps and applying Chebyshev's bound (e.g., $P[|Y-\mu_Y| \geq c] \leq \frac{Var[Y]}{c^2}$), the probability that the EF/AF flow's latency deviates from its mean by more than the target of 1ms is at most 0.6%/6%. The extremely low probability highlights the EF flow's exceptional consistency and reliability, making it particularly well-suited for critical applications where maintaining strict latency requirements is crucial. In our experiment, strict priority scheduling suffers from larger latency beyond 1ms. EF traffic maintains stable latency, while the latency of AF gradually increased. Overall, our proposed RL inference contributes to the reduction of latency of EF/AF flows by 90.7%/86.7% on average.

Jitter: In the subsequent evaluation shown in Fig. 7(b), we compare jitter with and without RL inference for EF and AF traffic over incremental BE random traffic. The results of this experiment demonstrate that RL inference effectively reduces jitter for both AF and EF flows, especially under higher BE traffic conditions, compared to traditional strict priority queue scheduling. These queue scheduling algorithms are typically implemented in configurable ASIC logic, enabling scheduling decisions at billions of packets per second on high-speed networks. Our programmable RL inference contributes to the reduction of jitter of EF/AF flows by 90.3%/88.0% on average through the allocation of bandwidth at the line rate. This dynamic allocation process also eliminates delays and variability associated with the frequent communication overhead from DP to CP. Consequently, our method facilitates swift and precise control over network traffic, thereby mitigating the impact of jitter on time-sensitive applications.

Packet Loss: The packet loss evaluations depicted in Fig. 7(c) show the relationship between packet loss rate and increasing BE random traffic with and without RL inference. In our test, EF and AF flows with RL inference exhibit a small packet loss rate (<1%) due to inherent quantization error associated with eDDPG hardcoded configurations. Meanwhile, the packet loss of queue-based scheduling is 0% when the EF and AF traffic flows do not exceed the port capacity, with the order of priority being EF, AF, and BE. Based on the available configuration for the real ASIC Tofino, a queue size of 200 cells, with each cell accommodating 80 bytes (~10 packets), is used in our

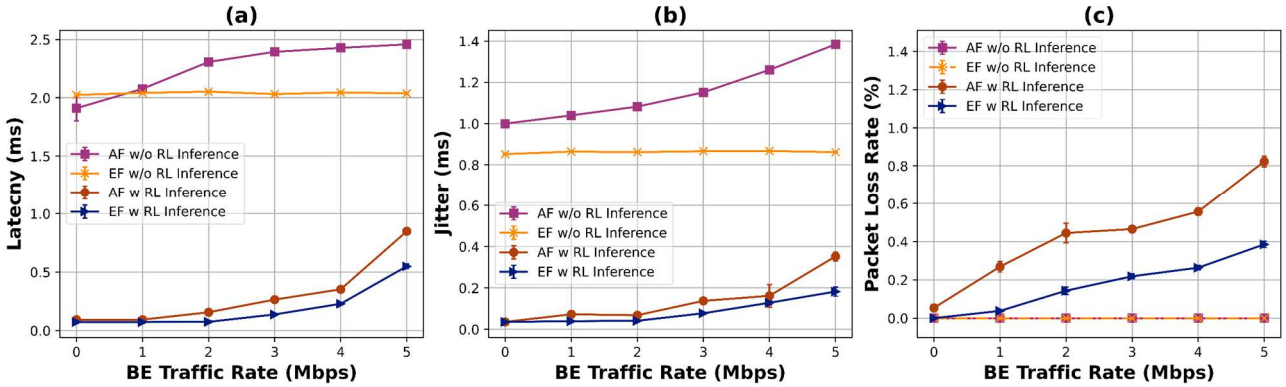

Fig. 7. System performance evaluation with and without RL inference over incremental BE random traffic. (a) Latency distribution of each flow. (b) Jitter comparison of each flow. (c) Packet loss rate of each flow.

experiment. Generally, the queue mechanism can buffer some arrived packets, reducing packet loss and handling traffic bursts, but may increase latency and jitter.

V. Conclusion

We introduced a novel in-network RL inference framework tailored to optimize bandwidth allocation at the line rate, employing advanced P4 metering and match-action table mapping techniques. Our method achieves fine precision in network control, facilitating demand-based bandwidth allocation to individual flows with negligible processing overhead. This innovative approach concurrently achieves a remarkable reduction in latency and jitter by up to 88.7% and 89.1% respectively. The efficacy of our framework has been rigorously validated through implementation on a Tofino-based P4 switch, advancing our previous research done on BMv2. This proposed framework not only solidifies its standing as an innovative solution but also advances the paradigm of efficient network resource allocation.

On the other hand, the limitation of the current version of RL inference is relying on trained eDDPG hardcoded configurations instead of online retraining. Our future work will focus on addressing this resource allocation challenges and deploying an efficient in-network RL inference suitable to NSF COSMOS testbed [20].

Acknowledgment

This work is supported by NSF PAWR COSMOS (#1827923) and NSF IRNC COSMIC (#2029295) grants.

References

[1] T. Dagiuklas, "The journey from 5G towards 6G," 2023 8th International Symposium on Electrical and Electronics Engineering (ISEEE), Galati, Romania, 2023, pp. 14-18.

[2] A. Liatifis et al., "Advancing SDN from OpenFlow to P4: A Survey," 2023 ACM Comput. Surv. 55, 9, Article 186 (September 2023), 37 pages.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, P4: Programming protocol-independent packet processors, SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 87–95.

[4] ONF/P4.org, "P4_16 Language Specification version 1.2.2," The P4 Language Consortium, Technical Report, 2021. Available: https://p4.org/p4-spec/docs/P4-16-v1.2.2.html. [Accessed: Aug. 7, 2024].

[5] INTEL, P416 Intel® Tofino™ Native Architecture – Public Version, 631348–0001, INTEL, 2021, Application Note – Public Version 2. Available at https://github.com/barefootnetworks/Open-Tofino. [Accessed: Aug. 7, 2024].

[6] [11] C. Busse-Grawitz et al., "pForest: In-Network Inference With Random Forests," 2019, doi: arxiv.org/abs/1909.05680.

[7] [12] G. Siracusano et al., "Running neural networks on the NIC," arXiv:2009.02353, 2020.

[8] [13] J. Luo et al., "Binary Neural Network with P4 on Programmable Data Plane," 2022 18th International Conference on Mobility, Sensing and Networking (MSN), Guangzhou, China, 2022, pp. 960-965, doi: 10.1109/MSN57253.2022.00155.

[9] [14] T. Swamy et al., "Taurus: a data plane architecture for per-packet ML," In Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '2022). Association for Computing Machinery, New York, NY, USA, 1099–1114, doi: 10.1145/3503222.3507726.

[10] Giuseppe Siracusano and Roberto Bifulco. 2018. In-network neural networks. arXiv preprint arXiv:1801.05731 (2018).

[11] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Gianni Antichi, Paolo Costa, Hamed Haddadi, and Roberto Bifulco. 2022. Re-architecting traffic analysis with neural network interface cards. In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22). USENIX Association, Renton, WA, 513–533.

[12] Coralie Busse-Grawitz, Roland Meier, Alexander Dietmüller, Tobias Bühler, and Laurent Vanbever. 2019. pforest: In-network inference with random forests. arXiv preprint arXiv:1909.05680 (2019).

[13] Changgang Zheng and Noa Zilberman. 2021. Planter: Seeding Trees within Switches. Association for Computing Machinery, New York, NY, 12–14.

[14] Changgang Zheng, Zhaoqi Xiong, Thanh T. Bui, Siim Kaupmees, Riyad Bensoussane, Antoine Bernabeu, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2022. IIsy: Practical in-network classification. arXiv preprint arXiv:2205.08243 (2022).

[15] Kyle A. Simpson and Dimitrios P. Pezaros. 2022. Revisiting the classics: Online RL in the programmable dataplane. In 2022 IEEE/IFIP Network Operations and Management Symposium (NOMS'22). 1–10.

[16] A. Qadeer, M. J. Lee, D. Nobayashi, "P4-based In-Network RL inference for Efficient Flow-Level Bandwidth Allocation", IEEE GLOBECOM, December 2023.

[17] A. Qadeer, M. J. Lee., "Flow-Level Dynamic Bandwidth Allocation in SDN-Enabled Edge Cloud using Heuristic Reinforcement Learning," 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, 2021, pp. 1-10.

[18] J. Heinanen and R. Guerin, "A two rate three color marker," IETF, RFC 2698, 1999.

[19] A. Qadeer, M. J. Lee., "Deep-Deterministic Policy Gradient Based Multi-Resource Allocation in Edge-Cloud System: A Distributed Approach," in IEEE Access, vol. 11, pp. 20381-20398, 2023.

[20] Chen T, Maddala P, Skrimponis P, Kolodziejski J, Adhikari A, Hu H, Gao Z, Paidimarri A, Valdes-Garcia A, Lee M, Rangan S. Open-access millimeter-wave software-defined radios in the PAWR COSMOS testbed: Design, deployment, and experimentation. Computer Networks. 2023 Oct 1; 234:109922.