Near Real-Time Autonomous Multi-Flow Routing with Dynamic Optical Bypass Management

S. Barzegar[1], H. Shakespear-Miles[1], F. Alhamed[2], F. Paolucci[3], P. González[1], M. Ruiz[1], and L. Velasco[1*]
Optical Communications Group - Universitat Politècnica de Catalunya. Barcelona, Spain
[2] *Scuola Superiore Sant'Anna (SSSA), Pisa, Italy* [3] *CNIT, Pisa, Italy e-mail: luis.velasco@upc.edu*

***Abstract*—Distributed autonomous flow routing is enabled by using multiple coordinated models and in-band network telemetry. A load manager organizes the operation of flow agents based on Deep Reinforcement Learning for optical bypass management. Delay constraints guarantees with optical bypass utilization is showcased.**

Keywords—Autonomous Routing, Optical Bypass

I. INTRODUCTION

Typically, solutions for autonomous network operation based on the application of Machine Learning (ML) have been proposed running in a centralized element, like the Software Defined Controller (SDN), to greatly reduce operational costs. However, when the target of automation is on decision problems that, like flow routing, need to be solved near real-time, the main challenge is on how to optimize resource utilization, while ensuring Quality of Service (QoS), e.g., keeping the end-to-end (e2e) delay under a maximum. In particular, the delay that packets experience along a route from ingress to egress includes three main components: *queuing* delay, *transmission* delay and *processing* delay. Transmission and processing delays are mostly predictable, while queuing delay depends on the instantaneous traffic load (i.e., burst of packets arriving at an output interface) and it is difficult to predict. Overprovisioning packet link capacity can minimize queuing delay by reducing the load of output interfaces, but it also increases the cost.

In our previous work [1], we proposed a distributed autonomous flow routing based on Deep Reinforcement Learning (DRL) agents running in the packet nodes. We assumed splitable traffic flows, i.e., traffic flows that consist of a large number of sub-flows, which can be routed independently. The system autonomously routes packet flows entering in the node considering the measured e2e packet delay, but without previous knowledge of traffic characteristics, which liberates the SDN controller from near-real time operations. In this paper, we massively relay on in-band network telemetry (INT) techniques [2] to coordinate the operation of several flows, specifically in the activation and deactivation of an optical bypass connecting two packet nodes. We also exploit the capabilities of a *Machine Learning Function Orchestrator* (MLFO) [3] to deploy and supervise the *flow agents*, as well as training ML models for the agents.

II. AUTONOMOUS MULTI-FLOW ROUTING OPERATION

The proposed autonomous flow routing procedure is presented in Fig. 1. For illustrative purposes, Fig. 1a shows a small 5-node network, a set of flows F with common ingress and egress nodes R1 and R5 (in general, R1 and R5 do not need to be in the border of the network), as well as other flows (R1-R3 and R2-R5) on the network (*background traffic*). We assume that every individual flow $f_i \in F$ has a particular maximum e2e delay that needs to be satisfied. At the control plane, an SDN controller is in charge of the network; specifically, during flow provisioning, the SDN controller computes a set of allowable routes $P(f_i)$ for every f_i, i.e., those that can potentially meet the required QoS in terms of predictable delay components only. In the example, for every f_i, the SDN controller computes two routes through the packet network: *i*) route R1-R2-R5 (*p1*); and *ii*) R1-R3-R4-R5 (*p2*), which can be used at any time. In addition, R1 and R5 can be connected on-demand using an *optical bypass*, i.e., a link in the packet network that is supported by an optical connection, which can be used if other routes get congested. The SDN controller gives different cost to each route and penalizes the optical bypass with a high cost.

Once the flows are established, the SDN controller gives up a *multi-flow agent*, running on top of ingress node R1, the near real-time control of the flows. Individual *flow agents* in the *multi-flow agent* make decisions dynamically on the amount of flow traffic for the flow under control to be sent through each of the allowable routes (i.e., output interfaces in R1) (*routing actions*). The *multi-flow agent* controls also the activation and deactivation of the optical bypass. Routing actions need to consider traffic conditions that result in delays below the maximum for the flow, which is a difficult task in dynamic traffic scenarios. In view of that, the multi-flow agent implements *flow agents* that are deployed and supervised by a MLFO. Flow agents are continuously fed with telemetry data provided by a *telemetry agent* deployed at the egress packet node, which measures (per-flow and aggregated) traffic and e2e delay for each of the allowed routes. Armed with such telemetry data, intelligent and accurate actions can be performed to meet complex operational objectives like ensuring per-flow maximum e2e delay with overall minimum routing cost. Note that the QoS of each flow f_i depends on its input traffic and routing actions, as well as on that of other flows in the network that use common network resources, e.g. flows R1-R3 and R2-R5 in Fig. 1a.

Fig. 1b details the workflow of autonomous multi-flow routing operation using a set of available routes, where every flow agent takes routing actions independently. The destination node R5 collects INT data and reports those records to the collector in the telemetry agent (1 in Fig. 1b). Then, per-flow

978-3-903176-62-1

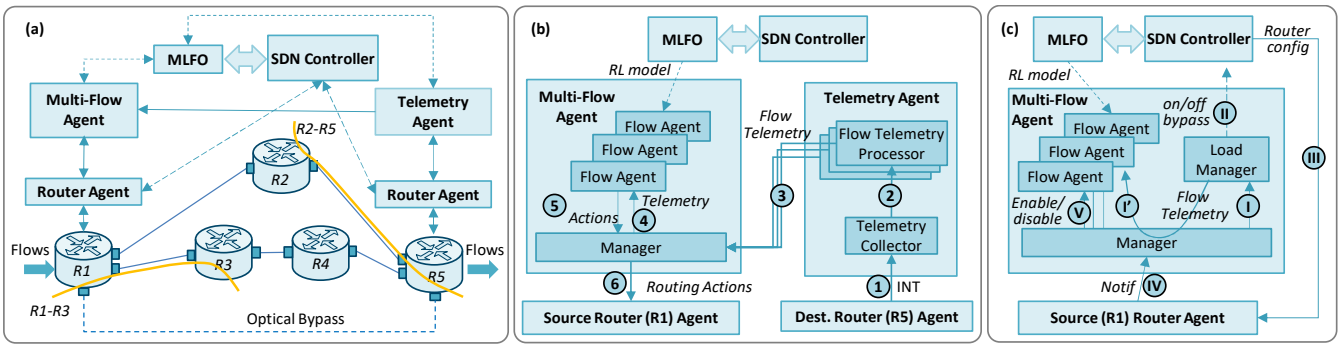

Fig. 1: Autonomous multi-flow routing (a). Independent flow control (b) and coordinated bypass utilization (c).

telemetry data is managed by individual processors (2) and sent to the multi-flow agent manager in the ingress (3). The manager sends telemetry data, together with other parameters to each of the flow agents (4). The flow agents compute the routing actions to be next performed (5) and finally, the manager forwards the actions to the router agent that translates them into specific configurations in the packet node (e.g., routing table updates) (6). In this approach, flow agents might require using the optical bypass, and based on the actions received the manager can realize this and activates it; once activated flow traffic can be routed through that interface. However, bypass activation takes time, which creates problems for the flow agent that realizes that requested actions were not implemented. In addition, once the bypass is activated, it could be used by other flows without any additional cost; then some coordination among the flow agents is needed. Our proposal is overviewed in Fig. 1c, where a *load manager* is in charge of the bypass operation; it receives collected telemetry data from the manager (I in Fig. 1c) and decides whether the bypass would be needed by analyzing the load of the routes together with the aggregated total traffic and the QoS of the flows. In the case of bypass activation, the load manager requests the SDN controller to set-up the optical connection and configure the packet nodes (II-III). Once a confirmation is received from the underlying router agent (IV), the flow agents are informed that a new route is available (V). In the case that the optical bypass can be deactivated, it first asks the flow agents to disable that route (I'), which might need to take the corresponding routing actions. Note that some of the flow agents might require new models to properly operate with a different number of routes; in that case, a request to the MLFO can be sent so as a new model is provided.

III. Models and Algorithms

This section describes the key models and algorithms involved in the proposed autonomous multi-flow routing operation, specifically during: *i*) *flow provisioning phase*, where the SDN controller finds resources that will support the packet flow; *ii*) *near-real time operation*, managed by flow agents that ensure the committed QoS, and *iii*) *load manager operation* to trigger activation/de-activation of optical bypass. Upon the reception of a new flow provisioning request, *req*, the SDN controller runs Algorithm 1 to compute the set of allowable routes P for the flow. Algorithm 1 receives the current network state, summarized in graph G, and the flow

Algorithm 1. Route computation algorithm

INPUT: $G=\{V,E\}$, $req=<s, t, x_{max}, d_{max}, p_{max}>$
OUTPUT: P

1: $G_{aux} \leftarrow G$
2: **for each** $e \in G_{aux}$.E **do**
3: **if** e.cap $< x_{max}$ **then** G_{aux}.E.pop(e)
4: $P \leftarrow k\text{-}SP(G_{aux}, s, t)$
5: **for each** $p \in P$ **do**
6: $p.\text{d}_{\min} \leftarrow transmDelay(p) + q_{min}$
7: **if** $p.\text{d}_{\min} > d_{max}$ **then** P.pop(p)
8: sort(P, <'d_{min}','*similarity*'>, ASC)
9: **return** $P[1..p_{max}]$

request including ingress s and egress t nodes, maximum traffic x_{max} and delay d_{max} to be guaranteed, and maximum number of allowed routes p_{max}. The algorithm discards those links with residual capacity below x_{max} and uses the k-shortest path algorithm compute routes on the resulting graph. The minimum expected delay d_{min} is computed and used to discard those routes that cannot meet d_{max}. Finally, the remaining routes are sorted by multiple criteria, such as their cost and capacity.

During flow provisioning, the MLFO initializes the flow routing agent with a pre-trained DRL model [4]. Flow routing agents include: *i*) a *DRL agent* in charge of learning the best actions to be taken based on the current routing state and the received reward; and *ii*) an *environment* that computes the state and a reward for the action taken. During near-real time operation, flow routing agents compute the state, the reward, and the actions periodically (e.g., 1 sec.). Each action $a(t)$ is a $|P|$-dimensional space, where every component specifies the fraction of input traffic to be routed through route p. The reward function $r(t)$ penalizes those actions that resulted into poor QoS or increased cost (each route has an associated cost). In parallel, the load manager uses the received telemetry to compute the combined load of the routes currently enabled, e.g., *p1* and *p2* in the example in Fig. 1a. A predictor is used extrapolate the overall maximum load of the routes based on the last observations, where the prediction target window is set to support optical bypass activation time. Based on the predicted maximum load, the bypass is activated when a threshold (*thr_up*) is exceeded. When the load reduces, an estimation of the load if the optical bypass is deactivated is compared to another threshold (*thr_down*) to decide if the resulting set of allowed paths would provide the required QoS for the flows. Both thresholds are defined based on the required QoS of the flows.

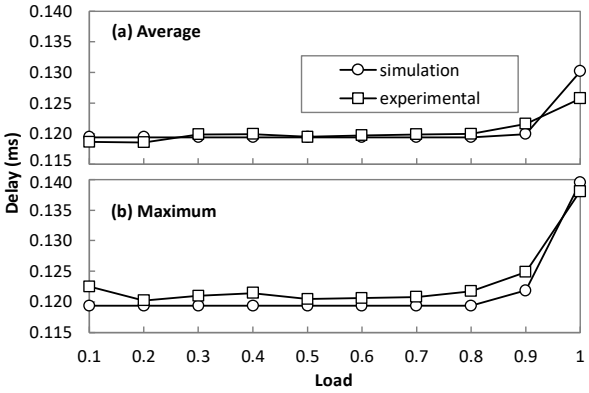

Fig. 2: Experimental Assessment

IV. Illustrative Numerical Results and Conclusions

The proposed near-real time operation was evaluated on an ad-hoc network simulator implemented in Python [5]. The simulator was firstly tuned with data from experiments, where two INT-enabled P4 software switches [6] connected to a Juniper M10i router were used to measure physical queueing times and emulate the routing between nodes R1 and R5 in Fig. 1a. Then, a Spirent SPT N4U Traffic Generator with 10Gb/s ports was connected to the software switches. The delay was measured between both switches for a total number of 2×10^5 packets generated with incremental traffic until reaching the full capacity of the system (load=1). The same traffic was used in the simulator. Fig. 2 present average and maximum delay results. Remarkable similarity between simulation and experimental results can be observed after tuning the simulator with the experimental measurements. These results validate the simulator as an accurate tool for evaluation purposes.

The scenario in Fig. 1a has been simulated with 3 flows between R1-R5. For the sake of simplicity in the results analysis, all the flows require d_{max} = 0.3 ms. The total amount of traffic ranges from 60 to 140 Gb/s and follows daily pattern oscillations [4]. Finally, background flows R1-R3 and R2-R5 were generated with random traffic around 30 and 50 Gb/s, respectively. With this configuration, two scenarios have been evaluated. The first scenario runs *without load manager*, i.e., workflow in Fig. 1c is not executed, and thus, one single DRL model per flow agent was trained to operate in the full range of traffic oscillations. Under this scenario, bypass is reactively activated/de-activated whenever any flow agent decides to route traffic through it. Routing costs are setup in order to foster using packet network routes (*p1*, *p2*) instead of optical bypass (*p3*). The second scenario runs *with load manager* and bypass is activated when load exceeds *thr_up* = 0.9 and de-activated when falls below *thr_down* = 0.85. Moreover, two different DRL models are available in the flow agents. When the optical bypass is activated, a model trained for low traffic range and operating only with packet network routes is loaded in the flow agents, whereas when the optical bypass is de-activated, another model trained for high traffic range using packet and bypass routes is loaded.

Fig. 3 shows the obtained performance along one typical day. Aggregated traffic (Fig. 3a) in packet network routes,

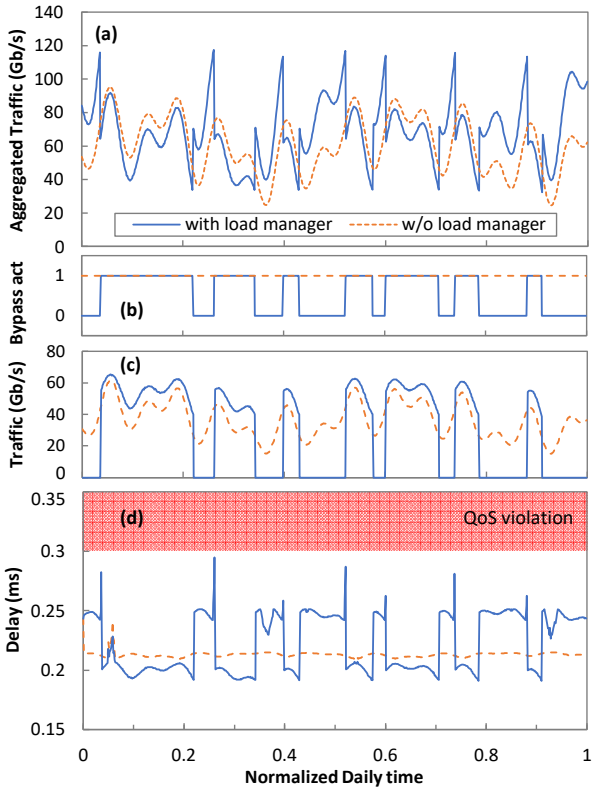

Fig. 3: Routing Performance and QoS Assurance

bypass activation (Fig. 3b), and bypass utilization (Fig. 3c) are shown. We observe that the intelligence added by the load manager achieves bypass utilization to 50% of the time, in contrast to a continuous bypass utilization and reduces network costs. However, the dynamicity generated by switching DRL models introduces traffic changes, which translates into higher delay variability (visible in Fig. 3a). Nonetheless, QoS is always guaranteed, as shown in Fig. 3d, where e2e delay fluctuates without violating committed requirement.

To conclude, an autonomous intelligence system has been proposed to provide near real-time operation, aimed to guarantee delay constraints and reduce network costs.

Acknowledgement

The research leading to these results has received funding from the Smart Networks and Services Joint Undertaking under the European Union's Horizon Europe research and innovation programme under G.A. 101096466 (DESIRE6G), the MICINN PID2020-114135RB-I00 (IBON) and from the ICREA Institution.

[1] S. Barzegar *et al*., "Distributed and Autonomous Flow Routing Based on Deep Reinforcement Learning," in Proc. PSC, 2022.

[2] F. Paolucci *et al.*, "Latency control in service chaining using P4-based data plane programmability," Elsevier Com. Net., 2022.

[3] A. Wassington *et al*., "Implementing a Machine Learning Function Orchestration," ECOC, 2021.

[4] S. Barzegar *et al*., "Packet Flow Capacity Autonomous Operation based on Reinforcement Learning," MDPI Sensors, 2021.

[5] A. Bernal *et al*., "Near Real-Time Estimation of E2E Performance in Converged Fixed-Mobile Networks," Comp. Comm.., 2020.

[6] T. Osiński *et al*. "A novel programmable software datapath for software-defined networking," CoNEXT, 2022.