Reinforcement Learning -based Autonomous Multilayer Network Operation

Sima Barzegar, Marc Ruiz, and Luis Velasco*

Optical Communications Group, Universitat Politècnica de Catalunya, Spain, * lvelasco@ac.upc.edu

Abstract *We propose an Intent-Based Networking solution based on Reinforcement Learning (RL). RL-based intents with specific operational objectives and coordinated operation between network layers are developed. Results show service assurance with automatic adaptation to changing traffic.*

Introduction

Advances in network automation [1] are receiving huge attention from the industry as the complexity of the network increases and the requirements from the services become more and more stringent and diverse. Network automation entails the collection of performance monitoring data that are analyzed (e.g., using Artificial Intelligence (AI) / Machine Learning (ML) techniques), and the extracted knowledge used to make decisions, thus defining *control loops*. However, several aspects and practices related to network automation need to be standardized and in fact, new concepts like the Intent-based Networking (IBN) [2] are currently under study by standardization bodies. IBN allows the definition of *operational objectives* that a network entity, e.g., a connection, has to meet without specifying how to meet them. IBN is in charge on implementing and enforcing those goals, often with the help of AI/ML.

In this paper, we go beyond multilayer network automation by demonstrating an IBN solution based on Reinforcement Learning (RL)[3], where customer-level operational objectives are propagated creating a hierarchy from the customer connections to the virtual links (*vlink*) and finally to the optical layer. RL fits well with IBN as it entails learning how to map situations to actions to maximize some reward, without specifically programming the learner.

RL-based autonomous operation

A reference example is presented in Fig. 1, where a multilayer transport network supports two customer connections A-D and B-C on top of the virtual network topology, where packet nodes (labelled Ri) are connected through vlinks, each supported by lightpaths on the optical layer. The customer connections can support services with different operational goals in terms of delay and throughput, so their capacity needs to be properly dimensioned. Instead of a fixed capacity dimensioning, however, such capacity can be dynamically adjusted to reduce overprovisioning. As a result of such dynamic capacity allocation, the vlink capacity can be also dynamically managed by establishing and releasing parallel lightpaths between the end packet nodes.

The inner graphs in Fig. 1 show the evolution of the traffic and the capacity of each entity (customer connection and vlink). For the sake of simplicity, let us assume that customer connections can adjust their capacity with a given granularity, e.g., 10 Gb/s, whereas each parallel lightpath supporting the aggregated capacity of the vlink has a capacity of 100 Gb/s. Therefore, the capacity of the entities can be adjusted to closely follow the input traffic in such entity by just establishing thresholds, so when the traffic exceeds or goes below a given threshold, their capacity is automatically increased or reduced. However, such threshold-based approach lacks the needed flexibility to introduce any different operational goal, in particular for customer connections, e.g., it is difficult to link maximum delay requirements with the required capacity of the connection. The inner graph for connection B-C in Fig. 1 shows the capacity adjustments performed assuming that the operational goal of the connection is to minimize the allocated capacity to reduce connectivity costs, by following as close as possible the input traffic while avoiding traffic loss. Note that this objective could be attained by means of ensuring the required threshold. However, in the case of connection A-D, the capacity was adjusted to ensure some maximum end-to-end delay; in spite of the subtle difference in the graphs in Fig. 1 compared to connection B-C, the capacity of connection A-D is always large enough with respect the input traffic to ensure that the delay added by the time spent in the queues is under the given maximum. With this capacity requirements, vlinks can be easily managed, i.e., the capacity of vlink R1-R2 varies after adding or releasing one lightpath.

To manage the capacity of the entities, in this paper we propose an IBN approach, where intents are based on RL. RL agents receive performance measurements from the network, as well as some reward that can be tuned as a function of the given operational goals, and are able to learn how to maximize such rewards in the long-term by generating the required actions to the environment. In our case, each intent consists of a RL agent managing the capacity of

each entity and an environment module that abstracts the connectivity entity and computes rewards from the measurements that are used by the RL agent to learn (Fig. 2). Each customer connection intent collects the amount of input traffic that is injected to the connection, as well as some other data as packet loss and delay and it determines the capacity of the connection that will be needed to meet the given operational goals for the next period (e.g., few minutes). A reward function uses the measured data and is programmable by tuning parameters that allow adjusting the behavior of the RL agent for e.g., keeping the total delay below a given maximum or to minimize the capacity; in both cases ensuring zero packet losses. In the case of the vlink intent, it receives as input the total capacity that every customer connection will require for the next period, as well as the aggregated amount of input traffic and the actual capacity of the vlink, and it is in charge of managing the vlink capacity to guarantee the aggregated capacity required by the customer connections by establishing/tearing down lightpaths. Note that while modifying the capacity of the customer connection entails programming some rules in the packet nodes and that capacity becomes immediately available, adding more capacity to the vlink entails establishing a new lightpath, which requires some time (e.g., one minute). Then, the vlink intents must make decisions with enough time to guarantee capacity availability.

In the next section, we present a RL model based on Q-learning[3] that fits with the needs of both customer connection and vlink intents.

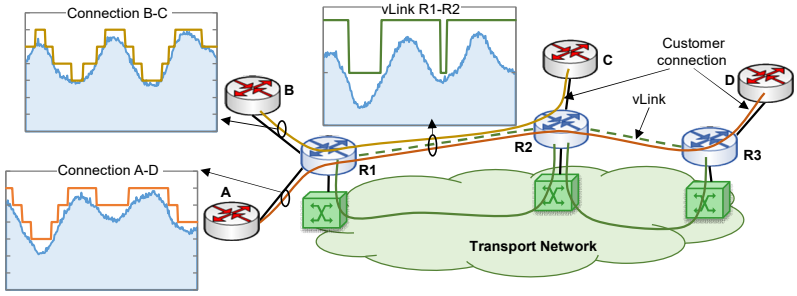

Fig. 1: A Multilayer scenario with two customer connections

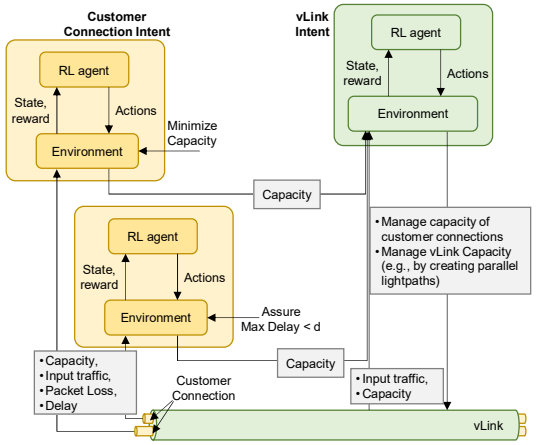

Fig. 2: RL-based connection and vlink intents

Table 1: State components for customer / vlink

(a) Customer [Min cap]	*y(t):*	max(input traffic) (b/s)
	z(t):	current capacity (b/s)
(b) Customer [Assure delay]	*y(t):*	max(delay) (s)
	z(t):	delay to be assured (s)
(c) vlink [Assure cap]	*y(t):*	max(input traf,Σ(cust. cap.)(b/s)
	z(t):	current cap (b/s)

Table 2: Reward function components

$f_i(X(t))$	φ(a)	φ(b)	φ(c)
Underprovisioning: packet loss > 0	-100	-100	0
Cap/delay limit violated: y(t) > *z*(t)	0	-100	-100
Capacity overprov: *z*(t) - *y*(t) > *k*	-10	-10	-10
Fitted provisioning: z(t) - y(t) < *k*	10	0	10

RL approach based on Q-learning

Q-learning is a model-free discrete RL technique that involves n^s states and n^a actions, and it is able to learn the optimal policy represented by a Q-table of pairs <state, action>. After taking an action, the RL agent observes the new state and gained reward and updates the Q-table accordingly [3]. In our approach, the computation of state and reward, as well as the actions to be taken, are evaluated periodically (e.g., every minute) when a new set $\mathcal{X}(t)$ of measurements, statistics (i.e., max, min and avg of the last monitoring samples), and parameters are available for the connection or vlink (we call this *episode*). The state at time *t* has been defined to reflect how close the entity is to its *limit*, see in eq. (1), where $y \in \mathcal{X}$ is the selected measurement and $z \in \mathcal{X}$ is its current limit. Table 1 defines *y* and *z* for the three identified problems, labelled (a), (b), and (c).

$$s(t) = \min\left(\lceil n^s \cdot y(t)/z(t) \rceil, n^s\right) \quad (1)$$

The reward function *r* is generalized as follows:

$$r(t) = \varphi_0 + \sum_{i \in m} \varphi_i \cdot f_i(\mathcal{X}(t)) \quad (2)$$

where the Boolean function $f_i(\cdot)$ returns 1 if the condition is met and 0 otherwise, and the coefficient φ_i penalizes (<0) or promotes (>0) the result of the operation; besides, φ_0 ensures a non-zero result. **Error! Reference source not found.** describes the components and coefficients for the three identified problems, where *k* refers to the size of a capacity unit and φ_0=1. For capacity over-provisioning (third row), we consider that both *x* and *z* refer to traffic and capacity, respectively, including for problem (b).

Finally, the action (number of *k*-size units to be added/subtracted) with highest reward with probability 1-ε (*exploitation*) or randomly with probability ε (*exploration*) is selected [3].

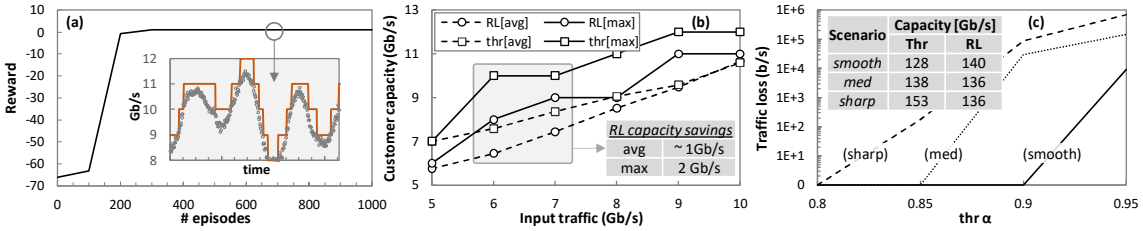

Fig. 3: RL convergence (a), capacity of customer connection for α=0.8 (b), and traffic loss in vlink (c)

Table 3: Customer operational objectives performance

Objective	*Capacity (Gb/s)*		*Delay (ms)*	
	avg	*max*	*avg*	*Max*
Minimize Capacity	**8.1**	**11**	1.5 (11%)	566 (27510%)
Delay < 5 ms	9.2 (13.6%)	13 (18.2%)	**1.35**	**2.05**

Illustrative Numerical Results

For numerical evaluation purposes, we have generated synthetic traffic flows for customer connection and vlinks according to the methodology and traffic models used in[4]-[5]. Using such generator, we emulated different number of traffic, generating more than 100,000 different samples to be collected and processed. We implemented Q-learning in Python3 and initially configured ε=0.2 with a decay strategy.

Fig. 3a shows the learning curve of the RL agent in terms of the average reward vs. the number of episodes for a customer connection. We observe that the RL agent quickly converges to the maximum reward, thus learning how to closely adjust the capacity to the actual traffic; similar learning curves for other customer and vlink traffic flows have been obtained. The inset in Fig. 3a shows a detail of such capacity adjustment (orange line) for an example of a customer connection flow (grey markers) with remarkably different daily traffic patterns.

To illustrate the impact of the operational objective on the actions taken by the RL agent, we run the same examples of customer connection traffic flows according to the two different objectives (a) and (b) defined in the previous section, while ensuring zero losses; the obtained average and maximum capacity and delay are summarized in Table 3. We observe from the results that the RL agent meets the selected objective; it either allocates minimum capacity at the expenses of increasing delay or it adds more capacity to assure that the max delay is not exceeded. Note the high max delay experienced by the traffic when the capacity minimization objective was configured, in contrast to the limited increment in the capacity allocation when max delay was assured.

Let us now compare the performance of the proposed RL agent against a reference *threshold-based* approach that adds/releases capacity to keep relative capacity utilization above parameter α ∈ [0,1]. Fig. 3b shows the required capacity for customer connection intents as a function of the magnitude of the input traffic. The threshold-based approach was configured with α=0.8 to ensure zero traffic loss. For the sake of fairness in the comparison, the operational objective has been configured to minimize the capacity, as the threshold-based approach cannot ensure maximum delay. In the results, we observe that the RL agent produces the best performance in the whole range of input traffic studied, significantly improving capacity utilization for low/moderated input traffic volume; the inner table shows savings as high as ~30% (2 Gb/s) per customer connection.

Finally, the hierarchical coordination of customer connection and vlink intents (see in Fig. 2) has been evaluated in terms of the allocated vlink capacity. To this end, we generated several scenarios by aggregating customer connections in a vlink; scenarios go from smooth to sharp variation between consecutive traffic measurements. A baseline method based on the threshold-based approach with no coordination between intents was implemented. Fig. 3c shows the maximum traffic loss (computed as the maximum amount of traffic exceeding capacity) for different values of α. We observe that different scenarios result into different optimal configuration of α, i.e., the largest α that produces zero loss. It is clear, in view of Fig. 3c, that a procedure to determine the best α according to input traffic characteristics is needed to achieve a high performance using a threshold-based approach. In contrast, the proposed RL agent adapts to different traffic characteristics in an automatic and robust way. The inner table compares the average capacity allocated by the RL agent with that of the threshold-based with the optimal α for every scenario (both operating with 0 losses). We observe that the RL agent allocates similar capacity regardless of the scenario, whereas the threshold-based approach overprovisions capacity when traffic changes are sharper.

Conclusions

Coordinated operation of RL-based intents in multiple layers has been proposed. Remarkable performance under traffic variations and different operational objectives has been shown.

Acknowledgements

The research leading to these results has received funding from the EC through the Spanish MINECO TWINS project (TEC2017-90097-R) and from ICREA.

References

[1] D. Rafique and L. Velasco, “Machine Learning for Optical Network Automation: Overview, Architecture and Applications [Invited Tutorial],” IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 10, pp. D126-D143, 2018.

[2] A. Clemm *et al*. (Eds.), “Intent-Based Networking - Concepts and Definitions,” IRTF draft work-in-progress, Mar. 2020.

[3] R. Sutton and A. Barto, *Reinforcement learning: an introduction*, MIT Press, 1998.

[4] M. Ruiz, F. Coltraro, and L. Velasco, “CURSA-SQ: A Methodology for Service-Centric Traffic Flow Analysis,” IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 10, pp. 773-784, 2018.

[5] A. Bernal, M. Richart, M. Ruiz, A. Castro, and L. Velasco, “Near Real-Time Estimation of End-to-End Performance in Converged Fixed-Mobile Networks,” Elsevier Computer Communications, vol. 150, pp. 393-404, 2020.