Towards network resiliency with AI driven automated load sharing in content delivery environments

Elkin Aguas, Anthony Lambert, Hervé Debar, Gregory Blanc

Towards Network Resiliency with AI Driven Automated Load Sharing in Content Delivery Environments

Elkin Aguas*†, Anthony Lambert*, Hervé Debar†, Grégory Blanc†
*Orange Labs, Châtillon, France
{prenom.nom}@orange.com
†Télécom SudParis, Institut Polytechnique de Paris, Evry-Courcouronnes, France
{prenom.nom}@.telecom-sudparis.eu

***Abstract*—Recent developments in orchestration and machine learning have made network automation more feasible, allowing the transition from prone-to-error, time consuming, manual manipulations to fast and refined automated responses in areas such as network security and management. This article investigates the capabilities of an RL agent to learn how to automatically distribute prefixes, correct undesired network behaviours and increase network resiliency and security. Our work focuses on network saturation, approaching the problem of network responsiveness in massive content delivery scenarios. Additionally, we propose a platform architecture to continuously monitor and deploy actions to the network.**

I. Introduction

New developments in machine learning, service deployment, and orchestration have drawn attention to the idea that automation in networks is needed and capable of achieving more than ever before [1]. Incorporating automation in networking areas such as security or management would indeed allow transitioning from error-prone, time-consuming, manual manipulations to faster and more refined automated responses, providing tools and methods to come up with a more resilient and secure network.

Network security threats have unique patterns and features, growing more complex and dynamic with time, rendering their identification and mitigation even more difficult. Automation has already proven applicable to detecting and correcting security threats in the network and application plane [2], [3], and thanks to novel learning approaches, e.g., neural networks and deep neural networks, facing new and more complex threats is becoming a possibility [4].

In particular, automation alleviates the control problem that Carriers and Internet Service Providers (ISPs) have when collaborating with content devlivery networks (CDNs). A direct consequence of the complex and opaque delivery strategies of these overlay networks and their actors, is Carriers and ISPs becoming "dumb pipes", as they have neither control nor insight on how traffic is delivered to their final users. This loss of control makes these entities vulnerable to not being able to react to events, such as traffic congestion and flash crowds, and certainly not attacks (such as Distributed Denial of Services (DDoS)), that put at risk the good, working state of the network.

There is still a way in which Carriers and ISPs can retain some control, that is by changing the BGP prefixes announced to the traffic sources so as to control and share the traffic coming from them. This however is a very error-prone and complex task, as finding and maintaining the good load sharing, that is in an optimal and timely manner, is hard. We believe that automation is able to reduce errors and delays of these manual prefix announcements.

This paper evaluates the capacity of automation through *Artificial Intelligence* to solve the aforementioned issue. Our work focuses on network saturation, approaching the problem of network responsiveness in massive content delivery scenarios and building a more resilient network, one that automatically handles events that put at risk its proper operation and user quality of experience (QoE). We use Deep Reinforcement Learning as our learning approach, firstly, because of its model-free nature, which enables to autonomously learn (by experience) the relation between the state of the environment and the impact of actions applied to it. This is motivated by the lack of datasets and models for the training process. Secondly, because of its inherent features (continuous adaptability and environment learning process), we believe RL can address the constantly changing and sometimes chaotic nature of computer networks.

II. State of the Art

Recent work in Reinforcement Learning (RL) [5], [6] continue to expand the limits of this field of machine learning, catching up researchers' attention and making us question ourselves about its applicability in different areas. Communication networks is one of these areas, and work is being already done to improve aspects such as QoE and traffic scheduling.

Xu et al. [7] propose experience-driven networking with a an RL approach. They present a traffic engineering (TE) framework that reduces end-to-end delay and improves network utility, without degrading the throughput (sometimes improving it). The RL algorithm they propose relies on the state of the network, represented by two components: throughput

and delay of each communication session. Their action space is formed by the split ratios of communication sessions.

An interesting congestion control problem with RL is presented in [8]. Jay et al. show they can monitor intricate patterns in network conditions and data traffic, by cosidering congestion control as an RL problem. They represent the network state by bounded histories of network statistics (latency gradient, latency ratio, and sending ratio). They apply changes in the sending rate as the actions on their environment.

III. Architecture

The building blocks of our proposed platform and its interaction with a service is shown in Fig. 1. The platform periodically monitors a service in the network, then these measurements are passed to a $Buffer$ block that stores it and organizes it in a chronological order with timestamps, a $Strategy$ block uses the collected information to create a representation of the state S_t of the network and chooses an action a_t, the latter modifying the transition of the former. This action is chosen from a finite set of predefined actions stored in the $Action$ block. The $Orchestrator$ block directs the process of choosing and action and passing it to the Deployment block. Finally, the $Deployment$ block applies the action to the network.

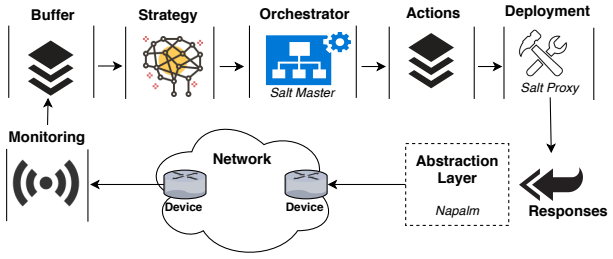

Fig. 1. Architecture diagram.

IV. Learning Approach

As previously said, Reinforcement Learning is the learning method chosen to approach our problem. Three factors represent the interaction between the agent and the environment in an RL problem: *state*, which represents the environment, *action*, which is the way the RL agent changes the environment state, and *reward*, which is value that says how good the action was based on the objective to be reached. The state of our environment is the maximum capacity, the number of prefixes and the traffic at a certain time, for each one of the links in Fig. 2, as well as the slope of the traffic. There are six actions for moving prefixes from one link to another, and one action for not moving any prefix. Finally, the reward is represented by a function (1), with $cap_{[link]}$ = [Link] maximum capacity and $pref_{[link]}$ = [Link] number of prefixes.

$$reward = -|cap_{peer} - pref_{peer}| \\ -|cap_{direct} - pref_{direct}| - |cap_{cache} - pref_{cache}| \quad (1)$$

V. Experimental Setting

Two scenarios are considered for the tests. The first one, the prefix distribution scenario, measures the performance of two algorithms to distribute prefixes between the interfaces in Fig. 2, based on their traffic capacity. The second one, the prioritized prefix distribution scenario, complexifies the task of the first scenario by prioritizing some traffic sources over others, as it is the case in more realistic cases where financial reasons are often the determinant factor.

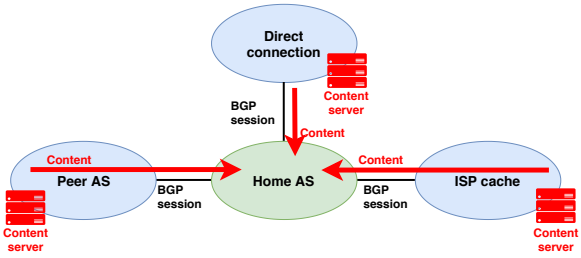

Fig. 2. CDN architecture.

A daily traffic model based on traffic characterized from a European tier-2 ISP in [9] is used in the training process. This traffic model is represented by the purple line in Fig. 5. A sample will be taken every five minutes, for technical reason and traffic wise convenience, giving a total of 288 measures taken per day. The number of actions per day will be between 0 and 288.

For both tests, the traffic capacity is set to 20%, 30% and 50% of the total AS incoming traffic, for peer, direct and cache link respectively. On each episode of the training process, a new set of initial randomly chosen prefixes for the three links is generated.

the prefix distribution scenario considers three cases. The first (no-algorithm case) does not consider any prefix redistribution algorithm, meaning that the initial random prefix distribution will stay during the whole day. We call the second case Naive Algorithm, and it is based on previous work we performed in [10]. It takes into consideration a set of conditional rules to verify the traffic in all the links and then distribute one prefix at a time based on this choice. Finally, the third case is our Deep Reinforcement Learning (DRL) algorithm that moves batches of prefixes.

The prioritized prefix distribution scenario, uses a special version of the DRL algorithm is used. We call it Priority Aware DRL (PADRL). This algorithm has a new reward function that reflects the desired prioritization between content sources.

The tests described above will allow to demonstrate that the RL agent is capable to learn the correct set of actions to reduce traffic loss, and therefore saturation in the network.

VI. Performance Analysis

Fig. 3 shows the maximum, mean, and minimum number of actions for the algorithms after one thousand executions of the test. Results show a proportional relation between the complexity of the algorithm and the number of actions it uses.

The no-algorithm case is not considered here because no action is ever taken. The average number of actions for DRL almost triples the number for the Naive algorithm, and for PADRL it is more than three times the number for DRL.

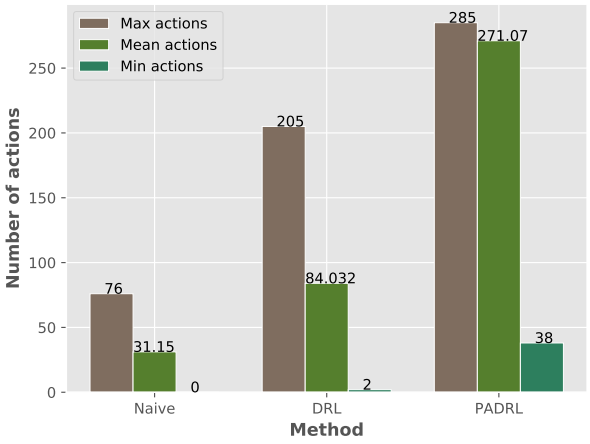

Fig. 3. Number of actions per method.

Fig. 4 presents the results of the traffic loss for all cases and for one thousand executions of the test. The loss percentage presented here is calculated from the total traffic entering the AS. Contrary to what we observed for the number of actions, a more complex algorithm means a lower traffic loss, however, using the most actions does not mean having the lowest loss percentage, which is the case of PADRL. Less traffic loss means less traffic congestion, which indicates the network learns to be resilient to events creating this undesired state.

The increase in the number of actions and decrease in the traffic loss, indicates that our RL algorithm can create a model that tries to reach the goal we gave to it, which is the distribution of prefixes to reduce congestion and traffic loss.

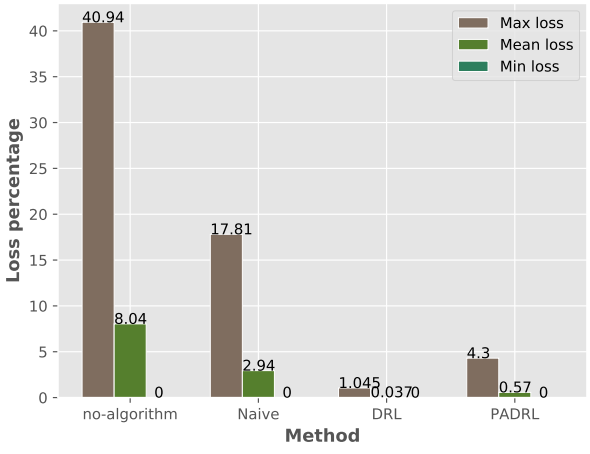

Fig. 4. Traffic loss per method.

Fig. 5 shows the distribution process for the DRL algorithm. The trained model brings the initial random number of prefixes on the left, represented by dashed and dotted lines, down to their expected values. After a convergence period of repeatedly moving prefixes, the model set the prefix distribution very close to the expected 20%, 30% and 50%.

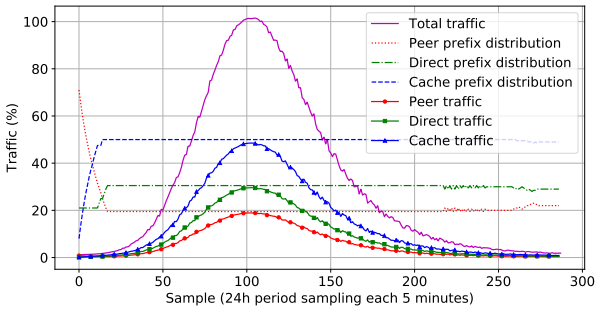

Fig. 5. Prefix distribution with DRL.

VII. Conclusion

Using RL to distribute BGP prefixes and increase network resiliency is feasible. Results show that, by increasing the number of actions taken and reducing the traffic loss, an RL agent is capable of learning the correct actions that lead to a better state of the network. Results also suggest that the base case of prefix distribution can be extended to more complex ad realistic cases, such as the prioritization case described, by changing the reward function of the RL algorithm.

References

[1] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "z-torch: An automated nfv orchestration and monitoring solution," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1292–1306, Dec 2018.

[2] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser, "Detecting spammers with snare: Spatio-temporal network-level automatic reputation engine," in *Proceedings of the 18th Conference on USENIX Security Symposium*, ser. SSYM'09. USA: USENIX Association, 2009, p. 101–118.

[3] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns," in *Proceedings of the 19th USENIX Conference on Security*, ser. USENIX Security'10. USA: USENIX Association, 2010, p. 18.

[4] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Cyberthreat detection from twitter using deep neural networks," in *2019 International Joint Conference on Neural Networks (IJCNN)*, July 2019, pp. 1–8.

[5] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," 2019.

[6] K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "The tools challenge: Rapid trial-and-error learning in physical problem solving," 2019.

[7] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 1871–1879.

[8] N. Jay, N. H. Rotman, P. B. Godfrey, M. Schapira, and A. Tamar, "Internet congestion control via deep reinforcement learning," 2018.

[9] Q. Grandemange, O. Ferveur, M. Gilson, and E. Gnaedinger, "A live network as-level traffic characterization," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2017, pp. 901–905.

[10] E. Aguas, T. Green, and A. Lambert, "Poster abstract: On the feasibility of event-driven network automation scenarios for bgp," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2019, pp. 1031–1032.