Reinforcement Learning for Datacenter Congestion Control

Chen Tessler, Yuval Shpigelman, Gal Dalal, Amit Mandelbaum, Doron Haritan Kazakov, Benjamin Fuhrer, Gal Chechik, and Shie Mannor

ABSTRACT

We approach the task of network congestion control in datacenters using Reinforcement Learning (RL). Successful congestion control algorithms can dramatically improve latency and overall network throughput. Until today, no such learning-based algorithms have shown practical potential in this domain. Evidently, the most popular recent deployments rely on rule-based heuristics that are tested on a predetermined set of benchmarks. Consequently, these heuristics do not generalize well to newly-seen scenarios. Contrarily, we devise an RL-based algorithm with the aim of generalizing to different configurations of real-world datacenter networks. We overcome challenges such as partial-observability, non-stationarity, and multi-objectiveness. We further propose a policy gradient algorithm that leverages the analytical structure of the reward function to approximate its derivative and improve stability. We show that this scheme outperforms alternative popular RL approaches, and generalizes to scenarios that were not seen during training. Our experiments, conducted on a realistic simulator that emulates communication networks' behavior, exhibit improved performance concurrently on the multiple considered metrics compared to the popular algorithms deployed today in real datacenters. Our algorithm is being productized to replace heuristics in some of the largest datacenters in the world.

1. INTRODUCTION

Most RL algorithms were designed under the assumption that the world can be adequately modeled as a Markov Decision Process (MDP). Unfortunately, this is seldom the case in realistic applications. In this work, we study a real-world application of RL to data-center congestion control. This application is highly challenging because of partial observability and complex multi-objective. Moreover, there are multiple decision-makers that have to make concurrent decisions that can affect each other. We show that even though that the problem at hand does not fit the standard model, we can nevertheless devise an RL framework that outperforms state-of-the-art tailored heuristics. Moreover, we show experimentally that the RL framework generalizes well.

We provide a full description of the Congestion Control (CC) problem in Appendix B, but from a bird's eye view, it is enough to think of CC as a multi-agent, multi-objective, partially observed problem where each decision maker receives a goal (target). The target makes it easy to tune the behavior to fit the requirements (i.e., how latency-sensitive the system is). We devise the target so that leads to beneficial behavior in the multiple considered metrics, without having to tune coefficients of multiple reward components. We model the task of datacenter congestion control as a reinforcement learning problem. We observe that standard algorithms (Mnih et al., 2015; Lillicrap et al., 2015; Schulman et al., 2017) are incapable of solving this task. Thus, we introduce a novel on-policy deterministic-policy-gradient scheme that takes advantage of the structure of our target-based reward function. This method enjoys both the stability of deterministic algorithms and the ability to tackle partially observable problems.

To validate our claims, we develop an RL GYM (Brockman et al., 2016) environment, based on a realistic simulator and perform extensive experiments. The simulator is based on OMNeT++ (Varga, 2002) and emulates the behavior of ConnectX-6Dx network interface cards (state of the art hardware deployed in current datacenters). Our experiments show that our method, Programmable CC-RL (PCC-RL), learns a robust policy, in the sense that it is competitive in all the evaluated scenarios; often outperforming the current state-of-the-art methods.

Our contributions are as follows. (i) We formulate the problem of datacenter congestion control as a partially-observable multi-agent multi-objective RL task. (ii) We present the challenges of this realistic formulation and propose a novel on-policy deterministic-policy-gradient method to solve it. (iii) We provide an RL training and evaluation suite for training and testing RL agents within a realistic simulator. Finally, (iv) we ensure our agent satisfies compute and memory constraints such that it can be easily deployed in future datacenter network devices.

2. REINFORCEMENT LEARNING FOR CONGESTION CONTROL

Due to the lack of space, the background is presented in Appendix A and Appendix C. Below we formulate the problem of CC as an RL task and present our solution.

The RL POMDP framework from Appendix C requires the definition of the four elements in $(\mathcal{S}, \mathcal{A}, P, R)$. The agent, a congestion control algorithm, runs from within the network-interface-card (NIC) and controls the rate of the flows passing through that NIC. At each decision point, the agent observes statistics correlated to the specific flow it controls, an observation o of the state s. The agent then acts by determining a new transmission rate and observes the outcome of this action.

Observations. As the agent can only observe information relevant to the flow it controls, we consider: the flow's transmission rate, RTT measurement, and number of *NACK* packets received. The NACK packets represent events occurring in the network. A NACK packet signals to the source host that packets have been dropped (e.g., due to congestion) and should be re-transmitted.

Actions. The optimal transmission rate depends both on the number of agents simultaneously interacting in the network, and on the network itself (bandwidth limitations and topology). As such, the optimal transmission rate will vary greatly across scenarios. Since it should be quickly adapted across different orders of magnitude, we define the action as a multiplication of the previous rate. I.e., $\text{rate}_{t+1} = \mathbf{a}_t \cdot \text{rate}_t$.

Transitions. The transition $\mathbf{s}_t \to \mathbf{s}'_t$ depends on the dynamics of the environment and on the frequency at which the agent is polled to provide an action. Here, the agent acts once an RTT packet is received. This is similar to the definition of a monitor interval by Dong et al. (2018), but while they considered fixed time intervals, we consider event-triggered (RTT) intervals.

Reward. As the task is a multi-agent partially observable problem, the reward must be designed such that there exists a single fixed-point equilibrium. Thus, we let

$$r_t = -\left(\mathbf{target} - \frac{\text{RTT}_t^i}{\text{base-RTT}^i} \cdot \sqrt{\text{rate}_t^i}\right)^2 ,$$

where **target** is a constant value shared by all flows, base-RTT^i is defined as the RTT of flow i in an empty system, and RTT_t^i and rate_t^i are respectively the RTT and transmission rate of flow i at time t. $\frac{\text{RTT}_t^i}{\text{base-RTT}^i}$ is also called the rtt inflation of agent i at time t. The ideal reward is obtained when $\mathbf{target} = \frac{\text{RTT}_t^i}{\text{base-RTT}^i} \cdot \sqrt{\text{rate}_t^i}$. Hence, when the **target** is larger, the ideal operation point is obtained when $\frac{\text{RTT}_t^i}{\text{base-RTT}^i} \cdot \sqrt{\text{rate}_t^i}$ is larger. The transmission rate has a direct correlation to the RTT, hence the two grow together. Such an operation point is less latency sensitive (RTT grows) but enjoys better utilization (higher rate).

Based on Appenzeller et al. (2004), a good approximation of the RTT inflation in a bursty system, where all flows transmit at the ideal rate, behaves like $\sqrt{N}$, where N is the number of flows. As the system at the optimal point is on the verge of congestion, the major latency increase is due to the packets waiting in the congestion point. As such, we can assume that all flows sharing a congested path will observe a similar $\text{rtt-inflation}_t \approx \frac{\text{RTT}_t^i}{\text{base-RTT}^i}$. As Proposition 1 shows, maximizing this reward results in a fair solution.

Proposition 1. *The fixed-point solution for all N flows sharing a congested path is a transmission rate of $\frac{1}{N}$.*

The proof is provided in Appendix H.

In practice we use the following approximation of the gradient (an exact derivation is provided in Appendix E):

$$\nabla_\theta G^{\pi_\theta}(\mathbf{s}) \approx \tag{1}$$
$$\left[\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \left(\mathbf{target} - \text{rtt-inflation}_t \cdot \sqrt{\text{rate}_t}\right)\right] \nabla_\theta \pi_\theta(\mathbf{s}) .$$

Using this derivation enables a deterministic on-policy policy-gradient solution, which we found detrimental to solving this complex task.

3. EXPERIMENTS

To show our method generalizes to unseen scenarios, motivating the use in the real world, we split the scenarios to train and test sets. We train the agents only in the many-to-one domain (see below), on the scenarios: $2 \to 1, 4 \to 1$, and $8 \to 1$. Evaluation is performed on many-to-one, all-to-all, and long-short scenarios. We provide additional details in Appendix G.1, with an extensive overview of the training process, including the technical challenges of the asynchronous CC task, in Appendix G.

3.1 Baseline comparison

The results for the many-to-one tests are presented in Table 1. The simulation settings (number of hosts and flows per host) are presented in Appendix G. We observe that while most methods are competitive at a small scale (small amount of flows with few interactions), at large-scale deployments, all methods aside from PCC-RL encounter extensive periods of packet loss. PCC-RL is the only method capable of maintaining high switch utilization, while keeping the latency low and fairness at a reasonable level.

Table 2: **All-to-all** test results. In these tests, none of the algorithms exhibited packet loss. For 4 hosts, we mark both PCC-RL and DC2QCN as best since it is up to the end-user to determine which outcome is preferred.

Alg.	**4 hosts**			**8 hosts**		
	SU	FR	QL	SU	FR	QL
PCC-RL	**94**	**77**	**6**	**94**	**97**	**8**
DC2QCN	**90**	**91**	**5**	91	89	6
HPCC	71	18	3	69	60	3
SWIFT	76	100	11	76	98	13

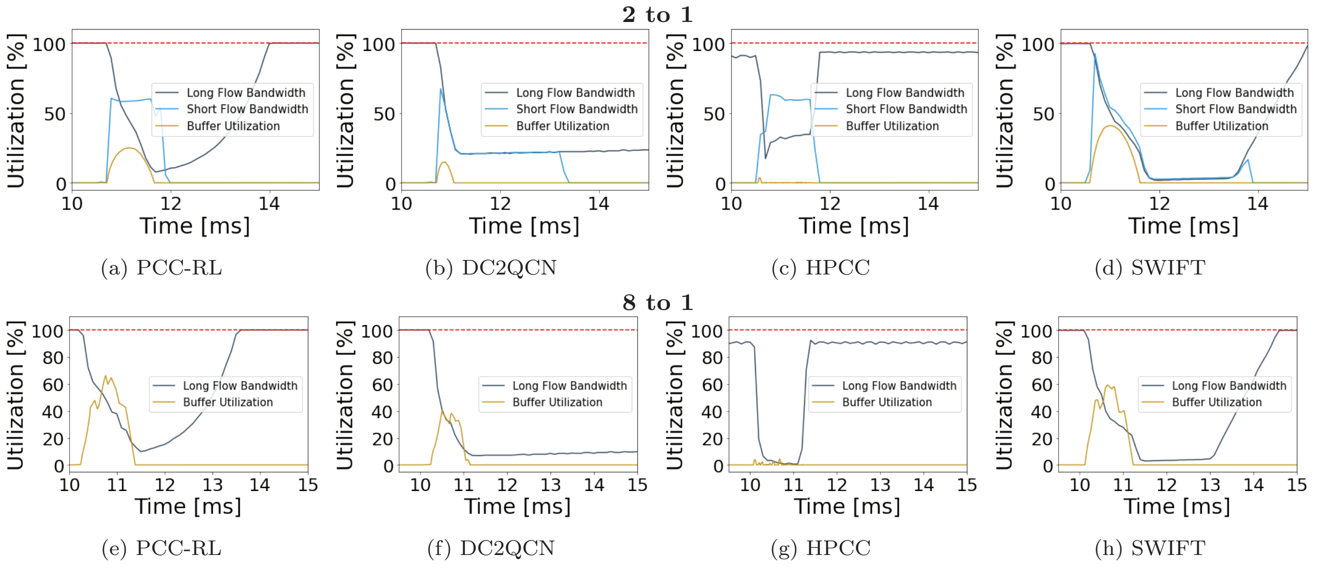
In addition, we evaluate the various methods on an all-to-all setting. Here, there are N hosts and $2 \cdot N$ flows running on each (a total of $2N^2$) . At each host, flow i constantly sends data towards host i mod N through switch i mod N. The results are presented in Table 2. Although SWIFT performed well at low scale many-to-one tests, when transitioning to the all-to-all setting it is incapable of retaining high switch utilization. Similarly to the many-to-one setting, PCC-RL performs competitively and outperforms at the higher scale setting.

Finally, we compared the methods on a long-short setting where the algorithms are tested on their ability to quickly react to changes, presented in Fig. 1 (numerical results in Appendix L). Although PCC-RL did not encounter this scenario during training, it is able to perform competitively. In this scenario, the fastest to react was HPCC, which also maintained minimal buffer utilization. We highlight, though, that HPCC was specifically designed to handle such long-short scenarios (Li et al., 2019). Nonetheless, as opposed to HPCC, PCC-RL achieves 100% utilization before and after the interruption and recovers faster than both SWIFT and DC2QCN.

Summary: We observe that PCC-RL is capable of learning a robust policy. Although in certain tasks the baselines

Table 1: **Many-to-one** test results. Numerical comparison of PCC-RL (our method) with DC2QCN (unpublished followup to Zhu et al. (2015)), HPCC (Li et al., 2019) and SWIFT (Kumar et al., 2020). The column legend is: **SU** Switch Utilization [%] (higher is better), **FR** Fairness defined as $\frac{minrate \cdot 100}{maxrate}$ (higher is better), **QL** Queue Latency [μ sec] (lower is better), and **DR** Drop rate [Gbit/s] (lower is better). As the goal of a CC algorithm is to prevent congestion, we color the tests that failed (extensive periods of packet loss) in red. We mark the best performing methods in each test in bold. For 1024, we mark both DC2QCN and SWIFT as it is up to the end user to determine which outcome is preferred.

Alg.	**128 to 1**				**1024 to 1**				**4096 to 1**				**8192 to 1**			
	SU	FR	QL	DR	SU	FR	QL	DR	SU	FR	QL	DR	SU	FR	QL	DR
PCC-RL	**92**	**95**	**8**	**0**	90	70	15	0	**91**	**44**	**26**	**0**	**92**	**29**	**42**	**0**
DC2QCN	96	84	8	0	**88**	**82**	**17**	**0**	85	67	110	0.2	100	72	157	1.3
HPCC	83	96	5	0	59	48	27	0	73	13	79	0.2	86	8	125	0.9
SWIFT	98	99	40	0	**91**	**98**	**66**	**0**	90	56	120	0.1	92	50	123	0.2



(a) PCC-RL (b) DC2QCN (c) HPCC (d) SWIFT

(e) PCC-RL (f) DC2QCN (g) HPCC (h) SWIFT

Figure 1: **Long Short** test results. The goal is to recover fast, but also avoid packet loss. Higher buffer utilization means higher latency and only when the buffer is fully utilized (100% utilization of the 5MB allocated) packets are dropped. In these tests, none of the algorithms encountered packet loss. The top row (Figs. 1a to 1d) presents the results of a long-short test with 2 flows, and the bottom row (Figs. 1e to 1h) presents a test with 8 flows. We plot the bandwidth utilization of the long flow and the buffer utilization in the switch. Recovery time is measured as the time it takes the long flow to return to maximal utilization. As can be seen, in both scenarios, DC2QCN does not recover within a reasonable time. In addition, in HPCC, the long flow does not reach 100% utilization, even when there are no additional flows.

seldom marginally outperformed it, PCC-RL always obtained competitive performance. In addition, in large scale scenarios, where thousands of flows interact in parallel, we observed that PCC-RL is the only method capable of avoiding packet loss and thus control network congestion. As PCC-RL was trained only on a low-scale scenario, it highlights the ability of our method to generalize and learn a robust behavior, that we believe will perform well in a real datacenter.

4. SUMMARY

In this work, we demonstrated the efficacy and generalization capabilities of RL, in contrast to the hand-crafted algorithms that currently dominate the field of CC. Our experiments utilized the realistic OMNeT++ network simulator that is commonly used to benchmark CC algorithms for deployment in real datacenters. While the various baselines exhibited outstanding performance in certain scenarios, there are others in which they catastrophically failed. Contrarily, PCC-RL learned a robust policy that performed well in all scenarios and often obtained the best results. In addition, we show that PCC-RL generalizes to unseen domains and is the only algorithm capable of operating at a large-scale without incurring packet loss.

PCC-RL was developed with the aim of easy deployment in real datacenters, and potentially even on-device training. We took into consideration memory and compute limitations. By limiting to relatively small and simple networks, combined with int8 quantization, we ensured that the method can run in real-time on a NIC. The next steps involve full productization of the algorithm and adaptive deployment in datacenters – such that enables customization to customers' needs via the tunable target parameter and additional possible reward components.

References

Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 conference*, pp. 63–74, 2010.

Appenzeller, G., Keslassy, I., and McKeown, N. Sizing router buffers. *ACM SIGCOMM Computer Communication Review*, 34(4):281–292, 2004.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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

Jay, N., Rotman, N., Godfrey, B., Schapira, M., and Tamar, A. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*, pp. 3050–3059, 2019.

Kumar, G., Dukkipati, N., Jang, K., Wassel, H. M., Wu, X., Montazeri, B., Wang, Y., Springborn, K., Alfeld, C., Ryan, M., et al. Swift: Delay is simple and effective for congestion control in the datacenter. 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. 514–528, 2020.

Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., and Legg, S. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

Li, Y., Miao, R., Liu, H. H., Zhuang, Y., Feng, F., Tang, L., Cao, Z., Zhang, M., Kelly, F., Alizadeh, M., et al. Hpcc: High precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication*, pp. 44–58. 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, C., Xu, X., and Hu, D. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398, 2014.

Mania, H., Guy, A., and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Mannor, S. and Shimkin, N. A geometric approach to multi-criterion reinforcement learning. *Journal of machine learning research*, 5(Apr):325–360, 2004.

Mittal, R., Lam, V. T., Dukkipati, N., Blem, E., Wassel, H., Ghobadi, M., Vahdat, A., Wang, Y., Wetherall, D., and Zats, D. Timely: Rtt-based congestion control for the datacenter. *ACM SIGCOMM Computer Communication Review*, 45(4):537–550, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 1994.

Ramakrishnan, K., Floyd, S., and Black, D. Rfc3168: The addition of explicit congestion notification (ecn) to ip, 2001.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *ICML*, 2014.

Spaan, M. T. Partially observable markov decision processes. In *Reinforcement Learning*, pp. 387–414. Springer, 2012.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Varga, A. Omnet++ http://www. omnetpp. org. *IEEE Network Interactive*, 16(4), 2002.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M. H., and Zhang, M. Congestion control for large-scale rdma deployments. *ACM SIGCOMM Computer Communication Review*, 45 (4):523–536, 2015.