Implementing Reinforcement Learning Datacenter Congestion Control in NVIDIA NICs

Benjamin Fuhrer
NVIDIA Networking

Yuval Shpigelman
NVIDIA Networking

Chen Tessler
NVIDIA Research

Shie Mannor
NVIDIA Research, Technion Institute of Technology

Gal Chechik
NVIDIA Research, Bar-Ilan University

Eitan Zahavi
NVIDIA Networking

Gal Dalal
NVIDIA Research

Abstract

Cloud datacenters are exponentially growing both in numbers and size. This increase results in a network activity surge that warrants better congestion avoidance. The resulting challenge is two-fold: (i) designing algorithms that can be custom-tuned to the complex traffic patterns of a given datacenter; but, at the same time (ii) run on low-level hardware with the required low latency of effective Congestion Control (CC). In this work, we present a Reinforcement Learning (RL) based CC solution that learns from certain traffic scenarios and successfully generalizes to others. We then distill the RL neural network policy into binary decision trees to achieve the desired μsec decision latency required for real-time inference. We deploy the distilled policy on NVIDIA NICs in a real network and demonstrate state-of-the-art performance, balancing all tested metrics simultaneously: bandwidth, latency, fairness, and drops.

1 Introduction

Modern datacenters support computationally intensive applications such as distributed data processing, heterogeneous and edge computing, and storage. With advancements in hardware and software, memory access management nowadays is often conducted directly by the network interface card (NIC) with Remote Direct Memory Access (RDMA) (Beck and Kagan, 2011). Consequently, the limiting factor in network performance becomes traffic congestion. Congestion occurs when traffic arrives at a node (switch or NIC) at a faster rate than it can be processed. As each node is equipped with a FIFO queue, the transmission latency grows proportionally to the congestion. Efficient CC is thus crucial to sustaining high throughput and low latency in datacenters. CC algorithms set a limit on the transmission rate or number of in-network bytes of each flow. By observing changes in the network, such as latency signals, these algorithms are tasked with preventing congestion in a diverse set of network topologies and traffic patterns.

Most literature on CC tackled the problem using hand-crafted heuristics. These methods tend to perform well in specific tasks yet underperform in others that they were not optimized for. For instance, DCQCN (Zhu et al., 2015) and SWIFT (Kumar et al., 2020) have been optimized for steady-state scenarios; but, as shown in (Tessler et al., 2022), their reaction time is slow for sudden bursts of short flows. Recently, Tessler et al. (2022) introduced a data-driven approach to learning a CC policy. They presented a reward function (measuring the latency and throughput) and devised an

Submitted to ML for Systems Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

algorithm that maximizes the cumulative reward throughout multiple steps. Their method resulted in a robust neural-net based policy capable of tackling a range of tasks in a *simulated* network.

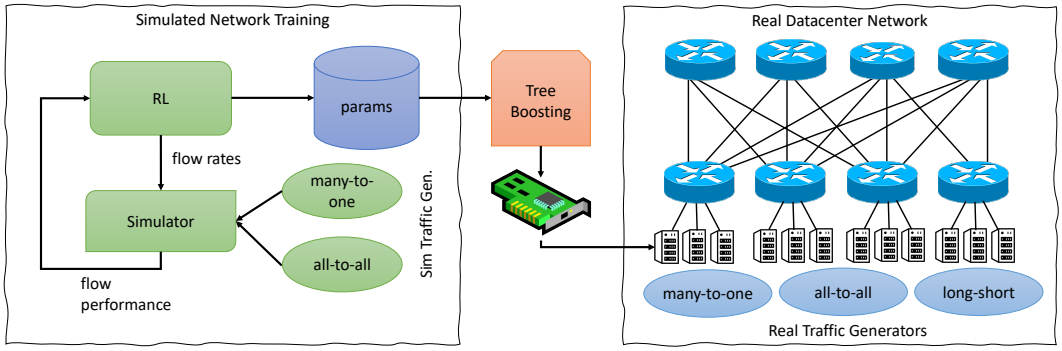

Figure 1: The entire flow: from reinforcement learning in simulation to live datacenter congestion control.

Similarly to previous ML-based CC approaches, the policy in (Tessler et al., 2022) is too computationally intense for real-world deployment directly on a NIC. In this work, we overcome the above issues that prevented ML-based CC approaches from reaching production pipelines. We design a lightweight version of Tessler et al. (2022)'s RL algorithm complying with present hardware limitations. We deploy it in an operational datacenter and achieve state-of-the-art (SOTA) performance. We also analyze our algorithm's behavior in a human-interpretable fashion and show how it matches known concepts in classic CC theory.

In summary, our main contributions are:

1. We show how to transform complex policies to low-compute low-latency architecture, gaining x500 latency reduction with a negligible effect on the quality of the policy. Specifically, we distill deep networks with $450\ \mu sec$ latency to decision trees with $0.9\ \mu sec$ latency.
2. We deploy our method on NVIDIA production NICs, ConnectX-6Dx, in an operational cluster of 64 hosts, and run extensive evaluations reaching SOTA results.

The RL-CC framework is visualized in Fig. 1.

2 Related Work

CC algorithms have been demonstrated to reduce packet drops, thus improving overall network performance (Bui et al., 2021; Zhu et al., 2015; Mittal et al., 2015; Shpiner et al., 2017; Kumar et al., 2020). These methods govern the transmission rate of each flow, balancing multiple, possibly conflicting, objectives. They aim to react quickly to changes and maximize the overall network utilization and fairness between flows, while minimizing packet latency and packet drops. The conflict between these objectives was explained by Kumar et al. (2020): when N flows share a congested path, and each transmits at the optimal rate (line rate$/N$), the average queue length is $\mathcal{O}(\sqrt{N})$. Hence, a low latency solution that is fair when N is large can only be achieved by reducing bandwidth, resulting in a bandwidth-latency tradeoff. Current SOTA CC relies on indications such as round-trip time (RTT) and switch queue-length to evaluate the network's status and adjust the transmission rate appropriately. Those deployed in practice use rule-based heuristics to react to such indications. A common drawback of conventional CC algorithms is the need for manual tuning of their multiple parameters by domain experts. And yet, the results are often unsatisfactory at properly balancing the tradeoffs.

Prior work considered ML-based CC (Jiang et al., 2020); however, those algorithms often require large memory and computational complexity (Jiang et al., 2020). Generally, for CC algorithms to successfully operate, their decision time must be $\mathcal{O}(\text{RTT})$. For modern datacenters utilizing RDMA, this amounts to a few μsec. Neural networks, on the other hand, require orders of magnitude larger inference time, relatively heavy compute power, and significant memory. These limitations partly explain why there is yet no learning-based RDMA CC in production. Recently, Tessler et al.

(2022) introduced an RL-based RDMA CC algorithm called Analytic Deterministic Policy Gradient (ADPG). In several network simulation benchmarks, ADPG outperformed SOTA rule-based CC algorithms, DCQCN, and SWIFT. Its success is attributed to its RTT-based reward that at its optimum depicts an optimal flow equilibrium. In line with our summary above, Tessler et al. (2022) state that for potential deployment, they would require dedicated hardware to accommodate the computational burden of deep-learning inference. Learning-based methods in general and RL in particular have many applications in networking. Nevertheless, the computational and memory requirements of NN training and inference are too demanding for implementation on NICs such as ConnectX-6Dx. In this work, we build upon (Tessler et al., 2022) and devise a lightweight variant based on decision trees that solves the computational challenge flagged by Tessler et al. (2022).

3 Problem Setup

We consider two environment settings – simulation and live. For simulation, we utilize a realistic OMNeT++ emulator (Varga, 2002) that models a shallow single-switch network with a varying number of flows. We experiment with different combinations of total flows, ranging between $2 - 8192$, distributed across multiple hosts. Each host is equipped with an NVIDIA NIC. With the simulator, we train the RL agent on a small set of benchmarks and then evaluate them on additional ones, with precise in-network measurements at pinpoint accuracy. While the simulations are rich, they do not precisely mimic real-world behavior. For instance, the simulations do not emulate complex network topologies and their effect on congestion and RTT. Thus, we also perform experiments on two operational clusters.

We consider the following benchmarks: **Many to one:** All hosts transmit data to a single receiver host. **All to all:** All hosts transmits data to all others. **Long Short:** A single long flow transmits unlimited data, while multiple short flows randomly appear and transmit for a short window of time. We evaluate our experiments with different metrics for the steady-state experiments and for the recovery experiments. **Steady-state**: (i) net data throughput, (ii) unfairness calculated as the coefficient of variation of the sent bandwidth, (iii) average packet latency, and (iv) average packet loss per flow. **Recovery**: (i) long flow bandwidth, (ii) average packet loss per flow (iii) average normalized completion time, (iv) normalized completion time 99% percentile.

We compare the performance of RL-CC to the Swift heuristic (Kumar et al., 2020), as it also relies on the **target** parameter. Additionally, we compare our implementation to the hardware-based DCQCN.

4 Lightweight RL-CC

Congestion control is a sequential decision-making problem. The decision maker (agent) in our case is an instance of the CC algorithm running within the NIC and controlling the rate of a single transmission flow. The agent acts upon recent information available to it: current and past transmission rate, RTT, and last actions taken. The agent's biggest challenge is that it is completely unaware of the existence of other concurrent agents and their state. It needs to act strictly based on its local state where all global network sensing is based solely on RTT packets the agent receives. To overcome these challenges, Tessler et al. (2022) utilize a novel analytic reward function. At it maximum, the reward function achieves a fair and bandwidth-efficient equilibrium among all flows.

Modern NVIDIA NICs support programmable CC engines and network state measurements such as RTT or switch telemetry. Leveraging this mechanism, we implement an RL agent that observes relevant statistics and acts by modulating its own transmission rate.

4.1 Model Distillation with Boosting Trees

Boosting-Trees is a supervised learning method that often achieve SOTA results on certain tasks and are comparable to deep learning on others (Caruana and Niculescu-Mizil, 2006; Roe et al., 2005; Anghel et al., 2018; Zhang et al., 2017). Boosting-Trees are robust (Einziger et al., 2019), deterministic, and once trained, can be easily converted to a sequence of if-else instructions suitable for hardware implementation. For this reason, we find them suitable for our hardware CC implementation. NN-based policies often require millions or even billions of interactions (Badia et al., 2020) to reach an optimal policy. These interactions are conducted with sub-optimal

	Small LSTM	MLP	Boosting-Trees
FLOPS	2600	200	-
Decision Latency [μsec]	450	17	0.9

Table 1: FLOPS and inference latency as calculated on ConnectX-6Dx. a) **Small LSTM**: state→fC→LSTM→fC→action. b) **MLP**: window of states→fc→action. c) **Boosting-Trees**: implemented as an if-else sequence.

Number of Flows	**8**		**64**		**512**		**1024**		**2048**	
Metric	GP	Latency	GP	Latency	GP	Latency	GP	Latency	GP	Latency
RL-CC (MLP)	0.96	8.85	0.92	12.19	0.90	17.82	0.90	21.70	0.90	27.62
RL-CC (Tree)	0.97	8.85	0.92	12.03	0.90	17.98	0.90	21.73	0.90	27.35

Table 2: Policy Distillation: Comparing MLP based policy vs distilled policy on many-to-one scenarios while varying the number of flows. GP denotes Goodput [normalized] (net traffic without packet loss), and Latency is measured in μsec.

policies during the learning stage, and mostly do not reflect optimal behavior. It is only after convergence to an optimal policy that such interactions are suitable to be used in a supervised learning setup. Thus, after training our RL policy with a NN, we can distill it to supervised learning model, Boosting Trees in our case, using a representative set of interactions.

Our goal is to teach a lightweight RL-CC tree-based policy that imitates a NN-based policy over a representative distribution of inputs. Previous work has shown that model distillation with trees can work well on various tasks (Che et al., 2017; Liu and Wang, 2018; Li et al., 2020; Song et al., 2021; Biggs et al., 2020). The biggest challenge in distilling the policy is the LSTM layer, which specializes in incorporating past information. We thus removed the LSTM layer and instead provide as input a window of previous states. The distillation process is illustrated in Fig. 2. Its goal is to minimize the loss between the outputs of the two models on data gathered during the NN RL policy inference stage. We restricted the number of boosting iterations and maximal tree depth per tree to satisfy the limits of ConnectX-6Dx. The resulting number of operations does not exceed 150.

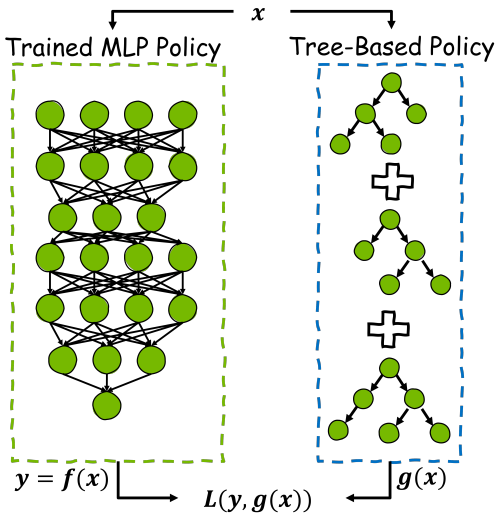

Figure 2: Model Distillation: Teaching tree-based student policy g to mimic the fixed NN-based policy f by minimizing $L(y, g(x)) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_i - g(x_i))^2}$.

In Table 1, we compare the decision time latency between the original NN and distilled tree-based policies. As seen, we obtained x500 speed-up, from $450\mu sec$ down to $0.9\mu sec$. In Table 2, we compare the performance differences between the MLP-based teacher model and our tree-based distilled student model. Our results show that using the distillation method, the student is capable of perfectly imitating the performance of the more complex teacher model.

5 Evaluation

By distilling the NN RL policy to an efficient decision tree, we now satisfy the decision-time constraint of 2μsec. We use the programmable CC interface of ConnectX-6Dx and deploy our tree-based policy on two live networking clusters. Our setup involves ConnectX-6Dx NICs connected through a Spectrum-2 switch over a lossy network with a link rate of 100 Gbps. We experimented on two operational clusters, detailed below. The tests were performed with the lightweight (tree-based) RL-CC version.

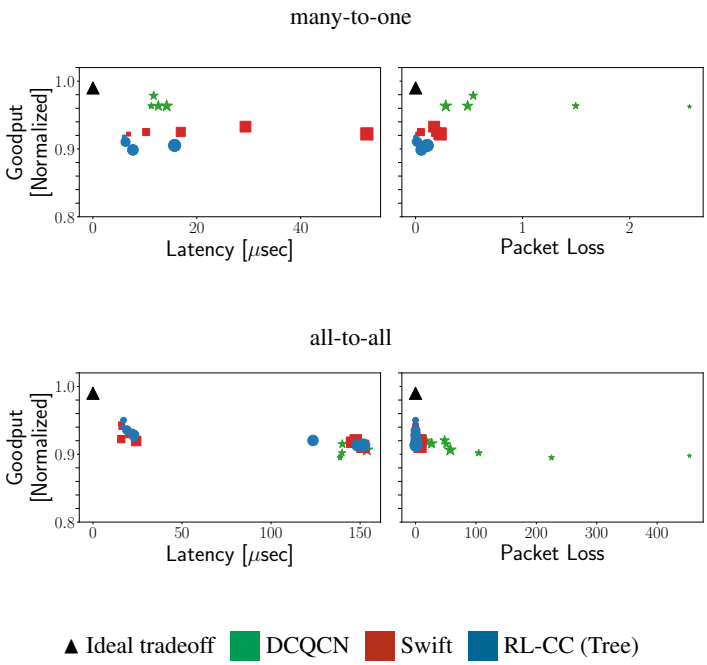

Figure 3: Large 64-host cluster. (Left) goodput vs. latency. (Right) goodput vs. packet loss. Marker size is proportional to the number of flows in the scenario. Packet loss is the total number of dropped packets divided by number of flows.

We compared the CC algorithms' ability to maintain a steady state and react to network changes. In the small cluster we performed many-to-one, all-to-all, and long-short tests. In the large cluster we performed many-to-one and all-to-all tests. We tested various configurations of number of flows per host and averaged across them. Beginning with steady-state scenarios, we compare RL-CC with DCQCN and Swift. In all cases, RL-CC sustains a high goodput bandwidth and low unfairness, similar to DCQCN and Swift. At the same time, RL-CC achieves a substantially lower packet latency and packet loss throughout most scenarios and performs competitively otherwise. Next, we evaluated RL-CC's ability to react to network changes and compare DCQCN, Swift, and RL-CC in the long-short test. RL-CC maintains the highest long bandwidth and competitive completion time throughout all the tests, outperforming Swift on both metrics. Additionally, while both Swift and RL-CC did not lose packets, DCQCN incurred packet loss in 100 flows. Moreover, DCQCN displays poor recovery performance with low long bandwidths in all scenarios. These results illustrate RL-CC's ability to rapidly adjust the transmission rate to changes in the network. Lastly, Fig. 3 depicts the tradeoff between goodput bandwidth and packet latency / packet loss. RL-CC presents the best overall tradeoff. In many-to-one, RL-CC achieves significantly lower latency and minimal packet loss at the expense of slightly lowering its goodput bandwidth. In all-to-all, RL-CC sustains high bandwidths while keeping similar packet latencies as DCQCN and Swift, with minimal packet loss. RL-CC is the only algorithm with consistent performance at different scales.

6 Conclusion and Future Directions

When considering real-time deployment in hardware, the agent, initially represented using a NN, required $450\mu sec$ to perform inference. Due to the rate of change within the datacenter, this resulted in an inability to control congestion. We introduced a method for distilling the agent to decision trees, reducing the inference time down to $0.9\mu sec$, a x500 improvement without performance loss. We then deployed RL-CC on a real multi-switch cluster, consisting of 64 hosts, running in real time on ConnectX-6Dx NICs. RL-CC demonstrated consistent high goodput and fairness while retaining low packet latency and minimal packet loss at different scales. Moreover, we showed the ability of RL-CC to generalize, out of the box, to new and unseen scenarios. An additional potential candidate to benefit from AI algorithms is the switch. There, not only congestion control is possible, but also power optimization, efficient routing, and more.

References

Ionut Anghel, Tudor Cioara, Dorin Moldovan, Ioan Salomie, and Madalina Maria Tomus. 2018. Prediction of Manufacturing Processes Errors: Gradient Boosted Trees Versus Deep Neural Networks. In *2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC)*. 29–36. https://doi.org/10.1109/EUC.2018.00012

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. 2020. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*. PMLR, 507–517.

Motti Beck and Michael Kagan. 2011. Performance evaluation of the RDMA over ethernet (RoCE) standard in enterprise data centers infrastructure. In *Proceedings of the 3rd Workshop on Data Center-Converged and Virtual Ethernet Switching*. 9–15.

Max Biggs, Wei Sun, and Markus Ettl. 2020. Model Distillation for Revenue Optimization: Interpretable Personalized Pricing.

Van-Phuc Bui, Trinh Van Chien, Eva Lagunas, Joël Grotz, Symeon Chatzinotas, and Björn Ottersten. 2021. Robust Congestion Control for Demand-Based Optimization in Precoded Multi-Beam High Throughput Satellite Communications. https://doi.org/10.48550/ARXIV.2109.02327

Rich Caruana and Alexandru Niculescu-Mizil. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. Association for Computing Machinery, New York, NY, USA, 161–168. https://doi.org/10.1145/1143844.1143865

Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. 2017. Interpretable Deep Models for ICU Outcome Prediction. *AMIA ... Annual Symposium proceedings. AMIA Symposium* 2016 (10 Feb 2017), 371–380. https://pubmed.ncbi.nlm.nih.gov/28269832 28269832[pmid].

Gil Einziger, Maayan Goldstein, Yaniv Sa'ar, and Itai Segall. 2019. Verifying Robustness of Gradient Boosted Models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence* (Honolulu, Hawaii, USA) *(AAAI'19/IAAI'19/EAAI'19)*. AAAI Press, Article 302, 8 pages. https://doi.org/10.1609/aaai.v33i01.33012446

Huiling Jiang, Qing Li, Yong Jiang, Gengbiao Shen, Richard O. Sinnott, Chen Tian, and Mingwei Xu. 2020. When Machine Learning Meets Congestion Control: A Survey and Comparison. *CoRR* abs/2010.11397 (2020). arXiv:2010.11397 https://arxiv.org/abs/2010.11397

Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan M. G. Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, David Wetherall, and Amin Vahdat. 2020. 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* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 514–528. https://doi.org/10.1145/3387514.3406591

Jiawei Li, Yiming Li, Xingchun Xiang, Shu-Tao Xia, Siyi Dong, and Yun Cai. 2020. TNT: An Interpretable Tree-Network-Tree Learning Framework using Knowledge Distillation. *Entropy* 22, 11 (2020). https://doi.org/10.3390/e22111203

Xuan Liu and Xiaoguang Wang. 2018. Improving the Interpretability of Deep Neural Networks with Knowledge Distillation. 905–912. https://doi.org/10.1109/ICDMW.2018.00132

Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-Based Congestion Control for the Datacenter. *SIGCOMM Comput. Commun. Rev.* 45, 4 (aug 2015), 537–550. https://doi.org/10.1145/2829988.2787510

Byron P. Roe, Hai-Jun Yang, Ji Zhu, Yong Liu, Ion Stancu, and Gordon McGregor. 2005. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 543, 2–3 (May 2005), 577–584. https://doi.org/10.1016/j.nima.2004.12.018

Alexander Shpiner, Eitan Zahavi, Omar Dahley, Aviv Barnea, Rotem Damsker, Gennady Yekelis, Michael Zus, Eitan Kuta, and Dean Baram. 2017. RoCE Rocks without PFC: Detailed Evaluation. In *Proceedings of the Workshop on Kernel-Bypass Networks* (Los Angeles, CA, USA) *(KBNets '17)*. Association for Computing Machinery, New York, NY, USA, 25–30. https://doi.org/10.1145/3098583.3098588

Jie Song, Haofei Zhang, Xinchao Wang, Mengqi Xue, Ying Chen, Li Sun, Dacheng Tao, and Mingli Song. 2021. Tree-Like Decision Distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13488–13497.

Chen Tessler, Yuval Shpigelman, Gal Dalal, Amit Mandelbaum, Doron Haritan Kazakov, Benjamin Fuhrer, Gal Chechik, and Shie Mannor. 2022. Reinforcement learning for datacenter congestion control. *ACM SIGMETRICS Performance Evaluation Review* 49, 2 (2022), 43–46.

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

Chongsheng Zhang, Changchang Liu, Xiangliang Zhang, and George Almpanidis. 2017. An Up-to-Date Comparison of State-of-the-Art Classification Algorithms. *Expert Syst. Appl.* 82, C (oct 2017), 128–150. https://doi.org/10.1016/j.eswa.2017.04.003

Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 523–536. https://doi.org/10.1145/2785956.2787484