Learning-Based Hierarchical Adaptive Congestion Control with Low Training Overhead

Jinbin Hu, Zikai Zhou, Shuying Rao, Yujie Peng, Bowen Bao, Chang Ruan
Changsha University of Science and Technology, ChangSha, China
{jinbinhu,baobowen,ruanchang}@csust.edu.cn
{zhouzikai,shuyingrao,pyj}@stu.csust.edu.cn

***Abstract*—Most congestion control mechanisms perform well in specific network environments, but none can consistently deliver good performance across all scenarios. Recently proposed frameworks based on reinforcement learning can flexibly select congestion control algorithms to adapt to dynamic changes in network conditions. However, frequently altering the congestion control mechanisms during relatively stable periods of the network actually leads to instability and unnecessary computational overhead. In this paper, we propose a hierarchical adaptive congestion control algorithm (HACC) to be resilient to the varying network. HACC dynamically selects the appropriate congestion control mechanism only when the current congestion control algorithm is not suitable for the current network state, rather than changing the congestion control scheme every training cycle to ensure network stability. The simulation results show that under different realistic workloads, HACC significantly reduces the computational overhead and improves throughput. Specifically, HACC reduces average overhead by 31% and improves throughput by up to 47%, 35%, 23%, and 15% compared to Cubic, Reno, BBR, and Antelope, respectively.**

***Index Terms*—Congestion Control, Deep Reinforcement Learning, Transport Protocols.**

I. Introduction

Since TCP was first introduced, a multitude of congestion control mechanisms have been developed. Nevertheless, it is important to understand that no single mechanism can consistently provide top-tier network performance across diverse environments and user demands. This can be primarily attributed to two reasons. First, most algorithms are generally optimized for specific network environments. Second, maintaining optimal performance with a single strategy is often difficult in dynamically changing conditions.

Machine learning-based congestion control approaches have the capability to autonomously adjust to a broad range of network scenarios, offering promising solutions to the challenges of network congestion. Examples of machine learning-based congestion control techniques are RemyCC [1], PCC-Vivace [2], DeepCC [3], and Orca [4], which each use distinct inputs and tactics to control transmitting rates.

Implementing these methods in real-world networks is challenging due to the unique characteristics of individual network environments and the continuous learning requirements, which compromise their reliability in new network setups. To address these issues, researchers propose a novel approach that uses deep learning to intelligently select suitable congestion control policies based on current network conditions. This approach, demonstrated by Antelope [5], does not directly configure transmission rates but instead dynamically adjusts congestion control for each data flow according to the network state. However, this adaptive strategy can introduce complexities, as traditional congestion control mechanisms stabilize networks gradually, and frequent adjustments may lead to network instability, ultimately affecting overall performance.

The HACC congestion control system is a novel idea presented in this research. It selects the optimal congestion control technique based on the state of the network using machine learning and a flexible hierarchical structure. Decision and policy modules are the two components involved in this. The decision module determines when the network isn't acceptable for the existing congestion control. When required, the policy module develops a fresh approach to congestion control that takes into account the state of the network.

The main contributions of this paper are as follows:

- We conduct experimental studies in gradually stabilizing network environments and bursty network environments. Traditional congestion control mechanisms experience performance degradation in bursty traffic scenarios, and existing congestion algorithms based on deep reinforcement learning often overreact to minor traffic changes in gradually stabilizing network environments, leading to performance degradation and excessive overhead.
- We propose HACC, a hierarchical adaptive network congestion control method that effectively addresses the issue of traditional congestion control mechanisms being unsuitable for highly variable network environments, reducing unnecessary computational overhead in the network. By utilizing DRL, HACC optimizes network performance while minimizing the computational burden caused by frequent switching of congestion control mechanisms.
- The simulation results show that HACC outperforms the Antelope scheme in terms of throughput and computational overhead, both in gradually stabilizing network environments and in environments with bursty traffic. For example, compared to Cubic, Reno, BBR, and Antelope, HACC demonstrates significant efficiency, with an average computational overhead reduction of approximately 31% and throughput improvements of 47%, 35%, 23%, and 15%, respectively.

This is the arrangement for the remainder of the paper. In Section II, we explain why we chose this particular design. We present the details of HACC and lay out its design in Section III. The experimental setup and NS-3 simulations, along with the findings and analysis, are detailed in Section IV. Correlated work is presented in Section V, and the paper is concluded in Section VI.

II. Motivation

In contemporary networks, the proliferation of internet-connected devices and the escalation of data-intensive applications have led to an unprecedented surge in network traffic. Traditional congestion control mechanisms often struggle to adapt efficiently to these fluctuating conditions, which can result in suboptimal performance and increased latency.

We carry out a comprehensive study in a simulated network environment to determine the effect of congestion control on performance in a different network setting than standard congestion management. As seen in Fig. 1, this assessment comprises a particular topology configuration with a K=4 Fat-Tree network topology. Under this configuration, 16 links overall, each with a 50 Gbps capacity, are added to each group of switches at the aggregation layer to enable connectivity to the server infrastructure. Furthermore, each link between the switches in the aggregation layer and the core layer has a bandwidth of 200 Gbps.

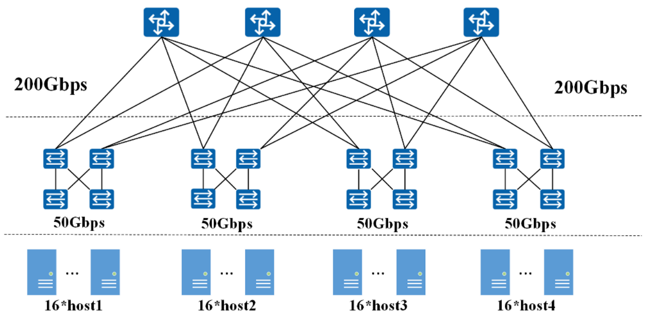


Fig. 1. Fat-Tree topology

A. Observation 1

Deploying deep reinforcement learning (DRL) in adapting congestion control strategies offers a promising solution to this challenge. DRL allows for real-time adjustments based on the current network conditions, theoretically improving overall network efficiency. However, this approach also brings significant challenges. Frequent adjustments to congestion control strategies can lead to network instability, as continuous changes may disrupt the balance needed for optimal performance. To illustrate this, experiments and evaluations are conducted using the Ns-3 simulation tool in the Fat-Tree topology shown in Fig. 1. Traffic is sent from different servers in host2, host3, and host4 to the first server in host1, gradually increasing the traffic load until a certain point where the load increase is stopped. During this period, we induce network congestion and observe the performance of different congestion control mechanisms, including Antelope and BBR.

The results shown in Fig. 2(a) indicate that both Antelope and BBR experience a significant decrease in throughput as the traffic load gradually increases, with BBR showing a larger decrease than Antelope. However, due to frequent policy adjustments, Antelope temporarily performs worse than BBR. This highlights the delicate balance between responsiveness and stability in adaptive congestion control mechanisms. Therefore, an effective DRL-based congestion control policy must carefully manage these adjustments to avoid introducing further instability.

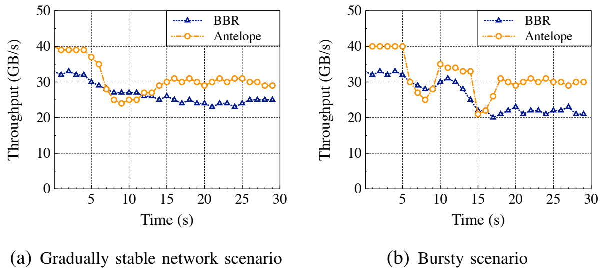


Fig. 2. Throughput performance in different scenarios

B. Observation 2

Network environments characterized by small data flows present a different set of challenges for congestion control mechanisms. Small flows are common in web browsing, IoT communications, and other applications where quick, short bursts of data are more typical than sustained large transfers. In these scenarios, the primary objective shifts from maximizing throughput to minimizing latency and ensuring swift delivery of packets.

Our experiment focuses on measuring the performance of Antelope and the traditional mechanism BBR in a burst traffic environment. The results reveal significant throughput fluctuations for all tested mechanisms. This variability is particularly detrimental in burst traffic environments where consistency and low latency are crucial. In the Fat-Tree topology shown in Fig. 1, servers from host2, host3, and host4 send steady traffic to the first server in host1, but periodically send a large amount of small bursts to test the performance of Antelope and BBR under burst traffic.

As shown in Fig. 2(b), the throughput of Antelope exhibites noticeable fluctuations at times of 5s and 15s. Although Antelope is generally effective in high-traffic environments, its throughput can experience significant variations in burst traffic situations. Due to the characteristics of its algorithm, Antelope can gradually recover its expected performance. However, such fluctuations may lead to delays and inefficiencies in practical applications.

III. System Design

A. HACC Overview

In the context of the HACC architectural framework, as depicted in Fig. 3, the system's functioning necessitates the acquisition and observation of data inputs for assessing the current state of the Reinforcement Learning (RL) agent. This assessment involves comparing the current state to the last state at which an action was executed. The system initiates an action only when it discerns a significant disparity, signaling the activation of the policy generator module to execute the next action. Furthermore, the RL agent maintains a connection to the environment, allowing for the retrieval of updated state and reward values. This connectivity serves to augment the agent's knowledge and enhances its overall performance.

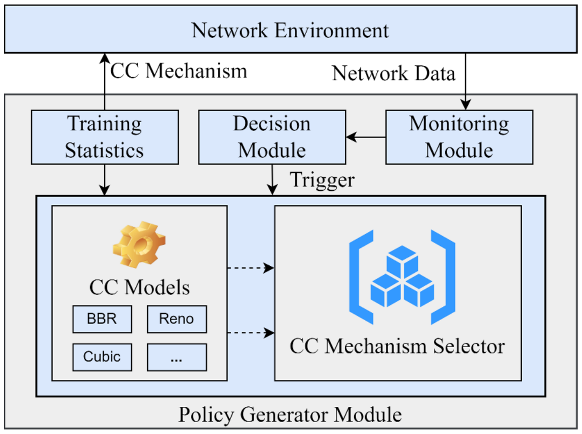


Fig. 3. HACC Overview

B. Design Details

HACC comprises three crucial modules: the Monitoring Module, the Decision Module, and the Policy Generator Module. The Monitoring Module is responsible for retrieving environmental data and transmitting it to the Decision Module for assessing the ongoing suitability of the current congestion control policy for the network environment. If adjustments to the congestion control policy are deemed necessary, the Policy Generator Module utilizes the current status information to formulate the most appropriate congestion control policy.

- **Monitoring Module:** The Monitoring Module is pivotal in gathering and analyzing real-time network data, ensuring that the system has an accurate and up-to-date understanding of current conditions. This module continuously monitors various network parameters, such as RTT, packet loss, throughput, minimum delay, and maximum pacing rate, as outlined in Table I. By collecting these statistics, the Monitoring Module provides a comprehensive view of network performance, identifying trends and anomalies that may indicate congestion or other issues. This data is then transmitted to the Decision Module for further analysis. The Monitoring Module operates continuously, ensuring that the system can respond swiftly to changing network conditions. By providing accurate and timely data, it plays a critical role in maintaining optimal network performance.
- **Decision Module:** Within the dynamic network context, the Decision Module assesses the current congestion control situation and determines if an adjustment to the congestion control policy is warranted. This module analyzes the data received from the Monitoring Module, comparing the current network state to predefined performance thresholds and historical data. If the analysis indicates a significant deviation from expected performance, the Decision Module triggers the Policy Generator Module to formulate a new congestion control strategy.The Decision Module meticulously documents each instance it triggers the Policy Generator Module, recording the current state and the implemented congestion control policy. This documentation allows for ongoing evaluation and refinement of the decision-making process, ensuring that the system learns from each adjustment and improves over time. The Decision Module's ability to make informed decisions based on real-time data is crucial for maintaining network stability and performance.

TABLE I
STATISTICS GENERATED BY THE MONITORING MODULE.

Category	Meaning
RTT	The mean value of the smoothed RTT
Loss	The number of lost packets
Throughput	The mean value of sending throughput
$\text{Delay}_{\text{min}}$	The minimum packet delay so far
Pacing Rate	The maximum pacing rate so far

- **Policy Generator Module:** In the HACC architecture, the Policy Generator Module plays a central role. This module includes sub-models for various congestion control strategies, allowing it to dynamically select the most appropriate one for the current network conditions. When the Decision Module indicates a need for policy adjustment, the Policy Generator Module evaluates the current status information and uses it to predict the reward for different congestion control strategies. For instance, if the system needs to predict the reward for a specific flow using the BBR control strategy, the BBR sub-model is utilized. This prediction helps the RL agent select the most suitable congestion control mechanism by choosing the one with the highest expected reward. The Policy Generator Module's ability to adaptively select and implement the best congestion control strategy ensures that the network can maintain optimal performance even under varying conditions.
- **Reward Function:** In the realm of deep reinforcement learning, the reward function assumes paramount significance, exerting a substantial influence on the system performance. Within our research, we introduce Equation 1 as a quantitative metric, denoted as the reward function, to assess the efficacy of each congestion control mechanism.

$$\hat{R} = \left(\frac{\text{TPS} \cdot \eta \cdot \text{loss}}{\text{delay}}\right)\left(\frac{\text{pacing\_rate}_{\text{max}}}{\text{delay}_{\text{min}}}\right) - \alpha \times \text{trigger} \quad (1)$$

In a related vein, Giessler's investigation introduces a compelling metric termed power, defined as throughput divided by latency. Additionally, we incorporate a "loss" parameter to fine-tune the reward function, with parameter n playing a pivotal role in calibrating the influence of packet loss on this function. The hyperparameter α holds a central role in our model training, with a fixed value of 0.05. This parameter governs the imposition of penalties during training, where an increase α value reduces policy generator activations, emphasizing system stability. The decision module exercises discretion in triggering the policy generator base on α. Conversely, a decrease in α leads to more frequent activations, affording finer control over network dynamics.

$$\text{delay}' = \begin{cases} \text{delay}_{\text{min}}, & \text{if delay}_{\text{min}} \le \text{delay} \le \delta \times \text{delay}_{\text{min}} \\ \text{delay}, & \text{o.w.} \end{cases} \quad (2)$$

However, optimizing both maximum throughput and minimal latency concurrently presents a complex challenge. Larger flows tend to prioritize throughput, while smaller flows prioritize latency. To tackle this issue, we introduce a factor δ (where $\delta \geq 1$) in Equation 2. Initially set to 2 when the connection is established, δ grows 5 exponentially as data units are received, with each unit doubling δ value. Consequently, the reward function's focus shifts from latency sensitivity to throughput sensitivity. It is worth noting that a similar approach is adopted in the Orca framework.

IV. Performance Evaluation and Analysis

A. Evaluation Setup

We employ four workloads: Web Server (WSv), Cache Follower (CF), Web Search (WSc), and Data Mining (DM) with distributions that closely resemble real-world workloads. These workloads span a wide range of average flow sizes, from 64KB to 7.41MB, with more than half of flows being less than 10KB.Four realistic workloads' flow size distributions are displayed in Table II.

TABLE II
Flow size distribution of realistic workloads

	Web Server	Cache Follower	Web Search	Data Mining
0-10KB	63%	50%	49%	78%
10KB-100KB	18%	3%	3%	5%
100KB-1MB	19%	18%	18%	8%
$>$ 1MB	0	29%	20%	9%
Average flow size	64KB	701KB	1.6MB	7.41MB

Our suggested approach's model design makes use of a hierarchical policy model built with the Pytorch [8] framework. The network environment's ability to make decisions and exercise control is greatly aided by this hierarchical policy paradigm. The NS-3 network simulator, a popular tool for network research and development, is utilized to build a realistic and comprehensive training environment.

B. Performance In Gradually Stable Network Environments

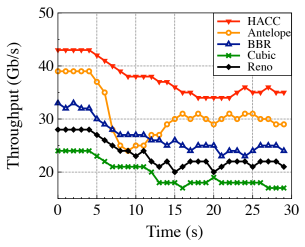
In this section, our goal is to evaluate the stability of HACC through experiments in four different gradually stabilizing network scenarios. We conducted the relevant experiments in the Fat-Tree topology shown in Fig. 1. In this topology, servers from host1, host2, and host3 send traffic to the servers in host4. The traffic is divided into four different scenarios: Web Server, Cache Follower, Web Search, and Data Mining, with traffic distribution ratios according to Table II. We conducted four throughput tests in these different environments for five congestion control mechanisms: HACC, Antelope, BBR, Reno, and Cubic.

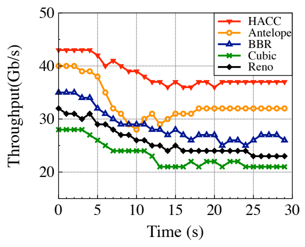
As shown in Fig. 4, we found that in scenarios with a higher proportion of small traffic, particularly in the Web Server and Data Mining scenarios, Antelope experienced a sharp drop in throughput after t=5s. Overall, compared to other congestion control mechanisms like Cubic and BBR, the Antelope algorithm generally demonstrates superior performance. However, in short traffic situations, its sensitivity to network fluctuations becomes more pronounced, leading to performance variations. Specifically, in scenarios involving short traffic, its overall performance at certain times is lower than that of Cubic and BBR. HACC addresses Antelope's overreaction issue by using a decision module to determine whether to change mechanisms. As shown in Fig. 6(a), in different scenarios, HACC reduces unnecessary overhead by approximately 31% compared to Antelope.

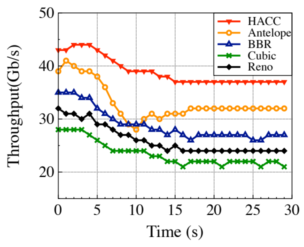
HACC consistently demonstrates superior and stable performance, consistently outperforming leading congestion control algorithms such as Cubic. HACC achieves throughput rates that are 47%, 35%, 23% and 8% over Cubic, Reno [8], BBR and Antelope, respectively. Its flexible layered control architecture, particularly its isolation of the policy module from the agent, proves advantageous as the network stabilizes. This approach ensures a thorough assessment of the network environment before making frequent adjustments to the congestion control mechanism, thereby enhancing stability and preventing network disruptions cause by sudden congestion control changes, which can negatively affect overall network performance.

C. Performance Under Bursty Scenario

In this section, our goal is to evaluate the stability of HACC after switching in different network scenarios. We conduct experiments in the Fat-Tree topology shown in Fig. 1. In this topology, servers from host2, host3, and host4 send traffic to the server in host1. We first send traffic based on the traffic ratio of the WebSearch scenario shown in Table II, then switch to the DataMining scenario, and finally switch back to the WebSearch scenario. We call this scenario SECN1. Similarly, we send traffic based on the traffic ratio of the CacheFollower scenario, then switch to the DataMining scenario, and finally switch back to the CacheFollower scenario. We call this scenario SECN2. In the same way, we send traffic based on the traffic ratio of the WebSearch scenario, then switch to the WebServer scenario, and finally switch back to the WebSearch scenario. We call this scenario SECN3. The final scenario is where we send traffic based on the traffic ratio of the CacheFollower scenario, then switch to the WebServer


(a) Throughput under Web Server


(b) Throughput under Cache Follower


(c) Throughput under Web Search

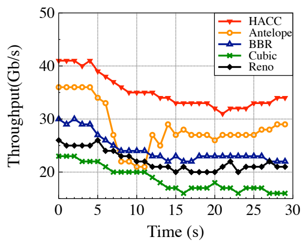
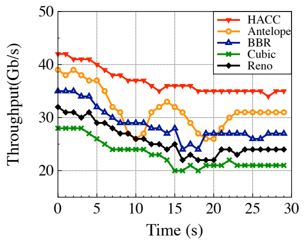
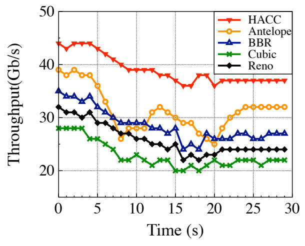
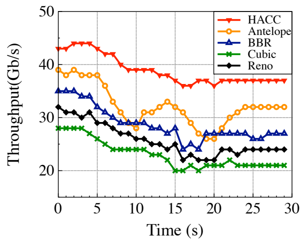

(d) Throughput under Data Mining

Fig. 4. Performance in different stable network environments


(a) Throughput under ($SECN_1$)


(b) Throughput under ($SECN_2$)


(c) Throughput under ($SECN_3$)

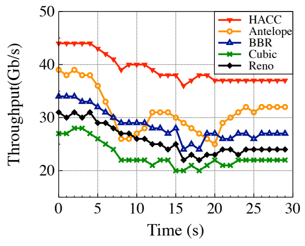
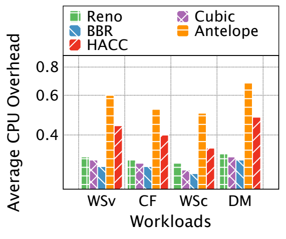

(d) Throughput under ($SECN_4$)

Fig. 5. Throughput performance in switching bursty scenarios


(a) Gradually stable network scenario

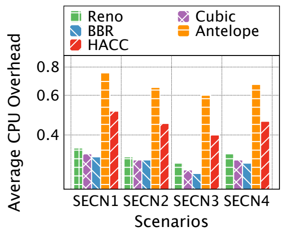

(b) Bursty scenario

Fig. 6. Average CPU overhead under different environments

scenario, and finally switch back to the CacheFollower scenario. We call this scenario SECN4. We conduct throughput tests for these four experimental scenarios.

As shown in Fig. 5, we find that at t1=10s and t2=15s, Antelope experienced a significant drop in throughput. Compared to other congestion control mechanisms like Cubic and BBR, the Antelope algorithm generally demonstrates relatively superior performance. However, after sudden traffic switches, its sensitivity to network fluctuations can sometimes be overly reactive or too slow, leading to performance variations. For a period after switching traffic patterns, Antelope's overall performance is lower than that of Cubic and BBR. Similarly, as depicted in Fig. 6(b), in these four distinct exchange traffic scenarios, HACC reduces the compute overhead by an average of approximately 30% when compared to Antelope.

In contrast, HACC consistently exhibits excellent and stable performance. Its flexible layered control architecture, especially its separation of the policy module from the agent, has proven advantageous when the network is stable. This approach ensures a thorough assessment of the network environment before making frequent adjustments to the congestion control mechanism, enabling quick switches when needed and discarding unnecessary switches while continuing to monitor.

V. RELATED WORK

TCP varieties. The fact that CC algorithms can be tailored to suit various situations is not a new discovery to us. Cellular networks are the target market for C2TCP [26] and PBE-CC [28], for example. Similar to this, Explicit Congestion Notification is used in the architecture of DCTCP [19], Timely [18], and Swift [29] for datacenter networks. Instead of trying to create new CC algorithms, we take advantage of this fact in our work and use it to dynamically choose the optimal method for the observed network conditions as needed.

Congestion Management. After more than thirty years, congestion control has continued to be a hot topic for research in the field of networking. Modern congestion control protocols, such DCTCP [19], DCQCN [20], and their improved versions, mostly depend on the ECN mechanism to enable rate control. For high-performance DCNs to continue operating, the ECN mechanism is essential. But most traditional solutions use static techniques to define the ECN thresholds, which are not flexible enough to handle dynamic changes in network conditions. This restriction frequently results in less than ideal performance consequences.

Learning-based Network Optimization. Optimization of network performance and parameterization of congestion management techniques have been popular applications of learning-based methodologies. Dynamic rate adjustment and DRL are two examples of such approaches used by Remy [1], Vivace [2], and Aurora [25]. For handling unpredictable traffic patterns, Orca blends learning-based techniques with classic TCP Cubic. While ACC [24] suggests employing DRL to automatically modify ECN settings, the majority of learning-based techniques concentrate on modifying the sending rate in response to feedback.

VI. Conclusion

This paper introduces HACC, an innovative congestion control solution that uses DRL for dynamic mechanism selection based on a decision module evaluation. HACC's flexible layered control structure, including the decision module and policy generation module, minimizes the need for frequent policy adjustments and enhances network stability. Empirical evaluations indicate that HACC significantly outperforms existing methods, achieving notable latency reduction and throughput improvement. In different network traffic patterns, HACC reduces average overhead by approximately 31% and improves throughput by 47%, 35%, 23%, and 15% compared to Cubic, Reno, BBR, and Antelope, respectively.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (62472050, 62102046), by the New Generation Information Technology Innovation Project 2023 (2023IT271), and by the open research fund of Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education (JZNY202308), and by the Science and Technology Project of Hunan Provincial Department of Water Resources (XSKJ2024064-36).

References

[1] K. Winstein, H. Balakrishnan. Tcp ex machina: Computer-generated congestion control. In proc. ACM SIGCOMM, 2013, pp. 123-134.
[2] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, M. Schapira. PCC Vivace: Online-Learning Congestion Control. In proc. USENIX NSDI, 2018, pp. 343-356.
[3] S. Abbasloo, C. Yen, H. Chao. Make tcp great (again?!) in cellular networks: A deep reinforcement learning approach. In proc. arXiv preprint arXiv, 2019, pp. 1912.11735.
[4] S. Abbasloo, C. Yen, H. Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In Proc. ACM, 2020, pp. 632-647.
[5] N. Cardwell, Y. Cheng, C. Gunn, S. Yeganeh, V. Jacobson. BBR: Congestion-based congestion control. In Communications of the ACM, 2017, pp. 60(2) 58-66.
[6] S. Ha, I. Rhee, L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. In proc. ACM SIGOPS operating systems review, 2008, pp. 42(5) 64-74.
[7] T. Henderson, S. Floyd, A. Gurtov, Y. Nishida. The NewReno modification to TCP's fast recovery algorithm. In Technical Report, 2012, No. rfc6582.
[8] A. Paszke, S. Gross, F. Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In proc. Advances in neural information processing systems, 2019, pp. 32.
[9] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. RDMA over Commodity Ethernet at Scale. In Proc. ACM SIGCOMM, 2016, pp. 202-215.
[10] W. Bai, A. Agrawal, A. Bhagat, M. Elhaddad, N. John, J. Padhye, et al. Empowering Azure Storage with 100×100 RDMA. In Proc. USENIX NSDI, 2023, pp. 49-67.
[11] J. Xue, M. Chaudhry, B. Vamanan, T. Vijaykumar, M. Thottethodi. Dart: Divide and Specialize for Fast Response to Congestion in RDMA-based Datacenter Networks. IEEE/ACM Transactions on Networking, 2020, pp. 28(1) 322-335.
[12] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and Thomas Moscibroda. multipath Transport for RDMA in Datacenters. In Proc. USENIX NSDI, 2018, pp. 357-371.
[13] C. Tian, B. Li, L. Qin, J. Zheng, J. Yang, W. Wang, G. Chen, and W. Dou. P-PFC: Reducing Tail Latency with Predictive PFC in Lossless Data Center Networks. IEEE Transactions on Parallel and Distributed Systems, 2020, pp. 31(6) 1447-1459.
[14] J. Hu, H. Shen, X. Liu, J. Wang. RDMA Transports in Datacenter Networks: Survey. IEEE Network, 2024, DOI: 10.1109/MNET.2024.3397781.
[15] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and Thomas Moscibroda. multipath Transport for RDMA in Datacenters. In Proc. USENIX NSDI, 2018, pp. 357-371.
[16] C. Tian, B. Li, L. Qin, J. Zheng, J. Yang, W. Wang, G. Chen, and W. Dou. P-PFC: Reducing Tail Latency with Predictive PFC in Lossless Data Center Networks. IEEE Transactions on Parallel and Distributed Systems, 2020, pp. 31(6) 1447-1459.
[17] J. Hu, Y. He, W. Luo, J. Huang , J. Wang. Enhancing Load Balancing with In-network Recirculation to Prevent Packet Reordering in Lossless Data Centers. IEEE/ACM Transactions on Networking, 2024, 32(5): 4114-4127.
[18] R. Mittal, V. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats. TIMELY: RTT-based Congestion Control for the Datacenter. In Proc. ACM SIGCOMM, 2015, pp. 537-550.
[19] M. Alizadeh, A. Greenberg, D. Maltz, et al. Data Center TCP (DCTCP). In Proc. ACM SIGCOMM, 2010, pp. 63-74.
[20] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion Control for Large-Scale RDMA Deployments. In Proc. ACM SIGCOMM, 2015, pp. 532-536.
[21] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren. Re-architecting Congestion Management in Lossless Ethernet. In Proc. USENIX NSDI, 2020, pp. 19-36.
[22] K. Qian, W. Cheng, T. Zhang, and F. Ren. Gentle Flow Control: Avoiding Deadlock in Lossless Networks. In Proc. ACM SIGCOMM, 2019, pp. 75-89.
[23] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye. ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY. In Proc. ACM CoNEXT, 2016, pp. 313-327.
[24] S. Yan, X. Wang, X. Zheng, Y. Xia, D. Liu, and W. Deng. ACC: Automatic ECN tuning for high-speed datacenter networks. In proc. ACM SIGCOMM 2021 Conference, 2021, pp. 384-397.
[25] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar. A deep reinforcement learning perspective on internet congestion control. In International Conference on Machine Learning, 2019, pp. 3050-3059.
[26] S. Abbasloo, Y. Xu, and H. Jonathan. C2tcp: A flexible cellular tcp to meet stringent delay requirements. IEEE Journal on Selected Areas in Communications, 2019, pp. 37(4) 918–932.
[27] J. Hu, C. Zeng, Z. Wang, J. Zhang, K. Guo, H. Xu, J. Huang, K. Chen. Load Balancing with Multi-level Signals for Lossless Datacenter Networks. IEEE/ACM Transactions on Networking, 2024, 32(3): 2736-2748.
[28] Y. Xie, F. Yi, and K. Jamieson. Pbe-cc: Congestion control via endpoint-centric. Physical-layer bandwidth measurements, 2002, pp. 03475.
[29] G. Kumar, N. Dukkipati, K. Jang, H. Wassel, X. Wu, B. Montazeri, Y. Wang, K. Springborn, C. Alfeld, M. Ryan, et al. Swift: Delay is simple and effective for congestion control in the datacenter. In Proc. ACM, 2020, pp. 514–528.
[30] J. Hu, Z. Zhou, J. Zhang. Lightweight Automatic ECN Tuning Based on Deep Reinforcement Learning with Ultra-low Overhead in Datacenter Networks. IEEE Transactions on Network and Service Management (TNSM), 2024, DOI: 10.1109/TNSM.2024.3450596.
[31] Y. Kang, X. Yang, B. Pu, X. Wang, H. Wang, Y. Xu, P. Wang. HWOA: An Intelligent Hybrid Whale Optimization Algorithm for Multi-objective Task Selection Strategy in Edge Cloud Computing System. World Wide Web, 2022, DOI:10.1007/s11280-022-01082-7.
[32] W. Dou, B. Liu, C. Lin, X. Wang, X. Jiang, L. Qi. Architecture of Virtual Edge Data Center with Intelligent Metadata Service of a Geo-distributed File System. Journal of Systems Architecture, 2022, DOI:10.1016/j.sysarc.2022.102545.
[33] J. HU, J. HUANG, J. WANG, et al. A Transmission Control Mechanism for Lossless Datacenter Network Based on Direct Congestion Notification[J]. Acta Electronica Sinica, 2023,51(9): 2355-2365.