SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

TT-PECN: Temporal Transformer Based Predictive Explicit Congestion Notification For Data Center Networks

Hardik K. Molia

Associate Professor, Computer Engineering Department, Government Engineering College, Bhavnagar, Gujarat, India. E-mail: hardik.molia@gmail.com

Abstract
Data Center Networks (DCNs) handle a large number of data flows, which can cause congestion, packet loss, and delay. Existing Explicit Congestion Notification (ECN) methods mainly react to the current queue condition and may not respond early enough to upcoming congestion. This research paper proposes TT-PECN (Temporal Transformer based Predictive Explicit Congestion Notification) to provide proactive congestion notification in DCNs. The proposed method uses a Temporal Transformer to learn network traffic patterns from time-series data and predict future queue conditions in terms of target queue length. The prediction is used to make ECN marking decision. The model is trained using data generated in NS-3.34 with DCTCP (Data Center TCP) and RED with ECN support. Different link bandwidths and TCP flow levels are used for evaluation. The performance of TT-PECN with DCTCP is evaluated with existing DCTCP and TCP NewReno. Average per-flow throughput and average per flow packet delivery ratio are used as performance metrics. It has been observed that under various distinct scenarios, TT-PECN with DCTCP performs better as compared to existing DCTCP and TCP NewReno.

Keywords: Data Center Networks, Temporal Transformer, Explicit Congestion Notification

1. Introduction

In recent years, Data Center Networks (DCNs) [1-3] have been emerged as important parts of modern cloud computing and online services. They connect a large number of servers and allow them to exchange data quickly. Many applications in data centers generate a large number of simultaneous network flows. As a result, network congestion can occur when the amount of incoming traffic becomes higher than the available network capacity. Network congestion can increase queue length, packet loss, and communication delay. It can also reduce the overall throughput of the network. Therefore, effective congestion management is important for achieving high performance and low latency in DCNs.

Explicit Congestion Notification (ECN) [4-6] is widely used to manage congestion without relying only on packet drops. ECN allows network devices to mark packets when congestion is detected. The receiver informs the sender about these marks, and the sender can then reduce its transmission rate. Data Center TCP (DCTCP) [7-10] uses ECN information to control the congestion window and improve network performance. However, conventional ECN-based mechanisms generally make decisions based on the current state of the queue. As a result, most of the time, congestion may already have increased before the ECN marking becomes effective. Active Queue Management (AQM) mechanisms such as Random Early Detection (RED) can mark packets before a queue becomes completely full. However, the marking decision is still mainly based on the current or recent queue condition. These approaches do not explicitly predict how the queue will change in the near future. A predictive approach can provide an opportunity to identify upcoming congestion earlier and take action before severe congestion occurs.

Machine learning techniques can be used to learn patterns from network traffic and congestion-related measurements. In particular, Transformer-based models are effective at learning relationships between observations in a sequence. This makes them suitable for time-series network data, where the current network condition is related to previous conditions. In this work, a Temporal Transformer based Predictive Explicit Congestion Notification (TT-PECN) is proposed for Data Center Networks. TT-PECN uses a Temporal Transformer to predict future queue condition from recent network observations. The predicted congestion information is then used to make proactive ECN marking decisions. Thus, instead of responding only to the current queue condition, TT-PECN attempts to take action before significant congestion develops. The performance of TT-PECN with DCTCP is evaluated with existing DCTCP and TCP NewReno. Average per-flow throughput and average per flow packet delivery ratio are used as performance metrics. It has been observed that under various distinct scenarios, TT-PECN with DCTCP performs better as compared to existing DCTCP and TCP NewReno.

This research paper is organized into 5 sections. Section 2 discusses the related work as a part of literature review. Section 3 discusses TT-PECN architecture along with various algorithms. Section 4 discusses performance analysis. Section 5 discusses conclusions and future work.

2. Related Work

2.1 Overview

Transmission Control Protocol (TCP) is a transport-layer protocol used to provide reliable data communication between two endpoints. It controls the rate at which data is sent and adjusts the transmission rate when network congestion occurs. Traditional TCP variants such as Tahoe, Reno, NewReno, and Cubic mainly use packet loss, delay, and acknowledgement information to detect congestion and adjust the congestion window. These approaches work well for general-purpose networks, but different types of networks have different traffic patterns and performance requirements, so a single TCP algorithm may not provide the best performance in every environment. TCP with Explicit Congestion Notification (ECN) provides an alternative by allowing network devices to mark packets when congestion is detected instead of waiting for packet drops. Active Queue Management (AQM) methods such as Random Early Detection (RED) can be used with ECN to mark packets before the queue becomes full. Machine learning-based TCP approaches further use network data to predict congestion and improve transmission decisions.

One important environment where congestion control is especially critical is the Data Center Network (DCN). DCNs connect a large number of servers, switches, and storage systems and are widely used in cloud computing, online services, distributed applications, and large-scale data processing. DCNs usually carry a large number of short and long flows simultaneously, making congestion control important for maintaining high throughput and low delay. TCP is therefore an important part of DCN communication because it provides reliable and controlled data transfer between servers. However, packet loss and queue buildup can significantly affect DCN performance. TCP with ECN is particularly useful in DCNs because it can provide early congestion information without requiring packet loss to react quickly to congestion and maintain low queue occupancy.

2.2 Recent ECN Advances

Explicit congestion notification (ECN) is a signaling mechanism that enables a router network path to mark packets for signalling incipient congestion to the sender so it can act accordingly. Over the years, ECN has been widely adapted.

The reassessment of ECN adoption on the Internet [11] specific focusing on infrastructure, internet paths, client networks done. The findings focused on majority of web servers are ECN supported, IPv6 is more ECN friendly and existence of ECN bleaching. DDT (Dual Dynamic Thresholds) [12] that is an active queue management algorithm (AQM) is proposed to achieve the flow-level fairness for DCNs with coexisting heterogeneous TCP traffic. It monitors real time switch queue and tunes distance between ECN-marking and packet-dropping thresholds to mitigate the aggressiveness difference between the ECN-enabled and ECN-disabled TCPs. Dual Congestion Control Signals (DCCS) [13] combines ECN and delay signals to handle burst traffic, reduce packet loss and end-to-end delay, and improve throughput fairness in DCNs. Proxy Buffer with Gradient-ECN [14] addresses RTT-induced unfairness between inter- and intra-DC flows by splitting the congestion-control loop at the receiver-side gateway. By enabling earlier ECN-Echo notifications based on queue-growth gradients, G-ECN stabilizes queue size and improves fairness compared with conventional ECN. DiffECN (Differential ECN) [15] uses differential ECN marking to identify and restrict congestion-causing flows while protecting unaffected flows from unnecessary throttling. AMRT [16]

uses Anti-ECN marking to detect spare bandwidth and dynamically increase the sending rate, achieving near-zero queueing delay while maintaining high link utilization. Micro-burst Aware ECN [17] mitigates micro-burst-induced ECN mismarking by decoupling threshold setting from marking and using a finer-grained double-threshold, dequeue-based ECN scheme. PET [18] is a learning-based automatic ECN tuning scheme using multi-agent IPPO to dynamically adjust ECN thresholds based on congestion, traffic patterns, queue conditions, and flow characteristics. ML-ECN [19] is a multi-level probabilistic ECN marking scheme designed for multi-queue switches, separating mice, medium, and elephant flows to improve fairness. It uses differentiated thresholds and probabilistic marking to achieve low latency for small flows and high throughput for large flows, outperforming existing schemes in simulations.

2.3 Reinforcement Learning based TCP Variants with / without ECN

This section discusses some of the recent Machine Learning (ML)-based TCP variants. TCP variants based on supervised learning have limited ability to perform effectively in completely unseen network scenarios, as their performance largely depends on the network conditions represented in the training data. Such approaches are therefore more suitable for known network environments similar to those encountered during training. However, modern networks can change rapidly due to variations in traffic patterns, congestion levels, and network topology. Consequently, TCP variants need to be capable of adapting their decision-making to dynamically changing and previously unseen network conditions. Reinforcement Learning (RL) provides a promising approach in this regard, as RL algorithms interact with the network environment, observe its changing state, and continuously improve their decision-making policy based on the resulting feedback. This adaptive learning capability makes RL-based approaches particularly suitable for TCP congestion control in diverse and dynamic network scenarios. Some of the recent Deep Reinforcement Learning (DRL) based TCP Variants are discussed below.

ASCRL [20] is an application specific congestion control for TCP based on DRL. This framework that adapts to changing network conditions while allowing applications to specify diverse objectives such as throughput, latency, packet loss, and jitter. ProCC [21] uses Programmatic RL to automatically discover interpretable TCP congestion-control policies instead of black-box neural networks. TCP-Int [22] uses RL with ns3-gym to dynamically regulate the congestion window in Wireless Mesh Networks (WMNs). A Deep Q-Network (DQN) [23] is proposed to adjust the congestion window based on real-time network conditions and evaluated for Dumbbell Network Topology. Pacemaker-CC [24] is a DQN-based TCP congestion controller that uses an adaptive, ceiling-normalized throughput reward to stabilize learning across varying bandwidth and RTT conditions. DECC [25] is for Data Center Networks that combines ECN feedback with multi-objective learning to achieve high bandwidth utilization and low queuing delay. Lightweight Automatic ECN Tuning based DRL with ultra-low overhead [26] focused avoiding consumption of a large number of computational resources that make deployment difficult.

3. TT-PECN Architecture

3.1 Overview

The proposed TT-PECN (Temporal Transformer based Predictive Explicit Congestion Notification) architecture is designed to predict future network congestion and use this prediction to perform proactive ECN marking in DCNs (Data Center Networks). The conventional ECN mechanisms react only after the queue length exceeds a predefined threshold whereas TT-PECN predicts future congestion using a Temporal Transformer model and dynamically adjusts the ECN mark probability before severe congestion occurs. TT-PECN consists of three major modules: Dataset Generation, Temporal Transformer Model Training, and Congestion Prediction for ECN Decision Making. The first two modules are performed offline, while the third module operates online during ongoing network traffic.

3.2 Time-Series Dataset Generation

TT-PECN model is trained using a comprehensive dataset generated through network simulations in NS-3.34. The dataset generation topology is inspired by the DCTCP evaluation topology, which is available as an example in NS-3.34. Fig. 1 illustrates the custom multi-bottleneck DCTCP topology adopted for dataset generation. The topology consists of three sender nodes (S_1, S_2, and S_3), two aggregation switches (T_1 and T_2), and two receiver nodes (R_1 and R_2). The original topology represents only a single network configuration; therefore, multiple variations are designed to improve the diversity of the training dataset. These variations include different link speeds and numbers of concurrent TCP flows while keeping the RED/ECN threshold settings fixed to represent a wide range of congestion scenarios in data center networks. Data Center TCP (DCTCP) is used as the transport layer protocol, while RED with ECN support is employed as the baseline Active Queue Management (AQM) mechanism. During

each simulation, network statistics are collected on reception of every Acknowledgement (ACK). The complete list of parameters considered for network configurations is presented in Table 1. There are 81 unique network configurations (3 × 3 × 3 × 3) generated by varying the access link bandwidth and the number of concurrent TCP flows between S1–R1, S2–R2, and S3–R1. Each configuration was simulated for 5 minutes, resulting in a total simulation time of 405 minutes (6 hours 45 minutes).

Fig. [1] - Custom Multi-Bottleneck DCTCP Topology

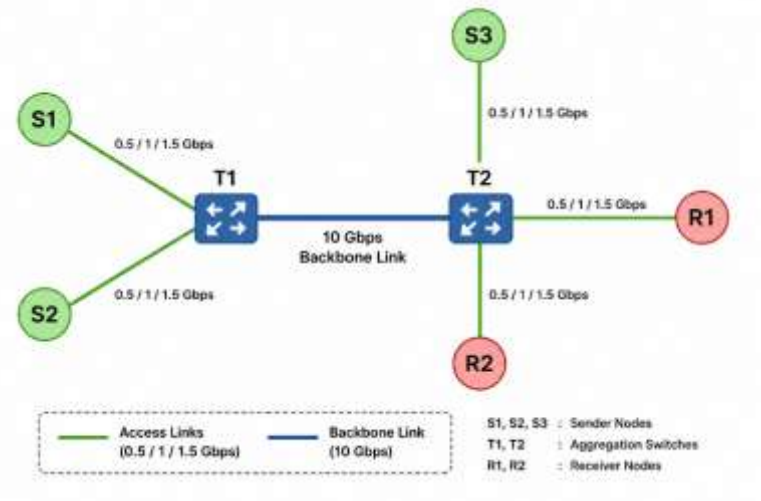


Table [1] - Parameters for Network Configurations - Training

Parameter	Value
Network Topology	Custom Multi-Bottleneck DCTCP Topology
Sender Nodes	S1, S2, S3
Intermediate Switches	T1, T2
Receiver Nodes	R1, R2
Access Link Bandwidth	0.5, 1, 1.5 Gbps
Backbone Link (T1–T2)	10 Gbps
Queue Management	RED with ECN support
Transport Protocol	Data Center TCP (DCTCP)
Simulation Time	5 minutes per configuration
Sampling Interval	Every Acknowledgement reception
Number of TCP flows from S1 to R1	5, 10, 15
Number of TCP flows from S2 to R2	5, 10, 15
Number of TCP flows from S3 to R1	5, 10, 15

For each network configuration, the values of the 17 selected parameters (16 features and 1 target) are recorded at every simulation interval (i.e., upon the reception of each acknowledgement). Each recorded observation forms one instance of the time-series dataset. The selected parameters are listed in Table [2]. Algorithm- 1 presents the entire process.

Table [2] - Parameters for Time-Series Dataset

Category	Parameter	Purpose
General	Timestamp	Current time of simulation.
TCP – Socket	Congestion_Window	Current TCP congestion window controlling the sender transmission rate.
TCP – Socket	Slow_Start_Threshold	Threshold at which TCP transitions from Slow Start to Congestion Avoidance.

TCP – Socket	Congestion_State	Current state of congestion.
TCP – Socket	ECN_State	Current ECN state.
TCP – Socket	Bytes_In_Flight	Number of unacknowledged bytes currently in the network.
TCP – Socket	Last_RTT	Latest value of Round Trip Time measurement.
TCP – Socket	Min_RTT	Minimum value of Round Trip Time observed so far.
DCTCP	Alpha	DCTCP congestion estimation parameter computed from ECN-marked packets.
DCTCP	Congestion_Experience_State	DCTCP Congestion Experienced state.
DCTCP	Acked_Bytes_ECN	Total number of acknowledged bytes which are marked.
DCTCP	Acked_Bytes_Total	Total number of acknowledged bytes.
RED	Queue_Length_Current	Instantaneous number of packets currently in the RED queue.
RED	Queue_Length_Average	Exponentially weighted moving average of the queue length.
RED	Probability_Packet_Mark	Current probability of ECN marking computed by RED.
RED	ECN_Mark_Count	Number of packets marked with ECN by RED.
RED	Target_Queue	Queue_Length_Current at next time step.

Algorithm 1: Time-Series Dataset Generation

Input: Network Simulations.
Output: Time-Series Dataset D.

1. Initialize the custom multi-bottleneck DCTCP topology as per Figure [$].
2. for each network configuration C{1 to 81} as per Table [$] do
   2.1. Configure the access link bandwidth.
   2.2. Configure the number of concurrent TCP flows.
   2.3. Start the network simulation.
   2.4. while simulation is running do
      2.4.1. At every acknowledgement (ACK) reception, Collect the values of all parameters listed in Table [$].
      2.4.2. Store the collected values as one dataset instance.
   end while
   end for
3. Generate Target_Queue by assigning the value of Queue_Length_Current on next acknowledgement reception.
4. Append Target_Queue to each dataset instance.
5. Save the generated time-series dataset.

3.3 Temporal Transformer

The Temporal Transformer is implemented using the PyTorch deep learning framework. The model is trained using the generated time-series dataset, where the 16 selected parameters are used as input features and Target_Queue is used as the prediction target. The complete model training procedure is presented in Algorithm 2.

Algorithm 2: Temporal Transformer Model

Input: Time-Series Dataset D.
Output: Temporal Transformer Model M.

1. Load Time-Series dataset D.
2. Chronologically split the dataset: Training set = 70%, Validation set = 15% and Testing set = 15%.

3. Normalize the input features using StandardScaler fitted on the training set, and apply the same scaler to the validation and testing sets.
4. Generate temporal input sequences by applying a sliding window of size W=8 independently within each dataset split.
5. Initialize the Temporal Transformer model: Embedding Dimension=64, Attention Heads=4, Encoder Layers=2, Feed Forward Dimension=128 and Dropout=0.1
6. Initialize the training hyperparameters: Loss Function=Mean Squared Error (MSE), Optimizer=Adam, Learning Rate=0.001, Batch Size=64 and Number of Epochs=100

for each epoch {1 to 100} do
7.1 Set the model to training mode.
7.2 for each mini-batch do
7.2.1 Reset the optimizer gradients.
7.2.2 Pass the input sequence through the embedding layer.
7.2.3 Add positional encoding.
7.2.4 Pass the encoded sequence through the Transformer Encoder.
7.2.5 Predict Target_Queue.
7.2.6 Compute the MSE loss.
7.2.7 Perform backpropagation.
7.2.8 Update the model parameters using the Adam optimizer.
end for
7.3 Set the model to evaluation mode.
7.4 Evaluate the model on the validation set and compute Current_Valid_Loss.
7.5 if Current_Valid_Loss < Best_Valid_Loss then
7.5.1 Best_Valid_Loss = Current_Valid_Loss
7.5.2 Save the current model as M.
end if
end for
8. Load M and evaluate it using test dataset.

3.4 Congestion Prediction and ECN Decision Making

The trained Temporal Transformer is used to predict future network congestion in terms of the target queue length. Based on the predicted queue length, the ECN marking decision is performed. The complete congestion prediction and ECN decision-making procedure is presented in Algorithm 3.

Algorithm 3: Congestion Prediction and ECN Decision Making

Input: Temporal Transformer Model M.
Output: ECN Packet Marking Decision

1. Load Temporal Transformer model M.
2. Initialize a sliding window of size W=8.
3. while the network simulation is running do
3.1 At every ACK reception, collect the values of the input parameters listed in Table I.

Normalize the collected input features using the normalization parameters obtained during model training and update the sliding window with the normalized observation.
3.3 if the sliding window contains W observations then
3.3.1 predict Target_Queue.
3.3.2 if Predicted_Target_Queue < Min_Threshold then
3.3.2.1 Do not apply ECN packet marking.
else if Predicted_Target_Queue < Max_Threshold then
3.3.2.2 Apply RED probabilistic ECN packet marking.
else
3.3.2.3 Mark all ECN-capable packets.
end if
end if
end while

4 Performance Analysis

4.1 Implementation

TT-PECN is implemented in two phases: Offline training phase and online deployment phase. The offline phase trains the Temporal Transformer model from the time-series dataset using Pytorch. The online phase integrates this model with NS-3.34 simulator to perform congestion prediction and ECN decision making during packet transmission. The implementation maintains compatibility with the existing DCTCP protocol by replacing only the queue-length estimation used for ECN marking while preserving the standard RED threshold-based marking strategy. Consequently, TT-PECN enables predictive congestion notification without requiring modifications to the end hosts or the DCTCP congestion control algorithm, thereby facilitating practical deployment in data centre networks.

4.2 Network Topology

TT-PECN was evaluated with the same topology that was used to generate the dataset to get overall idea about its effectiveness. Moreover, the topology has been modified with different values of access link bandwidth to facilitate unknown scenarios. The summary of the network scenarios is given in Table [3]. There are total 81 configurations for every Scenario S_i. The total of 81 * 3 = 243 configurations. Each configuration is simulated for 5 minutes so the total simulation time is 243 * 5 = 1215 minutes. The performance of TT-PECN is compared with Data Center TCP (DCTCP) and TCP NewReno.

Table [3] - Parameters for Network Configurations – Testing

Scenario for Topology Figure [$]	**Access Link Bandwidth**	**TCP Flows (S1→R1, S2→R2, S3→R1)**	**T1–T2 Link**
S1 - Original	0.5, 1, 1.5 Gbps	5, 10, 15	10 Gbps
S2 - Modified Low Access Link Bandwidth	0.25, 0.75, 1.25 Gbps	5, 10, 15	10 Gbps
S3 - Modified High Access Link Bandwidth	1.75, 2, 2.25 Gbps	5, 10, 15	10 Gbps

4.3 Performance Analysis

As discussed in section 4.2, There are total 243 independent network configurations were designed. To represent performance analysis of every network configuration independently would be complex and difficult in understanding. The better way is to derive performance metrics on average basis. Since TCP is an end-to-end transport layer protocol, performance analysis is done with two important metrics: Average Per-Flow End-to-End Throughput (Gbps) and Average Per-Flow End-to-End Packet Delivery Ratio (%). Table [4], Table[5], Figure[6] and Figure[7] shows the results. This paper presents only summary results to follow the number of pages requirements.

Table [4] - Average Per-Flow End-to-End Throughput (Gbps)

Scenario for Topology Figure [1]	**Average Per-Flow End-to-End Throughput (Gbps)**		
	TCP NewReno	Data Center TCP (DCTCP)	DCTCP with TT-PECN
S1 - Original	0.47	0.58	0.63
S2 - Modified Low Access Link Bandwidth	0.24	0.33	0.36
S3 - Modified High Access Link Bandwidth	0.77	0.86	0.89

Table [5] - Average Per-Flow Packet Delivery Ratio (%)

Scenario for Topology	**Average Per-Flow End-to-End Packet Delivery Ratio (%)**

Figure [1]	TCP NewReno	Data Center TCP (DCTCP)	DCTCP with TT-PECN
S1 - Original	98.65%	99.12%	99.34%
S2 - Modified Low Access Link Bandwidth	97.36%	98.42%	98.78%
S3 - Modified High Access Link Bandwidth	98.77%	99.25%	99.54%

Figure [6] - Average Per-Flow End-to-End Throughput (Gbps)

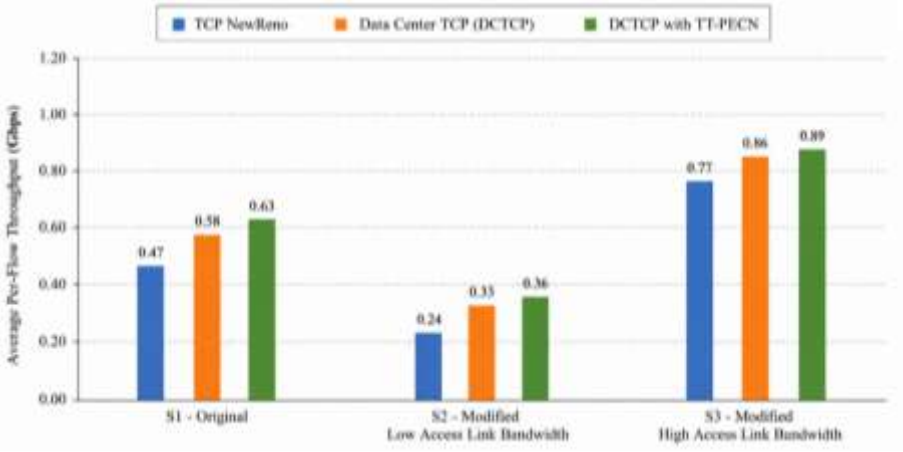
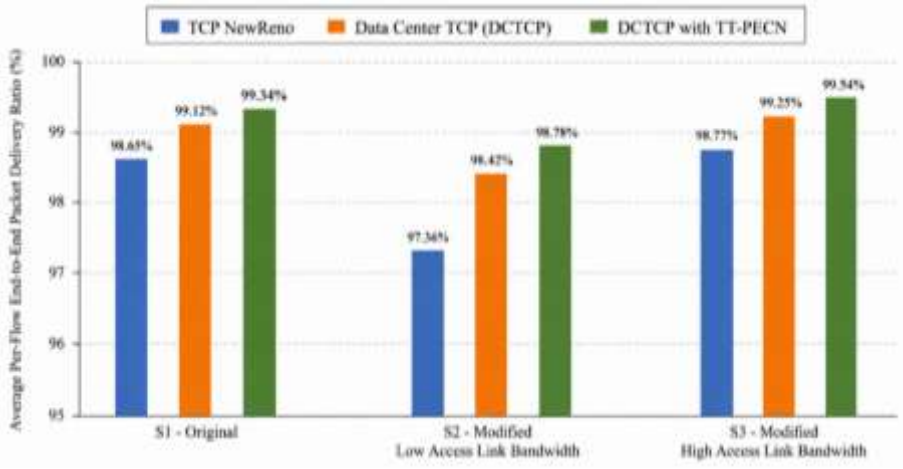


Figure [7] - Average Per-Flow Packet Delivery Ratio (%)



5. Conclusions and Future Work

The proposed TT-PECN (Temporal Transformer based Predictive Explicit Congestion Notification) provides a proactive approach for congestion management in Data Center Networks. Unlike conventional ECN mechanisms that mainly respond to the current queue condition, TT-PECN uses a Temporal Transformer to predict future congestion and make accurate ECN marking decision. The experimental results show that TT-PECN performs better than TCP NewReno and standard DCTCP in various scenarios. These scenarios are diverse interms of access link bandwidth as well as whether included in training dataset or not.The average per-flow end-to-end throughput achieved by TT-PECN is 0.63 Gbps, 0.36 Gbps, and 0.89 Gbps for S1, S2, and S3, respectively. Similarly, the packet delivery ratio reaches 99.34%, 98.78%, and 99.54%, respectively. These results indicate that predictive ECN marking can improve network performance under different bandwidth conditions.

In future work, TT-PECN can be evaluated on larger and more diverse data center topologies with a higher number of nodes and traffic flows. The model can also be tested under different traffic patterns and network conditions. Further work can explore online model updating, additional network features, and other Transformer architectures to improve congestion prediction. Real-time implementation and evaluation on a practical data center network can also be considered to study the applicability of TT-PECN in real-world environments.

References

1. Janovic, Jan. "Introduction: Datacenter network evolution." Cisco ACI: Zero to Hero: A Comprehensive Guide to Cisco ACI Design, Implementation, Operation, and Troubleshooting. Berkeley, CA: Apress, 2022. 1-12.
2. Han, Feixue, et al. "Future data center networking: From low latency to deterministic latency." IEEE Network 36.1 (2022): 52-58.
3. Avin, Chen, and Stefan Schmid. "Revolutionizing datacenter networks via reconfigurable topologies." Communications of the ACM 68.6 (2025): 44-53.
4. Gilliard, Ezekia, et al. "Explicit congestion notification-based congestion control algorithm for high-performing data centers." 2023 IEEE AFRICON. IEEE, 2023.
5. Meng, Qingkai, et al. "Explicit dropping notification in data centers." IEEE INFOCOM 2024-IEEE Conference on Computer Communications. IEEE, 2024.
6. Fathalli, Seifeddine, et al. "Network-Assisted Congestion Feedback." IEEE Transactions on Network and Service Management 23 (2025): 1797-1815.
7. Vivek Jain, Thomas R. Henderson, Shravya K. S., and Mohit P. Tahiliani. 2020. Data Center TCP in ns-3: Implementation, Validation and Evaluation. In Proceedings of the 2020 Workshop on ns-3 (WNS3 '20). Association for Computing Machinery, New York, NY, USA, 65–72.
8. Dhamija, Abhishek, et al. "A large-scale deployment of {DCTCP}." 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). 2024.
9. Wang, Haoyu, et al. "DCTCP-FQ: Enhancing Fairness and Convergence Time in Data Center Congestion Control." 2025 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2025.
10. Lu, Yifei, Xu Ma, and Changjiang Cui. "DCCS: A dual congestion control signals based TCP for datacenter networks." Computer Networks 247 (2024): 110457.
11. Zeynali, Danesh, et al. "The State of ECN Adoption in the Internet." (2026).
12. Zhang, Tao, et al. "Taming the aggressiveness of heterogeneous TCP traffic in data center networks." IEEE/ACM Transactions on Networking 32.3 (2023): 2253-2268.
13. Lu, Yifei, Xu Ma, and Changjiang Cui. "DCCS: A dual congestion control signals based TCP for datacenter networks." Computer Networks 247 (2024): 110457.
14. Aoki, Keita, and Miki Yamamoto. "Proxy buffer with gradient-ECN for congestion control to inter-data center networks." IEICE Transactions on Communications (2025).
15. Huang, Hanlin, et al. "DiffECN: Differential ECN marking for datacenter networks." IEEE Transactions on Networking 33.1 (2024): 210-225.
16. Hu, Jinbin, et al. "A receiver-driven transport protocol with high link utilization using anti-ECN marking in data center networks." IEEE Transactions on Network and Service Management 20.2 (2022): 1898-1912.
17. Zhang, Jinghui, et al. "Micro-burst aware ECN in multi-queue data centers: algorithm and implementation." IEEE Transactions on Network Science and Engineering 11.3 (2023): 2384-2398.
18. Cheng, Kai, et al. "PET: Multi-agent independent PPO-based automatic ECN tuning for high-speed data center networks." arXiv preprint arXiv:2405.11956 (2024).
19. Alanazi, Sultan, and Bechir Hamdaoui. "ML-ECN: Multi-level ECN marking for fair datacenter traffic forwarding." ICC 2022-IEEE International Conference on Communications. IEEE, 2022.
20. Xing, Jinming, and Muhammad Shahzad. "A reinforcement learning framework for application-specific TCP congestion-control." 2025 IEEE International Performance, Computing, and Communications Conference (IPCCC). IEEE, 2025.
21. Gu, Yin, et al. "Procc: Programmatic reinforcement learning for efficient and transparent TCP congestion control." Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining. 2025.
22. Mahajan, Smita, et al. "Reinforcement learning based congestion control technique for wireless mesh networks." International Journal of Networked and Distributed Computing 13.1 (2025): 20.
23. Ağlamazlar, Efe, Emirhan Eken, and Harun Batur Geçici. "A Deep Reinforcement Learning-Based TCP Congestion Control Algorithm: Design, Simulation, and Evaluation." arXiv preprint arXiv:2508.01047 (2025).

24. Chae, Heeju, et al. "Pacemaker Congestion Control: Ceiling-Normalized Reinforcement Learning for TCP." IEEE Access (2026).
25. Liu, Yi, et al. "DECC: Achieving low latency in data center networks with deep reinforcement learning." IEEE Transactions on Network and Service Management 20.4 (2023): 4313-4324.
26. Hu, Jinbin, Zikai Zhou, and Jin Zhang. "Lightweight automatic ECN tuning based on deep reinforcement learning with ultra-low overhead in datacenter networks." IEEE Transactions on Network and Service Management 21.6 (2024): 6398-6408.