energies

MDPI

Article

ACC-RL: Adaptive Congestion Control Based on Reinforcement Learning in Power Distribution Networks with Data Centers

Tairan Huang, Xiaojuan Lu, Dian Zhang, Haoran Cheng, Pingping Dong * and Lianming Zhang *

College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China; htr@hunnu.edu.cn (T.H.); luxiaojuanlll@hunnu.edu.cn (X.L.); 202070291677@hunnu.edu.cn (D.Z.); chenghaoran@hunnu.edu.cn (H.C.)

* Correspondence: ppdong@hunnu.edu.cn (P.D.); zlm@hunnu.edu.cn (L.Z.)

Abstract: Modern data center power distribution networks place greater demands on the stability and reliability of power supply. Growing network computing demands and complex network environments can cause network congestion, which in turn leads to network traffic overload and power supply equipment overload. Therefore, network congestion is one of the most important problems faced by data center power distribution networks. In this paper, we propose an approach called ACC-RL based on reinforcement learning (RL), which can effectively avoid network congestion and improve energy performance. ACC-RL models the congestion control task as a Partially Observable Markov Decision Process (POMDP). It is independent of the estimated value function and supports deterministic policies. It also sets the reward value function using real-time network information such as the transmission rate, RTT, and switch queue length, with the target transmission rate as the target equilibrium point. ACC-RL is highly general, can be trained on datasets running in different network environments, and generates a robust congestion control policy. The experimental results show that ACC-RL can solve the congestion problem without any predefined scenarios in different network environments. It can control the network traffic well, thus ensuring the stability and reliability of the power supply in the distribution network. We conduct network simulation experiments through NS-3. We set up different scenarios for experiments and data analysis in many-to-one, all-to-all, and long–short network environments. Compared with the popular rule-based congestion control algorithms such as TIMELY, DCQCN, and HPCC, ACC-RL shows different degrees of energy performance advantages in network metrics such as fairness, link utilization, and throughput.

Keywords: power distribution network; data centers; congestion control; reinforcement learning; energy performance

Citation: Huang, T.; Lu, X.; Zhang, D.; Cheng, H.; Dong, P.; Zhang, L. ACC-RL: Adaptive Congestion Control Based on Reinforcement Learning in Power Distribution Networks with Data Centers. *Energies* **2023**, *16*, 5385. https://doi.org/10.3390/en16145385

Academic Editors: Ahmed Abu-Siada, Heng Li and Kaiyang Liu

Received: 9 June 2023
Revised: 2 July 2023
Accepted: 13 July 2023
Published: 14 July 2023

1. Introduction

The data center power distribution network is a system designed to efficiently distribute and manage the power within a data center facility. It consists of various components and infrastructures that work together to provide reliable power to servers, network equipment, and other devices in the data center. When the network traffic in a data center is too heavy or uneven, it may result in excessive load pressure on the power distribution network. This can lead to the overloading of power supply devices, which can affect the stability and reliability of the power supply. The root cause of unbalanced and overloaded network traffic is network congestion. It is a situation where network traffic exceeds the processing capacity of network devices or the bandwidth capacity of network links, resulting in increased data transmission delays, increased packet loss, and degraded network performance. Therefore, the resolution of network congestion is one of the top issues to be addressed in data center power distribution networks [1].

The operation of data centers typically involves high energy consumption, including the energy required for processing equipment and the energy consumed by cooling systems.

Recent studies have improved the energy efficiency and overall performance of data centers from several perspectives. Salamai et al. [2] propose a dynamic voting classifier for identifying risks in Supply Chain 4.0. Since the classifier may require processing large amounts of data and performing complex computations, it requires high-performance computing power and storage capacity provided by data centers. Ibrahim et al. [3] discuss the use of deep learning and adaptive dynamic optimization algorithms for wind speed ensemble prediction. Data centers can use such wind speed prediction models to optimize their own wind energy utilization strategies, reduce reliance on the conventional grid, and improve energy efficiency. El-kenawy et al. [4] investigate the use of advanced artificial intelligence techniques to design dual T-shaped monopole antennas. Data centers can improve the overall performance of data centers by increasing the effectiveness and efficiency of wireless communications using this advanced antenna design.

Most of the current congestion control methods are designed with rule-based heuristics, such as TIMELY [5], Data Center Quantified Congestion Notification (DCQCN) [6], and High-Precision Congestion Control (HPCC) [7]. However, these algorithms cannot meet the demand for congestion control in complex and changing network environments. Therefore, Jiang [8] and Lei et al. [9] proposed a congestion control algorithm based on reinforcement learning (RL) [10,11]. It uses the real-time network environment and the amount of data to generate the best control policy to cope with the changing network environment by learning and generalizing the data. Most of the algorithms for RL can be modified by the Markov Decision Process (MDP) [12]. However, in practical data center congestion control, it cannot be constructed as a standard MDP due to security and partial observability, such as Aurora [13], REINFORCE [14], PPO [15], etc. This is one of the main reasons why RL cannot be deployed at scale in data centers.

To solve the above problem, we propose an approach called Adaptive Congestion Control based on Reinforcement Learning (ACC-RL). It is a stable congestion control method generated based on the deterministic policy gradient (DPG) model [16]. It is independent of the estimated value function and supports deterministic policies. We perform task modeling through a Partially Observable Markov Decision Process (POMDP). We set the reward value function using real-time network information such as the transmission rate of the flow, RTT, and switch queue length, with the target transmission rate as the target equilibrium point. In our design, the reward function is able to obtain the superior value. Moreover, our method is able to guarantee the optimal transmission rate without packet loss. ACC-RL can perform the network congestion task well, control the network traffic, and balance the load of the distribution network. Therefore, it can achieve a stable and reliable power supply while ensuring that power devices are not overloaded. We perform extensive simulation experiments based on NS-3 [17]. We use many-to-one, all-to-all, and long–short as the experimental network environments. In each environment, we set up different scenarios for experiments and data analysis. Simulation experiments show that ACC-RL provides a stable congestion control strategy that is competitive in all network environments. Our contributions are summarized as follows.

(1) We design ACC-RL, which is a congestion control algorithm based on reinforcement learning. It can use real-time network data for model training and online decision in different network environments, and generate robust congestion control policies. It can perform congestion tasks and control network traffic in multiple scenarios, thus ensuring the stability and reliability of power supply in distribution networks with high generality.

(2) We optimize the reward function. We choose transmission rate, RTT, and queue length as reward parameters to find a single equilibrium point. It has high convergence and can obtain the superior reward value accurately. It ensures that the transmission rate is kept at the best possible level.

(3) We compare ACC-RL with currently popular congestion control algorithms under different environments through simulation experiments on the NS-3 simulation platform. The results show that ACC-RL demonstrates its unique superiority in several metrics under different scenarios.

2. Backgrounds

In this section, we present the current research status in terms of both congestion control and RL, respectively.

2.1. Congestion Control Algorithms

In this subsection, we discuss several rule-based congestion control strategies and briefly analyze their principles and advantages and disadvantages.

TIMELY: It monitors the congestion in the network by measuring the RTT and uses a computational engine to convert the RTT signal into a target rate, and inserts delays between segments to achieve the target rate. TIMELY uses RTT as the congestion signal and is able to directly measure the delay due to network queuing. However, the latency problem will become more severe during its queue build-up. In addition, it is unable to handle lost messages and needs to rely on PFC [18] to avert message loss.

DCQCN: It combines Quantized Congestion Notification (QCN) [19] and Data Center TCP (DCTCP) [20]. It prevents packet loss by PFC and adjusts the transmission rate by Explicit Congestion Notification [21]. However, in some specific network environments, such as the long–short, DCQCN will trigger PFC frequently, which leads to problems such as low link utilization.

HPCC: It obtains link load information and controls traffic via In-band Network Telemetry [22]. It can make full use of free bandwidth to obtain a very low queue length and thus avoid congestion. However, in a many-to-one environment, when the number of senders keeps increasing, the link load increases. HPCC will experience severe packet loss and cannot guarantee normal communication.

These rule-based algorithms are able to demonstrate their advantages well in specific network environments but are unable to apply them to all domains for congestion control tasks under various scenarios.

2.2. RL Algorithms

In this subsection, we present several classical RL algorithms and analyze the problems that arise in their congestion control tasks.

Aurora: It generates a policy based on the deep RL [11] and controls the transmission rate by observing the collected network performance data such as latency, etc. Aurora can generate good congestion control policies in simple network simulation environments and exhibits good network performance, even exceeding general rule-based control policies. However, it is not capable of performing communication tasks in complex network environments or environments with large data volumes.

REINFORCE: It designs statistical gradient tracking algorithms that can compute RL networks with random output units in essentially any arbitrary way. REINFORCE is a gradient-based method that works well with other gradient computation techniques such as Stochastic Gradient Descent [23]. However, the algorithm lacks a suitable convergence theory and is prone to the pseudo-optimal convergence state common in gradient algorithms.

PPO: It is an RL-based policy gradient method that collects data and performs stochastic gradient ascent optimization while interacting with the environment. PPO inherits some of the advantages of trust region policy optimization [24]. It is optimized for easier implementation and greater versatility. However, the application of PPO to the congestion control environment leads to problems such as decreased stability and convergence speed, which cannot accomplish the congestion control task.

It can be seen that these classical RL algorithms are prone to problems in terms of stability and convergence, and thus cannot accomplish the congestion control task. Therefore, we should fully consider these factors when designing the relevant algorithms.

We show a comparison of whether the above algorithms can complete congestion control under different scenarios, as shown in Table 1. A (✓) indicates that the congestion control task has been completed in this experimental setting, whereas a cross mark (×)

indicates that it has not been completed. TIMELY, DCQCN, and HPCC can only complete the congestion control task in some of the scenarios, whereas Aurora, REINFORCE, and PPO end up with failure in congestion control tasks under all scenarios.

Table 1. Analysis of different congestion control algorithms.

	Many-to-One	**All-to-All**	**Long–Short**
Aurora/REINFORCE/PPO	×	×	×
TIMELY	×	✓	×
DCQCN	✓	✓	×
HPCC	×	✓	✓
ACC-RL (ours)	✓	✓	✓

3. Our Approach: ACC-RL

In this section, we present the ACC-RL. It is designed based on DPG. We illustrate the architecture of the intelligence and the design of the framework.

3.1. The Architecture of the Intelligence in ACC-RL

In this subsection, we show the architecture of intelligence in ACC-RL. We model the congestion control task as a POMDP. The model is based on a multi-intelligence implementation with multiple objectives and continuous actions.

The POMDP simulates the decision-making process of an intelligence supposing that the system dynamics are dictated by the MDP, but the intelligence cannot directly observe the state. Conversely, it must deduce the allocation of states based on whole area and fractional area observations of the model. The POMDP is composed of a five-tuple of (O, S, A, T, R). The environment is in state $s \in S$ for a certain period of time. It represents information about the network state in the current environment, including transmission rate and switch queue length, etc. After receiving an observation $o \in O$, the intelligence selects an action $a \in A$. This results in an environmental probability of $T(s'|s, a)$ of switching to another state s'. Meanwhile, the intelligence accepts another observation $o \in O$ with probability $O(o|s', a)$ according to the new environmental state. Eventually, the intelligence accepts reward $r \in R(s, a)$.

In this paper, the intelligence is used to complete the congestion control task to adjust the sending rate, running in the network interface card. To meet the demands of data center power distribution networks with zero packet loss, high throughput, etc., the intelligence observes the statistics of the flows it controls at each decision point and performs actions, rewards, and rate transformations according to our design. Thus, we define the quadruplet (O, A, T, R).

Observations: The intelligence observes the statistics of the flows. In our method, we specify the transmission rate of the flows, RTT, and switch queue length.

Actions: The optimal aim of our designed algorithm is to make the transmission rate optimal, but this depends on the network environment itself (network topology, experimental scenarios, etc.). In different network environments, the optimal transmission rate can vary significantly. Therefore, we start from the Markov property, so as to make the changing transmission rate correlate with the current transmission rate (defined as E). The action represents a change in the transmission rate, defined as $E_{t+1} = a_t E_t$. We refer to the parameter design in the experiments of Silver et al. [16]. Therefore, in our experiments, $a_t \in [0.8, 1.2]$ will obtain the best results.

Transitions: The frequency of the state $s_t \to s'_t$ is determined by a combination of the status of the decision circumstance and the action executed by the intelligence. When the intelligence receives an RTT message, it performs the corresponding action to change the transmission rate. In PCC [25], they are transformed by a fixed time interval. In contrast,

we use the network real-time information such as RTT for detection, which is more accurate and efficient.

Rewards: In pursuit of the observability task for the multi-intelligence part, when designing the reward, we want to ensure the existence of a stable equilibrium point. After research, we chose the transmission rate in the network data as the target equilibrium point. The network utilization reaches optimal saturation when the aggregate of transmission rates of each flow continuously converges to target rate. We define the reward function as

$$r_t = -[G - (\sum_{i=1}^{N} E_{t,i} + \frac{U}{I})]^2, \tag{1}$$

where G is a static value representing the desired target rate of all flows. $E_{t,i}$, U, and I represent the transmission rate of the i flow at time t, the switch queue length, and minimum RTT, respectively. In a typical network environment, the transmission rate will be as close to the target rate as possible. However, when the transmission rate exceeds the G, it will cause the switch queue to grow, which results in problems such as packet loss. Therefore, in our algorithm, $\frac{U}{I}$ is used as the network tolerance to adjust the rate in time. The optimal transmission rate is guaranteed while no packet loss occurs.

To better explain the reward function we designed, we introduce the Bandwidth Delay Product (BDP), defined as B. BDP uses the volume of the datalink (bits per second) and the multiplication of RTT (in seconds) as the performance criteria. We derive Equation (1) to conclude that

$$r_t = -[D - (\sum_{i=1}^{N} B_{t,i} + U)]^2, \tag{2}$$

where D is used to express the target BDP.

When $\sum_{i=1}^{N} Et, i$ is larger equal to G and U is smaller equal to 0, the intelligence will obtain the optimal reward 0. In other cases, where the rate is too low to reach the G, or too high to cause queue growth, the appropriate action is taken for rate control. The RTT of each sender we observe through the intelligence is approximately the same, so it is a fair solution that can guarantee a maximum rate of $\frac{X}{N}$ for N flows, where X stands for the max rate.

We extend the Actor–Critic algorithm of DPG. It replaces the true action value function $Q^{\mu}(s, a)$ with the differentiable action value function $Q^{w}(s, a)$. δ_t exists as a time error. Critics estimate the action value function $Q^{w}(s, a) \approx Q^{\mu}(s, a)$ using the appropriate strategy evaluation algorithm. γ represents the withholding factor, which is used to measure the importance of current rewards versus future rewards. μ_θ is a parameter of the strategy network, which represents the weight parameter of the strategy network. The goal of the policy network is to output a deterministic action that results in the maximum reward in the current state. μ_θ formats the policy network and controls how the policy network selects an action based on the input state. w is a parameter of the value function network and denotes the weight parameter of the value function network. The goal of the value function network is to estimate the state value or the state–action value function. w formats the value function network for computing the state value or the state–action value estimate. We use

$$\delta_t = r_t + \gamma Q^{w}(s_{t+1}, a_{t+1}) - Q^{w}(s_t, a_t) \tag{3}$$

to calculate the Temporal Difference Error (TDE) for time step t, which is used to measure the difference between the reward of the current state action pair and the reward of the next state action pair. The equation

$$w_{t+1} = w_t + a_w \delta_t \nabla_w Q^{w}(s_t, a_t) \tag{4}$$

is used to update the parameters of the value function network. It indicates that the update of the weight value is the product of the current weight plus the learning rate multiplied by the gradient of the residual sum value function over the weights. The equation

$$\theta_{t+1} = \theta_t + a_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu\theta(s)} \tag{5}$$

is used to update the parameters of the policy network. It indicates that the update of the policy parameters is the product of the current parameters plus the learning rate multiplied by the gradient of the policy gradient and the gradient of the value function over the actions.

3.2. The ACC-RL Framework

Our proposed ACC-RL framework is shown in Figure 1. We corroborate our design by analyzing the reasons for the failure of existing methods in performing congestion control tasks.

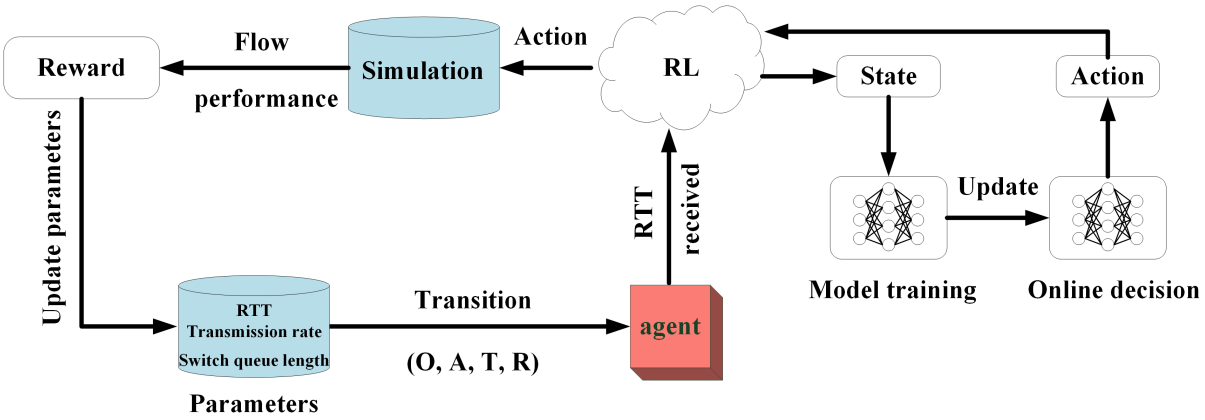


Figure 1. The framework of ACC-RL.

When multiple intelligences are trained in the same environment, non-stationarity can occur due to the constantly changing activities of other intelligences. Therefore, a non-policy approach cannot be used, but only a policy approach that ensures that each intelligence receives the trained data. Second, partial observability has an impact on the estimation of the value function. Generating correct gradient estimates via policy gradient methods for value functions requires straightforward visits to the state, as opposed to using observations [26]. Therefore, we refer to the situational reward trajectories in REINFORCE for the design. However, although REINFORCE solves the problem of value function estimation, it requires stochastic policies [14]. Stochastic policies lead to a highly unstable state during the intelligence's operation, so stability can only be satisfied by deterministic policies. From a comprehensive analysis of these problems, DPG as a policy-based method can solve these problems well. Therefore, our ACC-RL is designed based on DPG. We estimate the gradient directly by deriving the reward function to produce a deterministic policy gradient. It is a policy-based algorithm that is not dependent on the estimated value function and supports deterministic policies. For $s \in S$, Azizzadenesheli et al. [26] show that the gradient estimation is unbiased, despite the presence of partial observability in the policy. The deterministic gradient analysis approximation we obtained by optimizing the derivation and variant analysis of the gradient function is defined as

$$\nabla\theta\rho_{\pi\theta}(s) \approx [\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} (G - (\sum_{i=1}^{N} E_{t,i} + \frac{U}{I}))] \nabla\theta\pi\theta(o(s)). \tag{6}$$

When $G - (\sum_{i=1}^{N} E_{t,i} + \frac{U}{I}) < 0$, the gradient of the policy receives a negative weight, effectively reducing the transmission rate in the current state, and vice versa for $G - (\sum_{i=1}^{N} E_{t,i} + \frac{U}{I}) > 0$.

Model training: We perform model training by collecting information about the current state of each flow. Then, the local collector starts collecting current network information and saving it in the data pool during online decision-making and environmental interaction operations. The collected network information can reflect whether the network communication state is good or not. Therefore, the server decides the model training process based on the collected network information as a way to guarantee that the model can make an adaptation to the dynamic network.

Online decision: ACC-RL takes the active network status as input and computes the optimized sending rate in real time. The decision procedure is as shown below; in state s, the dispatcher performs the optimum action a and adjusts the transmission rate by calculation. Afterward, we can obtain another reward r and network state s'.

For the algorithm complexity analysis, we focus on two aspects: time complexity and space complexity. For the time complexity, the main overhead comes from the training of the strategy network and the gradient values. For each iteration, it needs to update the policy network and the gradient values at each time step. This means that its time complexity is $O(T)$, where T is the number of iterations. For the space complexity, it needs to store the parameters of the policy network and the gradient, so its space complexity is proportional to the number of parameters of these networks.

Overall, in our framework, the network parameters used for online decision are updated in parallel to adapt to changes in networks so that results in three different instances indicate a better performance than other methods (details in Section 4). Moreover, the model is positioned on the servers without any influence on a client's resources. These details are described in Algorithm 1.

Algorithm 1: ACC-RL

Input: transmission rate of the flows, RTT, and switch queue length
Output: reward r
Initialize O according to probability $O(o|s', a)$
for *each flow i* **do**
  Initialize state s_t according to an environmental probability of $T(s'|s, a)$
  Model training by current state to obtain factor a_t:
    $a_t \in [0.8, 1.2]$
  Update network transmission rate based on factor a_t:
    $E_{t+1} = a_t E_t$
  Store new transition (O_t, A_t, T_t, R_t)
  Set $\delta_t = r_t + \gamma Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t)$
  Update the reward r_t according to the function:
    $r_t = -[G - (\sum_{i=1}^{N} B_{t,i} + U)]^2$
  Update actor policy by the policy gradient:
    $\nabla\theta\rho_{\pi\theta}(s) \approx [\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} (G - (\sum_{i=1}^{N} E_{t,i} + \frac{U}{I}))] \nabla\theta\pi\theta(o(s))$
  Update the parameter vector θ:
    $\theta_{t+1} = \theta_t + a_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu\theta(s)}$
end

4. Experiments

In this section, we simulate the network structure as well as the data communication behavior by conducting network simulation in NS-3. We validate the superiority of our proposal and algorithm.

4.1. Experiment Settings

In this subsection, we describe the scenarios used in the experiments. We set the bottleneck link in the experimental scenario to 40 Gbps and the link latency to 5 µs. We focus on the following three environments, as shown in Figure 2.

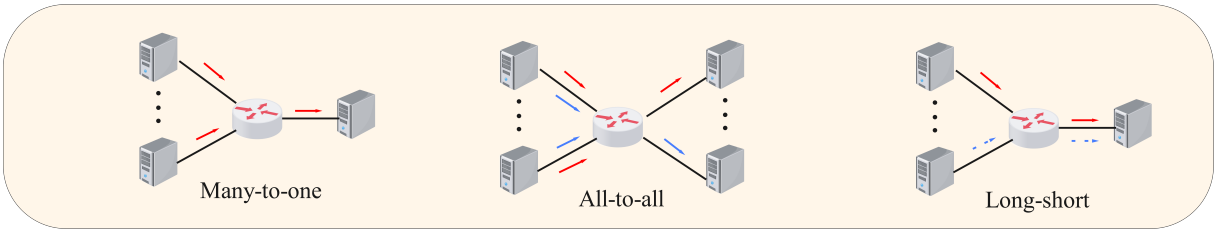


Figure 2. The three environments tested in the experiment are many-to-one, all-to-all, and long–short. The red and blue arrow lines all represent the flow of data. The blue dashed arrows indicate the direction of the short flow in the long–short environment.

(1) **Many-to-one**: In this environment, multiple nodes send data to a single target node simultaneously, i.e., N to 1. This scenario simulates multiple users sending data to a central node at the same time. In our experiments, we can set up multiple source nodes to send data to a single target node at the same time and observe the performance of the proposed algorithm in handling multiple data flows. In this case, the switch's data-processing capabilities are greatly tested. Since multiple flows share the same egress, traffic fairness becomes the primary performance metric to consider. We use the Jain Fairness Index to calculate fairness, which is a common metric used to calculate fairness. The function is defined as

$$FI = \frac{(\sum_{i=1}^{N} \frac{T_i}{O_i})^2}{n \sum_{i=1}^{N} (\frac{T_i}{O_i})^2} \tag{7}$$

where FI is the fairness index and O_i is the actual throughput when all links in the link are performing data transmission. T_i is the transmission capacity of the ith link in the network.

(2) **All-to-all**: In this environment, all nodes communicate with each other, i.e., N to N. This scenario simulates a large-scale data exchange between all nodes. In the experiment, we can connect all nodes together so that they can communicate with each other and observe the performance of the proposed algorithm in an all-to-all communication environment. This approach can lead to overloading or underutilization of some switches due to the increased amount of synchronization signals. Therefore, in this case, the main performance metric we observe is switch utilization, defined as $\frac{real\_throughput}{max\_throughput}$.

(3) **Long–short**: In this environment, the streams are sent sequentially and with different lengths. In our experiments, we can set the sending time and length of the streams between nodes and observe the performance of the proposed algorithm in the long- and short-distance environment. Therefore, we focus on evaluating the performance of our algorithm in the case of burst short flows. Each sender dispatches a continuous flow of the same length from the start moment. During the transmission process, we randomly add short flows to each sender to transmit data. Thus, we test the throughput variation in the continuous flow. When the short flow completes data transfer, the continuous flow should quickly return to its peak throughput.

4.2. Experiment Results

In this subsection, we collect and analyze the experimental data under three environments and compare them with other algorithms on different metrics to illustrate the superiority of our algorithm.

Figure 3 gives the experimental results in a many-to-one environment. We collect data in three scenarios: 128 to 11,024 to 1 and 4096 to 1. We find that multiple methods exhibit their competitiveness when the number of senders is small and the traffic is low.

We analyze whether the proposed algorithm can fairly allocate bandwidth to each source node by using the Jain Fairness Index. At the 128 to 1 scenario, REINFORCE, HPCC, and ACC-RL show their strong superiority with 99%, 97%, and 97% fairness data, respectively. As the number of senders increases to 1024, REINFORCE and HPCC's fairness metrics drop to 70% and 48%, respectively. Furthermore, when the number of senders increases to 4096, both methods exhibit varying degrees of packet loss, resulting in unguaranteed fairness.

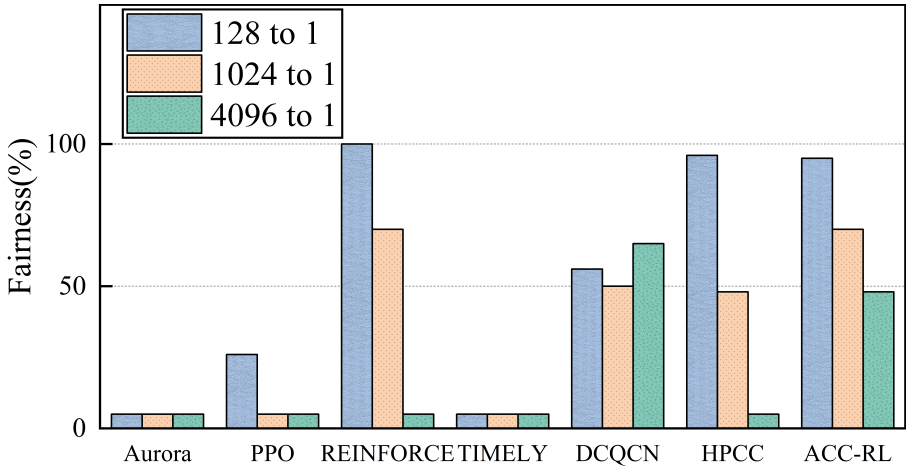


Figure 3. Comparison of fairness between different approaches in many-to-one environment, with data in percentages.

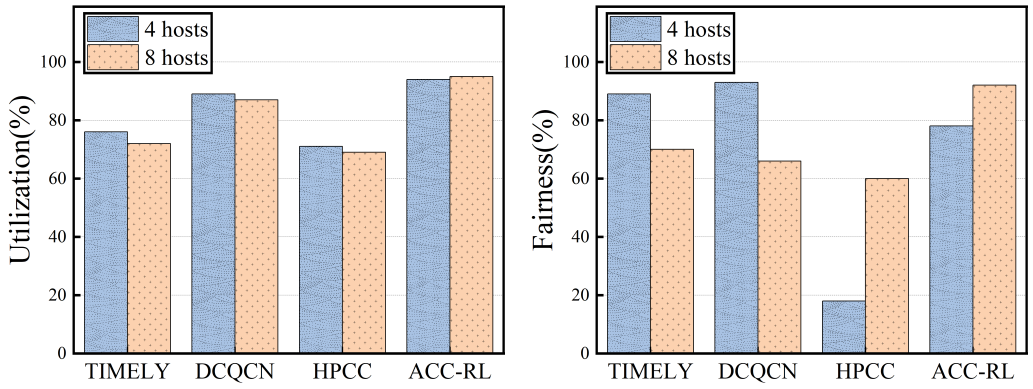
Meanwhile, we found that the other RL methods (Aurora and PPO) exhibit packet loss for all three scenarios in a many-to-one environment. This confirms our previous view that they are unable to perform the congestion control task. Among the rule-based congestion control algorithms, TIMELY is unable to accomplish the congestion control task in the many-to-one environment. Furthermore, although DCQCN can perform the task well, its fairness is uniformly distributed with 56%, 50%, and 62% results under three test scenarios, which cannot achieve the optimal solution and thus is not the best metric we expect. In contrast, our ACC-RL algorithm, with 97%, 70%, and 49% results under three scenarios presents the expected stepped data performance, keeping the best fairness as much as possible within a manageable range.

Table 2 illustrates the average memory usage percentages for the three scenarios. The optimal memory utilization interval should be one that makes full use of memory resources as much as possible while satisfying system stability and avoiding too high or too low utilization. Lower memory utilization may indicate underutilized memory resources, whereas higher memory utilization may take up more system resources. Since TIMELY, DCQCN, and HPCC focus mainly on real-time network congestion control, the memory usage is relatively small in the 128 to 1 scenario. As the number of senders increases, memory usage will gradually rise. Reinforcement learning-based algorithms like REINFORCE and ACC-RL may generate some intermediate data for calculating gradients, updating parameters and other operations during the training process, and these intermediate data will also occupy a certain amount of memory space. Therefore, in general, they have a higher memory usage than other rule-based algorithms. REINFORCE shows higher memory usage of 71% and 89% for 128 to 1 and 1024 to 1 scenarios. ACC-RL shows moderate and stable memory usage with 49%, 59%, and 67% for three scenarios.

In an all-to-all environment, RL algorithms (Aurora, PPO, REINFORCE) also suffer from packet loss and fail to complete their tasks. Therefore, in this section, we investigate rule-based data comparison congestion control algorithms. We chose two experimental scenarios: four hosts and eight hosts. The data plots we obtained after analyzing the performance metrics switch utilization and fairness are shown in Figure 4.

Table 2. Average memory usage percentages (%) for three scenarios in many-to-one environment.

	128 to 1	**1024 to 1**	**4096 to 1**
Aurora	Packet loss	Packet loss	Packet loss
REINFORCE	71%	89%	Packet loss
PPO	Packet loss	Packet loss	Packet loss
TIMELY	21%	43%	59%
DCQCN	32%	48%	63%
HPCC	45%	68%	76%
ACC-RL (ours)	49%	59%	67%



(**a**) Switch Utilization (**b**) Fairness

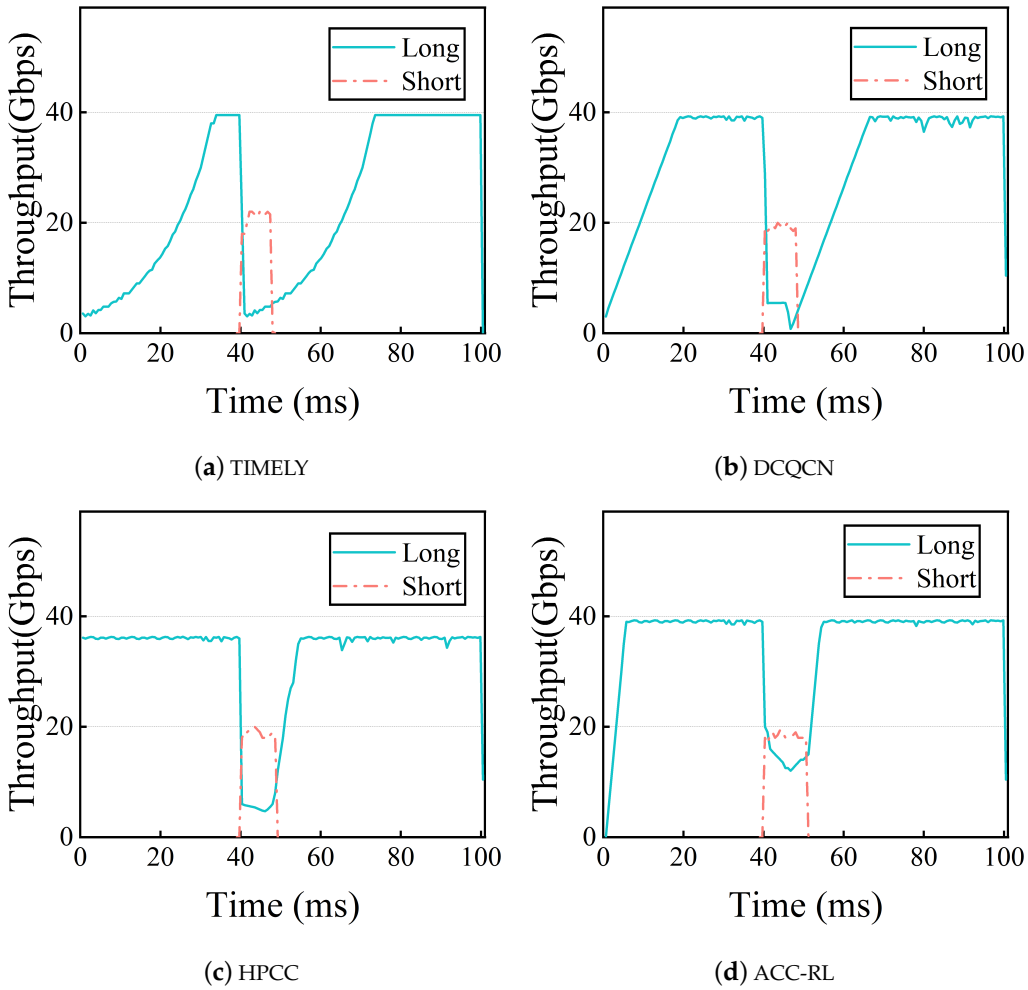
Figure 4. Comparison of switch utilization and fairness between different approaches in all-to-all environment, with data in percentages.

As we mentioned earlier, in an all-to-all environment, the metric we focus on is switch utilization. From the results, we can visually see that ACC-RL's switch utilization of 94% is significantly better than TIMELY's 76%, DCQCN's 89%, and HPCC's 71% for the four-host scenario. Meanwhile, for the eight-host scenario, ACC-RL's switch utilization is 95% ahead of TIMELY's 72%, DCQCN's 87%, and HPCC's 69%. For fairness, TIMELY and DCQCN outperformed HPCC's 18% and ACC-RL's 79% with 89% and 93%, respectively, for the four-host scenario. However, when the number of hosts increases and the amount of synchronization signals rises, the fairness of TIMELY and DCQCN drops significantly to 70% and 66% under the eight-host scenario. On the contrary, ACC-RL is unaffected by it and even the fairness improves to 92%. This demonstrates the superiority of our algorithm in a large-scale network environment.

In the long–short environment, our setting is such that each sender sends a long flow from the starting time 0 and sends a short flow at time 40. The simultaneous transmission of the two flows causes congestion to occur. We analyze the experimental results and focus on observing the throughput variation in each congestion control algorithm from time 40 onwards. This is shown in Figure 5.

As can be observed from the results, TIMELY and DCQCN largely affect the overall throughput when short flows are sent due to the long flow pauses caused by the PFC mechanism that comes with their algorithms. Moreover, we can intuitively see that the recovery speed of these two algorithms is slow and cannot reach the fast convergence to allow the throughput to reach the peak level as soon as possible. In addition, HPCC is more responsive, as it is specifically designed for long–short scenarios, so this is as expected. However, it is remarkable that the long-flow throughput cannot peak in HPCC even if no short-flow is present. In contrast to these algorithms, ACC-RL is equally capable of

guaranteeing partial long-flow communication when it is affected by short flows. Furthermore, it can converge to the peak throughput as fast as possible after the short flows end their transmission. Therefore, ACC-RL clearly outperforms other algorithms in this environment.



(**a**) TIMELY (**b**) DCQCN

(**c**) HPCC (**d**) ACC-RL

Figure 5. Comparison of throughput between different approaches in long–short environment, with data in Gbps.

We analyze data on the flow completion time (FCT) of each algorithm in this environment. ACC-RL is based on RL, which generates a certain amount of training time and decision time. To ensure the accuracy of the comparison experiments with TIMELY, DCQCN, and HPCC, we only consider the communication time of the data in the network when counting the FCT. Table 3 illustrates that the FCTs of TIMELY, DCQCN, and HPCC are 100,012 ms, 100,018 ms, and 100,019 ms, respectively, whereas the FCT of ACC-RL is 100,002 ms. It can be seen that our algorithm outperforms the other algorithms in terms of runtime.

Table 3. FCT (time in ms) for each algorithm in the long–short environment.

	Start Time	Burst Time	Completion Time
TIMELY	0	40	100,012
DCQCN	0	40	100,018
HPCC	0	40	100,019
ACC-RL (ours)	0	40	100,002

Overall, it is shown through our experiments that ACC-RL is able to autonomously learn a robust approach to congestion control tasks under different scenarios in different environments. The results demonstrate that ACC-RL is capable of meeting the performance demands of data center power distribution networks for a zero packet loss, high-throughput network environment. It outperforms other rule-based CC or RL algorithms in several important network performance metrics, such as fairness, switch utilization, and throughput, respectively. This is a major breakthrough compared to other algorithms that only optimize for a single environment.

5. Related Work

As data center networks (DCNs) are developing rapidly, both domestic and foreign scholars have undertaken many studies on the congestion control mechanism of DCNs. Among them, the core constructions can be divided into two categories. One is to optimize the rule-based congestion control method to enhance the capabilities of the network in a single environment. The second is to continuously optimize its core algorithms and models based on RL and apply them to congestion control tasks.

For rule-based approaches, most of them guarantee packet loss-free through the PFC mechanism and achieve lower latency and higher throughput by reducing PFC and the congestion queue. Recent studies such as Ternary Congestion Detection [27] perform inter-state transitions by dividing the port state into three, using features such as switch send mode and queue length changes, and combining congestion control mechanisms such as TIMELY and DCQCN for congestion regulation. Photonic Congestion Notification [28] provides congestion signals for rate-regulation purposes by detecting congestion on the switch and monitoring the internal state of the network for traffic identification. The congestion control mechanism of Backpressure Flow Control [29] is per-hop per-flow control, and although it should be used with caution in the presence of bounded states, constant-time switching operations, and buffers, it enables high bandwidth links, shortens tail delays in burst network environments, and improves network utilization.

For RL-based algorithms, optimizing the training model to adapt it to the congestion control network environment is a problem that must be addressed. We also observe relevant research advances in this area, such as the development of a new artificial intelligence in Deep Q-network [30] with the use of trained deep neural networks. It can directly acquire strategies for success from highly-dimensional sensibility feeds via end-to-end RL, but its application in the congestion control domain is not ideal. Analytic Deterministic Policy Gradient (ADPG) [31] is able to improve stability by approximating its inverse through the analysis architecture of the reward function to accomplish the congestion control task. Based on ADPG, Tessler et al. refine the model to justify a tree-based policy and apply the algorithm to a real network environment to accomplish the congestion control task and obtain positive performance metrics.

All of the above efforts exist to solve network congestion problems and improve the performance of data center networks. They all have their own characteristics and superiority. However, network renewal will gradually accelerate in the future, and the network congestion problem is unpredictable. Therefore, the method of solving network congestion through RL will gradually become prevalent and adaptable to any network environment.

6. Conclusions

In this paper, we propose a congestion control method based on reinforcement learning called ACC-RL in order to solve the network congestion of data center distribution networks and improve energy efficiency. It is independent of the estimated value function and supports deterministic policies. ACC-RL performs task modeling by a Partially Observable Markov Decision Process (POMDP) and designs superior reward value functions using real-time network information. ACC-RL is highly versatile in the face of variable network environments, supporting network testing with multiple scenarios. It also generates a robust strategy to converge the data transmission rate to the optimal value quickly

and maintain zero data buildup in the network link at all times. We evaluate ACC-RL performance with the NS-3 network simulation platform. The results indicate that ACC-RL shows superiority in terms of fairness, switch utilization, and throughput compared to other algorithms in complex network scenarios such as many-to-one, all-to-all, and long–short. It can achieve energy-efficient goals for power distribution networks with data centers. In our future work, we will integrate ACC-RL with real hardware and apply it to practical engineering applications.

Author Contributions: Methodology, T.H.; Software, T.H. and D.Z.; Validation, T.H.; Formal analysis, T.H. and D.Z.; Investigation, X.L.; Data curation, X.L.; Writing—original draft, T.H.; Writing—review & editing, P.D. and L.Z.; Visualization, H.C.; Supervision, P.D. and L.Z.; Project administration, P.D. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Hunan Provincial Natural Science Foundation of China (Nos. 2022JJ40277, 2022JJ30398 and 2021JJ30455) and Scientific Research Fund of Hunan Provincial Education Department (Nos. 22B0102 and 22A0056).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, W.; Yu, N. Phase balancing in power distribution network with data center. *ACM Sigmetrics Perform. Eval. Rev.* **2017**, *45*, 64–69. [CrossRef]
2. Salamai, A.A.; El-kenawy, E.S.M.; Abdelhameed, I. Dynamic voting classifier for risk identification in supply chain 4.0. *Comput. Mater. Contin.* **2021**, *69*, 3749–3766. [CrossRef]
3. Ibrahim, A.; Mirjalili, S.; El-Said, M.; Ghoneim, S.S.; Al-Harthi, M.M.; Ibrahim, T.F.; El-Kenawy, E.S.M. Wind speed ensemble forecasting based on deep learning using adaptive dynamic optimization algorithm. *IEEE Access* **2021**, *9*, 125787–125804. [CrossRef]
4. El-kenawy, E.S.M.; Abutarboush, H.F.; Mohamed, A.W.; Ibrahim, A. Advance artificial intelligence technique for designing double T-shaped monopole antenna. *Comput. Mater. Contin.* **2021**, *69*, 2983–2995. [CrossRef]
5. Mittal, R.; Lam, V.T.; Dukkipati, N.; Blem, E.; Wassel, H.; Ghobadi, M. TIMELY: RTT-based congestion control for the datacenter. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 537–550. [CrossRef]
6. Zhu, Y.; Eran, H.; Firestone, D.; Guo, C.; Lipshteyn, M.; Liron, Y. Congestion control for large-scale RDMA deployments. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 523–536. [CrossRef]
7. Li, Y.; Miao, R.; Liu, H.H.; Zhuang, Y.; Feng, F.; Tang, L. HPCC: High precision congestion control. In Proceedings of the ACM SIGCOMM, Beijing, China, 19–23 August 2019; pp. 44–58. [CrossRef]
8. Jiang, H.; Li, Q.; Jiang, Y.; Shen, G.; Sinnott, R.; Tian, C.; Xu, M. When machine learning meets congestion control: A survey and comparison. *Comput. Netw.* **2021**, *192*, 108033. [CrossRef]
9. Lei, K.; Liang, Y.; Li, W. Congestion control in SDN-based networks via multi-task deep reinforcement learning. *IEEE Netw.* **2020**, *34*, 28–34. [CrossRef]
10. Zhu, H.; Gupta, V.; Ahuja, S.S.; Tian, Y.; Zhang, Y.; Jin, X. Network planning with deep reinforcement learning. In Proceedings of the ACM SIGCOMM, Virtual Event, 23–27 August 2021; pp. 258–271.
11. Zhang, Q.; Ng, K.K.; Kazer, C.; Yan, S.; Sedoc, J.; Liu, V. MimicNet: Fast performance estimates for data center networks with machine learning. In Proceedings of the ACM SIGCOMM, Virtual Event, 23–27 August 2021; pp. 287–304. [CrossRef]
12. Puterman, M.L. Markov decision processes. In *Handbooks in Operations Research and Management Science*; Elsevier: Amsterdam, The Netherlands, 1990; Volume 2, pp. 331–434. [CrossRef]
13. Jay, N.; Rotman, N.; Godfrey, B.; Schapira, M.; Tamar, A. A deep reinforcement learning perspective on internet congestion control. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 3050–3059.
14. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256.
15. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
16. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 387–395.
17. Network Simulator 3. Available online: https://www.nsnam.org (accessed on 8 June 2023). [CrossRef]
18. IEEE 802.1 Qbb—Priority-Based Flow Control. Available online: https://1.ieee802.org/dcb/802-1qbb/ (accessed on 8 June 2023).
19. Ruan, C.; Zhang, T.; Li, H.; Xi, Y. Nonlinear control analysis of quantized congestion notification in data center networks. *IEEE Access* **2020**, *8*, 125401–125411.

20. Alizadeh, M.; Greenberg, A.; Maltz, D.A.; Padhye, J.; Patel, P.; Prabhakar, B. Data center tcp (dctcp). In Proceedings of the ACM SIGCOMM, New Delhi, India, 10 May 2010; pp. 63–74.
21. Floyd, S. TCP and explicit congestion notification. *ACM SIGCOMM Comput. Commun. Rev.* **1994**, *24*, 8–23.
22. Tan, L.; Su, W.; Zhang, W.; Lv, J.; Zhang, Z.; Miaom, J.; Liu, X.; Li, N. In-band network telemetry: A survey. *Comput. Netw.* **2021**, *186*, 107763. [CrossRef]
23. Ketkar, N. Stochastic gradient descent. In *Deep Learning with Python*; Apress: Berkeley, CA, USA, 2017; pp. 113–132.
24. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897. [CrossRef]
25. Dong, M.; Meng, T.; Zarchy, D.; Arslan, E.; Gilad, Y.; Godfrey, B.; Schapira, M. PCC Vivace: Online-Learning Congestion Control. In Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation, Renton, WA, USA, 9–11 April 2018; pp. 343–356. [CrossRef]
26. Azizzadenesheli, K.; Yue, Y.; Anandkumar, A. Policy gradient in partially observable environments: Approximation and convergence. *arXiv* **2018**, arXiv:1810.07900.
27. Zhang, Y.; Liu, Y.; Meng, Q.; Ren, F. Congestion detection in lossless networks. In Proceedings of the ACM SIGCOMM, Virtual Event, 23–27 August 2021; pp. 370–383.
28. Cheng, W.; Qian, K.; Jiang, W.; Zhang, T.; Ren, F. Re-architecting Congestion Management in Lossless Ethernet. In Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation, Santa Clara, CA, USA, 25–27 February 2020; pp. 19–36.
29. Goyal, P.; Shah, P.; Sharma, N.K.; Alizadeh, M.; Anderson, T.E. Backpressure flow control. In Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation, Virtual Event, 12–14 April 2021; pp. 1–26.
30. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
31. Fuhrer, B.; Shpigelman, Y.; Tessler, C.; Mannor, S.; Chechik, G.; Zahavi, E.; Dalal, G. Implementing Reinforcement Learning Datacenter Congestion Control in NVIDIA NICs. *arXiv* **2022**, arXiv:2207.02295.