electronics

MDPI

Article

P4QCN: Congestion Control Using P4-Capable Device in Data Center Networks

Junjie Geng, Jinyao Yan * and Yuan Zhang

Key Laboratory of Media Audio&Video Ministry of Education, Communication University of China, Beijing 100024, China; gjjtianxia369@163.com (J.G.); yuanzhang@cuc.edu.cn (Y.Z.)

* Correspondence: jyan@cuc.edu.cn; Tel.: +86-1350-128-8422

Received: 12 February 2019; Accepted: 26 February 2019; Published: 2 March 2019

Abstract: Modern data centers aim to offer very high throughput and ultra-low latency to meet the demands of applications such as online intensive services. Traditional TCP/IP stacks cannot meet these requirements due to their high CPU overhead and high-latency. Remote Direct Memory Access (RDMA) is an approach that can be designed to meet this demand. The mainstream transport protocol of RDMA over Ethernet is RoCE (RDMA over Converged Ethernet), which relies on Priority Flow Control (PFC) within the network to enable a lossless network. However, PFC is a coarse-grained protocol which can lead to problems such as congestion spreading, head-of-the-line blocking. A congestion control protocol that can alleviate these problems of PFC is needed. We propose a protocol, called P4QCN for this purpose. P4QCN is a congestion control scheme for RoCE and it is an improved Quantized Congestion Notification (QCN) design based on P4, which is a flow-level, rate-based congestion control mechanism. P4QCN extends the QCN protocol to make it compatible with IP-routed networks based on a framework of P4 and adopts a two-point algorithm architecture which is more effective than the three-point architecture used in QCN and Data Center QCN(DCQCN). Experiments show that our proposed P4QCN algorithm achieves the expected performance in terms of latency and throughput.

Keywords: congestion control; P4; data center transport; PFC; lossless; programmable data plane

1. Introduction

Application and storage architecture within data centers are becoming more complicated in recent years. As shown in Reference [1], the large-scale online data intensive services, high-performance deep learning networks, modern telecommunication central office networks and high-speed distributed pools of Non-Volatile Memory Express storage are the four critical data center use cases which are stressing the data center network. These applications raise increasingly demand for high throughput and low latency in data center network, which are generally regarded as two important goals for data center design [2]. Traditional network protocol, such as standard TCP/IP stacks, is designed for the wide-area networks, with high latencies and high CPU overhead [3] and cannot meet the demands in the data center. To address this problem, much work has been done around these two goals [4–6] for data center.

RDMA is an approach that can meet these demands in a data center. It is designed to solve the delay of server-side data processing in network transmission. At present, the mainstream transport protocol of RDMA over Ethernet is RoCE, which is based on the UDP. Compared with TCP protocol. UDP protocol is faster and occupies less CPU resources but it does not achieve reliable transmission. Once packet loss occurs, it can only rely on the upper layer protocol to check and recover, which will greatly reduce the performance of RDMA. The protocol only achieves good performance when running over a lossless network. RoCE relies on PFC to support a lossless environment in the layer 2 networks.

PFC is a coarse-grained port-based mechanism to avoid packet loss. It can cause many network problems such as congestion spreading [7] and deadlocks [8]. Congestion is the primary source of packet loss in the network. It can lead to serious performance degradation [9]. Therefore, PFC scheme should be triggered as a last resort. A congestion control which operates efficiently can alleviate these problems of PFC.

Two broadly adopted practical approaches to support congestion control are end-to-end congestion control and network-assisted congestion control. In an end-to-end congestion control approach, the network layer does not provide explicit support to the transport layer for congestion control purposes, such as the congestion control mechanism in TCP protocol. With network-assisted congestion control, network-layer components (i.e., routers) provide explicit feedback to the sender regarding the congestion state in the network. An example of network-assisted congestion control is the ATM available bit-rate (ABR) congestion control [10]. Several new congestion control schemes are proposed to improve performance in data centers, such as QCN [11], DCQCN [12] and so on. QCN is a flow-level congestion control scheme which is enabled within layer 2 networks and is incompatible with layer 3 networks. DCQCN is an improvement of QCN, extending the QCN to L3 networks. However, DCQCN uses a three-point algorithm architecture which sends congestion feedback from the receiver to sender. It may suffer longer round-trip-time for the ECN (Explicit Congestion Notification) control loop when the data center becomes larger with more switches and more layers in the network.

We design a congestion control protocol—called P4QCN—which extends the QCN based on P4 for the purpose of alleviating problems of PFC within a lossless network. P4 [13] is a programming language mainly used for data planes to provide instructions to the data forwarding plane equipment (such as switches, network cards, etc.) how to handle data packets. By the aid of these features of the programmable data plane, P4QCN can achieve a flow-level, rate-based, network-assisted congestion control protocol. Extending QCN protocol to IP-routed networks is not easy under the traditional network architecture. However, in the context of programmable data plane, we can flexibly achieve such extending. P4QCN extends the QCN protocol to make it compatible with IP-routed networks based on framework of P4 and adopts a two-point algorithm architecture which is more effective than the three-point architecture used in DCQCN.

The paper is organized as follows. In Section 2, we list the related work of congestion control in the data center network and the inadequacies of the existing design. The detail design and the implementation model of P4QCN is presented in Section 3. We introduce the CP-RP architecture of algorithm and the implementation model of P4QCN based on programmable data plane and PISA (Protocol Independent Switch Architecture). In Section 4, we introduce the simulation experiment using mininet and the performance results of P4QCN. In Section 5, we summarize our work and discuss the future work.

2. Related Work

Research revolving around congestion control, applied to high performance data center networks has already been in progress for many years, some of which have been widely used in the data centers network. We will present the related work historically, as it may be useful for us to understand the relevant research.

2.1. RDMA Networks: InfiniBand, RoCE, iWarp

Compared with the traditional network protocol, RDMA does not need the intervention of an operating system. So RDMA can easily achieve ultra-low latency and ultra-high throughput transmission between endpoints. In a word, RDMA has three main characteristics: CPU offload, kernel bypass, zero-copy.

RDMA has three different hardware implementations: (1) InfiniBand [14] is a network designed specifically for RDMA that guarantees reliable transmission at the hardware level. InfiniBand network has a good performance in the throughput and latency, however it uses the custom switch and NIC,

so the cost is very high. Meanwhile, the InfiniBand networking stack cannot be easily deployed in modern data centers. InfiniBand is not compatible with IP and Ethernet technologies. (2) RDMA over Converged Ethernet (RoCE) [15] has been defined to enable the RDMA over the IP and Ethernet. RoCEv2 [16] inherited the philosophy of InfiniBand which needs a lossless Layer 2 network to simplify the transport protocol. PFC [17] is been used in the RoCEv2 to achieve the lossless layer 2 network. RoCE is deployed on a large scale in the data center in reality due to technical and economic reasons [18]. (3) iWarp [19] is also an Ethernet-based RDMA technology which involves a different concept to InfiniBand and RoCE. iWarp protocol is designed to handle packet loss efficiently rather than making the Layer 2 network lossless. To ensure the generality, iWarp implements the entire TCP/IP protocol stack on the NIC. However, the full implementation of TCP/IP requires multiple translation between RDMA abstraction and TCP abstraction. As a result, the iWarp is more complex than the RoCE and has a higher cost and a lower performance [20].

2.2. Priority Flow Control

PFC, which is also referred to as Class-based Flow Control, is a mechanism that prevents frame loss that is due to congestion. When the queue length exceeds the set threshold, the switch sends a pause frame to the upstream switch. When the congestion is mitigated (the queue length comes under the threshold), a resume frame is generated to restart the transmission. PFC can achieve a lossless network when configured correctly.

PFC is a coarse-grained mechanism which can lead to some problems as shown in Reference [21]. For example, as shown in Figure 1, when the congestion occurs in the S1 switch and the queue size of the ingress port exceeds the threshold, a PFC PAUSE frame will be sent to the upstream switch (L1) due to the cascading effect of PAUSE frame and switch L1 stops sending data to switch S1. Because PFC will stop all traffic in a particular traffic class at the port, packet from h21 are also suspended, even if the destination port of the data stream from h21 is not congested. The phenomenon is known as head-of-line blocking. The cascade effect of the PAUSE frame can also lead to congestion spread.

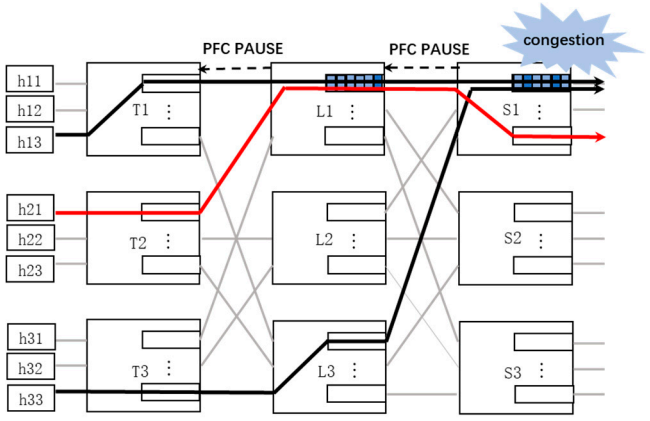


Figure 1. Priority Flow Control (PFC) Head-of-Line Blocking.

The PFC mechanism itself provides the class of priority to address the Head-of-Line blocking problem. However, the protocol only supports 8 priority classes. When the topology and senders are expanding, the performance will be bad. Furthermore, flows within the same class will still have the same problems shown above.

2.3. Quantized Congestion Notification

QCN is a congestion control mechanism by sending congestion feedback from switches to end hosts. QCN provides congestion control based on network feedback (similar to DCTCP [22]/ECN [23]) but operated at the Ethernet layer and does not isolate incast [24].

Basic QCN algorithm is consist of 2-point architecture: Congestion Point (CP) and Reaction Point (RP). CP is the point where the congestion occurs, it is responsible for sampling frames that are being sent in the buffer, computing and quantizing the feedback (Fb) value, RP is the point where the sending rate of a flow is changed due to congestion signals. When congestion message arrives, the data source performs multiplicative decrease of the sending rate. Also, when there is no congestion message arrives, the data source will gradually increase its transmission rate to detect the available bandwidth or recover the "lost" rate due to congestion. QCN also provides a three-point algorithm. The algorithm using three-point architecture shifts part work (generating feedback packets) of CP in two-point architecture to the destination as the Notification Point (NP) point.

QCN is a congestion control mechanism which is easy to deploy and has light resource requirement. QCN enables flow-level congestion control which is fine-grained compared to PFC, it resolves some problems of PFC mentioned above. However, QCN is designed within an L2 domain which identify flows using source/destination MAC address, therefore, it is not compatible with IP-routed networks.

2.4. Other Relevant Protocol

Some other congestion control protocols are also proposed, such as DCQCN [12], TIMELY [25] and IRN [21]. DCQCN is a rate-based, end-to-end congestion control protocol, that builds upon QCN and DCTCP. It consists of RP, CP and NP. The arriving packets are marked by the CP algorithm when the queue length exceeds the threshold. NP receives marked packets indicating congestion in the network and then sends Congestion Notification Packets (CNP) [14] back to the sender. When the RP gets an CNP, it reduces its sending rate of traffic injection into the network. DCQCN uses the ECN to feedback the congestion information to the sender. It may suffer longer round-trip-time for the ECN control loop when the data center becomes larger with more switches and more layers in the network. Larger networks also exist more data in-flight, making it difficult to absorb bursts of traffic before ECN congestion control takes effort.

IRN (Improved RoCE NIC) [21] provides another solution, which allows packet loss and improves network performance by adopting a more effective packet loss recovery mechanism. It makes two key changes to current RoCE NICs: (1) improving the loss recovery mechanism; (2) basic end-to-end flow control mechanism. IRN eliminates the need for PFC. In the diversified network environment, the performance degradation caused by packet loss is different, especially in some elephant-dominated networks, packet loss recovery will aggravate congestion.

PFC, ETS (Enhanced Transmission Selection) [26], QCN and DCBX (Data Center Bridging Exchange Protocol) [26] are a set of enhancements to the Ethernet local area network communication protocol for use in data center environments. These technologies are indeed complementary and typically used in combination. ETS is to enable multiple traffic types to coexist in the same network. DCBX is an extension protocol based on LLDP (Link Layer Discovery Protocol) used to automatically negotiate and configure PFC, ETS and CN among devices.

Some other congestion control protocols such as DCTCP [22]/TCP-Bolt [27] are window-based algorithms. They are complex in implementation and implemented in end host stacks and thus have high CPU overhead and high latency.

3. Modeling of P4QCN

In the past few years, one of the most active areas in computer networking is Software Defined Networking (SDN) [28] which separates the control-plane and the data-plane. The separating of the control-plane and data-plane brings additional flexibility of the networks. However, SDN still assumes that the behavior of the network data-plane is fixed. This hinders the further innovation of the network. In this context, programmable data plane technology has emerged. Programmable data plane has the characteristics of openness, flexibility and protocol independence. Based on the programmable data plane, we design P4QCN algorithm which is a network-assisted and flow-based congestion control

protocol and extends the QCN protocol to IP-routed networks. P4QCN is based on P4 [29] and is implemented in the P4-Capable switch.

As mentioned in Section 1, one of the main contributions of P4QCN is extending the QCN protocol to L3 networks within the framework of P4. Flows are defined using source/destination either MAC address or IP address and then we achieve a congestion control algorithm based on QCN. The P4QCN architecture is consist of two points: RP and CP.

3.1. Overall Architecture of P4QCN

P4QCN is a two-point congestion control architecture including RP and CP as shown in Figure 2. Compared with three-point architecture, CP on switches in the two-point architecture sends congestion feedback packet directly to RP on the source. Therefore, it is more effective and less complex than three-point architecture, where feedbacks are sent via NP. Next, we introduce the CP and RP algorithms in detail.

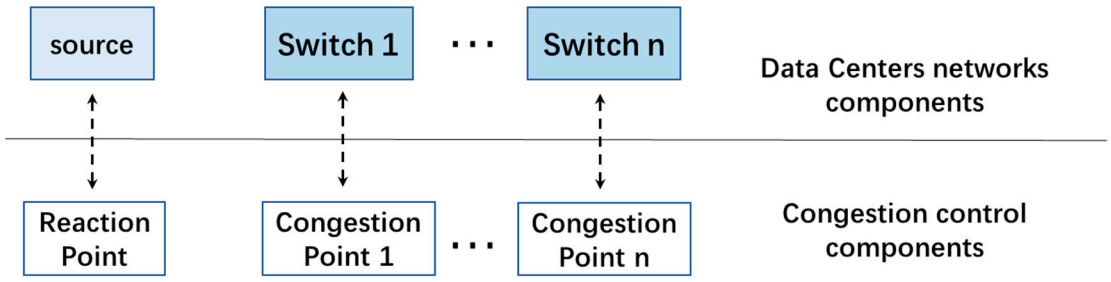


Figure 2. Architecture of P4QCN algorithm.

3.1.1. CP Algorithm

As shown in Figure 2, the CP algorithm deploys on switches in the network. It mainly achieves two functions: (1) predicating the congestion status; and (2) generating feedback packets. The probability to send feedbacks is shown in Equation (1). When the queue size in the egress port exceeds Q_{min}, the feedback packets are generated at a certain probability. When the queue size reaches to Q_{max}, the feedback packets will be generated and sent to RP at a probability of one hundred percent. $P_{feedback}$ is the probability of sending a feedback packet, q is the current queue size on the egress port indicating the current congestion status.

$$P_{feedback} = \begin{cases} \alpha(q - Q_{min}) & Q_{min} < q < Q_{max} \\ 1 & Q_{max} < q < Q_{pfc} \end{cases} \quad (1)$$

As shown in Figure 3, the probability of feedback packet generation is defined similar as in QCN [24]. Since PFC protocol is the last resort to ensure the achievement of lossless network, P4QCN should been triggered before PFC. It means that packets in flight during P4QCN message taking effect should been taken into account. PFC protocol is triggered when the queue size in the ingress port exceeds the Q_{pfc}. P4QCN is designed to reduce this occurrence.

When the P4QCN mechanism is triggered as the growth of queue size, the feedback packet (FBP) is generated according to CP algorithm. ECN field of FBP is set to "11" to indicate congestion. We can extract the source/destination IP address from the arriving packet at egress queue based on PISA (Protocol independent switch architecture) in programmable data plane. The congestion flow is defined by the source/destination IP address. Then the information of the congestion is sent to the source of the flow. In order to avoid frequent transmission of feedback information, the congestion point generates at most one feedback packet every N micro-second for a same congestion flow when the P4QCN is triggered.

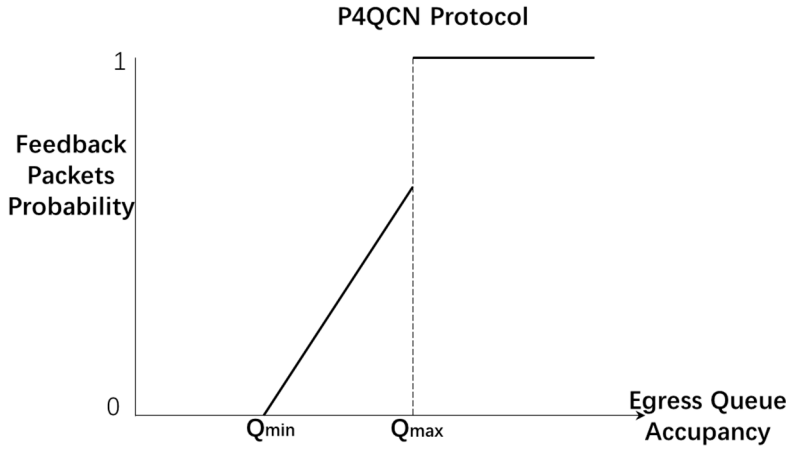


Figure 3. Feedback Packet Generation Probability.

The algorithm of CP is implemented by the original P4 program. The P4 program of P4QCN is then compiled and configured onto the P4 target (bmv2 in our testbed). We provide some definitions and phrase of P4 [30] so that the description of the model can be understood easily.

- **Header definitions:** the format (the set of fields and their sizes) of each header within a packet.
- **Parser:** the permitted header sequences within packets.
- **Pipeline layout and control flow:** the layout of tables within the pipeline and the packet flow through the pipeline.

Figure 4 describes the CP algorithm based on protocol independent switch architecture (PISA) of P4. It is based upon an abstract forwarding model consisting of a parser and a set of match+action table resources, divided between ingress and egress. P4 is a declarative language for expressing how packets are processed by the pipeline of a network forwarding element such as a switch, NIC, router or network function appliance.

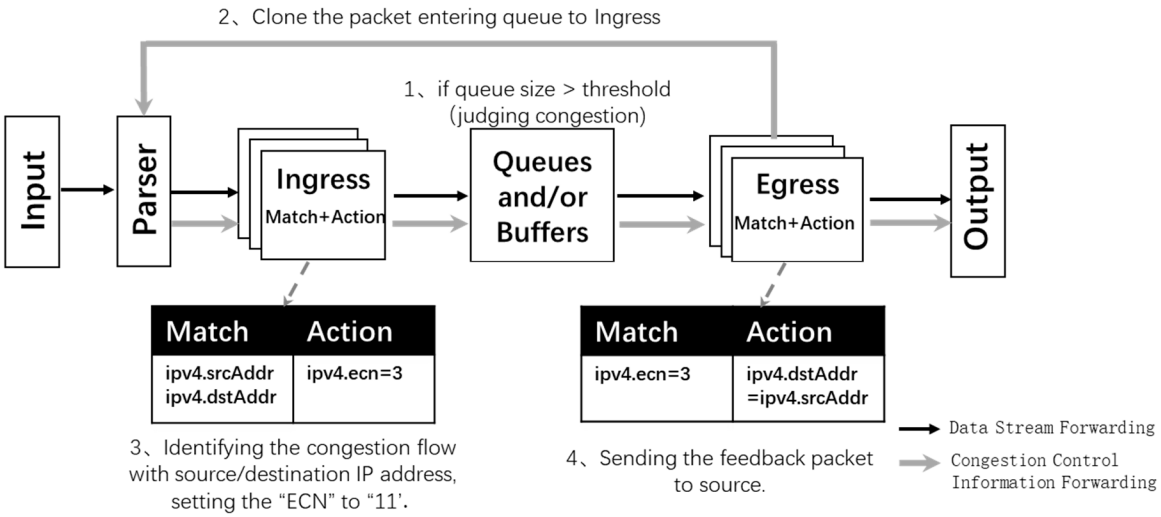


Figure 4. CP algorithm.

When the queue size of egress exceeds the threshold of P4QCN, the packet entering the egress queue is cloned to ingress. In ingress, the ECN field of the packet is set to "11" and then the packet is sent to the source of the packet by modifying the destination IP address. Furthermore, when the ingress queue size exceeds the threshold of PFC, the packet is cloned to ingress and then be modified to the upstream switch to pause the sending of the upstream. The above operation is implemented

by defining match+action table and controlling the flow logic in P4. The CP point is implemented in P4 language. The code of CP algorithm is compiled into a configuration file and loaded on the forwarding device. Table 1 shows the header and metadata fields extracted from packet by parser in P4 terminology. The parser identifies the headers present in each incoming packet.

Table 1. Header and metadata fields.

Phrase	Header Definitions in P4
Queue size	queueing_metadata.enq_qdepth
Source	ipv4.srcAddr
Destination	ipv4.dstAddr
ECN	ipv4.ecn
Ingress port	ig_intr_md_ingress

3.1.2. RP Algorithm

As shown in Figure 2, the RP algorithm deploys on ending hosts in the network. When the RP receives a FBP, it reduces sending rate of traffic injection into the network and records the rate before receiving feedback information as a target rate for fast recovery. The algorithm of reducing rate is shown in Equation (2), where R_{old} is the rate before receiving FBP, R_t is the target rate for fast recovery and R_c is the current rate. β is the rate reduction factor which is a constant, representing the degree of rate reduction. The maximum value of β is set to 1, so the maximum rate reduction is half of the original sending rate when a feedback packet is received.

$$\begin{aligned} R_t &= R_{old} \\ R_c &= R_c(1 - \beta/2) \\ R_d &= R_t - R_c \end{aligned} \tag{2}$$

The source limits its transmission rate based on the FBP received from CP. Then, the source will gradually increase its transmission rate to recover the "lost" rate due to congestion to detect the available bandwidth. The process of increasing the rate includes two stages: the fast recovery and the active increase.

$$R_c = (R_t + R_c)/2 \tag{3}$$

When the sending rate is reduced, a byte counter is reset and enters the fast recovery process. The fast recovery stage consists of N cycles. At the end of each cycle, R_t remains unchanged and R_c is updated according to Equation (3). The byte counter records the number of sending byte during one cycle and when the value of bytes counter reached a threshold, it records one cycle. The value of N is suggested to set to 5 in QCN. The process of rate increase in fast recovery is shown in Figure 5. When the N cycles has been completed, the stage of active increase begins to detect the available bandwidth.

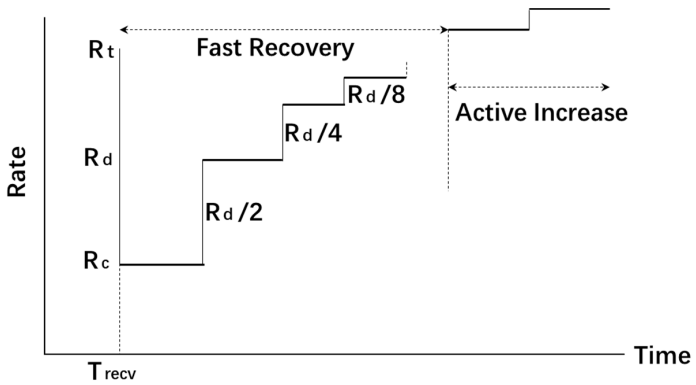


Figure 5. Fast recovery and active increase.

After five cycles of fast recovery, the byte counter enters the active increase. This stage is used to find redundant available bandwidth. RP performs operations as shown in Equation (4). R_{AI} is a constant which is set to 5Mbps in P4QCN protocol.

$$\begin{aligned} R_t &= R_t + R_{AI} \\ R_c &= (R_t + R_c)/2 \end{aligned} \tag{4}$$

The RP algorithm is implemented by a finite state machine (FSM). The FSM contains three states and the migration between them. Three states of the FSM represent three processes of RP algorithm: rate reduction, fast recovery and active increase.

Each host runs an instance of the RP process, which receives the feedback message from CP point. When receiving the feedback packet, the RP processor reduces the sending rate of the flow injection into the network and records the rate before receiving feedback information as a target rate for fast recovery. When RP process enters the fast recovery, the sending rate increases as the Equation (3) describes. After N cycles of the fast recovery, RP process enters the active increase to detect the available bandwidth of the network. The instance of RP process is implemented in python file, the script achieves the function of the FSM described in Figure 6.

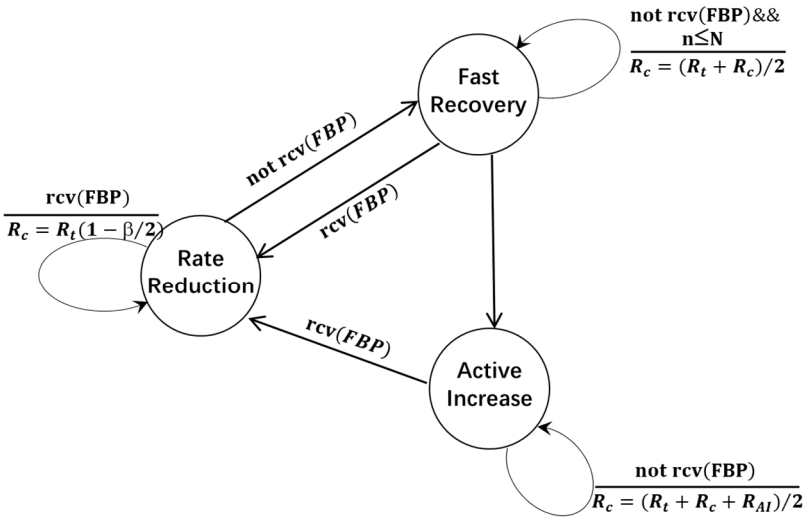


Figure 6. FSM of RP algorithm.

Figure 6 shows the operation of RP algorithm. The finite state machine contains three states and the migration between them. The arrows in the FSM description indicate the transition of the algorithm from one state to another. As shown in Figure 6, the event causing the transition is shown above the horizontal line labeling the transition and the actions taken when the event occurs are shown below the horizontal line. Rcv (FBP) represents the event of receiving FBP. The FSM is implemented in Python.

3.2. Analysis of P4QCN

As mentioned above, the targets of congestion control algorithm of P4QCN are: (1) ensuring lossless of packet working with PFC and reducing the negative impact of PFC. (2) maintaining cache queue occupancy at a desired value to increase link utilization. In a sense, these two objectives are contradictory. We must ensure that we do not lose packets and also seek for the higher link utilization. It means that the trigger of congestion control mechanism is neither too early nor too late. So the setting of the threshold is important to P4QCN. In addition, packet loss due to internal cache structure of switches should also be taken into account.

3.2.1. Switch-Inner Lossless Mechanism

In order to ensure lossless network, packet loss within the switch should also been taken into account. First of all, let us take a look at where queues will appear in the switch. Theoretically, it is clear that packet queues may form at both the input ports and the output ports. The location and extent of queueing (either at the input port queues or the output port queues) will depend on the traffic load, the relative speed of the switching fabric and the line speed. Input-Queued switch and Output-Queued switch have their own merits and flaws, respectively. Discussions on queuing design is out of the scope of this paper and much research has revolved around this topic. Details of the Input-Queued switch and Output-Queued switch design can be found in Virtual Output Queuing (VOQ) [31] and Combined Input/Output Queueing (CIOQ) [32] and so forth.

PFC is implemented on the ingress queues of a switch. An example of packet loss within switch is shown in Figure 7. In the scenario shown in Figure 7, both input 0 and input N are forwarded to output 1, it is possible that the egress queue has overflowed before the ingress queues triggering PFC. A coordination mechanism between ingress queue and egress queue to ensure a lossless environment within a switch should be taken into account. This coordination mechanism depends on the structure of the switch cache. In this paper, the buffer of the bmv2 switch is shared among all ports.

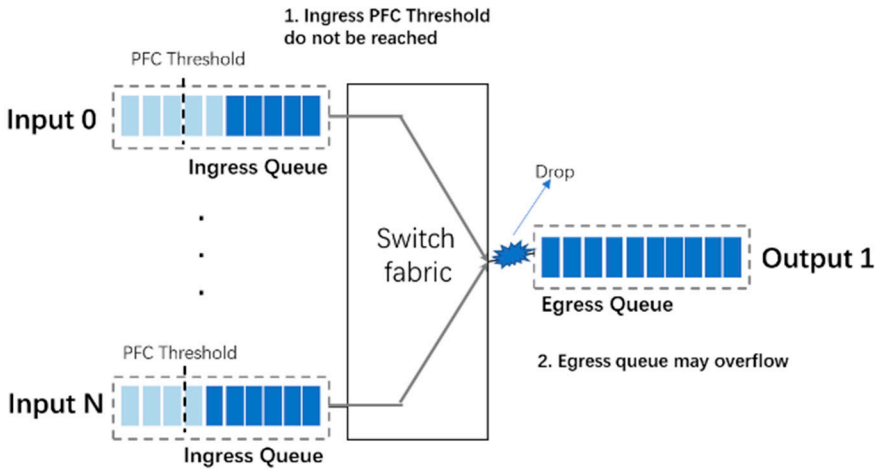


Figure 7. Switch-Inner packet loss.

3.2.2. Setting of Threshold

We now discuss the threshold for triggering P4QCN on egress queue and PFC on ingress queue.

- **PFC threshold (Q_{PFC}):** PFC is triggered as the last resort to ensure the lossless of network. PFC threshold is the maximum value of the queue size can grow to. To avoid the overflow of the queue, packets that are in flight and sent by upstream switch during processing the PFC PAUSE message should been taken into consideration. Some buffer space (Q_{flight}) should been reserved to receive these packets. Packets in flight for a flow is defined by the bandwidth-delay product (BDP) of the network in P4QCN, as suggested in Reference [33]. The total switch buffer size is Q, there are n ports in the switch and support 8 PFC priorities. The threshold should follow: $Q_{PFC} \leq \left(Q - 8nQ_{flight}\right)/8n$.
- **P4QCN threshold (Q_{P4QCN}):** P4QCN should been triggered before PFC. Furthermore, PFC should not be triggered before P4QCN take effect, it means that packets in flight during P4QCN message taking effect should be taken into account. So $Q_{P4QCN} \leq Q_{PFC} - Q_{flight}$. The switch buffer is shared among all ports in bmv2 switch which used in our testbed. P4QCN adopts the threshold setting method used in algorithm DCQCN [12] which is compatible for shared cache architecture. $Q_{P4QCN} < \alpha\left(\mathrm{B} - 8\mathrm{n}Q_{flight}\right)/(8n(\alpha+1))$; we use $\alpha = 8$.

3.2.3. IP-Enabled Protocol Extension

P4QCN is implemented based on programmable data plane and P4(Programming Protocol-Independent Packet Processors). P4 is a declarative language for expressing how packets are processed by the pipeline of a network forwarding element. It is based upon an abstract forwarding model consisting of a parser and a set of match+action table resources. The parser identifies the headers present in each incoming packet. Each match+action table performs a lookup on a subset of header fields and applies the actions corresponding to the first match within each table.

Since QCN is designed within an L2 domain which identify flows using source/destination MAC address, it is not compatible with IP-routed networks. P4QCN extends QCN protocol to L3 networks within the framework of P4. P4QCN defines flows with either source/destination IP address or MAC address enabled by the parser in P4 model. P4 models the parser as a state machine. This can be represented as a parse graph with each state a node and the state transitions as edges. Figure 8 shows a simple example of the parser, the source/destination IP address can be extracted from the header of the packet.

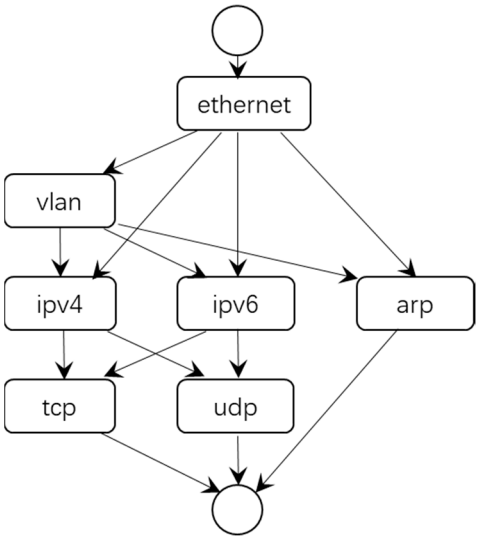


Figure 8. Parser graph.

The parser produces the representation of the packet on which match + action stages operate. It is the set of header instances which are valid for the packet. Therefore, we can define the headers and parser according to different protocol stack.

The source/destination IP address extracted from the header of the packet will be used in match + action stages to identify the congestion flow.

4. Experiments Results

The system and its implementation of P4QCN is described in Figure 9. We implement the CP algorithm based on the abstract model of programmable data plane as described in the right part of Figure 9. The parser graph defines how headers are identified within a packet and produces the representation of the packet on which match + action stages operate. The match + action table and the control flow achieve the logic of CP algorithm. The CP algorithm written in P4 language is compiled into a JSON configuration file and loaded into the data plane forwarding device. At the same time, the API is opened to the control plane. The data plane and the control plane communicate with each other using thrift protocol. RP algorithm implements three processes: rate reduction, fast recovery and action increase on the host ending of networks as described in the left part of Figure 9. RP algorithm is implemented with Scapy library of Python language.

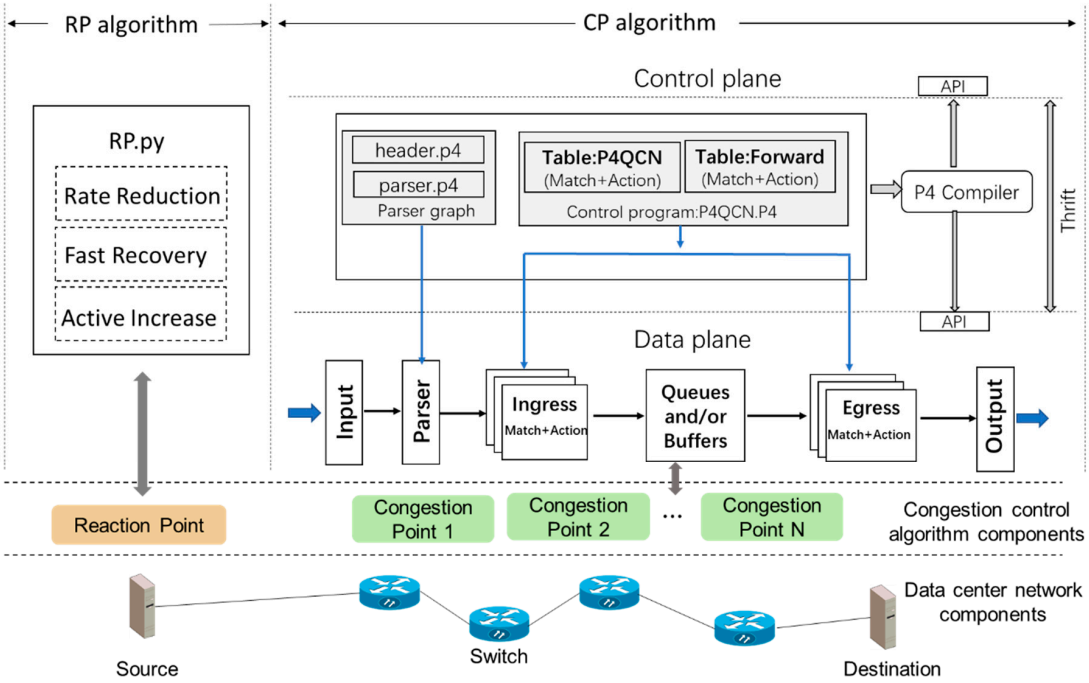


Figure 9. P4QCN system implementation.

We simulate a 3-tier datacenter topology in mininet to represent the current data center network. As shown in Figure 10, there are two spines (S1–S2), four leaves (L1–L4) and four ToRs (T1–T4) in the topology. We use bmv2(behavior model version 2) as the P4 software switch in mininet which is written in C++. We implement each switch as CP Point and each host as RP point.

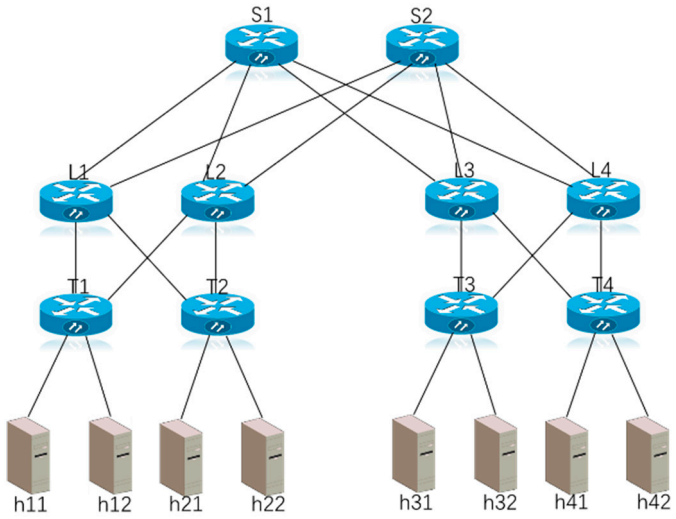


Figure 10. Testbed topology.

Limited by the performance of the simulation environment, we conclude that after many experiments the link bandwidth is the most stable when it is below 10 Mbps. We set the bandwidth of the link 7Mbps in our testbed; $Q_{P4QCN} = Q_{min} = Q_{max} = 11$; $Q_{PFC} = 14$ (the value of Q_{P4QCN} and Q_{PFC} is the quantization of queue occupancy in the virtual switch model); $N = 5$; $\beta = 1$, $R_{AI} = 5$Mbps respectively. We extend mininet by modifying the CLI class and Mininet class of mininet and use the extended iperf tools of mininet with a random traffic model for the data center environment. In the random traffic model, any host in the mininet topology randomly initiates a UDP data stream to another host with a random flow size. We use random traffic model to generate background traffic

in the data center network in our experiments. The congestion may happen on T4 switch in our experiments. We send a UDP flow with 4M/s from host11 to host41 via iperf and meanwhile set the sending rate of background traffic with 2M/s, 3M/s, 4M/s and 5M/s respectively to introduce different level of congestion on T4 switch.

4.1. Preliminary Verification of P4QCN

We deploy the P4QCN on the testbed. The CP algorithm is deployed on the bmv2 switch and the RP algorithm is deployed on the ending host. When congestion occurs, we capture the packet which is marked with "ECN" with Wireshark. As shown in Figure 11, "ECN" is set to "11." The sending rate of the source is recorded in the Figure 12, we can see the process of "rate reduction" "fast recovery" and "active increase."

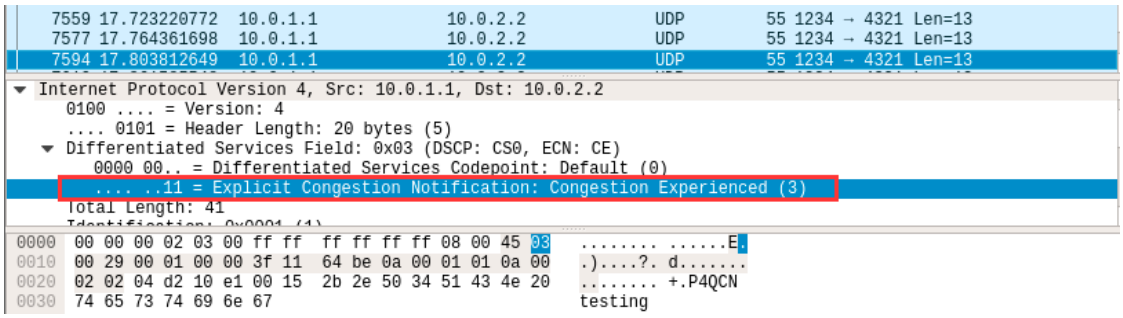


Figure 11. Congestion mark.

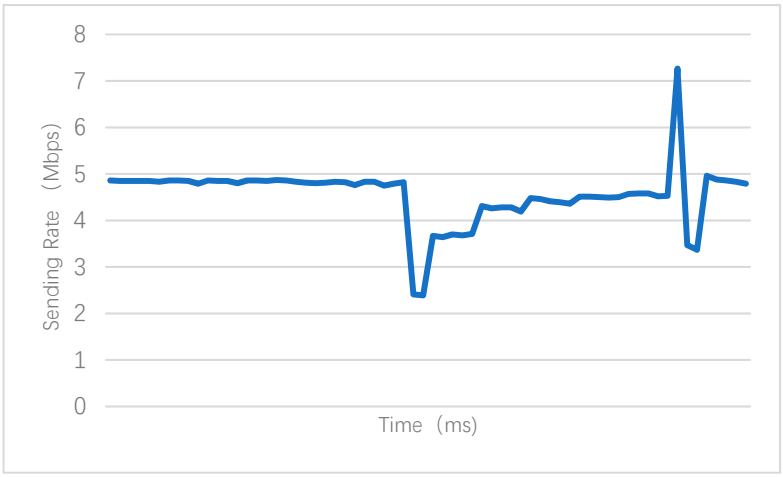


Figure 12. Sending rate of source.

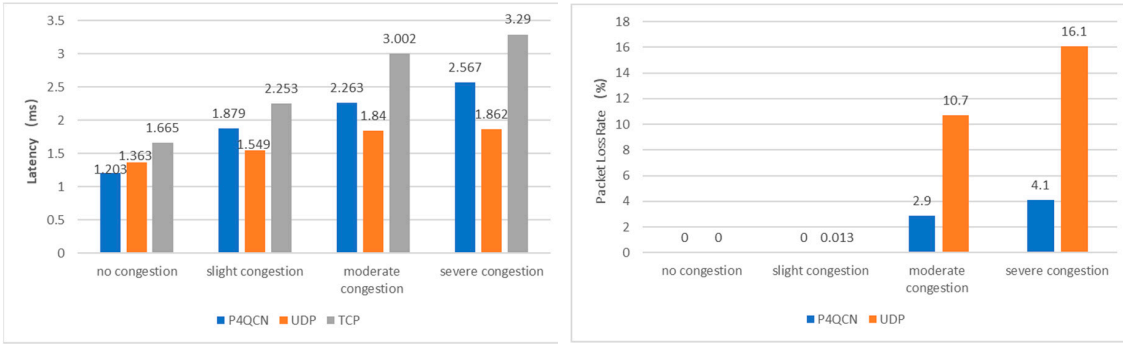
4.2. Comparative Analysis

We set up five groups of comparative experiments to verify the performance of P4QCN from different perspectives.

4.2.1. P4QCN Compared with UDP and TCP

We compare the performance metrics of UDP flows with and without P4QCN. As shown in Figure 13a,b, we can see that when congestion occurs, the packet loss with P4QCN is much lower than the case without P4QCN, in particular, when congestion level is high. It shows that P4QCN can effectively reduce the packet loss. In Figure 13a, the results indicate the delay with P4QCN increases a bit compared with UDP on the other hand.

We also compare the latency of TCP flows with P4QCN in Figure 13a. The results show that the TCP congestion control and loss recovery mechanism causes higher latency than P4QCN.



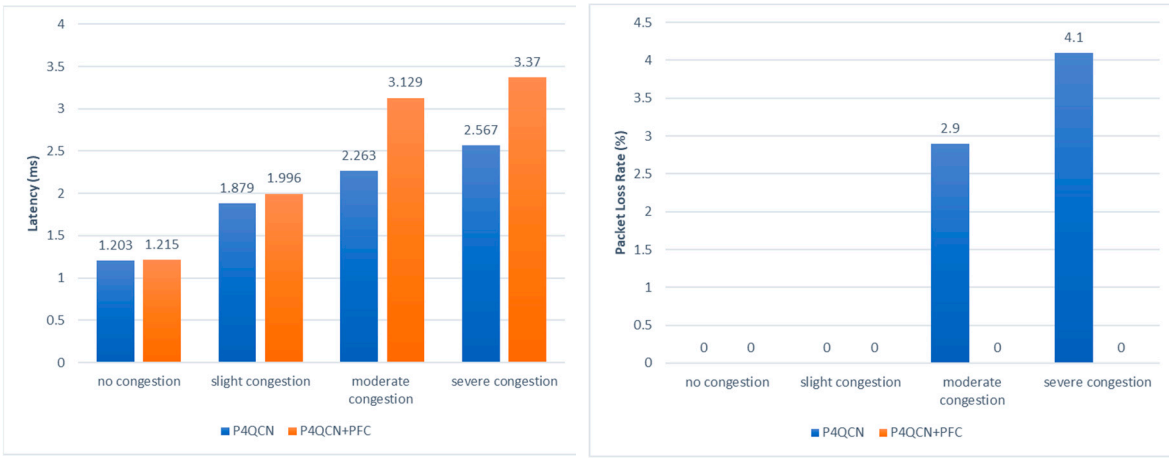
(a) **(b)**

Figure 13. (**a**) Latency comparison of P4QCN, UDP and TCP; (**b**) Packet loss comparison of P4QCN and UDP.

4.2.2. P4QCN Compared with P4QCN + PFC

We implement the PFC mechanism in the programmable data plane. In this group of experiments, we compare the network performance of P4QCN only with P4QCN and PFC both deployed simultaneously.

As shown in Figure 14a,b, when P4QCN and PFC are deployed together, the latency increases compared with P4QCN is deployed alone. This is due to additional delay introduced by PFC. The results of Figure 14b show packet losses with P4QCN, therefore, P4QCN alone cannot achieve lossless.



(a) **(b)**

Figure 14. (**a**) Latency comparison of P4QCN and P4QCN + PFC; (**b**) Packet loss comparison of P4QCN and P4QCN + PFC.

From this set of comparative experiments, we can conclude that: (1) P4QCN deployment alone cannot achieve lossless, it must be deployed with PFC to achieve a lossless network; (2) P4QCN can only reduce the number of PFC triggers but it cannot completely avoid PFC triggers.

4.2.3. P4QCN + PFC Compared with PFC

In this set of experiments, we compare the case of both P4QCN and PFC deployed with the case only PFC deployed in terms of latency. Both cases can achieve lossless but P4QCN + PFC can reduce the latency compared with PFC as shown in Figure 15. This is because P4QCN effectively reduces the number of PFC triggers. The results of experiments show that P4QCN is effective in improving some of the negative effects of PFC. Therefore, P4QCN is helpful to improve the performance of RoCE network when combined with PFC.

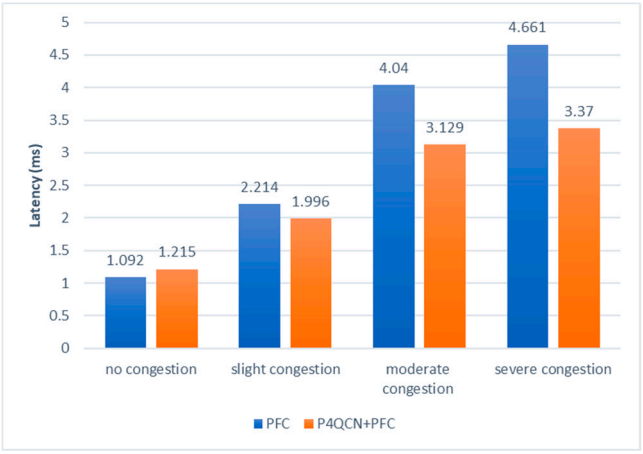
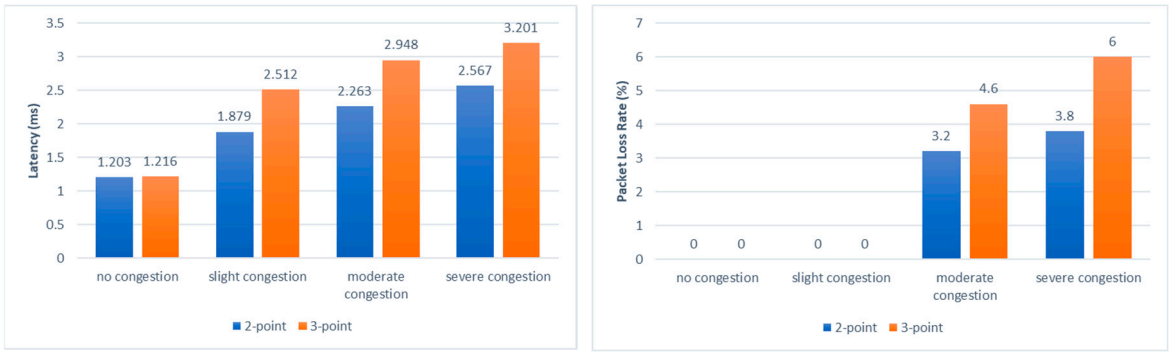


Figure 15. Latency comparison of PFC and P4QCN + PFC.

4.2.4. CP-RP Architecture Compared with CP-NP-RP Architecture

P4QCN uses two-point architecture which contains CP and RP, while some algorithms such as DCQCN use a three-point architecture which contains CP, RP and NP (Notification Point). We implement a three-point architecture algorithm in the programmable data plane model. The algorithm using three-point architecture shifts part work (generating feedback packets) of CP in two-point architecture to the destination as the NP point.

The results of experiments are shown in Figure 16a,b. Compared with three-point, two-point architecture has lower latency and smaller packet loss rate. In three-point architecture, the congestion information needs to go through a longer path and then feed back to the source. Therefore, the congestion control algorithm takes longer time to take effect. So, two-point architecture is more efficient.



(a) **(b)**

Figure 16. (**a**) Latency comparison of 2-point and 3-point; (**b**) Packet loss comparison of 2-point and 3-point.

4.2.5. Bandwidth Utilization Comparison under Different Thresholds

Next, we compare the bandwidth utilization of P4QCN and PFC and the impact of threshold on bandwidth utilization by setting different thresholds of P4QCN (Q_{P4QCN}) and PFC (Q_{PFC}). The value of threshold is the quantization factor of the queue occupancy in the virtual switch model. In the case of slight congestion, the results of experiments are shown in Figure 17.

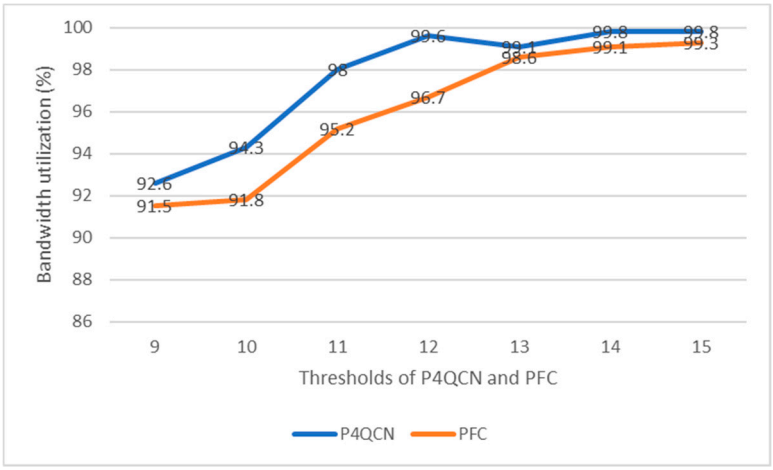


Figure 17. Bandwidth utilization comparison of P4QCN and PFC.

We compare the bandwidth utilization of P4QCN and PFC deployed separately under different thresholds to further analyze the impact of these two different mechanisms on network performance. The results in Figure 17 show that the P4QCN has a higher bandwidth utilization than PFC when the threshold is lower. They are getting closer when the threshold is getting higher. This is because, when the PFC is triggered, the upstream switch will be paused to send packets while P4QCN reducing the sending rate of the source. Therefore, PFC is more coarse-grained, which has a negative impact on network performance.

The results in Figure 17 also show that the higher the threshold of Q_{P4QCN} is, the more the bandwidth utilization would be. P4QCN is triggered more frequently when the threshold is getting smaller, as P4QCN reduces the sending rate of source which leads to a reduction of the bandwidth utilization. PFC has the same situation as P4QCN.

As the resource needed for computing and communication in P4QCN is comparable to that in QCN/DCQCN with either virtual or hardware switches, we expect our proposed P4QCN algorithm in real networks will achieve the same performance as in mininet environment in terms of network latency and bandwidth utilization, however higher processing speed.

5. Conclusions and Future Work

RDMA over Ethernet namely RoCE over the UDP is the current transport protocol designed to achieve high performance in the datacenter network. In the paper, we propose P4QCN, which is a flow-level, rate-based, network-assisted congestion control protocol, implemented in the framework of P4. It extends the QCN protocol to IP-routed networks and uses two-point (CP-RP) architecture to reduce the end-to-end latency and the packet loss rate.

We verify the performance of the P4QCN algorithm and compare it with other algorithms in the simulation network environment. The experiments are limited by the performance of virtual switches. In future work, we will verify P4QCN in the real network.

Author Contributions: Conceptualization, J.Y. and J.G.; methodology, J.Y. and J.G.; software, J.G.; validation, J.G. and Y.Z.; formal analysis, J.Y. and J.G.; investigation, J.Y. and J.G.; resources, J.Y., J.G. and Y.Z.; data curation, J.Y. and J.G.; writing—original draft preparation, J.G.; writing—review and editing, J.Y., J.G. and Y.Z.; visualization, J.G.; supervision, J.Y. and Y.Z.; project administration, J.Y.; and funding acquisition, J.Y.

Funding: This research was funded in part by National Natural Science Foundation of China, grant number 61472389.

Acknowledgments: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that improved the quality of this paper.

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

References

1. Nendica: IEEE 802 "Network Enhancements for the Next Decade" Industry Connections Activity. 2018. Available online: https://1.ieee802.org/802-nendica/ (accessed on 20 September 2018).
2. Handley, M.; Raiciu, C.; Agache, A.; Moore, A.W.; Antichi, G.; Wojcik, M. Re-architecting datacenter networks and stacks for low latency and high performance. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 20–25 August 2017; pp. 29–42.
3. Marinos, I.; Watson, R.; Handley, M. Network stack specialization for performance. In Proceedings of the 2014 ACM Conference on SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 175–186.
4. Alizadeh, M.; Kabbani, A.; Edsall, T.; Prabhakar, B.; Vahdat, A.; Yasuda, M. Less is more: trading a little bandwidth for ultra-low latency in the data center. Proceedings of 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012; pp. 253–266.
5. Greenberg, A.; Hamilton, J.R.; Jain, N.; Kandula, S.; Kim, C.; Lahiri, P.; Maltz, D.A.; Patel, P.; Sengupta, S. VL2: A scalable and flexible data center network. In Proceedings of the ACM SIGCOMM 2009 Conference on Data communication, Barcelona, Spain, 16–21 August 2009; pp. 51–62.
6. Perry, J.; Ousterhout, A.; Balakrishnan, H.; Shah, D.; Fugal, H. Fastpass: A centralized "zero-queue" datacenter network. In Proceedings of the 2014 ACM conference on SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 307–318.
7. Hu, S.; Zhu, Y.; Cheng, P.; Guo, C.; Tan, K.; Padhye, J.; Chen, K. Deadlocks in Datacenter Networks: Why Do They Form and How to Avoid Them. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 9–10 November 2016; pp. 92–98.
8. Shpiner, A.; Zahavi, E.; Zdornov, V.; Anker, T.; Kadosh, M. Unlocking Credit Loop Deadlocks. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 9–10 November 2016; pp. 85–91.
9. Jalaparti, V.; Bodik, P.; Kandula, S.; Menache, I.; Rybalkin, M.; Yan, C. Speeding up distributed request-response workflows. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013; pp. 219–230.
10. Jain, R.; Kalyanaraman, S.; Fahmy, S.; Goyal, R.; Kim, S. Tutorial Paper on ABR Source Behavior, ATM Forum/96-1270. 1996. Available online: http://www.cse.wustl.edu/~{}jain/atmf/ftp/atm96-1270.pdf (accessed on 12 October 2018).
11. IEEE. 802.11Qau. Congestion Notification. 2010. Available online: https://1.ieee802.org/dcb/802-1qau/ (accessed on 12 October 2018).
12. Zhu, Y.; Eran, H.; Firestone, D.; Guo, C.; Lipshteyn, M.; Liron, Y.; Padhye, J.; Raindel, S.; Yahia, M.H.; Zhang, M. Congestion Control for Large-Scale RDMA Deployments. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, London, UK, 17–27 August 2015; pp. 523–536.
13. Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; Mckeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; Walker, D. P4: Programming Protocol-Independent Packet Processors. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–95. [CrossRef]
14. InfiniBand Architecture Volume 1, General Specifications, Release 1.2.1. Available online: https://cw.infinibandta.org/document/dl/7143 (accessed on 12 October 2018).
15. Recio, R.; Metzler, B.; Culley, P.; Hilland, J.; Garcia, D. A Remote Direct Memory Access Protocol Specification. *RFC 5040*. 2007. Available online: https://tools.ietf.org/html/rfc5040 (accessed on 12 October 2018).
16. Infiniband Trade Association. Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.2 Annex A17: RoCEv2 (IP Routable RoCE). 2014. Available online: https://cw.infinibandta.org/document/dl/7781 (accessed on 26 November 2018).
17. IEEE. 802.11Qbb. Priority Based Flow Control. 2011. Available online: https://gestaltit.com/syndicated/ivan/introduction-802-1qbb-priority-based-flow-control-pfc/ (accessed on 26 November 2018).
18. Guo, C.; Wu, H.; Deng, Z.; Soni, G.; Ye, J.; Padhye, J.; Lipshteyn, M. RDMA over commodity ethernet at scale. In Proceedings of the 2016 ACM SIGCOMM Conference, Florianopolis, Brazil, 22–26 August 2016; pp. 202–215.
19. Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.2 Annex A16: RDMA over Converged Ethernet (RoCE). 2010. Available online: https://cw.infinibandta.org/document/dl/7148 (accessed on 26 November 2018).

20. RoCE vs. iWARP Competitive Analysis. 2017. Available online: http://www.mellanox.com/related-docs/whitepapers/WP_RoCE_vs_iWARP.pdf (accessed on 26 November 2018).
21. Mittal, R.; Shpiner, A.; Panda, A.; Zahavi, E.; Krishnamurthy, A.; Ratnasamy, S.; Shenker, S. Revisiting Network Support for RDMA. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, Budapest, Hungary, 20–25 August 2018; pp. 313–326.
22. Alizadeh, M.; Greenberg, A.; Maltz, D.; Padhye, J.; Patel, P.; Prabhakar, B.; Sengupta, S.; Sridharan, M. Data Center TCP (DCTCP). In Proceedings of the ACM SIGCOMM 2010 Conference, New Delhi, India, 30 August–03 September 2010; pp. 63–74.
23. Ramakrishnan, K.; Floyd, S.; Black, D. RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP. Technical Report. 2001. Available online: https://tools.ietf.org/html/rfc3168 (accessed on 5 January 2019).
24. Alizadeh, M.; Kabbani, A.; Atikoglu, B.; Prabhakar, B. Stability analysis of QCN: The averaging principle. In Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, San Jose, CA, USA, 7–11 June 2011; pp. 49–60.
25. Mittal, R.; Lam, V.; Dukkipati, N.; Blem, E.; Wassel, H.; Ghobadi, M.; Vahdat, A.; Wang, Y.; Wetherall, D.; Zats, D. TIMELY: RTT-based Congestion Control for the Datacenter. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, London, UK, 17–21 August 2015; pp. 537–550.
26. IEEE. 802.1Qaz. Enhanced Transmission Selection. 2011. Available online: https://1.ieee802.org/dcb/802-1qaz/ (accessed on 8 February 2019).
27. Stephens, B.; Cox, A.; Singla, A.; Carter, J.; Dixon, C.; Felter, W. Practical DCB for improved data center networks. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014.
28. Haleplidis, E.; Pentikousis, K.; Denazis, S.; Salim, J.H.; Meyer, D.; Koufopavlou, O. Software-Defined Networking (SDN): Layers and Architecture Terminology. 2015. Available online: https://tools.ietf.org/html/rfc7426 (accessed on 5 January 2019).
29. The P4 Language Consortium. 2016. Available online: https://www.p4.org (accessed on 5 January 2019).
30. McKeown, N.; Izzard, M.; Mekkittikul, A.; Ellersick, W.; Horowitz, M. The tiny tera: A packet switch core. *IEEE Micro* **1997**, *17*, 26–33. [CrossRef]
31. Chuang, S.; Goel, A.; McKeown, N. Matching Output Queueing with a Combined Input/Output-Queued Switch. *IEEE JSAC* **1999**, *17*, 1030–1039.
32. Alizadeh, M.; Yang, S.; Sharif, M.; Katti, S.; McKeown, N.; Prabhakar, B.; Shenker, S. pFabric: Minimal Near-optimal Datacenter Transport. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013; pp. 435–446.
33. Kim, C.; Sivaraman, A.; Katta, N.; Bas, A.; Dixit, A.; Wobker, L.J.; Networks, B. In-band Network Telemetry via Programmable Data-planes. 2015. Available online: https://pdfs.semanticscholar.org/a3f1/9dc8520e2f42673be7cbd8d80cd96e3ec0c1.pdf (accessed on 5 January 2019).