Traffic Control for RDMA-enabled Data Center Networks: A Survey

Zehua Guo *Member, IEEE,*, Sen Liu, Zhi-Li Zhang *Fellow, IEEE*

***Abstract*—Data centers, the infrastructure of cloud computing, have been widely deployed around the world to accommodate to the increasing cloud computing demands. A Data Center Network (DCN) connects tens or hundreds of thousands of servers in the data center and uses a traffic control scheme to enable the data transmission among the servers. Various new applications in cloud present new requirements on traffic control of DCNs, such as low latency, high throughput. Existing traffic control schemes in DCNs suffer from the complicated kernel processing and cannot satisfy the requirements. Remote Direct Memory Access (RDMA), which bypasses the kernel processing to enable fast memory moving across a network, is recognized as a promising solution. In this paper, we present a survey of traffic control schemes for traditional RDMA, traditional DCNs, and RDMA-enabled DCNs and explain their limitations. We also differentiate the existing schemes from congestion control, performance, and components. In order to encourage future research, we point out some potential research directions of this research.**

***Index Terms*—Survey, RDMA, Data Center Networks, Traffic Control, Congestion Control**

I. Introduction

Traffic control is a critical research topic in Data Center Networks (DCNs). Many efforts have been conducted to provide low latency and high throughput traffic transmission for DCNs, such as DCTCP [1], TIMELY [2], pFabric [3], QJUMP [4]. However, the emerging new applications in cloud require much higher performance [5]. For example, a Machine Learning (ML) application (e.g., parameter server [6]) distributes its computation load into computation units for parallel processing, and the units demand low transmission latency to frequently transmit and synchronize a large amount of data with each other. Existing traffic control schemes cannot satisfy the new requirements, and new techniques are needed.

Remote Direct Memory Access (RDMA) is recognized as a promising solution to meet the requirements, and many data center operators have started to deploy it. RDMA is a memory access technique that enables Network Interface Cards (NICs) to transfer data directly to/from application memory. The NIC-based fast data transfer, which prevents data copies between application memory and data buffers by bypassing the kernel processing in the operating system, significantly reduces Central Processing Unit (CPU) overhead and overall latency, compared to the traditional traffic control schemes in DCNs. The low latency and minimal processing overhead provided by RDMA can greatly accelerates overall application performance of ML applications that frequently move massive amounts of data, exchange messages, and compute results. In [7], a deep neural network-based speech training shows that RDMA outperforms Transmission Control Protocol (TCP) by reducing the communication time of the total training time from 72% to 44%. Using a 40G NIC, TCP respectively consumes 6% and 12% CPU for the sender and receiver, while RDMA only consumes 2% for the sender and the receiver. Many state-of-the-art ML systems (e.g, TensorFlow [8]) have employed RDMA.

This paper was supported in part by National Science Foundation under Grants CNS-1618339, CNS-1617729, CNS-1814322, and CNS-1836772, the National Key Research and Development Program of China under Grant 2018YFB1003700, the National Natural Science Foundation of China under Grant 61836001, the Natural Science Foundation of Beijing under Grant Z170003, and the China Scholarship Council under Grant 201706370143.

Z. Guo is with the School of Automation, Beijing Institute of Technology, Beijing 100081, China, and was with the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN 55455 USA (e-mail: guolizihao@hotmail.com).

S. Liu is with the School of Information Science and Engineering, Central South University, Changsha 410083, China (e-mail: sen.liu@csu.edu.cn).

Z.-L. Zhang is with the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN 55455 USA (e-mail: zhzhang@cs.umn.edu).

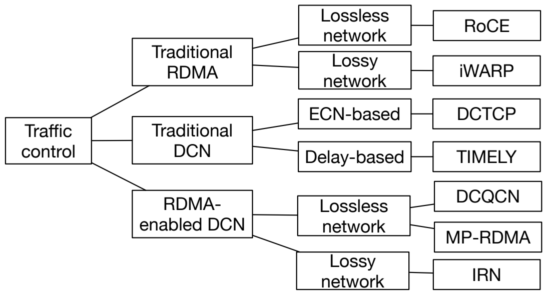


Fig. 1: Classification of traffic control schemes of traditional RDMA, traditional DCNs, and RDMA-enabled DCNs.

Many studies are working on deploying RDMA in DCNs. Some works present RDMA-based applications in DCNs, such as HERD [9], FaSST [10], and several studies efficiently utilize limited RDMA NIC cache, such as FaRM [11], LITE [12], INFINISWAP [13]. Actually, one of the fundamental problems is how to use RDMA to efficiently transmit traffic in DCNs. RDMA is originally designed for supercomputing networks (e.g., InfiniBand) rather than Ethernet or Internet Protocol (IP)-based DCNs. RDMA over Converged Ethernet (RoCE) and Internet Wide-area RDMA Protocol (iWARP) have been proposed to enable RDMA over Ethernet and IP networks. However, both of them cannot be directly used to DCNs. RoCEv2 depends on Priority Flow Control (PFC) to achieve the lossless Ethernet, but the PFC brings many undesirable results (Section III-A). Due to complicated NIC design, iWARP requires high on-die CPU and memory resource and results in low performance (Section III-B). Most existing traffic control schemes for DCNs are not compatible with RDMA because they rely on TCP/IP stack in the kernel layer of operating systems to control the traffic.

In this paper, we present a survey on traffic control schemes for RDMA. Fig. 1 categorizes the existing works. In Sec-

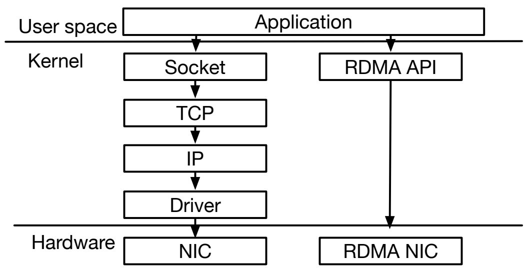


Fig. 2: Comparison of socket programming with RDMA programming.

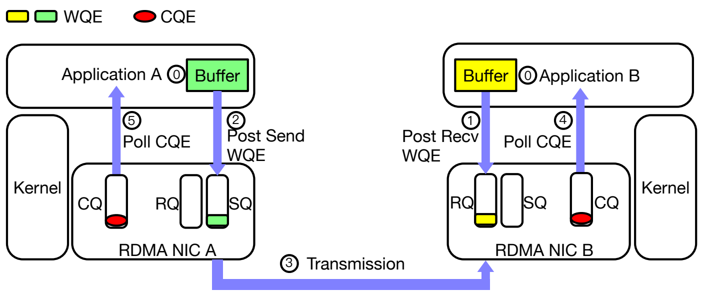


Fig. 3: An example of transmitting data using RDMA.

tion II, we briefly introduce RDMA and its operations. In Sections III and IV, we introduce traffic control schemes on traditional RDMA and traditional DCNs and explain why the existing schemes cannot be directly used. In Section V, we introduce state-of-the-art traffic control schemes of RDMA-enabled DCNs and analyze their limitations. Section VI differentiates existing schemes from congestion control, performance, and components. We propose potential research directions in Section VII and summarize our paper in Section VIII.

II. RDMA

A. Socket vs RDMA

Socket programming is usually used for transmitting data in a TCP/IP network. In the socket programming, an application from the user-space needs to initialize a socket in the kernel, which sends data through Transmission Control Protocol (TCP), Internet Protocol (IP), and NIC driver before accessing to the NIC. The data transmission between layers of the kernel requires CPU processing and memory copy. Different from the socket programming, the RDMA programming bypasses the multiple processing in the kernel and can directly use RDMA Application Programming Interface (API) to send data to the RDMA NIC from the application, reducing CPU overhead and overall latency for traffic transmission. Fig. 2 shows the processing difference between the socket programming and the RDMA programming.

B. RDMA Composition

RDMA transmission works in NICs with three queues: Send Queue (SQ), Receive Queue (RQ), and Completion Queue (CQ). When an RDMA application starts to work, it must create its SQ, RQ, and CQ in the RDMA NIC, and registers regions in memory for its processing. The work scheduling unit of RDMA is a Queue Pair (QP), which consists of one SQ and one RQ. Traffic of the same QP follows the same path, while traffic on different QPs traverses different paths. In the QP, a Work Queue Element (WQE) is placed as an instruction pointing to the memory buffer where the data will be placed or transmitted, and the RDMA NIC executes the WQE without involving the kernel. A Completion Queue (CQ) is used to notify the application when the transmission is done. Every time a WQE completes, a Completion Queue Element (CQE) is generated and placed into the CQ. RDMA supports two kinds of transmission semantics for WQEs: channel semantic including SEND and RECV verbs[1], and memory semantic including READ and WRITE verbs.

C. A Transmission Example

Fig. 3 shows an example of using verb SEND in RDMA to transmit data from application A to application B. The process in this figure consists of six steps. ⓪ RDMA NICs A and B create their QP's Completion Queues and register memory regions. NIC A registers a buffer for the data of application A to be moved to application B, and NIC B allocates an empty buffer for application B to receive the data from application A. ① Application B creates a WQE and posts it in the RQ. The WQE contains a pointer to an empty memory buffer allocated for the data to be placed. ② Application A creates a WQE and posts it in the SQ. The WQE points to the memory buffer that application A will move to application B. ③ RDMA NIC A consumes the WQE in the SQ of application A, and the data from the memory region of application A begins to transmit to application B over a network. When data arrives at RDMA NIC B, the NIC consumes the WQE in the RQ of application B to learn the memory location to place the data, and the data bypasses the kernel process and is directly placed in the assigned memory location. ④ When the data transmission completes, a CQE is created in the application B's CQ. The application B polls the CQE from its CQ and identifies that the transmission completes. ⑤ Similarly, when the data transmission completes, a CQE is created in the application A's CQ, and the application A polls the CQE from its CQ and identifies that the transmission completes.

III. Traffic Control in Traditional RDMA

RDMA was previously used in the lossless Infiniband networks. To use RDMA in Ethernet and IP networks, two protocols are introduced: RoCE and iWARP. RoCE follows the original design of RDMA for the lossless network and uses PFC to achieve lossless network on Ethernet and IP networks, while iWARP enables RDMA in the lossy networks by fully deploying TCP/IP stack in the NIC. In this section, we briefly introduce the two protocols and their limitations.

A. RoCE

RoCE v2[2] is a network protocol that makes RDMA compatible with existing networking infrastructure. With RoCE, an

[1]In RDMA, APIs to establish data channels are called verbs.

[2]In the paper, we use RoCE to represent RoCE v2.

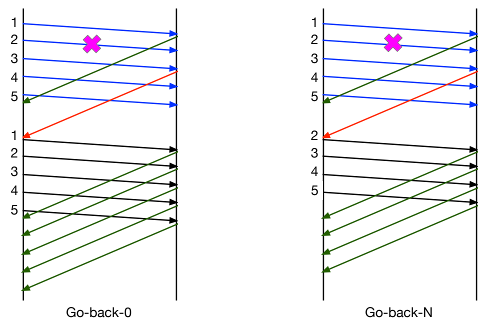


Fig. 4: An example of comparing go-back retransmission schemes in RoCE.

RDMA packet can be transformed into an Ethernet/IP/User Datagram Protocol (UDP) packet. The UDP header is used for Equal-Cost Multi-Path (ECMP) routing. The destination UDP port is always set to 4791, while the source UDP port is randomly chosen. RoCE handles two scenarios for realizing the lossless network transmission: preventing out-of-order packets with go-back retransmission and preventing packet loss with PFC.

1) Go-back Retransmission: In Infiniband networks, packet drops are rare but could happen due to some unexpected situations, such as software or hardware bugs. The original design of RDMA employs a *go-back-0* retransmission to handle this scenario. Assume a sender sends the data consisting of multiple packets to a receiver. When the receiver's NIC receives an out-of-order packet of the data, it discards this packet and requests the sender to retransmit all packets of the data. This go-back-0 scheme suffers from the live-lock problem: the links of switches are fully utilized, but the application's throughput could be very low [14]. This is because the application cannot use the data for processing until it receives all packets of the data. However, with the go-back-0 retransmission, one packet loss requires the retransmission of all packets of the data. To solve the problem, a modified go-back scheme named go-back-N retransmission is introduced. With the go-back-N scheme, when the receiver's NIC receives an out-of-order packet of the data, it discards this packet and asks the sender to retransmit all packets that are sent after the last acknowledged packet. Fig. 4 shows compares go-back-0 and go-back-N when packet 2 is an out-of-order packet.

Limitation: Go-back-N retransmission solves the live-lock problem but still wastes the time and bandwidth for sending redundant packets, potentially increasing the probability of congestion.

2) PFC: The PFC is a hop-by-hop flow control mechanism to prevent buffer overflow on Ethernet switches and NICs. It works in the queue granularity and sends PAUSE/RESUME frames from downstream devices to notify upstream devices to pause/resume sending packets. A downstream device receives packets from its upstream device and monitors its ingress queues. When one ingress queue of the downstream device reaches the PFC threshold, it sends a PAUSE frame to its corresponding upstream device's egress queue. Upon receiving the PAUSE frame, the upstream device stops sending packets from the corresponding egress queue. Once the ingress queue's length falls below the PFC threshold, the downstream device sends a RESUME frame to the upstream device to resume the transmission.

Limitation: Because of a coarse-grained queue level operation, the PAUSE frame could unfairly impact many flows on the same queue, including flows that are not relevant to congestion, and lead to poor performance for individual flows. Two typical results are unfairness flow transmission and head-of-line blocking. Fig. 5 shows an example of unfairness flow transmission resulted from PFC. In this figure, one switch has incoming flows from ingress queues 1 and 2 to egress queue 1. Ingress 1 is used by flow f1, and ingress 2 is used by flows f2, f3. Initially, the three flows equally share egress 1, as shown in Fig. 5(a). In Fig. 5(b), when egress 1 starts building up and reaches the PFC threshold, the switch pauses its ingress 1 and ingress 2. In Fig. 5(c), when the congestion is eliminated, ingress 1 and ingress 2 receive RESUME frames and resume to their transmission. In ingress 2, f2 and f3 compete each other, while ingress 1 only has f1. After a while, in egress 1, flow f1 transmits faster than flows f2, f3 and finally has the higher throughput than each flow on egress 2, as shown in Fig. 5(d).

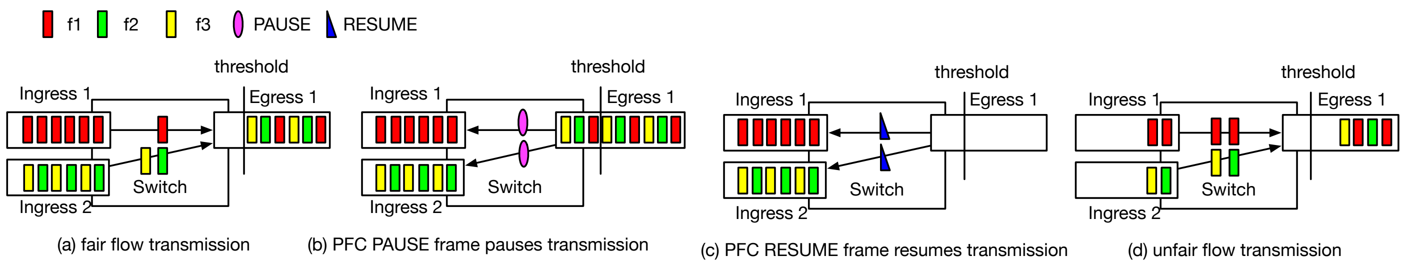
Fig. 6 shows an example of head-of-line blocking also resulted from PFC. In this figure, flows f1 and f2 share ingress queue 1 of the switch, but they are toward two different egress queues: f1 goes to egress 1, and f2 goes to egress 2. When egress 1 receives a PAUSE frame from its downstream device, it further sends a PAUSE frame to ingress 1 to pause packets of flow f1. Due to the coarse-grained queue level operation, flow f2, which is toward egress 2, is also affected by the PAUSE frame from egress 1.

Unexpected interaction between PFC and Ethernet packet flooding can break the up-down routing and could lead to occasional deadlocks [14][15]. Some existing works try to solve the above problems, but most solutions require to modify NIC. The hardware modification could cost months and years to complete and delay the wide deployment of RDMA.

B. iWARP

iWARP [16] implements RDMA over TCP/IP networks by putting full TCP/IP stack on the NIC. By offloading transport processing from the server's CPU to NIC, iWARP eliminates CPU overhead attributed to networking and removes system memory bandwidth reservation for intermediate TCP/IP stack buffer copies. One obvious advantage of iWARP is that it can directly use existing standard Ethernet-based network equipment and the route across IP networks with existing network infrastructure.

Limitation: iWARP is compatible with existing network structure at the cost of multiple protocol layers of translation between RDMA abstractions and traditional TCP bytestream abstractions. Fig. 7 compares the NIC implementation of iWARP and RoCE [17]. In this figure, iWARP NIC requires three layer translations, including Marker PDU Aligned framing (MPA, PDU is short for Protocol Data Unit), Direct Data Placement (DDP), and a separate RDMA Protocol (RDMAP). Compared to RoCE NICs, the complicated design of iWARP



f1
f2
f3
PAUSE
RESUME
threshold
Ingress 1
Egress 1
Switch
Ingress 2
(a) fair flow transmission
(b) PFC PAUSE frame pauses transmission
(c) PFC RESUME frame resumes transmission
(d) unfair flow transmission

Fig. 5: An example of unfairness flow transmission using PFC.

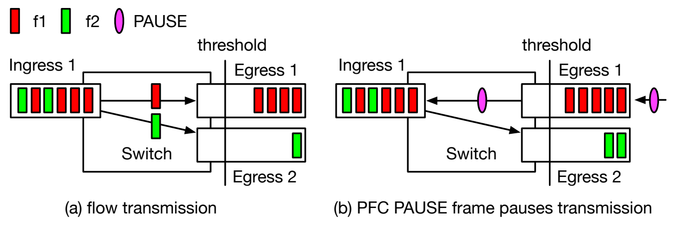


Fig. 6: An example of head-of-line blocking using PFC.

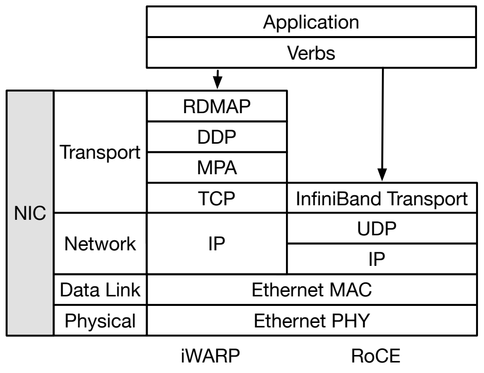


Fig. 7: Comparison of iWARP NIC and RoCE NIC.

NICs requires higher processing cost on NIC and achieves lower performance [18].

IV. Traffic Control in Traditional DCNs

Many traffic control schemes are designed for traditional DCNs. We categorize the existing schemes into two classes based on their different congestion signaling: Explicit Congestion Notification (ECN)-based and Delay-based. In this section, we present the requirements of traffic control in DCNs, introduce one representative scheme for the two classes, and point out their limitations on RDMA.

A. Traffic Control Requirements in DCNs

Data centers operate applications with diverse workload. The recent study categorizes the workload into two classes: delay-sensitive flows and throughput-intensive flows [1]. The delay-sensitive flows (e.g., web search, distributed memory caches, large-scale machine learning) are usually small flows and require low latency and high burst tolerance. Throughput-intensive flows (e.g., application upgrade) are typically large flows with high throughput demand.

The diverse workload proposes requirements for traffic control in DCNs:

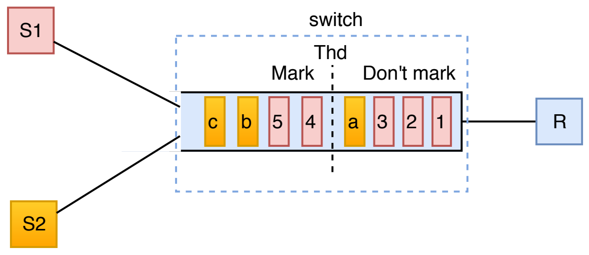
1) Fine-grained traffic control: because of the different goals, delay-sensitive flows and throughput-intensive flows should be treated differently, especially when they experience congestion.
2) Fast congestion detection: a congestion should be quickly identified and notified in order to timely adjust traffic sending rate.
3) Quick start: ML applications (e.g., parameters server) usually generate short flows (e.g., a few hundreds of bytes or less [19]). In order to achieve lower latency, the sender should start transmission at full speed (i.e., quick start) rather than probing the available bandwidth from a small transmission rate (e.g., 2 packets at slow start phase in TCP).
4) Low CPU consumption: CPUs on a server are involved in traffic transmission. Reducing the resource consumption for a traffic transmission enables more traffic transmission on a server.

B. Datacenter TCP (DCTCP)

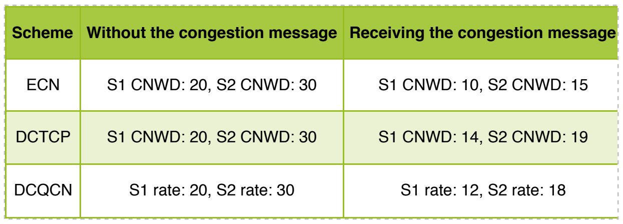
ECN is widely deployed in DCNs [1], [20], [21], [22], [23], [24], because ECN can notify senders that the queue in buffer is building up, while senders in other traditional schemes have to wait for packet loss. Thanks to ECN, senders can reduce their transmission in advance, rather than overflowing the buffer, which leads to packet loss, retransmission, greater transmission latency, and worse network performance.

DCTCP [1] is the first ECN-based traffic control scheme for DCNs. In DCTCP, each flow achieves not only quick congestion notification by ECN but also fine-grained per-flow congestion control, which adjusts each flow's congestion window according to its notified congestion severity. DCTCP works as follows: when a switch's queue occupancy is greater than the ECN marking threshold, the switch will mark subsequent packets with Congestion Encountered (CE) codepoint to record the congestion on the packets. Upon receiving an IP packet with the Congestion Experienced (CE) codepoint, the receiver sets the ECN-Echo (ECE) bit in the corresponding Acknowledgement (ACK) of this packet.

In conventional ECN, the receiver uses a single or multiple ECNs to notify the occurrence of a congestion and simply half the congestion window when a congestion happens. In DCTCP, however, the receiver precisely notifies the sender the number of packets that are experienced congestion by ECN

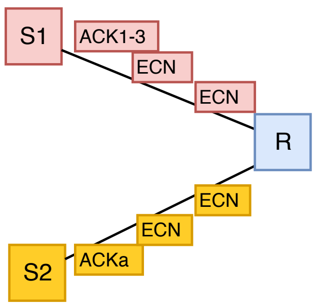


(a) S1 sends pink flow (packets 1-5) to R, and S2 sends yellow flow (packets a-c) to R. The switch's queue length reaches ECN threshold Thd, and the switch marks packets 4, 5, c, and d to inform R the congestion.

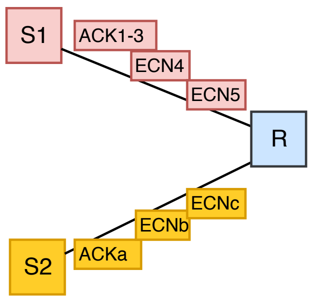


Scheme	Without the congestion message	Receiving the congestion message
ECN	S1 CNWD: 20, S2 CNWD: 30	S1 CNWD: 10, S2 CNWD: 15
DCTCP	S1 CNWD: 20, S2 CNWD: 30	S1 CNWD: 14, S2 CNWD: 19
DCQCN	S1 rate: 20, S2 rate: 30	S1 rate: 12, S2 rate: 18

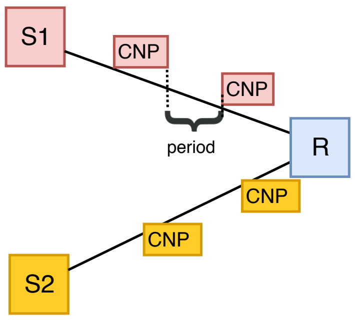
(b) Congestion control of ECN-based schemes when S1 and S2 receive an ECN, respectively. For ECN, S1 and S2 cut the congestion window (CNWD) of each flow by half. For DCTCP, the bitmaps showing the congestion impact are 00011 for the pink flow and 011 for the yellow flow. The two flows' estimate congestion severity are 2/5=40% and 2/3=66%. Under $g = 0.5$, $\alpha = 0.8$, their CNWDs are reduced to 14 and 19. For DCQCN, under $g = 0.5$, $\alpha = 0.8$, the sending rates of the two flows decrease to 12 and 18.



(c) ECN congestion notification: S1 receives one ACK for packets 1-3 and two ECN-marked ACKs, and S2 receives one ACK for packet a and two ECN-marked ACKs. The duplicated ECNs only show the occurrence of a congestion.



(d) DCTCP congestion notification: S1 receives one ACK for packets 1-3 and two ECN-marked ACKs for packets 4 and 5, and S2 receives one ACK for packet a and two ECN-marked ACKs for packets b and c. ACKs are used to generate two bitmaps showing the congestion impact on the two flows.



(e) DCQCN congestion notification: R periodically sends S1 and S2 a Congestion Notification Packet (CNP) until R receives a congestion notification echoing from S1 and S2. CNP is defined by RoCE to notify the occurrence of a congestion.

Fig. 8: Comparison of ECN, DCTCP, and DCQCN.

marks. Every Round-Trip Time (RTT), the sender calculates the congestion severity, which is inferred by the percentage of ECN-marked packets in all packets sent in this RTT, and updates the corresponding congestion window according to this congestion severity. Fig. 8 shows an example that differentiates ECN and DCTCP in terms of congestion notification and congestion window adjustment. Compared with ECN, DCTCP adjusts each flow's transmission rate dynamically according to the severity of congestion.

Moreover, in order to achieve better network performance, other DCTCP-like schemes, e.g., D^2TCP [22], L^2DCT [20], and TaTCP [24], are proposed. Unlike DCTCP which fairly shares the network bandwidth among flows, these new schemes argue that the user experiences could be improved through differentiating flows according to their own priorities (e.g., short flows first in L^2DCT, imminent deadline first in D^2TCP). To this end, these DCTCP variants leverage not only the congestion severity α but also their own priority factors to adjust the transmission rate of flows and control the network traffic.

Limitation: DCTCP needs switches to support ECN and does not fit RDMA because it relies on the TCP/IP stack to realize traffic control. Additionally, it has a slow start phase and performs poorly under bursty storage workloads [25]. Besides, its variants also suffer from the same issues, since all of them rely on the congestion severity α.

C. *Delay-based Traffic Control*

Many works use ECN to identify congestion since DCTCP is proposed. However, ECN can only identify a congestion that when the number of packets in a queue exceeds a ECN threshold at a single switch. It does not provide the accuracy of end-to-end congestion severity if multiple switches are congested at the same time. Besides, ECN also cannot detect congestion at NICs, which do not support ECN. In the case of multiple priority queue, a low priority flow could also experience a large queueing delay before triggering ECN. To solve the problems, many works argue that RTT is also an important measurement, or performs even better, for network congestion detection [2], [26], [27], [28], since it contains end-to-end information rather than a single switch's. Therefore, some delay-based congestion control schemes for DCNs are proposed. One representative of them is TIMELY [2].

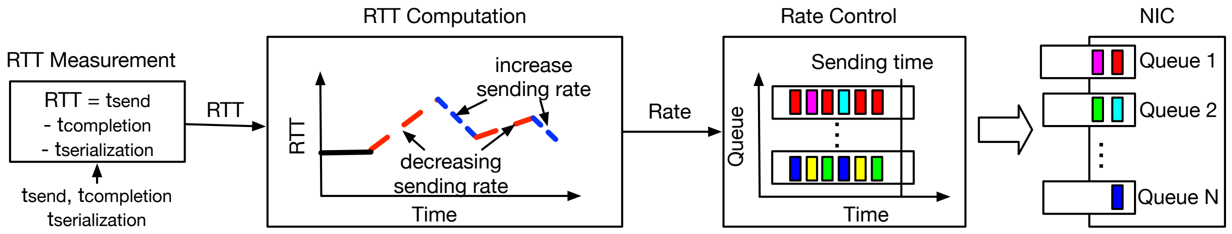


Fig. 9: The working procedure of TIMELY.

TIMELY uses RTT to reflect end-to-end latency, and existing NICs, which can provide accurate measurement of the time of packet transmission and reception, makes TIMELY feasible. Fig. 9 shows the processing procedure of TIMELY. TIMELY works at the sender and consists of three components: RTT Measurement, Rate Computation, and Rate Control. In the RTT Measurement, an RTT for each segment of data is calculated. An RTT equals $t_{completion} - t_{send} - t_{serialization}$, where $t_{completion}$ is the time the ACK is received as the segment of data transmission completes, t_{send} is the time when the first packet is sent, and $t_{serialization}$ is the serialization delay to transmit all packets in the segment. In the Rate Computation, the delay gradient (i.e., derivative of the queuing with respect to time) is used to decide the sending rate. An increasing RTT leads to a positive delay gradient, which indicates an increasing queue, while a negative gradient indicates a decreasing queue. If the gradient is less than or equals zero, the network can have a higher rate, and the sending rate is increased with a constant value to probe for more bandwidth. If the gradient is positive, the sending rate exceeds the network capacity, and a multiplicative rate decrement is performed. In the Rate Control, a scheduler is used to handle flows. It computes the sending time for the current segment based on the segment size, flow rate, and time of the last transmission, and then places the segment in a priority queue. When reaching on sending times, the segments are passed to the NIC in the round robin fashion for immediate transmission.

Moreover, DX [27] is another typical delay-based congestion control scheme in data centers. Thanks to the accurate queueing delay measurement in either software-based or hardware-based way, DX adjusts the transmission rate in a fine-grained manner, successfully achieving both high utilization of the link and low queueing delay. Specifically, DX increases the congestion window by one, if the average queueing delay is zero during the current window. Otherwise, the congestion window reduces by $1 - Q/V$, where Q is the measured average queueing delay. And V is a self-updated coefficient calculated by $R \cdot W^*/(W^* - 1)$. Here, R denotes the base RTT, while W^* is the previous congestion window size. Note that, in order to adjust transmission rate quickly and correctly, DX requires accurate RTT measurement through either software (i.e., Linux kernel modification) or hardware (i.e., DPDK-based NIC) solution.

Limitation: TIMELY tries to become a generic solution to control traffic in DCNs but has some limitations. First, TIMELY is implemented in each host, and such a distributed protocol, which uses only delay as the feedback signal, cannot simultaneously achieve fairness or a guaranteed steady-state delay [29]. Second, some jitters could introduce delay and noise in the feedback signal and affect the accuracy of congestion detection. Deploying TIMELY in RDMA scenario also needs PFC to prevent the performance degradation from packet drops. Though DX can work without PFC, it faces the same issue like TIMELY. That is modification of end-hosts' operating system or hardwares, leading to complex deployment.

V. Traffic Control in RDMA-enabled DCNs

The previous two sections explain the limitation of the existing traffic control schemes of RDMA and DCNs, and why they cannot be directly used. Deploying RDMA in DCNs also has some specific requirements. In this section, we present the requirements and introduce three state-of-the-art traffic control schemes for RDMA-enabled DCNs.

A. Traffic Control Requirements in RDMA-enabled DCNs

Because of low cost and high performance, RoCE dominates the RDMA market, and most recent works focus on RoCE-based solutions for RDMA-enabled DCNs. Besides the four requirements of DCNs explained in Section IV-A, efficient RDMA deployment in DCNs has the following requirements.

1) Low NIC resource consumption: the data transfer of RDMA is completely implemented by NIC, which has limited computing resource and on-chip memory. Reducing the NIC consumption for an RDMA transmission can enable more transmission for different applications.
2) Easy configuration and implementation: new schemes should be easily configured and deployed to existing RoCE NICs.
3) High performance: the PFC and go-back-N retransmission work together to enable lossless networks for RDMA, but they also bring some undesirable results and demands high overhead. An efficient solution should not depend on the two schemes.

B. DCQCN

DCQCN [25] is the first congestion control solution for utilizing RDMA in DCNs. DCQCN takes advantages of quick congestion notification from ECN, fine-grained rate control from DCTCP, and fast sending rate increase from Quantized Congestion Notification (QCN) [30]. DCQCN processing relies on the switch, the sender NIC and the receiver NIC. When a switch's egress queue's occupancy exceeds a given ECN

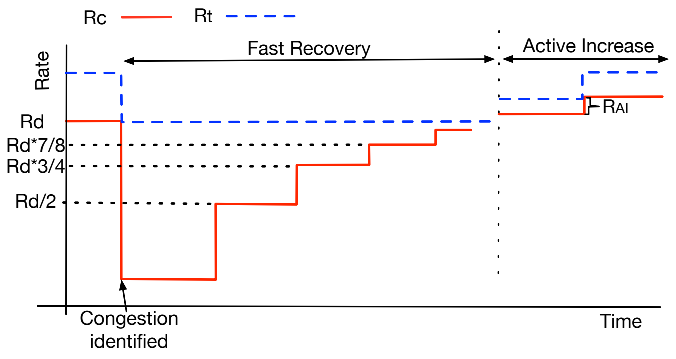


Fig. 10: Rate increase in QCN.

threshold, the switch marks arriving packets in the queue with CE codepoint to notify the receiver of a congestion. If the marked packet arrives at the receiver NIC, the NIC periodically sends the sender a Congestion Notification Packet (CNP) defined by RoCE until the NIC receives a congestion notification echoed from the sender. Fig. 8(e) shows an example that compares DCQCN with ECN and DCTCP in term of using ECN and congestion control.

A sender's NIC adjusts the sending rate of flows using two rates: R_c, the current sending rate, and R_t, the target rate that records the sending rate before the last congestion feedback message arrives at the sender NIC. The rate control consists of the sending rate decrease and the sending rate increase. Rate decrease is very simple. Upon getting a CNP, the sender updates R_c to R_t for recovery and reduces R_c similar to the change of the congestion window in DCTCP. However, the rate increase is complicated. Unlike TCP, which increases rate when receiving ACKs, Ethernet does not have positive rate-increasing signals. DCQCN employs the similar rate increasing method in QCN [30], where a sender uses two rate increasing patterns at two phases. In the Fast Recovery phase, the sending rate slowly gets back to the loss rate. R_t does not change, and R_c always increases to $(R_c + R_t)/2$. In the Active Increase phase, the sending rate grows fast through quick bandwidth probes. R_c still increases to $(R_c + R_t)/2$, but R_t increases to $R_t + R_{AI}$, where R_{AI} is a constant value. Fig. 10 shows the trend of the rate increase of QCN. In this figure, the rate is R_d when the congestion is identified. During the Fast Recovery phase, $R_t = R_d$, and R_c increases $R_d/2^n$ (e.g., $R_d/2$, $R_d/4$, $R_d/8$) at the end of interval n until R_c recovers to R_d. During the Active Increase phase, both R_t and R_c increase, and the increasing rate is proportional to the time.

Limitation: DCQCN provides per-flow congestion control and alleviates PFCs limitations. However, DCQCN still needs PFC to prevent packet loss and poor performance when flows begin transmitting at line rate. If we want to use ECN to send congestion feedback and reduce the probability of triggering PFC, we need to carefully set parameters for switches, such as the buffer threshold for storing PFC pause packet, the PFC threshold in ingress queue, and ECN threshold in egress queue. The parameter setting depends on switches case by case, and there is not a generic way for configuring switches.

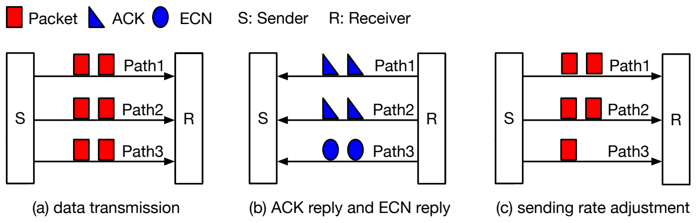


Fig. 11: Congestion control in MP-RDMA.

C. MultiPath-RDMA

In DCNs, transmitting traffic via rich and parallel paths is an efficient method to improve transmission efficiency. For example, Multipath TCP (MPTCP) allows a TCP connection to use multiple paths to maximize resource utilization and increase redundancy. Efficient multiple path transmission should dynamically distribute traffic based on the congestion state of each path and maintain high throughput by preventing out-of-order packets for flows. However, the RoCE NIC does not have enough memory to track and store the congestion state of each path and packet order of each flow. The NIC can upload path state and flow state to host memory, but frequent swapping data between on-chip memory and host memory increase Peripheral Component Interconnect express (PCIe) bus latency and the contention on the bandwidth, eventually degrading NIC throughput.

MultiPath-RDMA (MP-RDMA) [31] first proposes to use resource-limited RoCE NIC and Field-Programmable Gate Array (FPGA) to enable multi-path routing in RDMA-enabled DCNs. MP-RDMA uses one congestion window for all paths instead of maintaining per-path states and performs congestion control with an ECN-based multipath ACK-clocking. MP-RDMA works at switch, sender's NIC, and receiver's NIC. RoCE uses the UDP header for ECMP routing. In RoCE, the destination UDP port is always set to 4791, while the source UDP port can be selectively chosen. Thus, a UDP source port can be viewed as a virtual path, and a switch uses ECMP to pick up the path for a flow based on the UDP source port in the packet header. ECMP guarantees that packets with the same UDP source port are always mapped to the same path. Upon receiving a packet, the receiver's NIC with FPGA instantly generates a MP-RDMA ACK and sends it to the sender. In the ACK, its UDP header encodes the same virtual path ID of the received packet, and its RoCE header encodes the packet sequence number of the received packet, the cumulative sequence number at the receiver, and the ECN signal.

The sender's NIC mainly achieves congestion control with a congestion window-based traffic adjustment. Initially, the sender's NIC randomly spreads an initial window of N packets to N virtual paths by selecting a specific source port in the UDP header for each packet. When a MP-RDMA ACK arrives from the virtual path vp, the sender's NIC calculates its congestion window $cwnd$. If the ACK is marked with ECE bit, the sender's NIC decreases its $cwnd$ by 1/2 segment; otherwise, the NIC increases its $cwnd$ by $1/cwnd$ segment. If the $cwnd$ has space M, M packets will be forwarded on virtual path vp. Fig. 11 shows an example of congestion control in MP-RDMA. In this figure, paths 1, 2, and 3 are

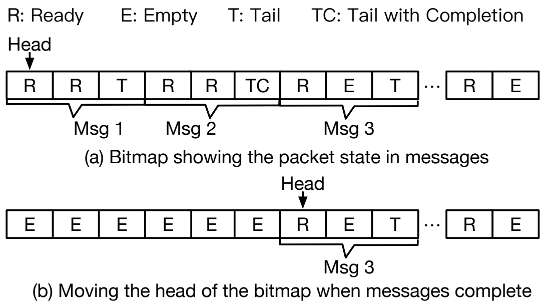

R: Ready
E: Empty
T: Tail
TC: Tail with Completion
Head
R R T R R TC R E T ··· R E
Msg 1
Msg 2
Msg 3
(a) Bitmap showing the packet state in messages
Head
E E E E E E R E T ··· R E
Msg 3
(b) Moving the head of the bitmap when messages complete

Fig. 12: MP-RDMA's bitmap in the NIC to track path state.

between sender S and receiver R. The $cnwd$ is six, and two packets are forwarded on each path. Paths 1 and 2 have the same loaded, and the receiver replies two ACKs for the received packets on the two paths. The congestion occurs on path 3. Two ECN-marked ACKs are replied to the sender. Following the above $cwnd$ adjustment principle, the sender decreases the $cwnd$ by one and sends only one packet on path 3.

The receiver's NIC prevents out-of-order packets with a bitmap-based path selection. The NIC tracks arrived packets using a 2-bit bitmap with four states: Empty, Received, Tail, and Tail with Completion, and encapsulates the state in each packet. When the NIC receives a head-of-line message with a continuous block of slots and the last slot is Tail or Tail with Completion state, it identifies that the message is completely transmitted. Then, it clears these slots to Empty state and moves the head point after the last block of the completed message. A Tail with Completion state requires an extra completion notification. Fig. 12 shows an example on monitoring path state using the bitmap. The tracking bitmap only has a limited number of slots used to monitor the path state. To efficiently use the limited slots, the NIC adaptively removes slow paths and only forward packets on the selected fast paths with a similar delay. Fig. 13 shows an example on dynamically selecting forwarding paths. In this figure, paths 1, 2, and 3 are between sender S and receiver R, and two packets are forwarded on each path. Path 3 is relatively congested, compared to paths 1 and 2, and the ACKs of path 3 arrive later than paths 1 and 2. Thus, path 3 is identified as a slow path. To make sure packets arrive in order, no packets will be forwarded on path 3.

Limitation: The implementation of MP-RDMA is based on FPGA and relies on PFC. Thus, the real deployment of MP-RDMA increases hardware cost and cannot be easily employed by data centers.

D. Improved RoCE NIC (IRN)

PFC and go-back-N transmission have obvious bad effects, but DCQCN and MP-RDMA require them to guarantee lossless network transmission. To remove RDMA's dependency on PFC, one recent work proposes IRN for deploying RDMA in the lossy network [32]. IRN follows the idea of iWARP, which handles packet loss of the TCP stack in hardware, but in a very simple method using selective retransmission and packet-level traffic control.

Selective retransmission improves packet loss recovery efficiency by only retransmitting lost packets. The IRN maintains a bitmap to track different types of packets: received packet are cumulatively acknowledged through ACKs, while lost packets are selectively acknowledged through Negative Acknowledgement (NACK). When receiving every out-of-order packet, the receiver's NIC sends a NACK, which carries both the cumulative acknowledgment that indicates its expected packet sequence number and the packet sequence number for triggering the NACK. When receiving a NACK, the sender's NIC stops sending new packets and starts to selectively retransmit lost packets indicated by the bitmap. The number of retransmitted packets is from the cumulative acknowledgment value to the sequence number of the packet triggered the NACK. If the sequence of the cumulative acknowledgment is greater than the sequence of the last regular packet that is sent before the retransmission of a lost packet, the loss recovery ends and the sender's NIC continues to transmit new packets. Fig. 14 shows an example that compares go-back-N retransmission with selective retransmission when packet 2 is out of order.

Packet-level traffic control reduces unnecessary queuing in the network by strictly setting an upper bound for the number of out-of-order packets. The sender's NIC calculates a Bandwidth-Delay Product (BDP) of the network and uses the BDP to bound the number of sending packets in flight. A new packet is sent by the sender only if the number of packets in flight is less than the BDP cap. This traffic control reduces a NIC's workload to maintain the state information for tracking packet losses.

Limitation: Since the design of IRN focuses on lossy networks, which is not originally supported by RoCE, it has to add some new features to the NIC. Selective retransmission and packet-level traffic control rely on per-packet ACKs. RoCE NICs support four data transfer messages, i.e., SEND, RECV, READ and WRITE but only supports per-packet ACKs for two messages. Thus, other two types of ACKs should be added. Many packets in a data message could be out of order, such as first packet, last packet, WQE matching packet. To prevent out-of-order packet delivery, IRN should handle all the scenarios.

VI. Comparison of Different Schemes

The comparison of different schemes are listed in Tables I, II, and III. In this section, we differentiate the existing schemes by explaining items in the three tables one by one.

A. Congestion Control

Tables I and II differentiate the schemes' congestion control from three aspects, i.e., detection, notification, and adjustment of the congestion. The details are explained below:

1) Congestion Detection and Notification: In Table I, the seven schemes use four congestion detection methods. RoCE implements PFC to notify the sender of the congestion when the queue length of switches or NICs exceeds the PFC threshold. iWARP and IRN use packet loss to detect congestion. iWARP fully deploys TCP in its NIC and relies on ACKs to

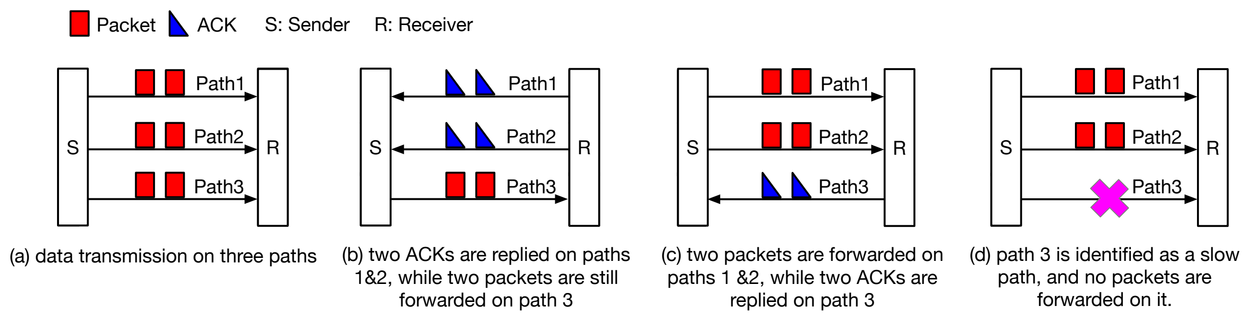

Packet
ACK
S: Sender
R: Receiver
S
R
Path1
Path2
Path3
(a) data transmission on three paths
(b) two ACKs are replied on paths 1&2, while two packets are still forwarded on path 3
(c) two packets are forwarded on paths 1 &2, while two ACKs are replied on path 3
(d) path 3 is identified as a slow path, and no packets are forwarded on it.

Fig. 13: Prevention of out-of-order packets in MP-RDMA.

TABLE I: Comparison of seven traffic control schemes' congestion detection and notification

	Congestion detection			Congestion notification	
Scheme	**Category**	**Method**	**Location**	**Signaling**	**Location**
RoCE	Traditional RDMA	PFC	NIC	Pause	Switch, NIC
iWARP		Packet loss	NIC	ACK	NIC
DCTCP	Traditional DCN	ECN, packet loss	Kernel	ECN-marked ACK	Kernel
TIMELY		RTT gradient, PFC	Application	PFC	Switch, NIC
DCQCN	RDMA-enabled DCN	ECN, PFC	NIC	CNP, PFC	Switch, NIC
MP-RDMA		ECN, PFC	FPGA	MP-RDMA ACK, PFC	NIC+FPGA, Switch, NIC
IRN		Packet loss	FPGA	NACK	NIC+FPGA

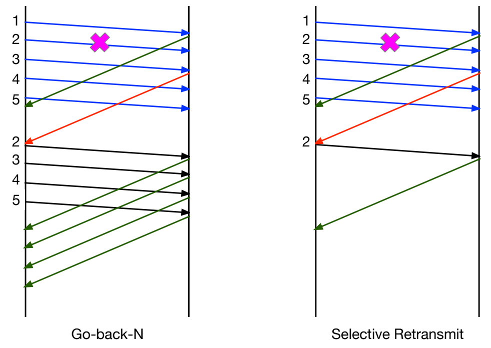


Fig. 14: Comparison of go-back-N and selective retransmission.

signal congestion from the receiver's NIC. IRN implements its customized congestion control on FPGA and uses NACKs to echo the sender congestions. Since DCTCP is a variant of TCP, packet loss also triggers DCTCP to react to the congestion.

DCQCN, MP-RDMA, and DCTCP employ ECN to detect congestion on switches but use different congestion signals. DCQCN uses CNP from NIC, and MP-RDMA uses its customized MP-RDMA ACK from FPGA. Besides packets loss, DCTCP also utilizes ECN-marked ACKs. Since DCQCN and MP-RDMA send traffic at line rate, they also use PFC to prevent heavy congestion and massive packet loss. Any scheme that uses PFC is involved with switch and NIC.

PFC has some bad effects and can degrade the network performance significantly. One efficient solution is to use ECN to reduce the dependency on PFC. ECN's threshold is usually smaller than PFC's threshold, and thus it is triggered before PFC. PFC only works when a busty traffic builds up queues of switches quickly but the sender has not reacted to ECN feedback yet [25]. We illustrate ECN and PFC's thresholds in Fig.15.

TABLE II: Comparison of seven traffic control schemes' congestion adjustment

	Congestion adjustment			
Scheme	**Adjustment unit**	**Sending rate increasing method**	**Sending rate decreasing method**	**Location**
RoCE	Sending rate	RESUME frame	PAUSE frame	NIC
iWARP	Congestion window	Congestion window+1	Congestion window/2	NIC
DCTCP	Congestion window	Congestion window+1	Congestion severity α	Kernel
TIMELY	Sending rate	additive increase, gradient-based increase	gradient-based decrease, multiplicative decrease	Application
DCQCN	Sending rate	QCN	Congestion severity α	NIC
MP-RDMA	Congestion window	Congestion window+ 1/(Congestion window)	Congestion window-1/2	FPGA
IRN	Sending rate	Bounded by bandwidth-delay product	Bounded by bandwidth-delay product	FPGA

TIMELY consists of NIC and application. The NIC calculates real time RTT, and the application identifies congestion based on variation of RTT gradient. TIMELY also uses PFC to prevent heavy congestion and massive packet loss. We classify the seven traffic control schemes based on congestion detection methods. Fig.16 shows the classification.

2) Congestion Adjustment: In Table II, the seven schemes use two adjustment units for flows: the flow's congestion window and sending rate. iWARP, DCTCP, and MP-RDMA use the congestion window. iWARP follows the standard TCP. DCTCP follows the standard TCP to increase a flow's sending rate but uses a congestion severity α to decrease a flow's rate. MP-RDMA implements its rate adjustment in FPGA.

TABLE III: Comparison of seven RDMA-based traffic control schemes' performance and requirement

	Performance		**Components**		
Scheme	**Traffic control granularity**	**Quick start**	**Implementation**	**Resource**	**PFC dependency**
RoCE	Per-queue	√	—	NIC	√
iWARP	Per-flow	×	—	NIC	×
DCTCP	Per-flow	×	ECN-enabled Switch, Kernel	Kernel	×
TIMELY	Per-flow	√	Application, NIC, Switch	Application, NIC	√
DCQCN	Per-flow	√	ECN-enabled Switch, NIC	NIC	√
MP-RDMA	Per-flow	√	ECN-enabled Switch, NIC+FPGA	NIC+FPGA	√
IRN	Per-flow	√	NIC+FPGA	NIC+FPGA	×

— : the scheme is already implemented in the product.

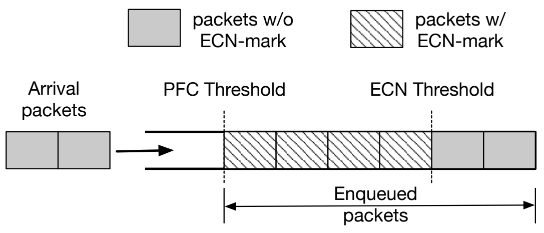


Fig. 15: Thresholds of PFC and ECN.

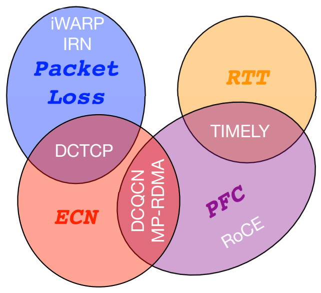


Fig. 16: Congestion detection classification.

The rest four schemes use the sending rate. With RoCE, a NIC uses RESUME and PAUSE frames to adjust a flow's sending rate. IRN uses FPGA to increase a flow's rate with a upper bound of bandwidth-delay product. DCQCN follows the standard QCN to increase a flow's sending rate but uses a congestion severity α to decrease a flow's rate. For increasing/decreasing a flow's rate,TIMELY uses two different methods for different RTT ranges.

B. Performance

Table III lists the performance of different schemes. In the table, RoCE relies to PFC and performs in a coarse-grained per-queue fashion, while other schemes work in a per-flow manner. As shown in Section III-A, RoCE suffers from PFC with unfairness flow transmission, head-of-line blocking, and deadlock.

ML applications need quick start to reduce communication latency significantly by starting the transmission of short flows at full speed without probing the available bandwidth from a small transmission rate. Therefore, iWARP and DCTCP, which follow the standard TCP legacy, suffer from the TCP's slow-start. By contrast, other schemes can send flows at the line rate at the beginning of transmission, resulting in a better performance.

C. Components

1) Implementation and Resource: In Table III, we list the implementation of all the seven schemes, which require four different types of hardware or software, i.e., switches or dedicated switches, kernel, application, and NIC or NIC with FPGA. Specifically, RoCE and iWARP are implemented in commodity NIC. DCTCP, DCQCN, and MP-RDMA requires ECN-enabled switches for congestion notification. TIMELY needs NIC support for precise RTT measurement. Both IRN and MP-RDMA use FPGA to adjust flows' sending rates. DCTCP and TIMELY respectively use kernel and application to adjust transmission rate while all other five schemes require NIC resources for rate adjustment.

2) PFC Dependency: Table III shows each scheme's dependency on PFC. Without loss of generality, all shemes expect iWARP and IRN need PFC to achieve lossless transmission. iWARP can recover from packet loss using its TCP/IP stack embedded in NIC, while IRN utilizes selective retransmission against packet loss. DCTCP does not use PFC since it handles the packet loss in the kernel like other TCPs.

VII. Open Research Issues

Based on our analysis, we expect the following trends for designing advanced traffic control schemes for RDMA-enabled DCNs.

1) Removing PFC: PFC can dramatically degrade performance and hinders the wide deployment of RDMA in industry. Some studies propose to remove PFC but require new hardware, e.g, FPGA. We need to find an efficient and practical method to reduce or remove the RDMA's dependency on PFC.
2) Practical design: some existing works control RDMA traffic in DCNs with new hardware (e.g., FPGA). These hardware-based solutions are not cost-efficient for DCNs with existing commercial RDMA NICs (e.g., RoCE). An efficient solution is to use existing RDMA NICs to achieve a similar performance of new hardware-based solutions, such as IRN and MP-RDMA. One possible solution is to deploy the traffic control in the user-space and has two benefits: (1) RDMA is only used for data transmission without any modification, and (2) we can flexibly use existing or customized congestion control schemes on demand.
3) Integrating RDMA NIC with Ethernet NIC: in DCNs, many servers could be equipped with both RDMA NIC with traditional Ethernet NIC and run different applications. RDMA NICs are typically used for ML

applications, while Ethernet NICs are suitable for other applications. If a server initializes VMs to run different applications, we need a solution to differentiate applications and dynamically assign the traffic of different applications to either RDMA NIC or Ethernet NIC.

4) Integrating RDMA NIC with QUIC: QUIC is a UDP-based transport layer network protocol that aims to be nearly equivalent to the TCP but with much-reduced latency [33]. We think the combination of RDMA with QUIC could be an interesting and promising solution since both RDMA and QUIC use UDP and bypass Linux kernel to enjoy the flexible user-space programming.

VIII. Conclusion

In this paper, we summarize the existing traffic control schemes for traditional RDMA, traditional DCNs, and RDMA-enabled DCNs, and discuss the pros and cons of the representative solutions of each type. In order to highlight the features and strengths of these existing schemes, we also compare them from three aspects: congestion control, performance, and required components. For encouraging the future study, we further list and discuss potential research directions and open issues.

References

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *ACM SIGCOMM*, 2010.

[2] R. Mittal, N. Dukkipati, and E. e. a. Blem, "Timely: Rtt-based congestion control for the datacenter," in *ACM SIGCOMM*, 2015.

[3] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *ACM SIGCOMM*, 2013.

[4] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. Watson, A. W. Moore, S. Hand, and J. Crowcroft, "Queues don't matter when you can jump them!" in *NSDI*, 2015, pp. 1–14.

[5] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Computer Networks*, vol. 91, pp. 528–547, 2015.

[6] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *OSDI*, 2014, pp. 583–598.

[7] C. Guo, "Rdma in data centers: Looking back and looking forward," in *ACM APNET*, 2017.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[9] A. Kalia, M. Kaminsky, and D. G. Andersen, "Using rdma efficiently for key-value services," *ACM SIGCOMM*, 2015.

[10] ——, "Fasst: Fast, scalable and simple distributed transactions with two-sided (rdma) datagram rpcs." in *OSDI*, vol. 16, 2016, pp. 185–201.

[11] A. Dragojević, D. Narayanan, O. Hodson, and M. Castro, "Farm: Fast remote memory," in *USENIX NSDI*, 2014, pp. 401–414.

[12] S.-Y. Tsai and Y. Zhang, "Lite kernel rdma support for datacenter applications," in *ACM SOSP*, 2017, pp. 306–324.

[13] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, "Efficient memory disaggregation with infiniswap." in *NSDI*, 2017, pp. 649–667.

[14] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in *ACM SIGCOMM*, 2016.

[15] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen, "Tagger: Practical pfc deadlock prevention in data center networks," in *ACM CoNEXT*, 2017, pp. 451–463.

[16] R. Recio, B. Metzler, P. Culley, J. Hilland, and D. Garcia, "A remote direct memory access protocol specification," Tech. Rep., 2007.

[17] P. MacArthur, Q. Liu, R. D. Russell, F. Mizero, M. Veeraraghavan, and J. M. Dennis, "An integrated tutorial on infiniband, verbs, and mpi," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2894–2926, 2017.

[18] "Roce vs. iwarp competitive analysis," http://www.mellanox.com/related-docs/whitepapers/WP_RoCE_vs_iWARP.pdf.

[19] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *ACM SIGCOMM*, 2018.

[20] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 2157–2165.

[21] T. Zhang, J. Wang, J. Huang, Y. Huang, J. Chen, and Y. Pan, "Adaptive-acceleration data center tcp," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1522–1533, 2015.

[22] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[23] E. Dong, X. Fu, M. Xu, and Y. Yang, "Dcmptcp: Host-based load balancing for datacenters," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 622–633.

[24] S. Liu, J. Huang, Y. Zhou, J. Wang, and T. He, "Task-aware tcp in data center networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 389–404, 2019.

[25] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," in *ACM SIGCOMM*, 2015.

[26] Y. Chatei, M. Hammouti, E. A. Reyouchi, and K. Ghoumid, "Downlink and uplink message size impact on round trip time metric in multi-hop wireless mesh networks," *Int J Adv Comput Sci Appl (IJACSA)*, vol. 8, no. 3, pp. 223–229, 2017.

[27] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 2015, pp. 403–415.

[28] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 253–266.

[29] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "Ecn or delay: Lessons learnt from analysis of dcqcn and timely," in *ACM CoNEXT*, 2016, pp. 313–327.

[30] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, "Data center transport mechanisms: Congestion control theory and ieee standardization," in *IEEE Annual Allerton Conference on Communication, Control, and Computing*, 2008, pp. 1270–1277.

[31] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and T. Moscibroda, "Multi-path transport for rdma in datacenters," in *USENIX NSDI*, 2018.

[32] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for rdma," *ACM SIGCOMM*, 2018.

[33] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.