A Receiver-Driven Transport Protocol with High Link Utilization Using Anti-ECN Marking in Data Center Networks

Jinbin Hu, Jiawei Huang, *Member, IEEE,* Zhaoyi Li, Jianxin Wang, *Senior Member, IEEE* and Tian He, *Fellow, IEEE, ACM*

***Abstract*—Existing reactive or proactive congestion control protocols are hard to simultaneously achieve ultra-low latency and high link utilization across all workloads ranging from delay-sensitive flows to bandwidth-hungry ones in datacenter networks. We present an Anti-ECN (Explicit Congestion Notification) Marking Receiver-driven Transport protocol called AMRT, which achieves both near-zero queueing delay and full link utilization by reasonably increasing sending rate in the case of under-utilization. Specifically, switches mark the ECN bit of data packets once detecting spare bandwidth. When receiving the anti-ECN marked packet, the receiver generates the corresponding marked grant to trigger more data packets. The testbed and simulation experiments show that AMRT effectively reduces the average flow completion time (AFCT) by up to 42% and improves the link utilization by up to 38% over the state-of-the-art receiver-driven transmission schemes.**

***Index Terms*—Data center, receiver-driven, link utilization.**

I. Introduction

MODERN data centers host diverse applications such as web search, social networking, deep learning and data mining. With the increasingly stringent demand on both low latency and high throughput in these datacenter applications, numerous of transport protocols are proposed to optimize the flow completion time by using reactive congestion control algorithms at sender side [2]–[7]. Since only reacting after congestion already happens, the sender-side transport protocols inevitably induce queue buildup, which adversely affects the performance of short or tiny flows in delay-sensitive applications such as remote procedure calls (RPCs) [8].

In recent years, receiver-driven transport protocols such as pHost [9], NDP [10], ExpressPass [11] and Homa [8] have been proposed to guarantee near-zero queueing delay by using proactive congestion control mechanism. These receiver-driven transport protocols conservatively trigger new data packets according to the data arrival rate at the receiver. Specifically, when a data packet arrives at the receiver, a corresponding grant packet is generated and returned to the sender to trigger only one new data packet called scheduled packet. Therefore, these receiver-driven transport protocols effectively achieve ultra-low queueing delay and significantly improve performance of delay-sensitive application.

However, the conservative receiver-driven transport protocols are not able to actively probe the available bandwidth and increase sending rate even in the case of spare bandwidth. As a result, the receiver-driven transport protocols potentially suffer from under-utilization problem, especially in the multi-bottleneck and dynamic traffic scenarios. (1) When a flow passes through multiple bottleneck links, its rate is limited by the most congested bottleneck. Due to the conservativeness of the receiver-driven transport protocols, when free bandwidth arises in the other bottleneck links, the coexisting receiver-driven flows are not able to grab the available bandwidth, leading to low link utilization. (2) Under the highly dynamic traffic in data center, the link bandwidth is potentially wasted in the receiver-driven transmission. When multiple flows to different receivers share the same bottleneck link, even though some flows finish transmission, the remaining ones are unable to fill up the available bandwidth, further reducing the link utilization.

Fortunately, the marking-based explicit feedback is an effective mechanism to address the link under-utilization problem. We introduce the explicit feedback into the receiver-driven transport protocol, called as AMRT, which uses anti-ECN [12] marked packets to notify the sender of link under-utilization and correspondingly increases sending rate to grab spare bandwidth. Specifically, when the time interval between two consecutive packets is greater than the transmission time of one packet, inspired by the AntiECN [12], the switch marks the ECN bit of the dequeued packet. Then the corresponding marked grant will be generated at the receiver to trigger more data packets. The receiver-driven sender increases its sending rate to match the available bandwidth.

The objective of AMRT is to achieve the ultra-low latency and ultra-high link utilization simultaneously. On the one hand, AMRT takes advantage of conservative receiver-driven transmission to guarantee near-zero queueing delay. Once receiving a grant without anti-ECN marking, the sender conservatively triggers only one new data packet. On the other hand, AMRT ensures full link utilization with the aid of explicit anti-ECN marking. Once receiving a grant with anti-ECN marking, the sender aggressively triggers two new data packets to grab the free bandwidth.

In summary, our major contributions are:

J. Hu is with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China.

J. Huang, Z. Li and J. Wang are with School of Computer Science and Engineering, Central South University, Changsha 410083, China. E-mail: jiaweihuang@csu.edu.cn

T. He is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA.

A preliminary version of this paper appears in ACM ICPP [1], Edmonton, Alberta, Canada, August, 2020.

- We conduct an extensive simulation-based study to analyze two key issues that lead to low link utilization issues in receiver-driven transmission: (1) when one flow passes through multiple bottleneck links, the spare bandwidth released by it at the bottlenecks other than the most congested one can not be utilized by the other coexisting flows, (2) when multiple flows share the same link in the dynamic traffic scenario, if some flows finish their transmissions, the other flows are not able to get more grants to seize the free bandwidth.
- We integrate the explicit feedback into receiver-driven transport protocol called AMRT, which uses anti-ECN marking to explicitly notify the sender of spare bandwidth at the bottleneck link. Therefore, AMRT increases aggressiveness of conservative receiver-driven transmission to guarantee ultra-low latency and high link utilization simultaneously.
- By using both testbed implementation and NS2 simulations, we demonstrate that AMRT performs remarkably better than the state-of-the-art receiver-driven transport protocols. AMRT reduces the average flow completion time (AFCT) by 19%-42% under heavy workload and yields up to 38%, 28%, 22% and 11% link utilization improvement over pHost, ExpressPass, Homa and NDP, respectively.

The rest of the paper is organized as following. In Section II, we present the related work. In Section III and IV-D, we respectively describe our design motivation and overview. In Section IV, we introduce the design details of AMRT. We give the model analysis of AMRT in Section V and discuss the implementation in Section VI. In Section VII and VIII, we show the testbed experimental and NS2 simulation results, respectively. We conclude the paper in Section IX.

II. Related Works

We compare AMRT with the sender-based, rate allocation, explicit flow control and receiver-driven mechanisms.

DCTCP [2] uses ECN marking [13]–[16] ratio to adjust the congestion window. D^{2}TCP [17] adjusts rate to maximize the deadline-meeting rate. DCQCN [18] uses fine-grained congestion control to adjust sending rate. TIMELY [19] leverages RTT to reduce congestion. Compared with the traditional TCP, these protocols achieve a good balance between low delay and high throughput. However, they may still suffer from buffer overflow under highly concurrent flows.

Several proposals use rate control techniques to achieve the target rate quickly [20]–[23]. PIAS [20] schedules packets based on the priority queues. Auto [21] schedules by deep reinforcement learning. Karuna [23] schedules a mix of flows to achieve good performance. TFC [24] allocates tokens to achieve near-zero queueing. ExpressPass [11] controls congestion by shaping the credit packets. However, these schemes require rate calculation or global scheduling, which incur delay overhead especially for short flows.

Lots of transport protocols accurately adjust rate to match the link capacity by using explicit feedback from switches. XCP [25] and VCP [26] leverage explicit feedback with multiple bits to regulate congestion window. Recently, HPCC [27] leverages in-network telemetry (INT) technique to obtain precise link load to control traffic precisely. However, it is hard for these protocols to make a tradeoff between low feedback overhead and accurate rate adjustment.

Recent receiver-driven transport protocols are proposed to achieve ultra-low queueing delay in DCNs [8]–[10], [28], [29]. pHost [9] performs distributed per-packet scheduling at the end hosts. NDP [10] cuts payloads and uses a receiver-pulled mechanism to control incoming traffic. Homa [8] uses a receiver-driven flow control mechanism and in-network priority queues to provide good performance. Aeolus [29] assigns a higher drop priority for unscheduled packets. Through the proactive congestion control mechanism, these conservative transport protocols guarantee the bounded queueing delay. However, they still potentially suffer from low link utilization under multi-bottleneck and high dynamic traffic scenarios.

In contrast with the above transport mechanisms, our solution AMRT works through a different perspective: AMRT uses anti-ECN marking to notify senders to increase sending rate to make a sufficient use of free bandwidth, thus achieving better transport performance in terms of low latency and high link utilization simultaneously without any traffic overhead.

III. Design Motivation

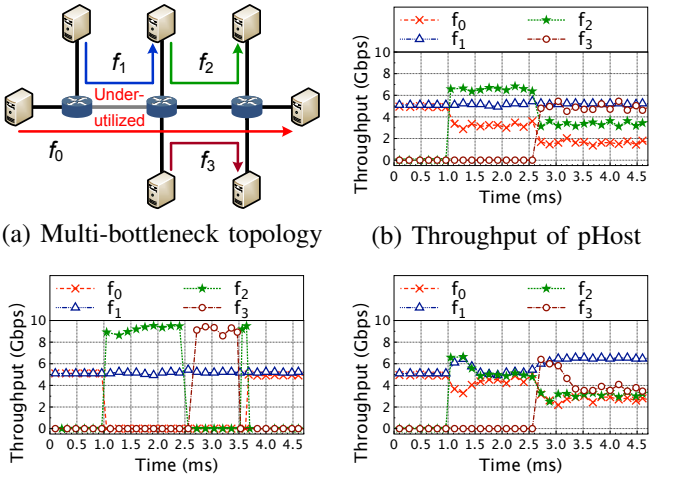
To motivate our design, we have investigated the impact of the receiver-driven transmission scheme on link utilization in multiple bottlenecks and dynamic traffic scenarios.

A. Link Under-utilization in Multiple Bottlenecks Scenario

The multiple bottlenecks widely exist in datacenter networks [30], [31], [32]. In multi-rooted tree topologies such as [33] and Fat-tree [34], it is common that cross-rack flows coexist with a variable number of background flows at each hop, resulting in multiple bottlenecks [35], [24]. For example, during the distributed training process of computation-intensive machine learning, the massive number of model parameters need to be updated synchronously by using a large number of cross-rack flows, which traverse multiple hops between thousands of servers at the end of each iteration [36], [27], [37], [38]. From the study in [39], about 4.3% packet drop rate caused by congestion at 80% load almost occurs at the last hop of the access destination link. Moreover, the observations from [31] and [40] show that the links with losses due to congestion have nearly 60% utilization at the edge layer, while the links with losses have less than 30% utilization in the core and aggregation layers.

In practice, it is common that the datacenter network traverses multiple bottlenecks and a flow coexists with a variable number of cross flows at each bottleneck link. As shown in Fig.12 in Section VII, in the typical multi-tier datacenter topology, the cross-rack flows f_0, f_1 and f_2 traverse multiple bottlenecks, which are shared with other cross flows. Unfortunately, since the receiver-driven transport protocol only generates one grant corresponding to each arrival packet at the receiver, such conservative receiver-driven transmission easily leads to low link utilization. Specifically, when a flow reduces its sending rate due to the flow competition at the

most congested bottleneck, the receiver-driven cross flows at the other bottlenecks are not able to get more grants to trigger more data packets to utilize the released bandwidth.



(a) Multi-bottleneck topology (b) Throughput of pHost

(c) Throughput of Homa (d) Throughput of ExpressPass

Fig. 1. Throughput loss for different receiver-driven transport protocols in the Multi-bottleneck topology.

We conduct NS2 simulation to illustrate the under-utilization problem under the multi-bottleneck scenario as shown in Fig. 1 (a). Four receiver-driven pHost flows f_0, f_1, f_2 and f_3 share two bottleneck links with respective senders and receivers. The bottleneck link has the rate of 10Gbps and the round trip propagation delay is 40μs. The switch buffer size is 128 packets.

Fig.1 (b) shows the throughput loss of pHost. At the beginning, we assume that f_0 and f_1 fairly share the 1st bottleneck link (from the left) without queueing buildup after experiencing congestion. The total number of their flighting data packets is worth of a bandwidth-delay product (BDP). Once a packet arrives at the receiver, a corresponding grant is generated and sent to the related sender to trigger a new data packet. At 1ms, f_2 starts at the line rate of 10Gbps, which is twice the rate of f_0. Consequently, f_0 experiences congestion at the 2nd bottleneck, and the sending rate of f_0 decreases to 3.3Gbps due to the bandwidth competing with f_2. While f_1 still triggers the new packets according to the arrival rate of packets at the receiver. Thus, the link utilization of the 1st bottleneck link drops to 83.3% because f_1 is not able to increase its sending rate to grab the spare bandwidth released by f_0. Similarly, at 2.5ms, the sending rate of f_0 decreases again due to the starting of f_3, resulting in the link utilization of the 1st bottleneck link is reduced to only about 66%.

Fig.1 (c) shows the throughput loss issue of Homa. Unlike pHost, Homa transmits the short flows at the high priority. At 1ms, f_2 is transmitted at the line rate with the highest priority. Even though there is 50% available link utilization released by f_0 at the 1st bottleneck link, f_1 cannot seize the spare bandwidth because it conservatively drives new packets according to the packet arrival rate at the receiver. At 2.5ms, Homa prioritizes the shortest flow f_3 to fully use the 2nd bottleneck link through the highest priority queue. Once f_3 is finished, f_2 is transmitted at the highest priority again.

We further add the result of ExpressPass, which is a receiver-driven transport protocol that can work with multi-bottlenecked topology. ExpressPass can saturate the bottleneck link by shaping the flow of credit packets at the switches, but the sender should wait for credits to start data transmission in the first RTT, resulting in bandwidth wastage. As shown in Fig.1 (d), ExpressPass performs better than pHost due to rate-limiting credits at switches. After f_2 and f_3 start, the flow f_1 can increase the sending rate to utilize the available bandwidth released by the flow f_0, which competes bandwidth with f_2 and f_3 at the other bottleneck link. However, since the new flows f_2 and f_3 need to wait one-RTT for credits to start transmission even though the network is under-utilized, resulting in a waste of bandwidth.

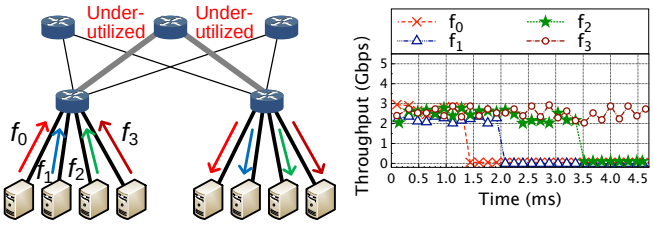
In datacenter networks, a destination usually connects with multiple sources. If a receiver only sends grants to one sender at a time, like pHost [9], the bottleneck link bandwidth will be wasted with unresponsive senders, resulting in poor network utilization especially under highly dynamic traffic scenario. To improve link utilization in this case, Homa [8] employs the overcommitment mechanism to allow a receiver grants multiple senders simultaneously. Consequently, even though some senders are not able to respond immediately to the grants, the link bandwidth is also effectively utilized by other active senders. However, this solution is hard to directly address the under-utilization problem in the multi-bottleneck and dynamic traffic scenarios because that, when some flows release bandwidth, the other coexisting flows can not proactively get more grants to trigger more data packets to utilize the free bandwidth. What's worse, as we show in our evaluation in Section VIII-D, the overcommitment mechanism easily causes queueing buildup under the highly dynamic workloads, leading to poor latency performance.

B. Link Under-utilization in Dynamic Traffic Scenario

Due to the continuous and instantaneous changing of traffic behavior, the highly dynamic is an intrinsic feature of data center network [41], [42], [43]. Recent data center traffic studies show that traffic fluctuates frequently over time and space [30], [31], [32]. Specifically, flows arrive and leave randomly in nature and most of them are short-lived, lasting less than 0.1s. For example, in the popular partition/aggregate communication pattern, a large number of flows are generated concurrently to exchange data among servers, incurring bursty transient traffic [2], [11]. Consider a dynamic traffic scenario in which multiple receiver-driven flows with different source/destination pairs share a bottleneck, limited by the conservative congestion control, even if some of flows complete and release the bottleneck bandwidth, the remaining flows are unable to actively increase sending rate to saturate the bottleneck link.

We use another NS2 simulation test to show the under-utilization issue in the dynamic traffic scenario. The simulation settings are the same as that in Section III-A. As shown in Fig. 2 (a), four receiver-driven pHost flows f_0, f_1, f_2 and f_3 share a bottleneck link with respective sources and destinations.

In Fig.2 (b), at the beginning, four flows share a bottleneck at the same rate and just make full use of the link capacity



(a) Dumbbell topology (b) Throughput loss

Fig. 2. Throughput loss with dynamic traffic in the Dumbbell topology.

without queueing buildup. Since the four flows have the same priority, the throughput of pHost and Homa are the same in this scenario. At 1.5ms, f_0 is finished, and the 3 flows to the other 3 receivers cannot use the grants corresponding to the last packets of f_0. The other 3 flows continue to drive new packets according to the rate of received packets at the destination. Therefore, the bottleneck link is not saturated by the remaining 3 flows, resulting in 25% reduction of link utilization. After f_1 and f_2 finish transmission successively, the link utilization of the bottleneck link drops to 25% because f_3 cannot increase the sending rate by driving more packets.

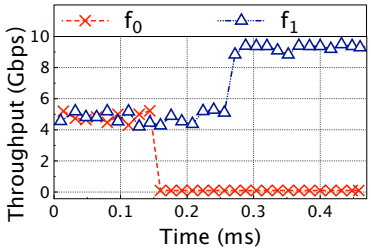


Fig. 3. Throughput of pHost when congestion occurs at the last hop.

Under the dumbbell topology, multiple receiver-driven flows with different source/destination pairs share a bottleneck, congestion occurs at the inter-switch hop. Since these flows to different receivers cannot share the tokens released by the finished flows in pHost, even though some of flows complete and release the bottleneck bandwidth, the remaining flows are unable to actively increase sending rate to saturate the bottleneck link.

We further add the simulation test for pHost in multiple-to-one scenario where congestion occurs at the last hop. We agree with the reviewer that pHost can work with in such a topology. However, when one flow is finished, pHost needs to wait for a timeout period (default setting to 3 RTTs) to downgrade the corresponding sender, and then sends tokens to other responsive senders. In the process of waiting for the sender to be degraded, there will be a waste of bandwidth. In this test, two flows f_0 and f_1 are sent to the same receiver, and pHost employs fairness scheduling at the receiver. As shown in Fig.3, at the beginning, the receiver-driven pHost flows f_0 and f_1 fairly share the bottleneck link. When f_0 is finished, the remaining flow f_1 cannot immediately utilize the available bandwidth released by f_0 because f_1 cannot obtain all tokens until the sender of f_0 is downgraded after three RTTs.

C. Summary

Our analysis of the under-utilization problem of receiver-driven transmission leads us to conclude that (1) though the receiver-driven transmission ensures low latency, the conservative congestion control is not able to make full use of link bandwidth at multiple bottlenecks, (2) since traffic is likely to be variable over both time and space, receiver-driven flows potentially waste bandwidth when flows dynamically come and go. These conclusions motivate us to design and implement a receiver-driven transport protocol to simultaneously achieve low latency and high link utilization.

IV. Design Details

We first describe the design details of AMRT, which consists of three parts: packet interval estimation and anti-ECN marking at switches in Section IV-A, grant generation and explicit feedback at receivers in Section IV-B, receiver-driven transmission and rate adjustment at Senders in Section IV-C. Then we illustrate an example for AMRT overview in Section IV-D.

A. Packet Interval Estimation and Anti-ECN Marking at Switches

To avoid low link utilization in the conservative receiver-driven transmission, the straightforward approach is to measure the spare bandwidth of bottleneck link and then send it back to senders to adjust their sending rates. Unfortunately, it requires multiple bits to encode the congestion level information to present how much the network is under-utilized, unavoidably introducing large overhead [25]. AMRT employs a simple anti-ECN marking mechanism, which uses only one bit to explicitly carry under-utilization signal by using available ECN bit in the IP header [12].

In our AMRT design, switches are mainly responsible for estimating inter-packet gap and anti-ECN marking. The switches monitor the inter-dequeue time t_{inv} by recording the time when packets are forwarded at egress ports, which is well supported by modern data center switches [27]. The value of t_{inv} is calculated as $t_{inv} = t_{current} - t_{last}$, where $t_{current}$ and t_{last} are the dequeue times of the current and last packets at the egress port, respectively.

Specifically, no matter which flow the packets belong to, when the inter-dequeue time t_{inv} of the current and last packets is greater than the transmission time of one packet, the link is deemed under-utilized and the current packet is marked by setting the Congestion Experienced (CE) code-point bit $CE_{current}$ to 1. Otherwise, the bottleneck link is saturated and the value of $CE_{current}$ is set to 0. That is,

$$\begin{cases} CE_{current} = 1 & t_{inv} \geq \frac{MSS}{C}, \\ CE_{current} = 0 & t_{inv} < \frac{MSS}{C}, \end{cases} \tag{1}$$

where C is the bottleneck link capacity and MSS is the TCP segment size. For the packets with different sizes, we use the default Ethernet MTU of 1500 Bytes as MSS in calculating the packet transmission time to avoid congestion. Note that the initial value of CE bit is set to 1 to indicate under-utilized bottleneck link.

During the end-to-end transmission through multiple bottleneck links, the sending rate of a flow is limited by the most congested bottleneck. AMRT performs "AND" operation

between the piggybacking marked value CE_{last} from the last switch and the value of $CE_{current}$ at the current switch to obtain the final marked value CE_{final}. Thus, we have $CE_{final} = CE_{current}$ & CE_{last}.

Fig. 4 shows the anti-ECN marking operation of AMRT. Specifically, switch 1 measures the time interval between two continuous dequeued packets at the egress port. Two packets are marked because the time interval satisfies Equation (1), meaning that the bottleneck link still has free bandwidth to transmit more packets. When the two packets arrive at switch 2, another packet from the other ingress port also arrives, making the idle time between packets not large enough to transmit a packet. Therefore, the 2nd packet is unmarked to indicate there is no spare bandwidth.

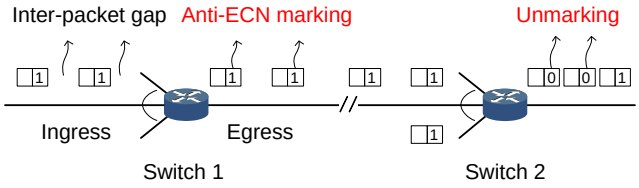


Fig. 4. Anti-ECN marking operation of AMRT. The dequeueing packet is marked by setting CE code-point to 1 once the inter-packet gap exceeds the transmission time of one packet. Otherwise, the packet is unmarked.

In the anti-ECN marking mechanism, only if all switches mark the CE bit of one data packet as 1, the corresponding marked grant packet will trigger two data packets. That is, for each final marked packet, the inter-packet gap between it and the previous packet on the bottleneck link during its end-to-end transmission is large enough to transmit another packet without causing queue buildup. Consequently, the sending rate is limited by the conservative receiver-driven congestion control and anti-ECN marking mechanism to avoid queue buildup and achieve high link utilization simultaneously. Note that since the marking operation is made on packet level and is independent of the flow, the problem of bandwidth over-allocation is avoided. Therefore, AMRT only fills the gap between packets and will not lead to bandwidth over-allocation.

In the AMRT's anti-ECN marking mechanism, some flows are likely to have more marked packets than others. Therefore, it is difficult for AMRT to ensure fairness among flows. However, although fairness cannot be guaranteed, the first advantage of marking based on packet level is that AMRT does not need to maintain per-flow state information at switch, which has very limited storage resources. Therefore, AMRT has good scalability because it is not limited by the number of flows. Furthermore, as a receiver-driven transmission mechanism, the primary goal of AMRT is to achieve low latency, and then improve link utilization by using anti-ECN marking. The issue of fairness is worth studying in our future work. Moreover, the anti-ECN marking mechanism provides explicit feedback from the switches to help senders aggressively increase sending rate by only employing the ECN capability on current commodity switches, without incurring large deployment overhead.

B. Grant Generation and Explicit Feedback at Receivers

At the receiver, AMRT is designed based on the core design of proactive congestion control. Specifically, upon arrival of a data packet, the receiver generates a corresponding grant packet and sends it back to the source end-host to drive a new data packet called scheduled packet. Consequently, the sending rate is limited by this conservative congestion control mechanism to avoid queue buildup. Meanwhile, the receiver uses the built-in ECN-Echo function to convey free bandwidth information back to the sender. When an ECN-marked data packet arrives, the receiver copies the marked CE code-point and sets the ECN-Echo flag in the corresponding grant packet to notify the sender of low link utilization.

Therefore, instead of relying on reactive congestion control to reduce sending rate after congestion occurs, AMRT takes advantage of proactive receiver-driven congestion control to obtain ultra-low queueing delay and adopts explicit anti-ECN marking to increase the sending rate and network utilization.

The other consideration is how AMRT handles lost packets including data packets and grant packets. In AMRT, the receivers are responsible for detecting the packet loss. Similar to Homa, AMRT employs a timeout-based mechanism to detect lost packets without traffic overhead. Specifically, when the data packet corresponding to a grant packet including an unmarked grant or an anti-ECN marked grant has not arrived at the receiver within a time period ($1\times RTT$ by default), whether it is caused by data packet loss or grant loss, the receiver reissues a same grant packet to ask the sender to retransmit the data packet.

C. Receiver-driven Transmission and Rate Adjustment at Senders

At the sender, AMRT performs receiver-driven rate adjustment. During startup, a new flow sends data packets at line rate like the existing state-of-the-art receiver-driven transport protocols such as Homa [8], pHost [9] and NDP [10] to avoid wasting bandwidth by waiting for the grant packets from the receiver. Then AMRT adjusts the sending rate in a cautious yet active manner. AMRT leverages grant packets from receivers to conservatively trigger new data packets rather than aggressively sending data packets in reactive congestion control. Meanwhile, AMRT reasonably increases sending rate in response to under-utilized bottleneck link feedbacked by anti-ECN marked packets to achieve a tradeoff between proactive and reactive congestion control.

Specifically, if the sender receives an ECN-marked grant packet, it indicates that the dequeueing time interval between the corresponding data packet and the previous one is large enough to transmit one data packet to fill the inter-packet gap. In this case, AMRT adds aggressiveness to the receiver-driven transmission by driving two packets to grab the free bandwidth. On the contrary, if a grant packet without ECN marking arrives at the sender, it means that the inter-packet gap is not large enough to add a data packet ahead of the corresponding data packet. In short, AMRT adjusts the number of sending packets according to the anti-ECN feedback to achieve both low latency and high network utilization.

D. An Illustrative Example for AMRT Overview

At a high level, AMRT retains the rationale of conservative rate control in the existing receiver-driven transmission solutions when the bottleneck link is saturated, and reasonably performs aggressive rate control when the bottleneck link is under-utilized. By giving full play to the advantages of proactive and reactive rate control, AMRT achieves ultra-low latency and high link utilization simultaneously.

The key point of AMRT is using anti-ECN marked packets to explicitly carry under-utilization information to make senders aggressively increase sending rate, while performing conservative transmission controlled by the receiver when the bottleneck link is saturated.

Specifically, the switches measure the inter-dequeue time between two consecutive data packets. When the interval time is large enough to transmit a packet (1500 Bytes by default), the switches mark the ECN bit in the packet header without any additional overhead. Then the receiver echoes back the under-utilization information to the sender. Finally, the sender increases the sending rate to improve link utilization according to the received marking information. AMRT conservatively transmits data driven by the receiver and aggressively increases the sending rate to utilize the free bandwidth to obtain both ultra-low latency and high utilization simultaneously. To show how AMRT works, we illustrate a simple scenario with two flows in Fig. 5.

Consider all links have same link capacity and 3 packets are able to saturate the bottleneck link without loss of generality. Two receiver-driven flows f_1 and f_2 are sent by senders S_1 and S_2 to receivers R_1 and R_2, respectively. At the switch, f_1 and f_2 share port 0. As shown in Fig. 5, sender S_1 and S_2 each send a data packet triggered by grants from the receiver R_1 and R_2 respectively. Then these two data packets pass through the same link. However, two data packets do not make full use of the bottleneck link. To utilize the spare bandwidth, if the inter-dequeue time between two packets is large enough to transmit a packet, AMRT marks the currently dequeued packet of f_2 to indicate under-utilization at the bottleneck link. R_2 receives one marked data packet and copies the marking information to the corresponding grant packet.

Once receiving the marked grant packet piggybacking under-utilized information, S_2 sends 2 data packets. S_1 sends 1 data packet triggered by the unmarked grant. Thus, 3 data packets make full use of the bottleneck link. Moreover, the additional one data packet does not introduce queue buildup due to the precise notification information.

Therefore, the key challenges of AMRT include: (1) measurement of the inter-dequeue time to judge whether there is spare bandwidth, (2) anti-ECN marking scheme without overhead, (3) adjustment of sending rate to timely and accurately grab the available bandwidth. In the following part, we present the design details to address the above challenges.

Next, we describe the design details of AMRT, which consists of three parts: packet interval estimation and anti-ECN marking, grant generation and explicit feedback, receiver-driven transmission and rate adjustment.

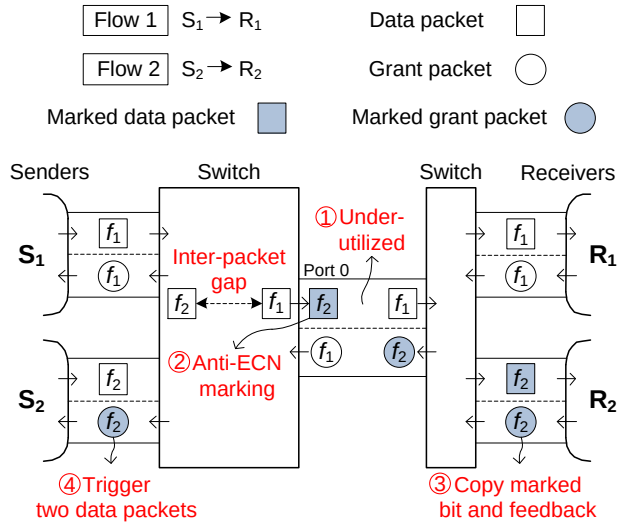


Fig. 5. An illustrative example of AMRT.

V. Model Analysis

AMRT uses anti-ECN marking mechanism to convey link utilization state and then notify senders of adjusting sending rate accordingly to avoid bandwidth wastage. Compared to the existing receiver-driven transport protocols, AMRT efficiently achieves low latency and high network utilization simultaneously. Here, we construct the theoretical model to analyze the performance gain of AMRT in link utilization.

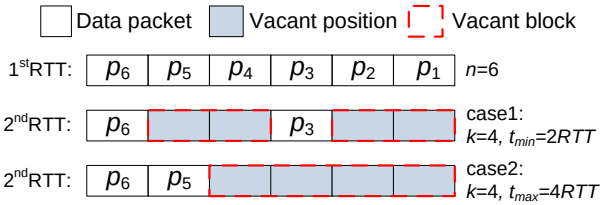


Fig. 6. A simple example shows the time required for AMRT to fill up the spare bandwidth when some packets release bandwidth.

As shown in Fig. 6, in the first round trip time (RTT) (i.e., 1^{st} RTT), we assume that n packets arrive at the switch back-to-back, and the output rate matches the bottleneck link capacity. In the next RTT (i.e., 2^{nd} RTT), we assume that there are k vacant positions representing the spare bandwidth among the remaining $n-k$ packets after some flows finish transmission. In the case1, k vacant positions are evenly distributed among the remaining $n-k$ packets, there are $\frac{k}{n-k}$ (k<n) consecutive vacant positions in each vacant block. In the case2, k vacant positions are consecutive among the remaining $n-k$ packets with only one vacant block. The time required for AMRT to reach the full link utilization in case1 and case2 is minimum and maximum, respectively.

Here, we further analyze the convergence time when AMRT fills up the spare bandwidth of bottleneck link. As shown in Fig. 6, the bottleneck link is saturated by 6 back-to-back packets (i.e. n=6) in the 1^{st} RTT. We assume that 4 packets are not sent in the 2^{nd} RTT (i.e. k=4), leaving 4 vacant positions. In our design AMRT, once the inter-packet gap between the

dequeued packet and its previous dequeued packet is more than the transmission time of one packet, the switch uses the anti-ECN signal to notify the sender to send one more packet in the next RTT. Specifically, in the case1, one anti-ECN marked packet in each vacant block notifies the sender to send one more packet in each round of RTT. Thus, 2 vacant positions are filled after one RTT, AMRT needs the minimal time of 2 RTTs to fill up the spare bandwidth of 4 vacant positions in 2 vacant blocks. In the case2, the maximum time for AMRT to achieve the full link utilization for 4 vacant positions in 1 vacant block is 4 RTTs.

Without loss of generality, when k vacant positions are evenly distributed among $n-k$ packets, there are $\frac{k}{n-k}$ (k<n) consecutive vacant positions in each of $n-k$ vacant blocks. In this way, $\lceil \frac{k}{n-k} \rceil$ RTTs are required to fill k vacant positions. Therefore, we get the minimum time t_{min} for AMRT to achieve the full link capacity as

$$t_{min} = \lceil \frac{k}{n-k} \rceil \times RTT, \tag{2}$$

where RTT is the base round trip time.

Please note that if $k = n$, the time required for AMRT to fill all vacant positions is $k \times RTT$, that is, the minimum time is equal to the maximum time.

If k spare positions are consecutive, the maximum time t_{max} to fill up the available link bandwidth is

$$t_{max} = k \times RTT. \tag{3}$$

In a word, AMRT is able to make full use of the bandwidth in the time $t \in [t_{min}, t_{max}]$, while the traditional receiver-driven transport protocols are not able to fill up the free bandwidth released by k packets. On the one hand, when the bottleneck link is saturated, AMRT performs proactive receiver-driven congestion control to maintain the low latency advantage of existing conservative receiver-driven transport protocols. On the other hand, when the bottleneck link is under-utilized, AMRT uses anti-ECN marking mechanism to convey under-utilized link state and then notify senders to increase sending rate accordingly to avoid bandwidth wastage, resulting in further reduced delay. Compared to the existing receiver-driven transport protocols, AMRT efficiently achieves low latency and high network utilization simultaneously.

Next, we use a simple theoretical model to quantify gain of AMRT in link utilization and flow completion time compared to the traditional receiver-driven protocol.

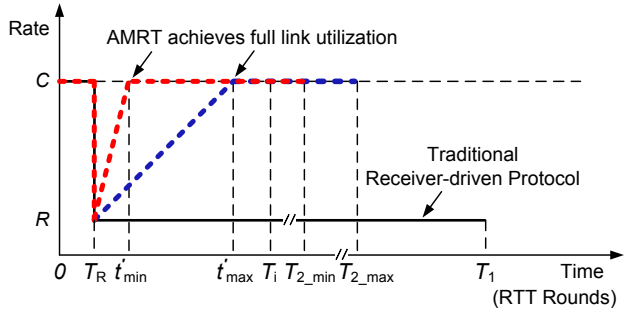


Fig. 7. AMRT vs. Traditional receiver-driven protocol.

As shown in Fig. 7, we use C to denote the bottleneck link capacity. We assume that the rate of a flow is reduced from C to R at time T_R due to network congestion. Since the existing receiver-driven protocols conservatively trigger new data according to the arrival rate at the receivers, the released free bandwidth is not utilized. Therefore, the flow completion time T_1 of current receiver-driven protocols is

$$T_1 = \frac{S - C \times T_R}{R} + T_R, \tag{4}$$

where S is the flow size.

AMRT adjusts the sending rate based on the anti-ECN marking at the bottleneck link. As shown in Fig. 7, the two dot lines show the maximum and minimum gains of AMRT. The earliest time t'_{min} for AMRT to increase the rate from R to C is

$$t'_{min} = \lceil \frac{C - R}{R} \rceil + T_R. \tag{5}$$

The latest time t'_{max} at which AMRT achieves the full rate C is

$$t'_{max} = C - R + T_R. \tag{6}$$

After time t' ($t' \in [t'_{min}, t'_{max}]$), AMRT makes full use of the bottleneck link. Then the flow size S is given by

$$S = C \times T_R + \frac{1}{2} \times (R + C) \times (t' - T_R) + C \times (T_2 - t'), \tag{7}$$

where T_2 is the flow completion time of AMRT. Then, we obtain T_2 as

$$T_2 = \frac{S - C \times T_R - \frac{1}{2} \times (R + C) \times (t' - T_R)}{C} + t'. \tag{8}$$

As shown in Fig. 7, for t'_{min} and t'_{max}, the corresponding values of T_2 are T_{2_min} and T_{2_max}, respectively.

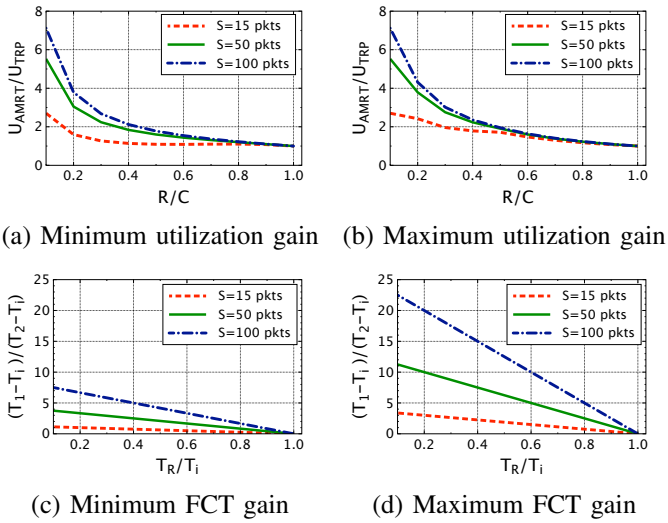
Let U_{AMRT} and U_{TRP} respectively denote the utilization ratios of AMRT and the traditional receiver-driven protocols at the bottleneck link. Then we quantify the utilization gain U_{gain} as

$$U_{gain} = \frac{U_{AMRT}}{U_{TRP}} = \frac{T_1}{T_2} = \frac{\frac{S - C \times T_R}{R} + T_R}{\frac{S - C \times T_R - \frac{1}{2} \times (R+C) \times (t' - T_R)}{C} + t'}. \tag{9}$$

We define T_i as the ideal flow completion time without network congestion as $T_i = \frac{S}{C}$. Then we get the gain in flow completion time FCT_{gain} as

$$FCT_{gain} = \frac{T_1 - T_i}{T_2 - T_i} = \frac{\frac{S - C \times T_R}{R} + T_R - \frac{S}{C}}{\frac{S - C \times T_R - \frac{1}{2} \times (R+C) \times (t' - T_R)}{C} + t' - \frac{S}{C}}. \tag{10}$$

By substituting t'_{max} and t'_{min} into Equation (9) and Equation (10), we can obtain the minimum and maximum gains of link utilization and FCT of AMRT as shown in Fig. 8. We set the capacity of bottleneck link C to 1Gbps, the round trip time to 100μs and the rate reduction time T_R to 0. The results in Fig. 8 illustrate that AMRT significantly outperforms the traditional receiver-driven transport protocols even if it converges to the full rate with the largest time t_{max}. Specifically, Fig. 8 (a) and (b) show that as the ratio of $\frac{R}{C}$ decreases, the utilization gain for AMRT increases



(a) Minimum utilization gain (b) Maximum utilization gain

(c) Minimum FCT gain (d) Maximum FCT gain

Fig. 8. Minimum and Maximum utilization gain and FCT gain with increasing $\frac{R}{C}$ and $\frac{T_R}{T_i}$.

considerably. Moreover, AMRT performs better with larger flow size. Fig. 8 (c) and (d) demonstrate that the FCT gain of AMRT increases with the decreasing value of $\frac{T_R}{T_i}$ and with the increasing flow size S since AMRT fills up more spare bandwidth to improve link utilization.

VI. Implementation

We have implemented a prototype of AMRT based on Intel Data Plane Development Kit (DPDK) 20.11 [46], which is a framework to accelerate packet processing by allowing the network stack directly communicate with NIC bypassing the OS kernel.

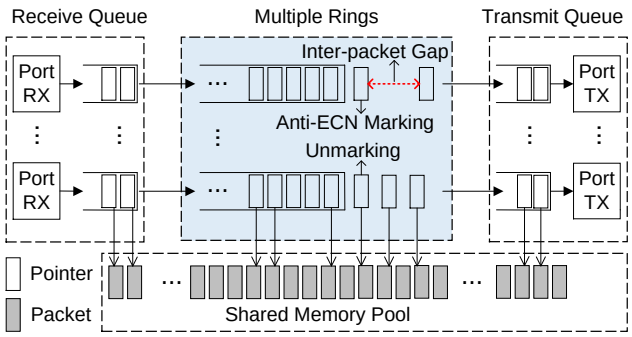


Fig. 9. AMRT's DPDK Implementation at switch.

Fig. 9 shows the structure of AMRT's DPDK implementation at switch, which consists of receiving, forwarding and transmitting components. Each RX/TX port has a receive and transmit queue, and the number of these queues are configured by `rte_eth_dev_configure()` function. The ports are served by different CPU cores. Each packet arriving at the RX port is retrieved by `rte_eth_rx_burst()` function and delivered to the receive queue, which is allocated and set up by `rte_eth_rx_queue_setup()` function. After that, the packet is enqueued a logic queue corresponding to the RX port, which is created by `rte_ring_create()` function with the form of ring structure.

In the packet forwarding component, the key point of AMRT is anti-ECN marking. The dequeueing inter-packet gap of two consecutive packets is calculated based on their dequeueing time obtained by `rte_rdtsc()` and `rte_get_timer_hz()` functions. When the packet interval exceeds the transmission time of one packet, the CE code-point in the IP header of the subsequent dequeueing packet is accessed by `rte_pktmbuf_headroom()` function and set to 1 to indicate link under-utilization. The checksum of IP header is processed by `rte_ipv4_cksum()` function.

In practice, we measure the burst gap. Specifically, we pop the burst packets from the logic egress queue at a time. The maximum number of burst packets is set to four in AMRT's implementation. If there is no interval between the burst packets, they are dequeued in the burst manner and fully utilize the link bandwidth. Moreover, if the logic egress queue length is larger than one, the burst packets are marked with ECN value of 0 at the logic ingress queue, which will not affect the dequeueing rate. Otherwise, if the interval between the arrival packets at the logic egress queue is larger than the transmission time of one packet, we can only pop one packet from the logic egress queue at a time, then the dequeued packet is marked with ECN value of 1 and sent to TX buffer one by one. The above operation can guarantee the maximum speed is almost at 10Gbps.

Finally, the transmitting component calls `rte_eth_tx_buffer()` function to deliver packets from each logical queue to the corresponding TX port's transmit queue, which is allocated and set up by `rte_eth_tx_queue_setup()` function. Since the above DPDK processing only manipulates the packet pointers, the real packets with `rte_mbuf` struct are driven from the shared memory pool and transmitted to the network by `rte_eth_tx_burst()` function.

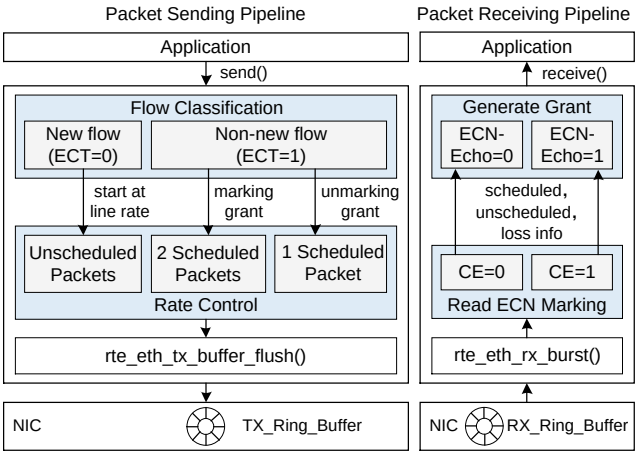


Fig. 10. AMRT's DPDK Implementation at end-hosts.

Fig. 10 shows the architecture of AMRT's DPDK implementation at end-hosts, which consists of packet sending and receiving pipelines at the sender and receiver, respectively. At the sender, applications start data transmission by calling the `send()` function. For new flows, they start immediately at line rate with bursting unscheduled packets in the first RTT.

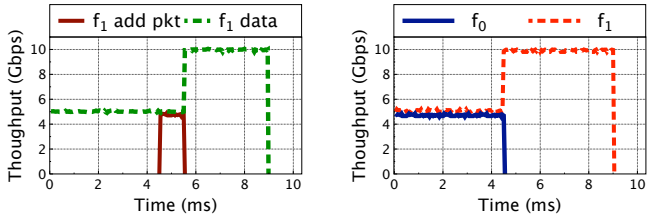
For other flows, the sender decides to drive 1 or 2 scheduled packets according to the ECN-Echo value of received grant. Then the packets are forwarded to the TX Ring buffer of NIC by `rte_eth_tx_buffer_flush()` function. At the receiver, packets are retrieved from RX Ring buffer of NIC by `rte_eth_rx_burst()` function. For each received packet, according to the value of CE code-point in IP header, a corresponding grant with or without ECN-Echo flag is generated. Then the data packet is delivered to the application by calling the `receive()` function.

VII. Testbed Evaluation

In this section, we use a real testbed to evaluate the feasibility and effectiveness of AMRT. In the testbed, each server has a Intel Core Xeon(R) E5-2687W V4 CPU and 32GB memory. The servers run Ubuntu18.04 with Linux 4.15.0-1090 kernel and are equipped with Intel 10GbE 2P X520 Network Interface Cards (NICs). The servers acting as the four-port DPDK switches with two Intel 10GbE 2P X520 NICs. The round-trip propagation time is 100μs.

We firstly test whether AMRT successfully grabs the spare bandwidth under the dynamic traffic scenario. The test topology is same as Fig. 2 in Section III-B. Flow f_0 and f_1 sharing a single bottleneck link are sent to two receivers, respectively. We show the throughput to the bottleneck link bandwidth in Fig. 11.

We run a test with two flows initiated at the same time. At the beginning, f_0 and f_1 fairly share the bottleneck link. Fig. 11 (a) shows the throughput of f_1. When the background flow f_0 is finished at about 4.5ms, f_1 adds new packets. The added packets driven by the marked grant packets in f_1 approximately take a half bandwidth of the bottleneck link released by the completed flow f_0. We show the throughput of two flows in Fig. 11 (b). The results illustrate that the bottleneck links are fully utilized by AMRT under the dynamic traffic scenario.



(a) Added packets in AMRT (b) Normalized throughput

Fig. 11. Throughput of AMRT under dynamic traffic scenario.

Next, we compare AMRT with the state-of-the-art receiver-driven transport protocols in a multi-bottleneck scenario. We run a test with four receiver-driven flows in the leaf-spine topology as shown in Fig. 12. Flow f_0 experiences two bottlenecks, which are shared with f_1 and f_2, respectively. In addition, f_2 and f_3 share a single bottleneck link.

Fig. 13 shows the throughputs of pHost, Homa, NDP and AMRT over time. At the beginning, f_0 and f_1 fairly share the bottleneck link. Fig. 13 (a) and (b) show that, when a new flow f_2 with the same destination as f_0 starts at 0.1s, f_2

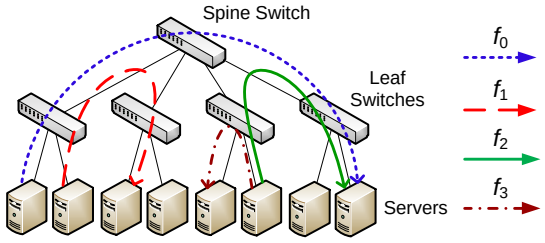
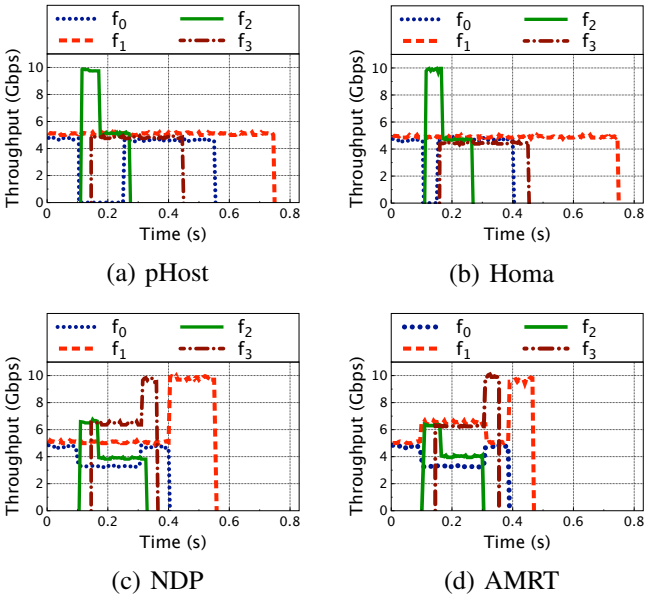


Fig. 12. Testbed topology for multi-bottleneck scenario.

gets full link capacity in pHost and Homa due to the shortest remaining processing time (SRPT) policy. In Fig. 13 (c), (d) and (e), since f_2 starts transmission with the link rate, f_0 and f_2 share the bottleneck link with a rate ratio of about 1:2 after flow competition.



(a) pHost (b) Homa

(c) NDP (d) AMRT

Fig. 13. Throughput of four receiver-driven transmission schemes under multi-bottleneck scenario.

For pHost, to maximize network utilization at the beginning of transmission, a few "free tokens" are assigned for per flow to trigger new data packets without waiting any tokens for the request to send (RTS) packet from the corresponding destination. Moreover, when the congestion occurs at the last hop, pHost employs a source downgrading mechanism to prevent a receiver from sending tokens to a source that does not respond with data packets. Specifically, if the number of unexpired tokens exceeds a threshold, the receiver downgrades the source and stops assigning tokens to it until the end of a timeout period (default being $3\times RTT$ [9]). However, if congestion does not occur at the last hop, but in the multi-bottleneck scenario as shown in Fig.12, it is difficult for pHost to seize the available bandwidth.

After f_2 starts, f_0 releases bandwidth at the first bottleneck link under all receiver-driven schemes. Fig. 13 (a) shows that, the spare bandwidth released by f_0 at the first bottleneck link is wasted in pHost. The reason is that, even if the sending rate of f_0 drops, the receiver-driven flow f_1 still triggers new data packet according to the arrival rate of data packet at its receiver. The f_1 sender can not get more grants to increase sending rate to utilize the spare bandwidth released by f_0.

After f_2 finishes, due to the conservativeness of pHost, f_3 is not able to get more grants to trigger more data packets and utilize the free bandwidth released by f_2.

For Homa, there are two schemes to improve link utilization. Firstly, to avoid under-utilization issue when some senders do not respond to grants in many-to-one scenario, the overcommitment scheme allows a receiver to grant multiple senders simultaneously. However, for the responsive senders, since Homa still drives new data packets based on the data arrival rate at the receiver, the overcommitment scheme will not send more grants to each sender to break the deterministic nature of proactive receiver-driven transport. Secondly, if there are multiple incoming messages at a receiver, once a grant has been sent for the last bytes of a message, data packets for that message may result in grants to other messages for which grants had previously been stopped. However, if multiple messages are sent to different receivers, even though the last data packets of a message result in grants, which cannot be used by the message at the another receiver to trigger new data packets.

Specifically, the throughput of Homa is shown in Fig.13 (b). Since f_1 does not go to the same receiver as other flows, f_1 cannot use the grants after the end of the flows at other receivers. Moreover, in the one-to-one scenario, the overcommitment mechanism does not work. Therefore, even though the flows f_0, f_2 and f_3 are finished, f_1 still triggers new data packets according to the arrival rate of data packets at its receiver, without any increase in its throughput.

For NDP, flows start at line rate. If the switch queue exceeds eight packets, the payloads of packets are trimmed. Once every packet header or data packet arrives at the receiver, a corresponding PULL packet is generated and added to the pull queue, which is shared by all connections at the receiver. When the congestion occurs at the last hop, if a flow terminates, the PULL packets will pull more data packets for other flows. However, when a flow experiences congestion under the multi-bottleneck scenario, the available bandwidth released by it is likely to be wasted, resulting in low link utilization.

Specifically, the throughput of NDP is shown in the Fig.13 (c). When f_2 starts, f_0 experiences congestion at the bottleneck link shared with f_2. Though the arrival rate of data packets at the f_0's receiver decreases, since both the data packets and trimmed packet headers of f_0 will trigger PULL packets, the sender of f_0 still receives a bandwidth-delay product (BDP) worth of PULL packets. Thus, the sender of f_0 still keeps the line sending rate and competes bandwidth with f_1, letting f_1 occupy only half of link bandwidth. However, once f_0 finishes its transmission at time 0.4s, the throughput of f_1 increases since the bandwidth competition disappears.

Fig.13 (d) shows that, after f_2 finishes, AMRT is able to flexibly grab the spare bandwidth by the anti-ECN marking feedback mechanism. Specifically, between 0.1s and 0.3s, f_1 under AMRT increases the throughput from 5Gbps to around 6.6Gbps compared with the other protocols. Similarly, after f_0 finishes transmission, the rate of f_1 achieves the link capacity in AMRT. AMRT reduces the FCT of flow f_1 by 31%, 30% and 12% over pHost, Homa and NDP, respectively. These results indicate that AMRT is able to fully utilize the spare bandwidth in the multi-bottleneck scenario to speed up flow transmission.

In addition, Fig. 13 (a) shows that f_0 stops transmission until f_2 finished as pHost adopts the SRPT policy. In Fig. 13 (b), when the rate of f_2 drops to half of the bottleneck bandwidth at the beginning of f_3, f_0 gets 50% of the link capacity due to overcommitment mechanism in Homa. Therefore, Homa reduces the FCT of flow f_0 by 28% compared with pHost. However, since the free bandwidths released by f_0 and f_2 are not utilized by f_1 and f_3, respectively, the link utilization is still lower than AMRT.

VIII. Simulation Evaluation

In this section, we firstly compare AMRT performance against the state-of-the-art receiver-driven transport protocols over a wide range of realistic datacenter workloads in the large-scale scenarios. Then we test AMRT performance in many-to-many communication and bursty scenarios.

A. Performance under Realistic Workloads

We perform NS2 simulations to evaluate AMRT performance on a large-scale network topology with varying workloads under the typical datacenter scenarios. We measure the flow completion time (FCT), 99^{th} percentile FCT and link utilization of AMRT, pHost, Homa and NDP.

Simulator: For pHost, Homa and NDP, we implement their transmission mechanisms in NS2 simulator according to the descriptions in [9], [8] and [10], respectively. pHost employs fair scheduling at the end-hosts. Each flow is assigned a BDP worth of free tokens. The sender is downgraded for 3 RTTs when the number of unexpired tokens exceeds a BDP worth of ones. Homa employs the shortest remaining processing time policy and 8 priority queues. We implement the timeout-based loss recovery for Homa instead of assuming infinite switch buffer. For ExpressPass, we use the corresponding open source code with NS2 simulator [47]. For all the above schemes, we use the default parameters and configuration options recommended in the related papers and simulators provided by the authors.

Network Topology: We use a common leaf-spine topology with 10 top-of-rack (ToR) switches, 8 core switches and 400 end-hosts. Each leaf switch connects to 40 hosts with 100Gbps links. Each network link delay is set to 4μs [45]. The switch buffer size is set to 128 packets. We employ Equal Cost Multi Path (ECMP) mechanism to support multipath routing.

TABLE I
FLOW SIZE DISTRIBUTIONS OF REALISTIC WORKLOADS.

	Web Server (WSv)	Cache Follower (CF)	Hadoop Cluster (HC)	Web Search (WSc)	Data Mining (DM)
0-10KB (S)	63%	50%	60%	49%	78%
10KB-100KB (M)	18%	3%	10%	3%	5%
100KB-1MB (L)	19%	18%	20%	18%	8%
>1MB (XL)	0%	29%	10%	20%	9%
Average flow size	64KB	701KB	1.05MB	1.6MB	7.41MB

Traffic workloads: We use five workloads with the same distributions as the realistic ones, including Web Server

(WSv), Cache Follower (CF), Hadoop Cluster (HC), Web Search (WSc) and Data Mining (DM) [8], [9], [11], which cover a wide range of average flow sizes from 64KB to 7.41MB and more than half of flows are less than 10KB. Table 1 shows the flow size distributions of five realistic workloads. We generate the traffic between randomly selected source and destination hosts. The flow arrival follows a Poisson process and the traffic load varies from 0.1 to 0.7.

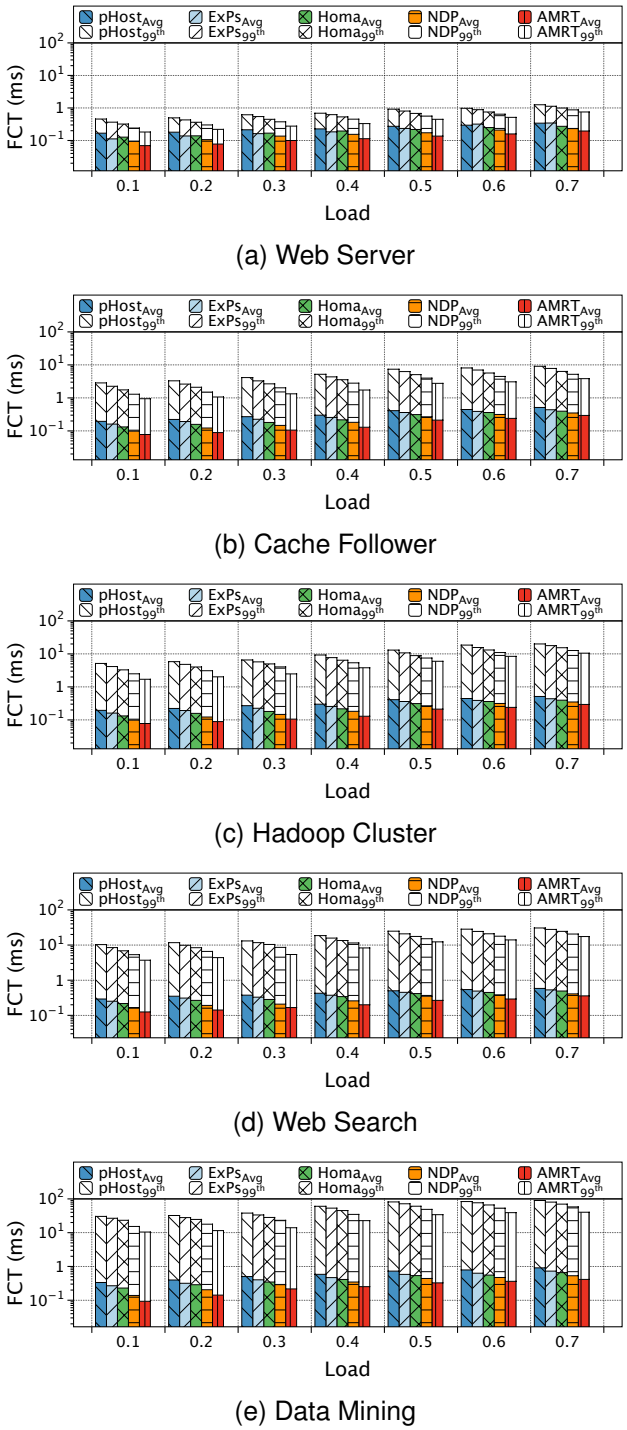


Fig. 14. The average FCT and 99^{th} percentile FCT of all flows for five different receiver-driven transport protocols with increasing load under five realistic workloads. ExPs stands for ExpressPass.

Fig. 14 shows the flow completion time of all flows with varying load from 0.1 to 0.7. The bottom and upper bar indicate the average and 99^{th} percentile FCT, respectively. AMRT performs better than the other protocols across all workloads, because it is able to timely grab the spare bandwidth through anti-ECN marking feedback. Among the five workloads, AMRT obtains the largest gain in Data Mining scenario. The reason is that more large flows experience multiple bottlenecks and are affected by dynamic traffic, potentially resulting in more opportunities for AMRT to fill up the spare bandwidth. Specifically, in Data Mining, AMRT reduces the AFCT and 99^{th} percentile FCT by 42%, 35%, 28%, 19% and 43%, 36%, 32%, 23% at 0.7 load over pHost, ExpressPass, Homa and NDP, respectively. Moreover, as the load increases, the higher network dynamic provides more chances for AMRT to seize the spare bandwidth. For example, when the Web Search load increases from 0.1 to 0.7, AMRT improves the average FCT from 31% to 49%, and the 99^{th} percentile FCT from 38% to 56% compared to pHost.

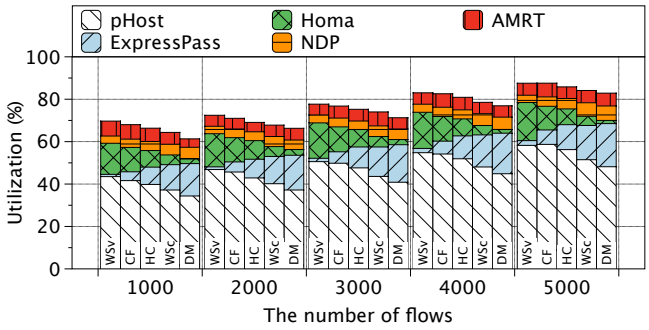


Fig. 15. The bottleneck utilizations of five different receiver-driven transport protocols with varying number of flows under five realistic workloads. Specifically, WSv, CF, HC, WSc and DM stand for Web Server, Cache Follower, Hadoop Cluster, Web Search and Data Mining, respectively.

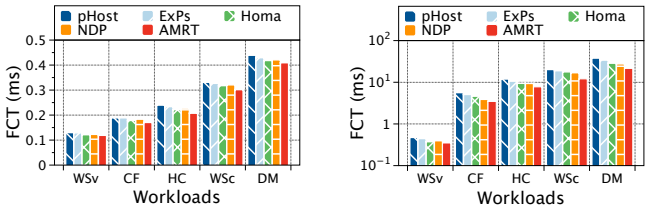
We also measure the bottleneck link utilization with increasing number of flows. As shown in Fig. 15, AMRT significantly outperforms the other receiver-driven mechanisms, because AMRT increases sending rates once detecting the spare bandwidth and thus achieves high rate close to the link capacity in a bounded time period. Specifically, AMRT improves link utilization by 38%, 28%, 22% and 11% in Data Mining workload with 5000 flows over pHost, ExpressPass, Homa and NDP, respectively.

NDP also obtains high link utilization because in addition to pacing pull packets at the link rate of receiver side, it trims payloads for the packets when queue length becomes large and retransmits them to recover the sending rate after the congestion is alleviated. Compared with pHost, Homa performs better because it uses the overcommitment mechanism to improve link utilization for the scenario that the senders do not respond to the receivers. However, these protocols are hard to make good use of the free bandwidth in the multi-bottleneck or dynamic traffic scenarios.

B. Performance under Non-oversubscribed Topology

We further conduct simulation under non-oversubscribed topology, which contains 10 ToR switches, 8 core switches and 80 end-hosts. Each leaf switch connects to 8 hosts with

100Gbps links. The other simulation settings are same as that in the oversubscribed topology in Section VIII-A.



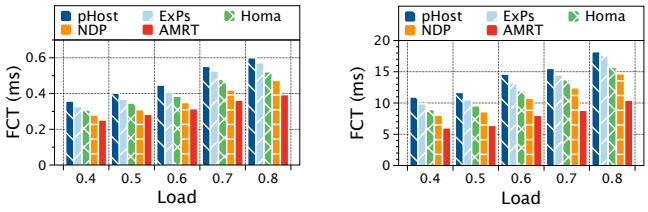
(a) The average FCT (b) The 99^{th}-ile FCT

Fig. 16. FCT for realistic workloads under non-oversubscribe topology.

Fig.16 shows the average and 99^{th}-ile FCT for five realistic workloads. Because congestion only occurs at the last hop, the flows to the same receiver can share grants in all receiver-driven mechanisms. Specifically, once some flows are finished, the other remaining flows can obtain more grants to drive more data packets, making the bottleneck link of the last hop fully utilized. There are also some differences between these receiver-driven mechanisms. For ExpressPass, new flows should wait for credits to start sending packets in the first RTT even though the bottleneck link is under-utilized. For pHost, Homa, NDP and AMRT, they start flows at line rate to make good use of link bandwidth. Homa transmits short flows at the highest priority to achieve low FCT. For pHost, the sender that does not respond with data packets is downgraded once the number of unexpired tokens exceeds a threshold. In brief, as shown in Fig.16 (a) and (b), the AFCT and tail FCT of the these receiver-driven mechanisms have little difference under different workloads.

C. Performance under Fat-tree Topology

Next, we evaluate the effectiveness of AMRT under 12-pod Fat-tree topology. Each pod consists of 6 edge switches, 6 aggregation switches and 84 end-hosts connected. There are 36 parallel paths between any pair of edge switches across pods. The other simulation settings are same as that in Section VIII-A.



(a) The average FCT (b) The 99^{th}-ile FCT

Fig. 17. FCT for Web Search under Fat-tree topology.

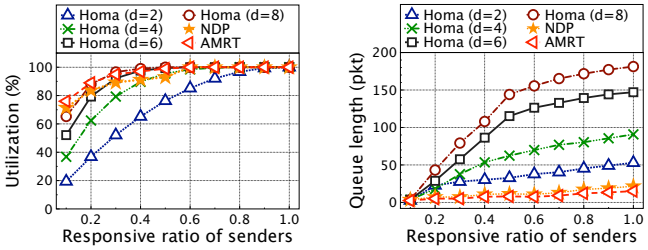
Fig. 17 (a) and Fig. 17 (b) show the average and 99^{th} percentile FCT for Web Search workload under Fat-tree topology, respectively. In this heavy-tailed workload, around 49% of tiny flows less than 10KB and about 20% long flows larger than 1MB cause multi-bottleneck congestion in the fabric. The experimental results show that AMRT outperforms the other receiver-driven schemes and achieves the lowest FCT. The reason is that AMRT can effectively seize the available bandwidth under the dynamic traffic scenarios by actively increasing the sending rate based on the anti-ECN marking. At 0.8 traffic load, compared with pHost, ExpressPass, Homa and NDP, AMRT reduces average FCT and 99^{th}-ile FCT by up to 35%, 32%, 25%,21% and 42%, 39%, 34%, 29%, respectively.

D. Performance in Many-to-many Communications

In many-to-many communication scenario, a source usually establishes multiple connections with multiple destinations at the same time. Similarly, a destination usually connects with multiple sources. If a receiver only sends grants to one sender at a time, the bottleneck link bandwidth will be wasted with unresponsive senders, resulting in poor network utilization.

In AMRT, even if some senders do not respond, the other senders leverage the explicit utilization feedback to increase sending rates to utilize the spare bandwidth without queueing buildup. In this section, we conduct experiment to show that AMRT outperforms the other protocols in many-to-many communication scenarios.

We generate many-to-many communication pattern in a leaf-spine topology with 3 leaf switches to compare AMRT with Homa's overcommitment mechanism. Each of the first two leaf switches connects with 20 senders, and each sender respectively establishes 2 connections with 2 receivers under the third leaf switch. The other simulation settings are same as that in Section VIII-A. We measure the bottleneck link utilization and maximum queue length with increasing responsive ratio of the senders from 0.1 to 1. In this test, we change the degree of overcommitment in Homa from 2 to 8. In AMRT, each receiver sends grants to the corresponding sender according to the received data packets. We repeated the test 50 times to get the average results.



(a) Bottleneck utilization (b) Maximum queue length

Fig. 18. The bottleneck link utilization and the queueing buildups with increasing ratio of responsive senders.

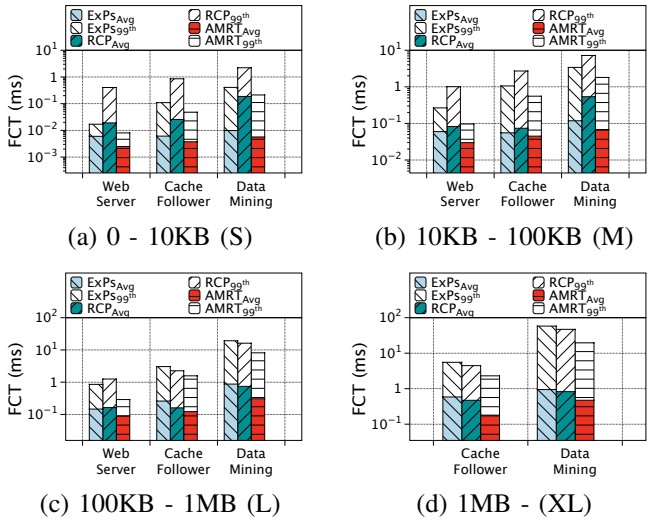
Fig. 18 shows the bottleneck link utilization and maximum queue length with varying responsive ratios of senders. As shown in Fig. 18 (a), compared with Homa, AMRT keeps higher link utilization since it flexibly adjusts the sending rate according to the network state. The key difference is that AMRT only increases the sending rate to match the target rate when the bottleneck link is under-utilized. On the contrary, Homa increases the degree of overcommitment to reduce the likelihood of wasted bandwidth at the cost of consuming more buffer space, causing larger queueing delay. As shown in Fig. 18 (b), when the response ratio of senders is 0.5 and the degree of overcommitment is set to 8, the average link utilization in

Homa is improved almost by 32% compared with degree of 2, but the average queue length also increases by about 4 times.

In brief, it is hard for Homa to achieve high link utilization and low queueing delay simultaneously by using a fixed degree of overcommitment under highly dynamic traffic scenario. AMRT provides high link utilization by reasonably adjusting the sending rate according to the anti-ECN feedback from the bottleneck link and guarantees low latency via conservative receiver-driven transmission.

E. Performance Compared to Switch-centric Solutions

Unlike the switch-centric solutions [24], [25], [48], AMRT is a receiver-driven transport protocol, which conservatively triggers new data packets according to the data arrival rate at the receiver. Specifically, when a data packet arrives at the receiver, a corresponding grant packet is generated and returned to the sender to trigger one new data packet. To improve link utilization due to the conservativeness of the receiver-driven transmission, AMRT uses anti-ECN marked packets to notify the sender of link under-utilization and correspondingly increases sending rate to grab spare bandwidth. Specifically, when the time interval between two consecutive packets is greater than the transmission time of one packet, the switch marks the ECN bit of the dequeued packet. Then the corresponding marked grant will be generated at the receiver to trigger two data packets. For RCP [48], each router assigns a specific rate to all flows that pass through it. In this way, RCP helps the flows to finish faster than XCP [25], which returns the congestion information carried in packets header back to the senders to increase or decrease window size. Thus, we added the results of RCP compared with ExpressPass and AMRT in the oversubscribed topology as same in Section VIII-A.



(a) 0 - 10KB (S) (b) 10KB - 100KB (M)

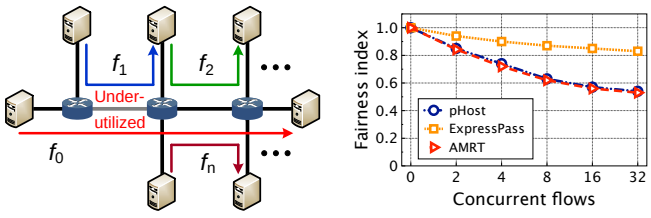
(c) 100KB - 1MB (L) (d) 1MB - (XL)

Fig. 19. The average and 99^{th}-ile FCT for realistic workloads (load 0.6).

Fig.19 shows the average and 99^{th}-ile FCT across three realistic workloads for a target load of 0.6. As shown in Table 1, S, M, L, and XL represent flows with sizes of 0-10KB, 10KB-100KB, 100KB-1MB and larger than 1MB, respectively. AMRT performs better than ExpressPass and RCP across workloads, because they achieve low latency by conservative receiver-driven transmission and high link utilization by anti-ECN marking. Since RCP assigns the same rate for a new flow as existing flows, ExpressPass performs better than RCP for short flows due to low queueing delay and ramp up time. In contrast, ExpressPass performs worse than RCP for long flows due to lower link utilization caused by more credit waste. In brief, AMRT reduces the average FCT by up to 82% and 68% compared to RCP for S and M flows, respectively, and the gap is larger at 99^{th}-ile FCT. For L and XL size flows, AMRT reduces the average FCT by up to 63% and 42% compared to ExpressPass. Since all flows are less than 1MB in Web Server workload, the FCT of Web Server workload is not presented in Fig. 19 (d) where the flow size is larger than 1MB.

F. Performance of Fairness

We conduct a test under multiple bottleneck scenario with different RTTs as shown in Fig. 20 (a). The bottleneck link capacity is 10Gbps, and the base round trip propagation delay is 40μs. The switch buffer size is set to twice bandwidth-delay product (BDP). We vary the number of flows from 2 to 32, which share with the flow f_0 at the second bottleneck link from the left. Fig. 20 (b) shows the Jain's fairness index, which is calculated by using the throughputs of f_0 and f_1 at every 100ms interval and taking the average value. The throughput of f_0 decreases as the number of flows passing through the second bottleneck increases, resulting in more free bandwidth and lower link utilization at the first bottleneck. AMRT marks the data packets for f_0 and f_1 to drive more packets and increase link utilization. However, since f_0 and f_1 have different RTTs, the number of marked packets is proportional to the rate of flows, resulting in much more marked data packets for f_1 than that for f_0. Therefore, although f_0 and f_1 make full use of the link bandwidth, f_1 grabs more bandwidth than f_0 at the first bottleneck, leading to unfairness. As the number of concurrent flows increases, the fairness of AMRT drops. Since pHost cannot actively increase sending rate, the fairness is also decreased with the increase of concurrent flows. ExpressPass achieves better fairness due to credit feedback loop.

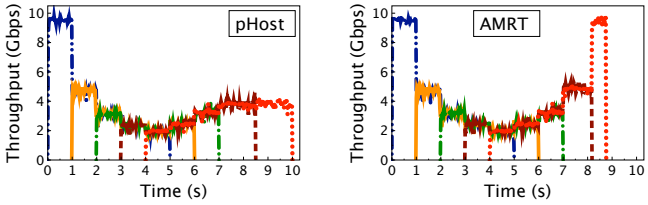


(a) Multi-bottleneck scenario (b) Fairness index

Fig. 20. Fairness under multiple bottlenecks scenario.

If the flow rates are different at the under-utilized bottleneck link, more marked packets for high-speed flows cause unfairness. In the above multiple bottlenecks scenario, to make full use of the available bandwidth as soon as possible, it is reasonable to mark more data packets for high-speed flow to trigger more new data packets. Because even if AMRT marks the flows f_0 and f_1 fairly, the packets from f_0 cannot grab the free bandwidth in time due to congestion at other bottlenecks.

However, if the flows sharing the bottleneck link have the same destination, we can utilize the flow information collected by the receiver to control the number of marked packets for each flow to improve fairness. Next, we run a test with five flows from independent senders to a same receiver over a single bottleneck. When the bottleneck link is under-utilized, AMRT reassign marked packets fairly for each flow at the receiver.



(a) Throughput of pHost (b) Throughput of AMRT

Fig. 21. Fairness under dynamic traffic scenario.

Fig. 21 shows the throughput of each flow averaged over 20ms. When multiple flows compete for the bottleneck link bandwidth with the same RTT from 1s to 7s, pHost and AMRT achieve good per-flow fairness performance. In Fig. 21 (a), since pHost cannot seize the available bandwidth released by the finished flows from other receivers, the throughput loss happens from 7s to 10s. In Fig. 21 (b), AMRT can effectively utilize the spare bandwidth by increasing data packets driven by each marked grant. Since each flow has almost the same marked packets controlled at the receiver, AMRT also achieves good fairness in seizing the spare bandwidth.

IX. Conclusion

We bring the explicit feedback into the proactive congestion control of receiver-driven transport protocol, called as AMRT, which uses anti-ECN marking feedback to indicate under-utilized link and notifies the sender to fill up the spare bandwidth without introducing traffic overhead. The test results of real testbed and large-scale NS2 simulations show that AMRT significantly outperforms pHost, ExpressPass, Homa and NDP by 38%, 28%, 22% and 11% respectively in terms of link utilization. AMRT effectively reduces the AFCT by up to 42% compared with the state-of-the-art receiver-driven transport protocols.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62132022, 62102046, 61872387, 61872403), the Natural Science Foundation of Hunan Province (2021JJ30867, 2022JJ30618), Key Research and Development Program of Hunan (2022WK2005), Science and Technology on Parallel and Distributed Processing Laboratory(PDL) Foundation under Grant (6142110200406).

References

[1] J. Hu, J. Huang, Z. Li, J. Wang and T. He, "AMRT: Anti-ECN Marking to Improve Utilization of Receiver-driven Transmission in Data Center," in Proc. ACM ICPP, 2020.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data Center TCP (DCTCP). In Proc. ACM SIGCOMM, 2010.

[3] Lisheng M, Wei S, et al. Joint Emergency Data and Service Evacuation in Cloud Data Centers Against Early Warning Disasters. IEEE Transactions on Network and Service Management, 2022, vol.19, no. 2, pp. 1306-1320.

[4] Feng H, Deng Y, Zhou Y, et al. Towards Heat-Recirculation-Aware Virtual Machine Placement in Data Centers[J]. IEEE Transactions on Network and Service Management, 2021, vol.19, no. 1, pp. 256-270.

[5] Feng H, Deng Y, Qin X, et al. Criso: An Incremental Scalable and Cost-Effective Network Architecture for Data Centers[J]. IEEE Transactions on Network and Service Management, 2020, vol.18, no. 2, pp. 2016-2029.

[6] Iqbal W, Berral J L, Erradi A, et al. Adaptive prediction models for data center resources utilization estimation[J]. IEEE Transactions on Network and Service Management, 2019, vol.16, no. 4, pp. 1681-1693.

[7] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang. PIAS: Practical Information-Agnostic Flow Scheduling for Commodity Data Centers. IEEE/ACM Transactions on Networking, vol. 25, no. 4, pp. 1954-1967, 2017.

[8] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout. Homa: A Receiver-driven Low-latency Transport Protocol Using Network Priorities. In Proc. ACM SIGCOMM, 2018.

[9] P. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker. phost: Distributed Near-optimal Datacenter Transport over Commodity Network Fabric. In Proc. ACM CoNEXT 2015.

[10] M. Handley, C. Raiciu, A. Agache, A.Voinescu, A. Moore, G. Antichi, and M. Wójcik. Re-architecting Datacenter Networks and Stacks for Low Latency and High Performance. In Proc. ACM SIGCOMM, 2017.

[11] I. Cho, K. Jang, and D. Han. Credit-scheduled Delay-bounded Congestion Control for Datacenters. In Proc. ACM SIGCOMM, 2017.

[12] S. S. Kunniyur. AntiECN Marking: A Marking Scheme for High Bandwidth Delay Connections. In Proc. IEEE ICC, 2003.

[13] W. Bai, L. Chen, K. Chen, and H. Wu. Enabling ECN in Multi-Service Multi-Queue Data Centers. In Proc. USENIX NSDI, 2016.

[14] W. Bai, K. Chen, L. Chen, C. Kim, and H. Wu. Enabling ECN over Generic Packet Scheduling. In Proc. ACM CoNEXT, 2016.

[15] J. Zhang, W. Bai, and K. Chen. Enabling ECN for Datacenter Networks with RTT Variations. In Proc. ACM CoNEXT, 2019.

[16] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, and Y. Zhu. Combining ECN and RTT for Datacenter Transport. In Proc. ACM APNet, 2017.

[17] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware Datacenter TCP (D2TCP). In Proc. ACM SIGCOMM, 2012.

[18] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion Control for Large-scale RDMA Deployments. In Proc. ACM SIGCOMM, 2015.

[19] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats. Timely: RTT-based Congestion Control for the Datacenter. In Proc. ACM SIGCOMM, 2015.

[20] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang. Information-Agnostic Flow Scheduling for Commodity Data Centers. In Proc. USENIX NSDI, 2015.

[21] L. Chen, J. Lingys, K. Chen, and F. Liu. AuTO: Scaling Deep Reinforcement Learning for Datacenter-Scale Automatic Traffic Optimization. In Proc. ACM SIGCOMM, 2018.

[22] Z. Li, W. Bai, K. Chen, D. Han, Y. Zhang, D. Li, and H. Yu. Rate-Aware Flow Scheduling for Commodity Data Center Networks. In Proc. IEEE INFOCOM, 2017.

[23] L. Chen, K. Chen, W. Bai, and M. Alizadeh. Scheduling Mix-flows in Commodity Datacenters with Karuna. In Proc. ACM SIGCOMM, 2016.

[24] J. Zhang, F. Ren, R. Shu, and P. Cheng. TFC: Token Flow Control in Data Center Networks. In Proc. ACM EuroSys, 2016.

[25] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-delay Product Networks. In Proc. ACM SIGCOMM, 2002.

[26] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit Is Enough. In Proc. ACM SIGCOMM, 2005.

[27] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, M. Yu. HPCC: High Precision Congestion Control. In Proc. ACM SIGCOMM, 2019.

[28] J. Hu, J. Huang, Z. Li, Y. Li, W. Jiang, K. Chen, J. Wang and T. He, "RPO: Receiver-driven Transport Protocol Using Opportunistic Transmission in Data Center," in Proc. IEEE ICNP, 2021.

[29] S. Hu, W. Bai, B. Qiao, K. Chen, and K. Tan. Augmenting Proactive Congestion Control with Aeolus. In Proc. ACM APNet, 2018.

[30] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken. The Nature of Data Center Traffic: Measurements & Analysis. In Proc. ACM IMC, 2009.

[31] T. Benson, A. Akella, and D. Maltz. Network Traffic Characteristics of Data Centers in the Wild. In Proc. IMC, 2010.

[32] A. Roy, H. Zeng, J. Bagga, G. Porter, A. C. Snoeren. Inside the Social Network's (Datacenter) Network. In Proc. ACM SIGCOMM, 2015.

[33] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In SIGCOMM, 2008.
[34] A. Greenberg et al. VL2: a scalable and flexible data center network. In SIGCOMM, 2009.
[35] P. Cheng, F. Ren, R. Shu, and C. Lin. Catch the Whole Lot in an Action: Rapid Precise Packet Loss Notification in Data Centers. In Proc. USENIX NSDI, 2014.
[36] J. Xia, G. Zeng, J. Zhang, W. Wang, W. Bai, J. Jiang, K. Chen. Rethinking Transport Layer Design for Distributed Machine Learning. In Proc. ACM APNet 2019.
[37] H. Zhu, D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and M. Erez. Kelp: QoS for Accelerators in Machine Learning Platforms. In Proc. IEEE HPCA, 2019.
[38] A. Eker, B. Williams, K. Chiu, and D. Ponomarev. Controlled Asynchronous GVT: Accelerating Parallel Discrete Event Simulation on Many-core Clusters. In Proc. ACM ICPP, 2019.
[39] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pFabric: Minimal Near-optimal Datacenter Transport. In Proc. ACM SIGCOMM, 2013.
[40] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding Data Center Traffic Characteristics. In Proceedings of Sigcomm Workshop: Research on Enterprise Networks, 2009.
[41] T. Wang, F. Liu, J. Guo, and H. Xu. Dynamic SDN Controller Assignment in Data Center Networks: Stable Matching with Transfers. In Proc. IEEE INFOCOM, 2016.
[42] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren. Re-architecting Congestion Management in Lossless Ethernet. In Proc. USENIX NSDI, 2020.
[43] X. Wang, A. Tumeo, J. D. Leidel, J. Li, and Y. Chen. MAC: Memory Access Coalescer for 3D-Stacked Memory. In Proc. ACM ICPP, 2019.
[44] J. Hu, J. Huang, J. Lv, Y. Zhou, J. Wang, and T. He. CAPS: Coding-based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center. IEEE/ACM Transactions on Networking, vol. 27, no. 6, pp. 2338-2353, 2019.
[45] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan and Y. Wang. Aeolus: A Building Block for Proactive Transport in Datacenters. In Proc. ACM SIGCOMM 2020.
[46] DPDK plane development kit, Intel DPDK, 2019.
[47] Expresspass simulator. https://github.com/kaist-ina/ns2-xpass.
[48] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor Sharing Flows in the Internet. In Proc. IEEE IWQoS, 2005.

Jinbin Hu received the B.E. and M.E. degrees from Beijing Jiao Tong University, China, in 2008 and 2011, respectively, and the PhD degree in computer science from Central South University, China, in 2020. She is currently a Post-Doc in the Department of Computer Science & Engineering, Hong Kong University of Science & Technology, and working in the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. Her current research interests are in the area of datacenter networks and systems.

Jiawei Huang obtained his PhD (2008) and Masters degrees (2004) from the School of Computer Science and Engineering at Central South University. He also received his Bachelor's (1999) degree from the School of Computer Science at Hunan University. He is now a professor in the School of Computer Science and Engineering at Central South University, China. His research interests include performance modeling, analysis, and optimization for wireless networks and data center networks.

Zhaoyi Li received the B.S. degree from Central South University, China, in 2019. He is currently pursuing the M.S. degree in the School of Computer Science and Engineering at Central South University, China. His research interests are in the area of data center networks.

JianXin Wang received the B.E. and M.E. degrees in computer engineering from Central South University, China, in 1992 and 1996, respectively, and the PhD degree in computer science from Central South University, China, in 2001. He is the chair of and a professor in the School of Computer Science and Engineering, Central South University, Changsha, Hunan, P.R. China. His current research interests include algorithm analysis and optimization, parameraized algorithm, Bioinformatics and computer network. He is a senior member of IEEE.

Tian He received the PhD degree under Prof. John A. Stankovic from the University of Virginia, Charlottesville in 2004. He is currently a professor with the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. His research includes wireless sensor networks, cyber-physical systems, intelligent transportation systems, real-time embedded systems and distributed systems, supported by the US National Science Foundation, IBM, Microsoft and other agencies. He is the author and coauthor of over 200 papers in journals and conferences with over 20,000 citations (H-Index 59). His publications have been selected as graduate-level course materials by over 50 universities in the United States and other countries. He has received a number of research awards in the area of networking, including five best paper awards. He is also the recipient of the NSF CAREER Award 2009 and McKnight Land-Grant Professorship. He served a few program chair positions in international conferences and on many program committees, and also currently serves as an editorial board member for six international journals including IEEE Transactions on Computer. He is an ACM and IEEE fellow.