AMRT: Anti-ECN Marking to Improve Utilization of Receiver-driven Transmission in Data Center

Jinbin Hu
Central South University
Changsha, China
jinbinhu@csu.edu.cn

Jiawei Huang
Central South University
Changsha, China
jiaweihuang@csu.edu.cn

Zhaoyi Li
Central South University
Changsha, China
lizhaoyi@csu.edu.cn

Jianxin Wang
Central South University
Changsha, China
jxwang@csu.edu.cn

Tian He
University of Minnesota
Minneapolis, MN, USA
tianhe@umn.edu

ABSTRACT

Cloud applications generate a variety of workloads ranging from delay-sensitive flows to bandwidth-hungry ones in data centers. Existing reactive or proactive congestion control protocols are hard to simultaneously achieve ultra-low latency and high link utilization across all workloads in data center networks. We present a new receiver-driven transport scheme using anti-ECN (Explicit Congestion Notification) marking to achieve 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 experimental results of small-scale testbed implementation and large-scale NS2 simulation show that AMRT effectively reduces the average flow completion time (AFCT) by up to 40.8% and improves the link utilization by up to 36.8% under high workload over the state-of-the-art receiver-driven transmission schemes.

CCS CONCEPTS

• **Networks** → **Data center networks**; **Transport protocols**.

KEYWORDS

Data center, receiver-driven, link utilization

ACM Reference Format:
Jinbin Hu, Jiawei Huang, Zhaoyi Li, Jianxin Wang, and Tian He. 2020. AMRT: Anti-ECN Marking to Improve Utilization of Receiver-driven Transmission in Data Center. In *49th International Conference on Parallel Processing - ICPP (ICPP '20), August 17–20, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404397.3404412

1 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 [1–6]. 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) [7].

In recent years, receiver-driven transport protocols such as pHost [8], NDP [10], Homa [7] and Aeolus [11] are 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. 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 co-existing 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 a 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.

To improve link utilization in many-to-many communication scenarios, Homa uses the overcommitment mechanism to allow multiple senders to simultaneously respond to a single receiver. 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 (Section 8.2), the overcommitment mechanism easily causes queueing buildup under the highly dynamic workloads, leading to poor latency performance.

Fortunately, the marking-based explicit feedback is an effective mechanism to address the above problem. We propose a new receiver-driven transport protocol called AMRT, which 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 dequeued packet, whose corresponding marked grant will be generated at the receiver to trigger more data packets. Then the receiver-driven sender increases its sending rate to match the available bandwidth. Therefore, AMRT takes advantage of conservative receiver-driven transmission to guarantee near-zero queueing delay and meanwhile ensures full link utilization with the aid of explicit anti-ECN marking.

In summary, our major contributions are:

- 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. For example, as shown in Section 8, pHost, Homa and NDP only obtain about 61%, 68% and 75% average link utilization under the dynamic data mining workload, respectively.
- We propose a new receiver-driven transport protocol 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. Moreover, AMRT can be easily deployed on the commercial switches using the built-in ECN function.
- 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 18.3%-40.8% under heavy workload and yields up to 36.8%, 22.5% and 11.6% link utilization improvement over pHost, Homa and NDP, respectively.

The rest of the paper is organized as following. In Section 2 and 3, we respectively describe our design motivation and overview. In Section 4 and 5, we introduce the design details and model analysis of AMRT, respectively. We discuss the implementation in Section 6. In Section 7 and 8, we show the testbed experimental and NS2 simulation results, respectively. In Section 9, we present the related work and then conclude the paper in Section 10.

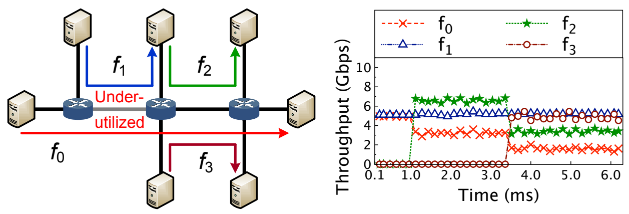
2 DESIGN MOTIVATION

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

2.1 Multiple bottlenecks scenario

The multiple bottlenecks scenarios widely exist in data center networks [12], [13], [14]. 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 [15], [16], [17], [18]. These cross-rack flows coexist with a variable number of cross flows at each hop, resulting in multiple bottlenecks [19], [20].

It is common that a flow traverses multiple bottlenecks and coexists with a variable number of cross flows at each bottleneck. Unfortunately, the conservativeness of current receiver-driven transmission easily leads to low link utilization. Since the receiver-driven transport protocol only generates one grant corresponding to each arrival packet at the receiver, 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 loss

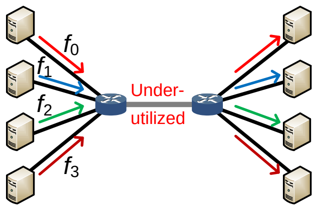
Figure 1: Multiple bottlenecks.

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 $100\mu s$. The switch buffer size is 128 packets.

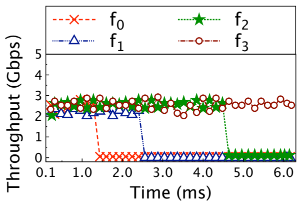
Fig. 1 (b) shows that, at the beginning, f_0 and f_1 fairly share the 1st bottleneck link (from the left) without queue buildup. At 1ms, f_2 starts at the line rate of 10Gbps. Consequently, the sending rate of f_0 decreases to 3.3Gbps due to the bandwidth competing with f_2 at the 2nd bottleneck. Thus, 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 3.5ms, the sending rate of f_0 decreases again due to the starting of f_3. Finally, the link utilization of the 1st bottleneck link is only about 66%.

2.2 Dynamic traffic scenario

Due to the continuous and instantaneous changing of traffic behavior, the highly dynamic is an intrinsic feature of data center network [21], [22], [23]. Recent data center traffic studies show that traffic fluctuates frequently over time and space [12], [13], [14]. 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 almost concurrently to exchange data among servers, incurring bursty transient traffic [1], [9]. 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.



(a) Dumbbell topology



(b) Throughput loss

Figure 2: Dynamic traffic.

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 2.1. 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. Firstly, four flows share a bottleneck at the same rate and just make full use of the link capacity. However, when f_0 is completed, the bottleneck link is not saturated by the remaining three 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.

2.3 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.

3 DESIGN OVERVIEW

In this section, we present an overview of AMRT. 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. 3.

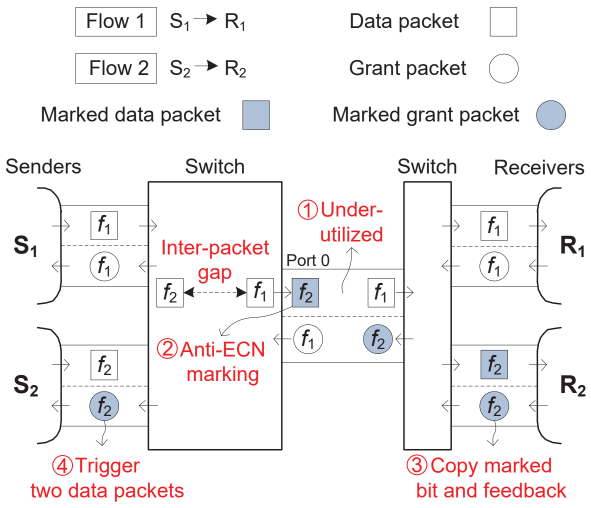


Figure 3: AMRT overview.

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. 3, 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.

4 DESIGN DETAILS

In this section, 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.

4.1 Packet Interval Estimation and Anti-ECN Marking

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 [24]. 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 [25].

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 [16]. The value of t_{inv} is calculated as

$$t_{inv} = t_{current} - t_{last}, \tag{1}$$

where $t_{current}$ and t_{last} are the dequeue times of the current and last packets at the egress port, respectively.

If the packet's inter-dequeue time t_{inv} is greater than the transmission time of one packet, then the link is deemed under-utilized and the Congestion Experienced (CE) codepoint bit $CE_{current}$ of the arriving packet is set to 1. Otherwise, the bottleneck link is saturated. In this case, since there is no need to add more packets, the value of $CE_{current}$ is set to 0. That is,

$$\begin{cases} CE_{current} = 1 & t_{inv} \geq \frac{MSS}{C}, \\ CE_{current} = 0 & t_{inv} < \frac{MSS}{C}, \end{cases} \tag{2}$$

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 initially value of CE bit is set to1 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. Therefore, AMRT obtains the final marked value CE_{final} of one dequeued packet as the "AND" operation result of the piggybacking value CE_{last} of the dequeued packet and the value of $CE_{current}$ at the current switch. Thus, we have

$$CE_{final} = CE_{current} \,\&\, CE_{last}. \tag{3}$$

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 (2), 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.

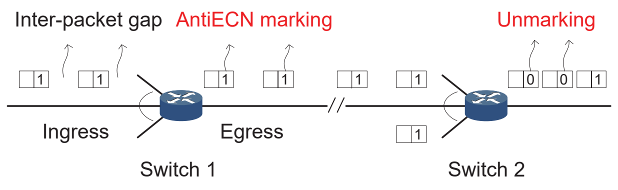


Figure 4: Anti-ECN marking operation of AMRT.

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. Note that since the marking operation is made on packet level and is independent of the flow, the switches do not need to maintain per-flow state information. 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.

4.2 Grant Generation and Explicit Feedback

At the receiver, AMRT uses the existing receiver-driven transmission mechanism for 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. 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 codepoint 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.

4.3 Receiver-driven Rate Adjustment

At the sender, 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-utilization feedback at the bottleneck link to achieve a tradeoff between proactive and reactive congestion control.

Specifically, if 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.

5 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.

Assuming that n packets arrive at the switch back-to-back in one round trip time (RTT), and the output rate matches the bottleneck link capacity. In the next RTT, we assume that there are k vacant positions among the remaining $n-k$ packets after some flows finish transmission.

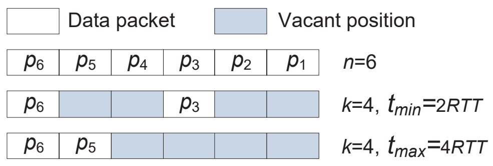


Figure 5: A simple example shows the time required for AMRT to fill up the spare bandwidth when some packets release bandwidth.

Here, we analyze the convergence time when AMRT fills up the spare bandwidth of bottleneck link. As shown in Fig. 5, the bottleneck link is saturated by 6 back-to-back packets (i.e. n=6) in the 1st RTT. We assume that 4 packets are not sent in the 2nd 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 make the sender to send one more packet in the next RTT. Thus, when the four vacant positions are evenly distributed, AMRT needs the minimal time of 2 RTTs to fill up the spare bandwidth. On the contrary, when the four vacant positions are consecutive, the maximum time to achieve the full link utilization is 4 RTTs.

Without loss of generality, when k spare positions are evenly distributed among $n-k$ packets, we get the minimum time t_{min} for AMRT to achieve the full link capacity as

$$t_{min} = \lceil \frac{k}{n-k} \rceil \times RTT, \tag{4}$$

where RTT is the minimum round trip time.

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

$$t_{max} = k \times RTT. \tag{5}$$

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. 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.

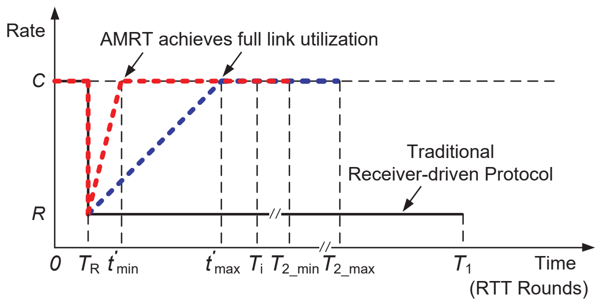


Figure 6: AMRT vs. Traditional receiver-driven protocol.

As shown in Fig. 6, 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

$$T_1 = \frac{S - C \times T_R}{R} + T_R, \tag{6}$$

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. 6, 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

$$t'_{min} = \lceil \frac{C-R}{R} \rceil + T_R. \tag{7}$$

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

$$t'_{max} = C - R + T_R. \tag{8}$$

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

$$S = C \times T_R + \frac{1}{2} \times (R+C) \times (t' - T_R) + C \times (T_2 - t'), \tag{9}$$

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

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

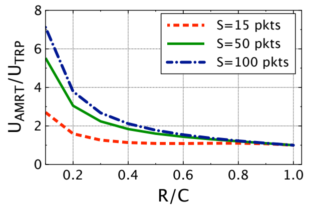
As shown in Fig. 6, for t'_{min} and t'_{max}, the corresponding values of T_2 are T_{2_min} and T_{2_max}, respectively.

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

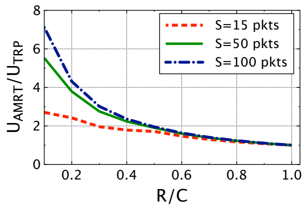
$$U_{gain} = \frac{U_{AMRT}}{U_{TRP}} = \frac{T_1}{T_2} = \frac{\frac{S - C \times T_R}{R} + T_R}{\frac{S - C \times T_R - \frac{1}{2} \times (R+C) \times (t' - T_R)}{C} + t'}. \tag{11}$$

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

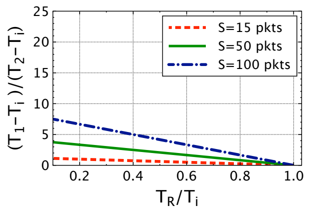
$$FCT_{gain} = \frac{T_1 - T_i}{T_2 - T_i} = \frac{\frac{S - C \times T_R}{R} + T_R - \frac{S}{C}}{\frac{S - C \times T_R - \frac{1}{2} \times (R+C) \times (t' - T_R)}{C} + t' - \frac{S}{C}}. \tag{12}$$



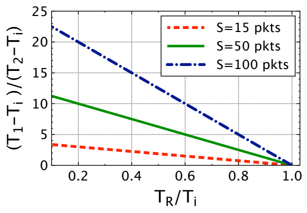
(a) Minimum utilization gain



(b) Maximum utilization gain



(c) Minimum FCT gain



(d) Maximum FCT gain

Figure 7: Minimum and Maximum utilization gain and FCT gain with increasing $\frac{R}{C}$ and $\frac{T_R}{T_i}$.

By substituting t'_{max} and t'_{min} into Equation (11) and Equation (12), we can obtain the minimum and maximum gains of link utilization and FCT of AMRT as shown in Fig. 7. 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. 7 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. 7 (a) and (b) show that as the ratio of $\frac{R}{C}$ decreases, the utilization gain for AMRT increases considerably. Moreover, AMRT performs better with larger flow size. Fig. 7 (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.

6 IMPLEMENTATION

We implement AMRT with three key considerations. The first one is how AMRT works well in many-to-many communication pattern, which may cause low link utilization due to unresponsive senders. In this scenario, a sender establishes connections with multiple receivers, while other senders also establish connections with these receivers at the same time.

For pHost, a receiver assigns tokens to one sender by using the shortest remaining processing time first (SRPT) policy. If the receiver does not receive a certain number of packets from the sender in succession, the receiver assigns tokens to other senders and stops sending tokens to the unresponsive sender for a short timeout period (default being 3×RTT [8]). This process potentially results in poor network utilization.

Homa uses the overcommitment mechanism to allow a receiver grants simultaneously to a few senders and each of sender is able to send a bandwidth-delay product (BDP) bytes [7]. However, the overcommitment mechanism easily leads to queue buildup when the senders simultaneously start transmissions.

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. The corresponding experiment results in Section 8.2 show that AMRT outperforms the other protocols in many-to-many communication scenarios.

The second consideration is how AMRT handles packet drops. In our design, 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 has not arrived at the receiver within a time period (1×RTT by default), the receiver reissues a grant packet for the lost packet to ask the sender to retransmit the data packet.

The last key point is how to ensure ultra-low latency when a new flow starts transmission. In order to avoid wasting bandwidth, the existing receiver-driven transport protocols start a new flow immediately without waiting for grants from the receiver. However, the blindly transmission for a new flow is hard to guarantee the ultra-low queueing delay especially under the high concurrency situations [11]. In our design, similar to NDP [10], the queue length at switch buffer is limited to a very small threshold (i.e., 8 packet [9],[10]). AMRT directly drops the packets beyond this threshold to maintain ultra-low latency.

7 TESTBED EVALUATION

In this section, we use a real testbed to evaluate the feasibility and effectiveness of AMRT. The testbed consists of 13 servers, each of which has a Intel Core Xeon(R) 3.00 GHz CPU and 32GB memory. The servers run CentOS 7.5 with Linux 3.10.0-862.el7.x86_64 kernel and are equipped with Intel Corporation 82580 Gigabit Ethernet Network Interface Cards (NICs). Eight servers are connected to five servers acting as switches, each of which has 1GbE quad-ports I350-T4 NICs.

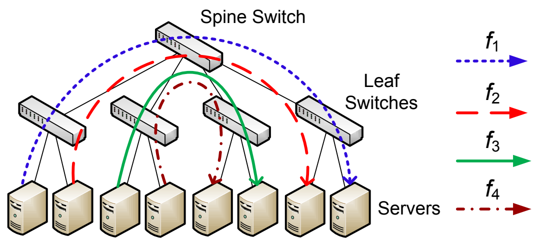
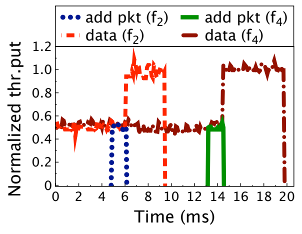


Figure 8: Testbed topology for dynamic traffic scenario.

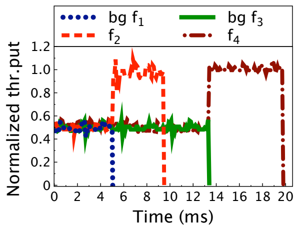
We firstly test whether AMRT successfully grabs the spare bandwidth under the dynamic traffic scenario. We create the test topology shown in Fig. 8. Flow f_1 and f_2 sharing a single bottleneck link are sent to two receivers, respectively. Similarly, flow f_3 and f_4 share another bottleneck link. We show the normalized throughput to the bottleneck link bandwidth in Fig. 9.

We run a test with four flows initiated at the same time. At the beginning, f_1 and f_2, f_3 and f_4 fairly share the two bottleneck links, respectively. Fig. 9 (a) shows the normalized throughputs of f_2 and f_4. When the background flow f_1 is finished at about 5ms, f_2 adds new packets to fully use the spare bandwidth within about 2ms. For the same reason, from about 13ms to 15ms, the added packets driven by the marked grant packets in f_4 approximately take a half bandwidth of the bottleneck link released by the completed flow

f_3. We show the normalized throughput of four flows in Fig. 9 (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

Figure 9: 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. 10. Flow f_1 experiences two bottlenecks, which are shared with f_2 and f_3, respectively. In addition, f_3 and f_4 share a single bottleneck link.

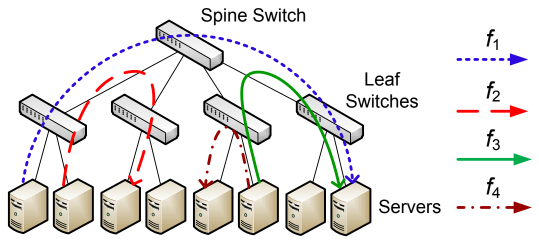
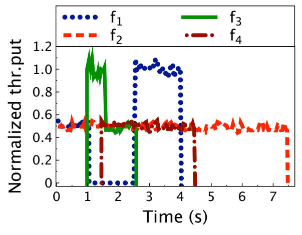


Figure 10: Testbed topology for multi-bottleneck scenario.

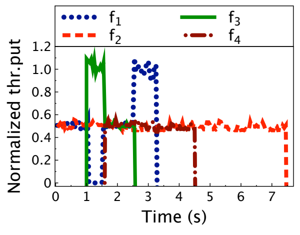
Fig. 11 shows the normalized throughput of pHost, Homa, NDP and AMRT over time. At the beginning, f_1 and f_2 fairly share the bottleneck link. Fig. 11 (a) and (b) show that, when a new flow f_3 with the same destination as f_1 starts at 1.0s, f_3 gets full link capacity in pHost and Homa due to the shortest remaining processing time (SRPT) policy. In Fig. 11 (c) and (d), since f_3 starts transmission with the link rate, f_1 and f_3 share the bottleneck link with a rate ratio of about 1:2 after flow competition.

Therefore, after f_3 starts, f_1 releases bandwidth at the first bottleneck link under the four receiver-driven schemes. In pHost, Homa and NDP, the spare bandwidth released by f_1 at the first bottleneck link is wasted. The reason is that, even if the sending rate of f_1 drops, the receiver-driven flow f_2 still triggers new data packet according to the arrival rate of data packet at its receiver. The f_2 sender can not get more grants to increase sending rate to utilize the spare bandwidth released by f_1. Only AMRT is able to flexibly grab the spare bandwidth by the anti-ECN marking feedback mechanism. Specifically, between 1.0s and 3.1s, f_2 in AMRT increases the normalized throughput from 50% to around 66% compared with the other protocols.

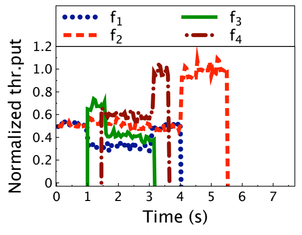
Fig. 11 (a) and (b) show that, after f_3 finishes at 3.1s, due to the conservativeness of pHost and Homa, f_4 is not able to get more grants to trigger more data packets and utilize the free bandwidth released by f_3. In Fig. 11 (c), since NDP cuts payloads of the packets when the queue length exceeds a given threshold instead of dropping packets, the sending rates of f_4 is recovered to the bottleneck link capacity. Fig. 11 (d) shows that f_4 in AMRT also achieves full bottleneck bandwidth by increasing sending rate based on the anti-marking feedback mechanism.



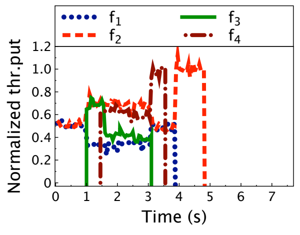
(a) pHost



(b) Homa



(c) NDP



(d) AMRT

Figure 11: Throughput of four receiver-driven transmission schemes under multi-bottleneck scenario.

Similarly, after f_1 finishes transmission, the rate of f_2 achieves the link capacity only in NDP and AMRT. Compared with pHost and Homa, NDP achieves about 26.6% and 20% reduction at FCT for f_2 and f_4, respectively. For AMRT, the rapid reaction to the under-utilization greatly improves the network efficiency. For example, AMRT reduces the FCT of flow f_2 by ~36%, ~36% and ~12.7% 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. 11 (a) shows that f_1 stops transmission until f_3 finished as pHost adopts the SRPT policy. In Fig. 11 (b), when the rate of f_3 drops to half of the bottleneck bandwidth at the beginning of f_4, f_1 gets 50% of the link capacity due to overcommitment mechanism in Homa. Therefore, Homa reduces the FCT of flow f_1 by 17.5% compared with pHost. However, because the free bandwidths released by f_1 and f_3 are not utilized by f_2 and f_4, respectively, the link utilization is still lower than AMRT.

In summary, the test results of testbed experiments show that AMRT generally outperforms pHost, Homa and NDP. With anti-ECN marking method and explicit feedback to adjust sending rate, AMRT significantly achieves higher throughput and reduces the flow completion time.

8 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 Incast scenarios.

8.1 Performance under realistic workloads

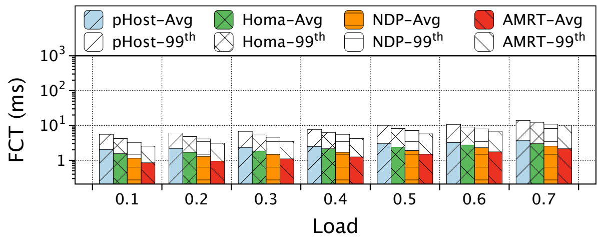
We perform NS2 simulations to evaluate AMRT performance on a large-scale network topology with varying workloads under the typical datacenter scenarios. We measure the flow completion time (FCT), 99^{th} percentile FCT and link utilization of AMRT, pHost, Homa and NDP.

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 10Gbps links. Each network link has a propagation delay of $100\mu s$. The switch buffer size is set to128 packets. We employ Equal Cost Multi Path (ECMP) mechanism to support multipath routing.

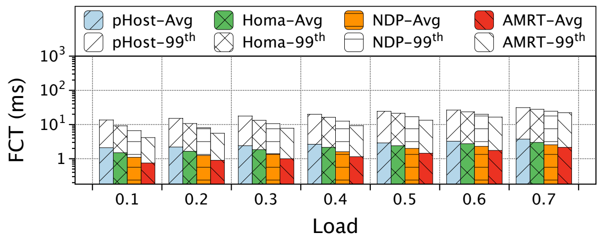
Traffic workloads: We use five workloads with the same distributions as the realistic ones, including web server (WSv), cache follower (CF), hadoop cluster (HC), web search (WSc) and data mining (DM) [7], [8], [9], which cover a wide range of average flow sizes ranging from 64KB to 7.41MB and more than half of the flows are less than 10KB. Specifically, in web server application, except for tiny flows smaller than 10KB, the size of the other flows is uniformly distributed from 10KB to 1MB, resulting in the smallest average flow size. While in the other four workloads, the flow size distributions are heavy-tailed with about more than 90% bytes contributed by the small fraction of large flows. We generate the traffic between randomly selected source and destination hosts. The flow arrival follows a Poisson process and the traffic load is changed from 0.1 to 0.7.

Fig. 12 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 ~40.8%, ~26.4%, ~18.3% and ~43.6%, ~32.5%, ~21.5% at 0.7 load over pHost, 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.

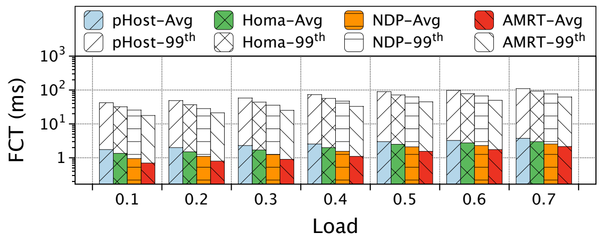
We also measure the bottleneck link utilization with increasing number of flows. As shown in Fig. 13, AMRT significantly outperforms the other receiver-driven mechanisms, because AMRT increases sending rates once detecting the spare bandwidth and thus achieves high rate close to the link capacity in a bounded time period. Specifically, AMRT improves link utilization by ~36.8%, ~22.5%, ~11.6% in data mining workload with 800 flows over pHost, 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



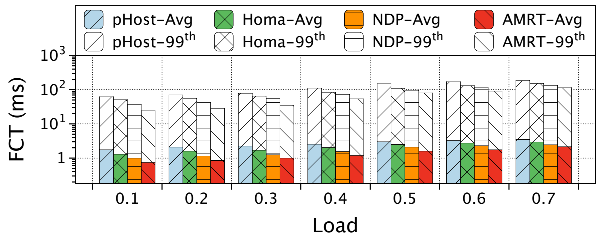
(a) Web Server



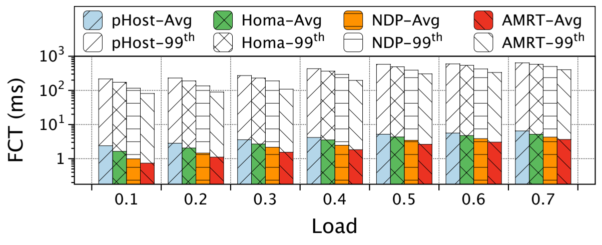
(b) Cache Follower



(c) Hadoop Cluster



(d) Web Search



(e) Data Mining

Figure 12: The average FCT and 99^{th} percentile FCT of all flows for four different receiver-driven transport protocols with increasing load under five realistic workloads.

large and retransmits them to recover the sending rate after the congestion is alleviated. Compared with pHost, Homa performs

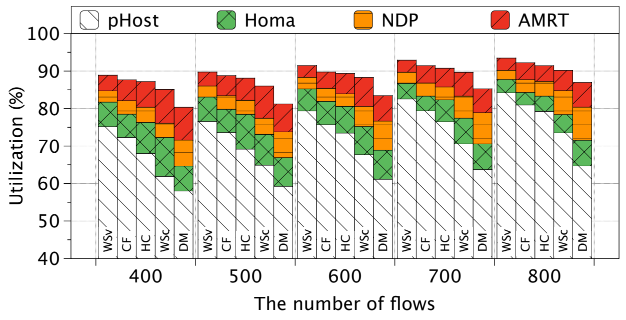


Figure 13: The bottleneck utilizations of four different receiver-driven transport protocols with varying number of flows under five realistic workloads. Specifically, WSv, CF, HC, WSc and DM stand for Web Server, Cache Follower, Hadoop Cluster, Web Search and Data Mining, respectively.

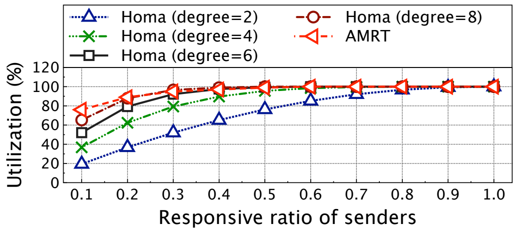
better because it uses the overcommitment mechanism to improve link utilization for the scenario that the senders do not respond to the receivers. However, these protocols are hard to make good use of the free bandwidth in the multi-bottleneck or dynamic traffic scenarios.

8.2 Performance in Many-to-many Communications

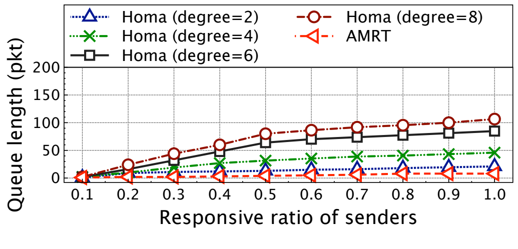
In many-to-many communication scenario, a source usually establishes multiple connections with multiple destinations at the same time. Similarly, a destination usually connects with multiple sources. If a receiver only sends grants to one sender at a time, like pHost, the bottleneck link bandwidth will be wasted with unresponsive senders, resulting in poor network utilization especially under highly dynamic traffic scenario. To address under-utilization problem in this case, Homa employs the overcommitment mechanism to allow a receiver grants multiple senders simultaneously (i.e., degree of overcommitment). 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.

In this section, 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 8.1. 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.

Fig. 14 shows the bottleneck link utilization and maximum queue length with varying responsive ratios of senders. As shown in Fig. 14 (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. 14 (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.



(a) Bottleneck utilization



(b) Maximum queue length

Figure 14: The bottleneck link utilization and the queueing buildups with increasing ratio of responsive senders.

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.

9 RELATED WORKS

In recent years, a variety of transport designs are proposed to obtain low latency or high throughput in data centers. In this section, we compare our design AMRT with the sender-based, rate allocation, explicit flow control and receiver-driven mechanisms.

DCTCP [1] makes use of the ECN marking ratio to adjust the congestion window to ensure low queueing delay and high throughput. D^2TCP [2] and D^3 [26] adjust transmission rate for deadline-sensitive flows to maximize the deadline-meeting rate. HULL [27] maintains near-zero buffer occupancy by using phantom queues to provide low flow completion time. To avoid packet losses, DCQCN [4] uses a fine-grained and end-to-end congestion control scheme to adjust sending rate. TIMELY [5] leverages the changes in RTT as a congestion signal to reduce queuing delay. 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. PDQ [28] calculates flow rate to implement preemptive flow scheduling to minimize flow completion time. pFabric [3] decouples flow scheduling from rate control and schedules packets based on the strict priorities at switches. Fastpass [29] uses a centralized controller to determine the scheduling time and transmission path for each packet. Karuna [30] focuses on how to schedule a mix of flows with and without deadline to achieve performance benefits for all types of traffic. TFC [20] explicitly allocates tokens to active flows to achieve near-zero queue occupancy. By shaping the flow of credit packets at the switch, ExpressPass [9] effectively controls congestion even before sending data. However, these rate control schemes require the rate calculation and global scheduling, which potentially incur delay overhead and performance degradation especially for short and tiny flows.

Lots of transport protocols accurately adjust the sending rate to match the bottleneck link capacity by using explicit feedback information from switches. XCP [24] uses explicit and precise congestion feedback from the switches with multiple bits to regulate congestion window. VCP [31] leverages two ECN bits to carry network congestion information to dynamically adjust the congestion window. Recently, HPCC [16] leverages in-network telemetry (INT) technique to obtain precise link load information at the switches and then sends the load information back to the sender to control traffic precisely. However, it is hard for these transport protocols to make a tradeoff between low feedback overhead and accurate rate adjustment.

Recent receiver-driven transport protocols are proposed to enhance network performance in terms of ultra-low queueing delay and near-zero packet loss in data centers. pHost [8] performs distributed per-packet scheduling at the end hosts to optimize flow performance. NDP [10] starts flows at full rate, cuts payloads for the packets exceeding the given queue length threshold and uses a receiver-pulled mechanism to control incoming traffic. Homa [7] uses a receiver-driven flow control mechanism and in-network priority queues to provide good performance especially for short messages. Aeolus [11] assigns a higher drop priority for aggressive unscheduled packets to maintain lossless for scheduled 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.

10 CONCLUSION

We propose a new receiver-driven transport protocol AMRT that 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, Homa and NDP by 36.8%, 22.5% and 11.6% respectively in terms of link utilization. In addition, AMRT effectively reduces the average flow completion time by up to 40.8% compared with the state-of-the-art receiver-driven transport protocols.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61872387, 61572530, 61872403), CERNET Innovation Project (Grant No. NGII20170107), Project of Foreign Cultural and Educational Expert (G20190018003).

REFERENCES

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, et al. Data center TCP (DCTCP). In Proc. ACM SIGCOMM, 2010.
[2] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter TCP (D2TCP). In Proc. ACM SIGCOMM, 2012.
[3] 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.
[4] Y. Zhu, H. Eran, D. Firestone, et al. Congestion control for large-scale RDMA deployments. In Proc. ACM SIGCOMM, 2015.
[5] R. Mittal, V. T. Lam, N. Dukkipati, et al. Timely: Rtt-based congestion control for the datacenter. In Proc. ACM SIGCOMM, 2015.
[6] A. Rucker, T. Swamy, M. Shahbaz, K. Olukotun. Elastic RSS: Co-Scheduling Packets and Cores Using Programmable NICs. In Proc. ACM APNet, 2019.
[7] 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.
[8] 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.
[9] I. Cho, K. Jang, and D. Han. Credit-scheduled delay-bounded congestion control for datacenters. In Proc. ACM SIGCOMM, 2017.
[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] S. Hu, W. Bai, B. Qiao, K. Chen, and K. Tan. Augmenting Proactive Congestion Control with Aeolus. In Proc. ACM APNet 2018.
[12] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken. The nature of data center traffic: measurements & analysis. In Proc. ACM IMC, 2009.
[13] T. Benson, A. Akella, and D. Maltz. Network traffic characteristics of data centers in the wild. In Proc. IMC, 2010.
[14] A. Roy, H. Zeng, J. Bagga, G. Porter, A. C. Snoeren. Inside the Social Network's (Datacenter) Network. In Proc. ACM SIGCOMM, 2015.
[15] J. Xia, G. Zeng, J. Zhang, et al. Rethinking Transport Layer Design for Distributed Machine Learning. In Proc. ACM APNet 2019.
[16] Y. Li, R. Miao, H. H. Liu, et al. HPCC: high precision congestion control. In Proc. ACM SIGCOMM, 2019.
[17] 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.
[18] 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.
[19] 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.
[20] J. Zhang, F. Ren, R. Shu, and P. Cheng. TFC: token flow control in data center networks. In Proc. ACM EuroSys, 2016.
[21] 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.
[22] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren. Re-architecting Congestion Management in Lossless Ethernet. In Proc. USENIX NSDI, 2020.
[23] 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.
[24] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for high bandwidth-delay product networks. In Proc. ACM SIGCOMM, 2002.
[25] S. S. Kunniyur. AntiECN Marking: A Marking Scheme for High Bandwidth Delay Connections. In Proc. IEEE ICC, 2003.
[26] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: Meeting deadlines in datacenter networks. In Proc. ACM SIGCOMM, 2011.
[27] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda. Less is more: trading a little bandwidth for ultra-low latency in the data center. In Proc. USENIX NSDI, 2012.
[28] C. Y. Hong, M. Caesar, and P. B. Godfrey. Finishing Flows Quickly with Preemptive Scheduling. In Proc. ACM SIGCOMM, 2012.
[29] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal. Fastpass: A centralized "zero-queue" datacenter network. In Proc. ACM SIGCOMM, 2014.
[30] L. Chen, K. Chen, W. Bai, and M. Alizadeh. Scheduling Mix-flows in Commodity Datacenters with Karuna. In Proc. ACM SIGCOMM, 2016.
[31] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit Is Enough. In Proc. ACM SIGCOMM, 2005.