Accurate Available Bandwidth Measurement with Packet Batching Mitigation for High Speed Networks

Vincent Tran[1], Jean Tourrilhes[2], K. K. Ramakrishnan[1], Puneet Sharma[2]
[1]*University of California, Riverside;* [2]*Hewlett Packard Enterprise*

2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) | 978-1-6654-4579-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/LANMAN52105.2021.9478812

***Abstract*—Measuring the Available Bandwidth (ABW) is an important function for traffic engineering, and in software-defined metro and wide-area network (SD-WAN) applications. Because network speeds are increasing, it is timely to re-visit the effectiveness of ABW measurement again. A significant challenge arises because of Interrupt Coalescence (IC), that network interface drivers use to mitigate the overhead when processing packets at high speed, but introduce packet batching. IC distorts receiver timing and decreases the ABW estimation. This effect is further exacerbated with software-based forwarding platforms that exploit network function virtualization (NFV) and the lower-cost and flexibility that NFV offers, and with the increased use of poll-mode packet processing popularized by the Data Plane Development Kit (DPDK) library.**

We examine the effectiveness of the ABW estimation with the popular probe rate models (PRM) such as PathChirp and PathCos++, and show that there is a need to improve upon them. We propose a modular packet batching mitigation that can be adopted to improve both the increasing PRM models like PathChirp and decreasing models like PathCos++. Our mitigation techniques improve the accuracy of ABW estimation substantially when packet batching occurs either at the receiver due to IC, DPDK based processing or intermediate NFV-based forwarding nodes. We also show that our technique helps improve estimation significantly in the presence of cross-traffic.

***Keywords*—Available Bandwidth Measurement, Interrupt Coalescence, Mitigation Technique, Network Congestion.**

I. Introduction

The Available Bandwidth of an end-to-end path is the minimum amount of its available capacity of all links at a given time [1]. ABW measurement plays an important role in traffic engineering applications such as admission control [2], SD-WAN [3], congestion control [4], and network capacity reconfiguration [5]. Bandwidth estimation [6] uses active probing to measure ABW of a network path. This problem has been studied extensively over many years. However, with the increasing use of software-based forwarding (e.g., using NFV [7]), and with link speeds going higher and higher, we find it worthwhile to re-examine this well-established issue.

One of the main challenges for bandwidth estimation on high speed network paths is Interrupt Coalescence (IC) [8]. IC is implemented in the NIC driver, it is needed to increase performance and reduce the overhead when processing received packets [9]. IC is also implemented in most software forwarding devices, especially those based on DPDK [10] or OpenNetVM(ONVM) [7]. IC adversly impacts TCP congestion control [11] and increase bandwidth estimation error [8].

Different ABW estimation methods have been proposed over the years [12]. In this work, we select a few of the most efficient methods, which have relatively short chirp train and relatively good accuracy. PathChirp [13] is very popular and a good representative of methods based on the Probe Rate Model (PRM) with an increasing rate chirp train. PathCos++ [14] and SLDRT [15] are part of a new class of methods based on the PRM with an decreasing rate chirp train, and greatly outperform older methods.

Some techniques have been proposed to mitigate the effect of IC on estimation methods [8], [16], and as results, to varying degrees of effectiveness. Those mitigation techniques can also be used for other types of measurements [8] or TCP congestion control [16]. The new class of decreasing rate methods have not been tested in depth in the presence of IC and on high speed links. Moreover, the benefit of the IC mitigation techniques has not been studied in context of this new class of methods. Finally, only effect of the IC in the receiver has been studied, the effect of IC in nodes within the network path has not been explored.

In this paper we evaluate the effectiveness of the PRM increasing and decreasing methods with high speed (1 Gbps and 10 Gbps) links, especially with end-systems and intermediate router hops performing IC. Based on our observation that they are not effective, we describe our framework for mitigation of the impact of IC for the PRM techniques, that include a Batching Train Modifier at the sender that cooperatively works with a Batching Detection component in the receiver to adjust the size of the probe packet train. Additionally, our framework includes a novel Packet Batching Filter technique at the receiver called Max-IAT, this technique identify packet batches based on maximums in Inter-arrival Time, before the PRM methods is used to estimate the ABW. Our paper compares different increasing and decreasing probe rate methods with the mitigation technique to accurately estimate the ABW.

II. Background and Related Work

A. Interrupt Coalescence

Most operating systems use interrupts to initiate the processing of received packets, motivated by the need to have low latency. However, the performance of the system degrades significantly at higher arrival rates due to overheads including that for context switching. At 10 Gbps, minimum sized packets arrive at 14.4 Mbps. Once the packet arrival rate gets above the point where it saturates the system, the system goes into receive livelock, as described in [17].

Modern operating systems (OSes) and NIC drivers implement IC or Poll Mode (PM), they process packets in a batch to mitigate packet processing overheads, especially due to interrupts [9]. Different OSes and NIC drivers use different strategies to reduce interrupt overheads, and as a result will batch packet differently [9]. Most often batching is based on

978-1-6654-4579-5/21/$31.00 ©2021 IEEE

time, so the number of packet batched depend on the rate of packets, but some also terminate a batch based on packet number. Network interfaces both at the end-systems as well as in the intermediate hops (especially with the use of software based routers and middleboxes) can implement IC. IC can decrease TCP throughput, especially when implemented on forwarding nodes [11]. IC distorts packet timing used by the ABW measurement and causes estimation error [8].

B. Forwarding using DPDK and OpennetVM

The adoption of Software Defined Networking (SDN) and NFV has led to an increase in software forwarding on network paths [7]. Building efficient packet processing software relies on careful management of system and network resources, and an understanding of the interfaces between software and hardware. I/O libraries such as the DPDK [10] achieve high performance by bypassing the kernel's networking stack to deliver packets directly from the NIC into user space memory. ONVM [7], [18] is an open source, BSD-licensed platform built on top of DPDK, but offers higher level abstractions and features such as naming, load balancing, and service chaining. Network functions run in lightweight Docker containers that can be dynamically combined to form complex chains.

One of the main principle of DPDK is to not use interrupts to receive packet, but instead to use PM, and process received packets as a batch at regular intervals [10]. As a result, any processing built on top of DPDK or ONVM will likely batch packets. The goal and effect of PM in DPDK are similar to IC in a NIC driver, so for simplicity we call it IC.

C. Available Bandwidth estimation

The ABW of an individual link is its unused capacity, i.e., the difference between its capacity and the current amount of traffic using it [5]. The ABW of a network path is the smallest ABW across its links. Most ABW estimation methods using active probing are in two categories: Probe Gap Model (PGM) and PRM.

The PGM model uses packet pair [19] for its estimation. When the packet pairs are sent, the PGM methods use the relationship between the time gap at the sender and output probe gap at receiver to estimate the available bandwidth. Because of the difficulty of PGM techniques to effectively measure the ABW when there are multiple congested links in the network [20], we instead focus on the PRM methods.

The PRM is based on the concept of self induced congestion. If the probe rate is below the path ABW, the probing packets face no queuing delay, whereas if it exceeds the ABW, congestion is created, probe packets are queued at the tightest link and experience an increase in the one-way delay (OWD). In PRM models, the unique inflection point can be found under some idealized conditions, such as sending an infinite length of probing packet trains. The well-known PRM-based estimation methods are TOPP [21], PathLoad [22], PathChirp [13]. Both TOPP and PathLoad use multiple rounds of constant bit rate streams and change the probe rate incrementally, until they find the probe rate separating the non-congested and congested regime. Both require sending multiple probing trains to produce a single estimate and are fairly intrusive and slow.

PathChirp uses a single chirp train to probe multiple rates, reducing overhead. The chirp train of PathChirp consists of a sequence of probe packets with increasing probe rates, by decreasing the time between sending packets. PathChirp measures the relative OWD of each received probe packet in the train, and estimates the ABW by finding the inflection point at which the OWD shows a consistent increasing trend. Despite being one of the earlier works, PathChirp is still considered one of the state-of-the-art methods for PRM [12], and new measurement methods derived from PathChirp have been proposed in recent years.

PathCos++ [14] was the first proposed method using a chirp train with decreasing probe rates, by increasing the time between packets. The goal of such a chirp train is to first congest and then decongest the path, creating a congestion peak. The additional intuition is that two packets with similar OWD usually experience similar congestion, and therefore the probe traffic between those packets should be congestion neutral and equal to the ABW. PathCos++ tries to find the widest spaced pair of packets that are on both sides of the congestion bump and with similar OWD, and then computes the received rate of probe packets between those two packets as the ABW estimate. SLDRT [15] also uses a chirp train with a decreasing rate. It searches the point at which the path becomes decongested, by picking the first packet for which the OWD returns to its minimum, and then uses the rate of the chirp train up to that point as the ABW estimate.

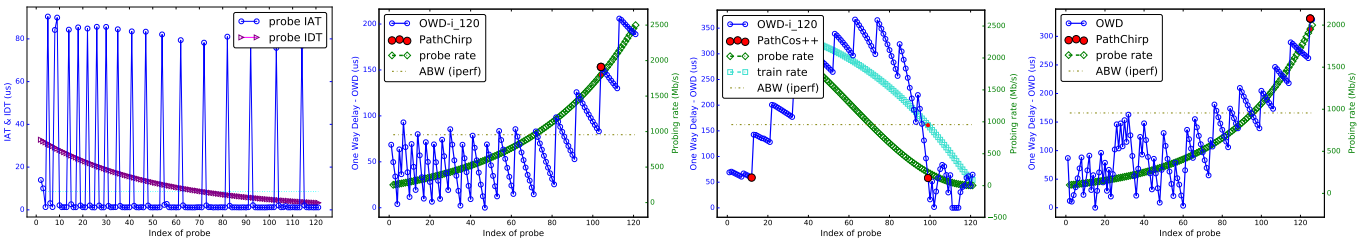
Voyager-D [23] is a new method derived from PathCos++. Voyager-D introduces a noise threshold based on the measured OWD noise, and modifies the pair selection to prefer probe pairs which are above the noise threshold. Voyager-D also adjusts the leading and trailing probes to find a pair with less difference in OWD. The chirp train of Voyager-D is designed for systems with rate adaptation, with reduced density at the edge of the rate window, and has an exponential decrease, rather than the linear decrease of PathCos++.

D. Interrupt Coalescence Mitigation

Most high-speed network interfaces (1Gbps and higher) use IC or PM to reduce the frequency of context switch and CPU load [8]. The driver processes the NIC receive queue less frequently, and incoming packets wait in the receiver NIC and are delivered to the OS as a *batch*. This extra queuing in the NIC distorts the received times, directly impacting the OWD used for ABW estimation, and decreasing estimation accuracy.

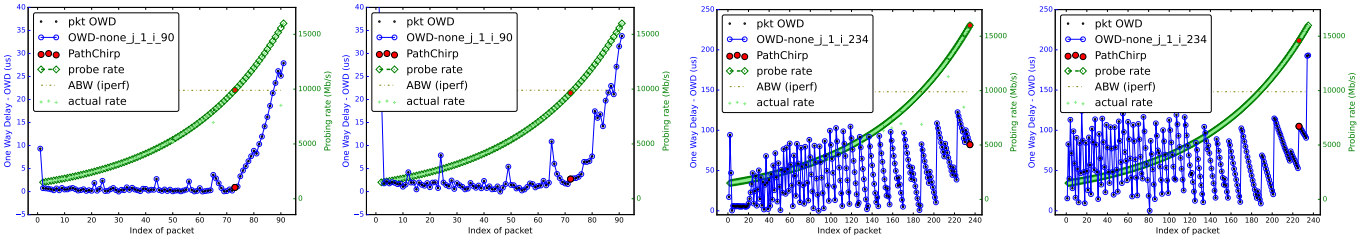
PathLoad [8] was one of the first methods to include a technique to mitigate IC. It detects IC when the Inter Arrival Time (IAT) between two packets is equal to the latency of a packet processing at the bottleneck rate. It filters received packets and keeps only packets at the end of a batch. The PathChirp tool [24] implements a similar technique, where it sends multiple packets for each rate probed and keep one packet per rate probed, which is the end of a batch.

A goal of those IC mitigation techniques is to improve the accuracy of ABW estimation. The techniques can apply to other network measurements (e.g., PathRate uses it for path



(a) *IAT of PathChirp* (b) *OWD of PathChirp* (c) *OWD of PathCos++* (d) *3 routers, PathChirp*

Fig. 1: *Batching effect from interrupt coalescence. IDT is the inter-departure time at the traffic source. Fig. (a)-(c) use a single idle 1Gb/s link. Fig. (d) uses four idle 1Gbps links.*



(a) *single hop, without IC* (b) *2 routers, without IC* (c) *single hop, with IC* (d) *2 routers, with IC*

Fig. 2: *Batching effect from with and without interrupt coalescence using PathChirp method on 10Gbps link.*

capacity measurement [8]) and for TCP congestion control (e.g., TCP-Rapid uses BASS [16]).

III. Experiments to Motivate Batching Mitigation

The IAT measured at the receiver between packets within a batch is primarily limited by the network protocol stack processing rate, and is therefore very small. However, the IAT between batches is dictated by the frequency of polling, and is much higher than average, as shown in Fig. 1a. Thus, the receive timing (IAT) is distorted compared to the inter-departure time (IDT) at the traffic source. These distorted received time values directly impact the OWD used for ABW estimation (Fig. 1b), and reduces the accuracy of the estimation of ABW.

Figure 1 is the results with IC from two experiments, without router and 3 routers on 1Gbps link. We use a longer chirp train (120 packets) so the effect of IC is clearer. The variability we observe in both the IAT (Fig.1a) and OWD (Fig.1b) due to IC causes the increasing-methods, such as PathChirp, to overestimate the ABW when no mitigation techniques are used, to overcome the noise in OWD. In the Fig. 1b, the bottom red point, the inflection point that is detected by Excursion Detection Algorithms, indicates the start of congestion. It happens around packet index 105 in the chirp train. The top red point along with probe rate is 1.7x of the actual ABW from PathChirp. PathChirp significantly overestimates ABW. This strongly motivates the need to improve the estimation of increasing methods like PathChirp, when there is IC.

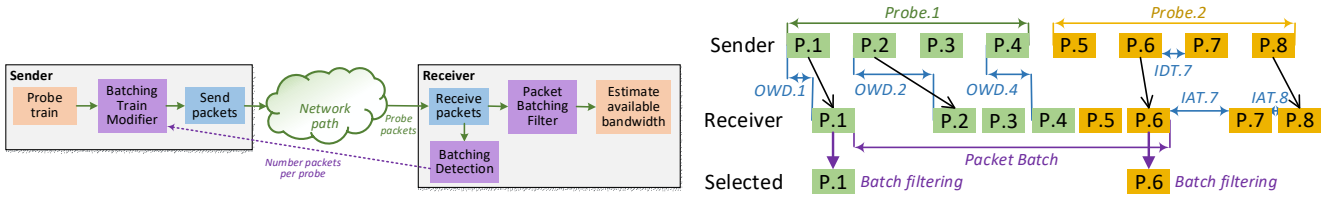
PathCos++ uses the Bump Detection Algorithm where two anchor points before and after the bump with similar OWD are selected to measure ABW. In Fig. 1c, the two red dots on the blue OWD curve at packet index 18 and 100 indicate the selected packets. In this case, the rate of probes between these packet is 878 Mb/s and underestimates the ABW. (Note: A red dot on the train rate is indicative, but it does not use the full PathCos++ for this approximate estimate.)

With multiple hops, if these include software routers (or middleboxes) that use IC to help achieve higher forwarding throughput, the impact of the IC on the OWD is exacerbated. For example, when we have a network with three intermediate routers as shown in Fig. 1d, IC happens at multiple points and the accumulated variability of the OWD is more complex, as the OWD trend starts to increase very early, well before the expected inflection point. The ABW estimate much less accurate (Fig. 1d).

We repeat the experiment with 10 Gbps links both for just the single hop, sender-receiver being connected back-to-back, and a configuration with 2 routers in-between the source and destination. We examine what happens when we do not have IC (Fig.2a, 2b) and when there is IC (Fig.2c, 2d). With the PathChirp increasing method, the results show that IC at the receiver impacts the ABW measurement. In the OWD plot without IC, the PathChirp methods are able to find the correct ABW almost exactly at 10 Gbps (the 'red dot' in Fig.2a. However, by enabling IC at the receiver, the results of OWD plot become more complex, and the PathChirp method overestimates the actual ABW Fig.2c. Hence, with the increasing methods, IC impacts ABW across the entire range of low to high bandwidth links. The problem continues to remain the same with the 3-hop case (2 software routers with IC in-between), as in Fig.2d.

IV. Packet Batching Mitigation Framework

To combat the effect of IC and to compare the performance of the different mitigatiton techniques, we implemented a modular Packet Batching Framework. This framework aims to address *Packet Batching*, regardless of its source, such as IC, PM or Frame Aggregation. This framework is an evolution of the solution implemented in PathChirp [24] and is comprised of 3 parts that are added to the bandwidth estimation tool (Fig. 3a). The Batching Train Modifier resides in the sender and modifies the chirp train to make it more robust to packet batching. The Packet Batching Filter processes the received packets and modifies the chirp train so that the bandwidth estimation can have better accuracy. Finally,



(a) *Packet Batching Framework* (b) *Multiple Packets per Probe*

Fig. 3: *Packet Batching Framework*

the Packet Batching Detector is a component we added to the framework, which measures the strength of the packet batching, and this information is used in the sender in the Batching Train Modifier. Our framework is modular, so that different techniques can be selected in each of these boxes, although not all techniques are compatible with each other. In particular, for most bandwidth estimation methods, the received chirp train processed by a given estimation method must have its original structure. Therefore most of the changes done by the Batching Train Modifier, such as adding packets, must be hidden by the Packet Batching Filter.

The default Batching Train Modifier is Multiple Packet per Probe Modifier (**MPPM**), and was adapted from PathChirp [24]. A chirp train is composed of a number of probes, each probe represents a rate that need to be tested. A normal chirp train has one packet for each probe. With MPPM, each probe is composed of multiple packets sent at the same rate, the number of packet for each probe is *mpp* (Fig. 3b). The ideal configuration of *mpp* is to have each probe longer than the batching introduced by the network path, including the receiver, so that each rate can be tested. If *mpp* is larger than the batch size, each probe will likely include a packet that is at the end of a batch. This multiplicative factor of MPPM makes the chirp train longer and increases overhead. Another option for the Batching Train Modifier is to create a **longer chirp train**, by multiplying the number of probes by *mpp*.

The first Packet Batching Filter is PathChirp Filter (**PCF**), which was adapted from PathChirp [24]. PCF assume a chirp train using MPPM, to restore the original chirp train it selects one packet in each probe (Fig. 3b). It uses a fixed 10 μs threshold to identified batched packets: if a particular packet is followed by an IAT smaller than 10 μs, this packet is considered batched. In each probe, PCF selects the last unbatched packet. If no unbatched packet is found in a probe, the chirp train is truncated before that probe. The second Packet Batching Filter uses the concepts of Buffering-Aware Spike Smoothing (**BASS**) [16] which was used to eliminate spikes in the IAT. Our estimation methods use OWD, and we adapted BASS to apply the same correction to the OWD. **Min-OWD** is a third, very simple Packet Batching Filter we created. It requires the use of MPPM. For each probe it selects the packet with the smallest OWD.

We designed **Max-IAT** to overcome some of the issues we discovered with PCF and Min-OWD. It requires the use of MPPM. For each probe, it identifies the maximum and minimum IATs following packets of the probe. If the maximum is at least four time higher than the minimum, batching is assumed and the packet before the maximum IAT is selected, otherwise the last packet of the probe is selected.

The main parameter of the Packet Batching Framework is *mpp*, the number of packets per probe. When the NIC on the receiver batches the processing of packets using polling, then the batch duration is determined mostly by the polling interval, with the number of packets in each batch depending on the received rate, which varies based on the chirp train and network conditions. If the value of *mpp* is too low, we may not find a end of batch in each probe, and mitigation will be less effective. But, a larger value of *mpp* increases the chirp train length and increases overhead (which we seek to avoid).

V. Experiments and Evaluation

A. Experimental Testbed

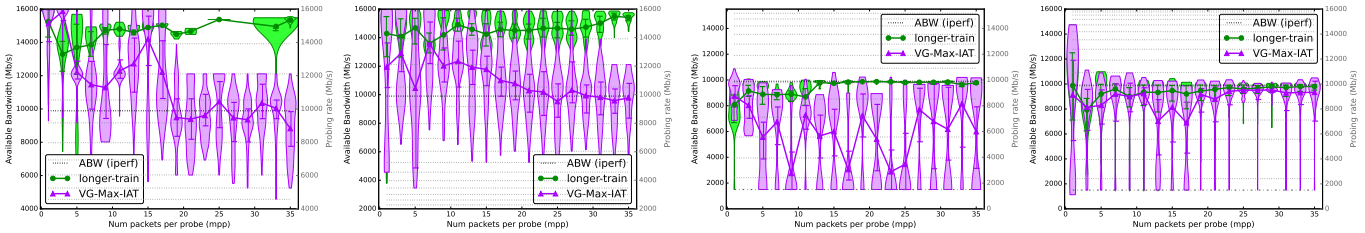
To measure the effectiveness of our IC mitigation techniques, we used a testbed of nodes interconnected using 10 Gbps links. The routers and sender node were Intel Xeon CPU E5-2640v4@2.4GHz with 20 Cores, 64 GB of RAM, and running Ubuntu 18.04.1 LTS (kernel4.15.0-137-generic), using Intel® 82599ES 10 Gigabit Ethernet NICs. The receiver node was using Intel® Ethernet Controller X710 NICs.

Our simplest configuration is just two nodes, a sender and receiver. We also used a slightly more complex topology with two intermediate routers. These routers were software-based nodes running OpenNetVM, with a simple forwarding NF. OpenNetVM uses poll mode drivers and batches packets to be processed and therefore we expect to see considerable distortion. We then measure the effectiveness of IC mitigation. A final configuration uses an separate node running OpenNetVM for cross traffic. All the links in our testbed were 10Gbps, and all the experiments used jumbo packets.

B. Evaluation of batching mitigation techniques

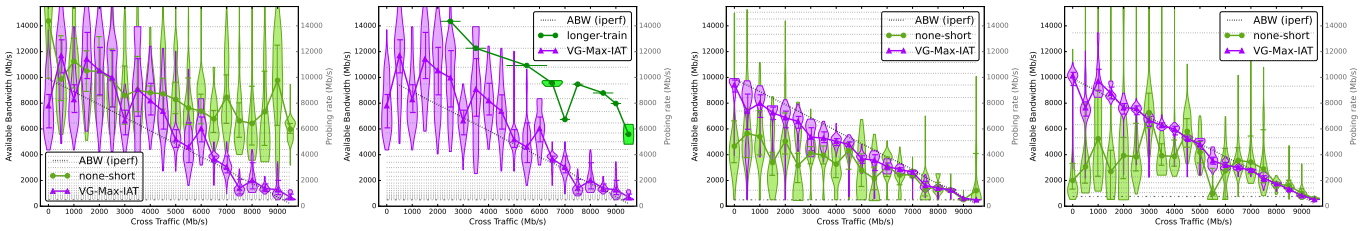
Because of the modular nature of our framework, we are able to apply **max-IAT** packet batching filtering to all the methods. **Max-IAT** provides some improvement in accuracy beyond those associated with making the chirp train longer.

Before we explain our results, let us describe the graph briefly. Each plot shows the relationship between number of packets per probe (horizontal axis) and ABW Mbps (vertical axis). Each point on the graph (the ABW estimate vs. the number of packets per probe) also shows a 'violin plot' for each data point to show the distribution of the values measured for that point. The center is the average, and the thickness is the extent of the probability density. At each x-axis point, the height of the violin represents the distribution for each ABW values in the experiment's data using a particular number of packets per probe. The green line (and violin) represents the



(a) *PathChirp, single hop* (b) *PathChirp, 2 routers* (c) *PathCos++, single hop* (d) *PathCos++, 2 routers*

Fig. 4: *Impact of interrupt mitigation technique on single hop and 3 hops 10Gb/s link using PathChirp and PathCos++*



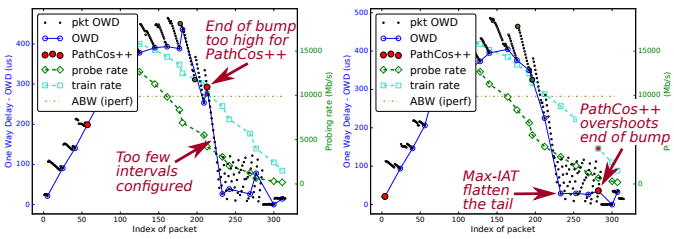
(a) *PathChirp* (b) *PathChirp-Long* (c) *PathCos++* (d) *Voyager-D*

Fig. 5: *Impact of interrupt mitigation technique with UDP Cross Traffic going through 1 ONVM software router (10Gb/s link)*

case when there is the only mitigation technique used it to make the chirp train longer, and the purple line (and violin) is for the case where the max-IAT filtering is used, in conjunction with MPPM on the sender. For instance, let us take a look at the VG-Max-IAT in the Fig. 4a where the number of packets per probe is 17. The center point is around 12 Gbps, the median, which shows the ABW estimate of this with the max-IAT filtering (batch mitigation method) in place. Moreover, more of the area of the purple is above 10Gbps, which results in an over-estimate of the PathChirp technique.

Figure 4 shows the performance of the increasing and decreasing methods on high-speed links (10Gbps) using the **max-IAT** mitigation technique. We use PathChirp and PathCos++ as the base methods. We have both the 1 hop case and one with 2 routers between source and destination.

Performance of Increasing method: Fig. 4a, 4b first shows the performance of PathChirp without any mitigation techniques (green line). The ABW estimate is consistently higher around 14 Gbps or higher. With the mitigation technique, VG-Max-IAT, the performance is somewhat better for both cases, with the ABW being closer to the nominal 10 Gbps when the *mpp* increases above 20 as seen in both Fig. 4a, 4b. With the 2 routers in the path, the VG-Max-IAT mitigation allows the ABW estimate to be slightlly better for higher *mpp*.



(a) *bump fails height filter* (b) *overshoots bump*

Fig. 6: *PathCos++ OWD Plot*

Performance of Decreasing method: Figs. 4c, 4d and show the performance of decreasing method without mitigation techniques, for the single hop and the network with 2 routers in-between. PatchCos++ by itself without the mitigation technique (none) actually performs better than when VG-Max-IAT filtering is used, in the single hop case as seen in Fig. 4c. The first cause of underestimation is that the number of intervals is too small, this increases the risk of PathCos++ selecting a pair of points that is too high and that fails the height filter of PathCos++ (Fig. 6a), in that case PathCos++ declares the path uncongested. The second cause is that **Max-IAT** flattens the noise of the OWD curve, in some cases the tail is so flat that PathCos++ overshoots the end of bump and include in it's measurement some idle time (Fig. 6b). However, once we go to the multi-hop (2 routers) case, (see Fig. 4d), PathCos++ with mitigation is much closer to the case with no mitigation ('none') when there are at least 20 packets per probe.

Overall, we see that the design of decreasing-chirp-train method is more resilient to the noise introduced by IC in estimating ABW. In fact, all the decreasing methods offer superior performance with IC across a range of conditions and work well on high-speed links. The increasing-chirp-train methods are not robust enough to accommodate the noise introduced by IC, especially on multiple links, even with **max-IAT** filtering. With a longer chirp train, PathCos++ with batching mitigation estimates ABW reasonably.

C. Cross Traffic

We now generate cross traffic to compete with the flow whose ABW is being measured. We show the effect of cross traffic on the ABW estimate of existing techniques and to demonstrate the effectiveness of our batching mitigation technique. We have 3 nodes (sender, receiver, and cross-traffic generator) connected to interfaces at a single router running ONVM-based forwarding. Iperf is used to generate the cross-traffic (using 5 UDP flows) flowing from the generator to the receiver, at rates from 0 to 9500 Mbps, in steps of 500 Mbps, with jumbo packets. Fig. 5 shows the results with the cross traffic using both the increasing method (PathChirp)

and decreasing methods (PathCos++, Voyager-D), as the cross traffic varies. We use a fixed *mpp* of 8.

Increasing methods: In Fig. 5a PathChirp without using any mitigation technique (PathChirp-none) is compared to PathChirp using Max-IAT (PathChirp-Max-IAT), and in Fig. 5b PathChirp with a longer chirp train (PathChirp-longer) is compared to using Max-IAT again. The performance of PathChirp-none and PathChirp-longer is poor, they overestimate the ABW, they do not track the actual residual bandwidth beyond the cross-traffic and the errors increase with increasing cross-traffic. The average difference between PathChirp-none with actual ABW is 175% with the error increasing when the cross traffic is greater than 3Gbps. Overall, Pathchirp-longer does more worse than PathChirp-none (Figs. 5a,5b) and does not improve the accuracy of ABW. The mitigation does help - PathChirp-Max-IAT has only a 21% average error difference, its curve follows the actual ABW, and the result is even better when the amount of cross traffic is higher than 6.5Gbps.

Decreasing methods: In Fig. 5c and 5d, not using the mitigation techniques, PathCos++ and Voyager-D (the two representative decreasing methods we study) substantially underestimate the ABW for the most part (except when the cross traffic is very high, and the residual available bandwidth is just less than 3 Gbps.)

Errors with varying ABW: The average error with just PathCos++ is about 37%. However, using the mitigation techniques with the decreasing methods provides excellent results, with the estimated ABW tracking reasonably closely to the actual ABW, especially with Voyager-D (Fig. 5d) and the error is a modest 4%. With decreasing methods, such as PathCos++ and Voyager-D, the distribution of ABW is quite tight throughout the range (seen by the range (max.- min.) of the estimate). The distribution (width of the violin) is also close to the actual ABW (Fig. 5c, 5d). On the other hand, with the increasing methods (Fig. 5a, 5b) the range is much larger and the distribution is widespread throughout.

VI. Conclusions

Interrupt Coalescence used by NIC drivers to mitigate packet processing overheads for high speed networks distort the receiver timing and impacts ABW estimation. We showed that existing techniques such as PathChirp and PathCos++ produce inaccurate ABW estimates in higher speed networks (e.g., at 1 Gbps or more), and these inaccuracies increase with software-based forwarding engines using DPDK.

We proposed a packet batching mitigation technique, Max-IAT, and show that it reduces the errors for both classes of probe rate models, increasing and decreasing methods. We further showed that, generally the decreasing methods perform better, and the mitigation technique is particularly useful for them in the presence of cross-traffic.

VII. Acknowledgments

We thank the US NSF for their generous support of this work through grant CNS-1763929.

References

[1] C. D. Guerrero and M. A. Labrador, "On the applicability of available bandwidth estimation techniques and tools," Computer Communication, vol. 33, no. 1, pp. 11–22, Jan. 2010.

[2] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint admission control: Architectural issues and performance," ACM SIGCOMM Computer Communication Review, vol. 30, no. 4, pp. 57–69, 2000.

[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," SIGCOMM Comput. Commun. Rev., 2013.

[4] T. Tuan and K. Park, "Multiple time scale congestion control for self-similar network traffic," Performance Evaluation, vol. 36, pp. 359–386, 1999.

[5] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. Mckeown, "Elastictree: Saving energy in data center networks," in NSDI, 2010.

[6] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, 2003, pp. 39–44.

[7] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. K. Ramakrishnan, and T. Wood, "OpenNetVM: A Platform for High Performance Network Service Chains," in Proceedings of the 2016 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization. ACM, Aug. 2016.

[8] R. Prasad, M. Jain, and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in Passive and Active Network Measurement. Springer, 2004, pp. 247–256.

[9] K. Salah, K. El-Badawi, and F. Haidari, "Performance analysis and comparison of interrupt-handling schemes in gigabit networks," Computer Communications, vol. 30, no. 17, pp. 3425–3441, 2007, special Issue Concurrent Multipath Transport.

[10] "Data plane development kit," http://dpdk.org/, 2014, [ONLINE].

[11] M. Zec, M. Mikuc, and M. Žagar, "M.: Estimating the impact of interrupt coalescing delays on steady state tcp throughput," in in Proceedings of 10-th SoftCOM, 2002.

[12] D. Salcedo, J. Guerrero Macias, and C. Guerrero, "Overhead in available bandwidth estimation tools: Evaluation and analysis," International Journal of Communication Networks and Information Security, vol. 9, pp. 393–404, 01 2017.

[13] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in Passive and active measurement workshop, 2003.

[14] H. Lin, M. Liu, A. Zhou, H. Liu, and Z. Li, "A novel hybrid probing technique for end-to-end available bandwidth estimation," in Local Computer Networks (LCN), 2010 IEEE 35th Conference on. IEEE, 2010, pp. 400–407.

[15] Z. Hu, D. Zhang, A. Zhu, Z. Chen, and H. Zhou, "Sldrt: A measurement technique for available bandwidth on multi-hop path with bursty cross traffic," Computer Networks, vol. 56, no. 14, pp. 3247–3260, 2012.

[16] Q. Yin, J. Kaur, and F. D. Smith, "Can bandwidth estimation tackle noise at ultra-high speeds?" in 2014 IEEE 22nd International Conference on Network Protocols, 2014, pp. 107–118.

[17] J. C. Mogul and K. K. Ramakrishnan, "Eliminating receive livelock in an interrupt-driven kernel," ACM Transaction Computer System, vol. 15, no. 3, 1997.

[18] J. Hwang, K. K. Ramakrishnan, and T. Wood, "Netvm: High performance and flexible networking using virtualization on commodity platforms," in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), Seattle, WA, 2014, pp. 445–458.

[19] S. Keshav, "Packet-pair flow control," AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, USA, Tech. Rep., 1995.

[20] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic," SIGCOMM Computer Communication Review, vol. 21, no. 5, 1991.

[21] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in Global Telecommunications Conference, 2000. GLOBECOM'00. IEEE.

[22] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in In Proceedings of Passive and Active Measurements (PAM) Workshop. Citeseer, 2002.

[23] C. Liu, J. Tourrilhes, C.-N. Chuah, and P. Sharma, "Voyager: Revisiting available bandwidth estimation with a new class of methods - decreasing-chirp-train methods." to be published.

[24] "PathChirp source code," http://www.spin.rice.edu/Software/pathChirp/, 2003.