HF^{2}T: Host-Based Flowlet Fine-Tuning for RDMA Load Balancing

Chuhao Chen
Fudan University

Jiarui Ye
Fudan University

Yongbo Gao
Fudan University

Sen Liu
Fudan University

Yang Xu*
Fudan University
Peng Cheng Laboratory

ABSTRACT

In modern data center networks, RDMA is widely applied in scenarios such as high-performance computing, distributed storage and machine learning. In recent studies, it has been observed that flowlet switching load balancers cannot fully unleash their robust capabilities due to an insufficient number of flowlets in RDMA networks. In this paper, we scrutinize the traffic pattern at the end hosts and meticulously analyze time gaps between packets. Our findings reveal that in RDMA, the proportion of time gaps between packets larger than the flowlet threshold is notably scarce, constituting only a fraction of those in TCP, averaging 1/300. Based on this observation, we propose **HF^{2}T**, a host-based method to improve the effectiveness of flowlet-level load balancing in RDMA. The core idea is to postpone a minimal number of specific packets at the host, actively elongating the time gaps between them, and promoting flowlet generation at the switch. The cost of postponing a minimal number of packets is far outweighed by the benefits of flowlets generation at the switch, improving the network performance. Simulation experiments confirm that **HF^{2}T**, when deployed in conjunction with the flowlet load balancing, achieves an average reduction of 37.32% in Medium FCT and an average reduction of 28.75% in 99-percentile FCT, compared to deploying the same flowlet load balancing scheme solely at switches.

CCS CONCEPTS

• **Networks → Data center networks**.

KEYWORDS

Load Balancing, Remote Direct Memory Access, Data Center Networks

ACM Reference Format:
Chuhao Chen, Jiarui Ye, Yongbo Gao, Sen Liu, and Yang Xu. 2024. **HF^{2}T**: Host-Based Flowlet Fine-Tuning for RDMA Load Balancing. In *The 8th Asia-Pacific Workshop on Networking (APNet 2024), August 3–4, 2024, Sydney, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3663408.3663410

*Corresponding Author.

APNet 2024, August 3–4, 2024, Sydney, Australia

ACM ISBN 979-8-4007-1758-1/24/08
https://doi.org/10.1145/3663408.3663410

1 INTRODUCTION

Remote Direct Memory Access (RDMA), exemplified by technologies like RoCEv2 [9], is revolutionizing data center networking by enabling low-latency, high-throughput communication with minimal CPU intervention. Through direct memory access between servers' RAM using dedicated RDMA-capable network interface cards (RNICs), RDMA reduces CPU involvement and enhances overall network efficiency.

Data center network topologies are typically crafted for scalability with ample redundancy. The CLOS topology [2] is widely adopted in the data center for its high aggregate bandwidth and low latency. In the CLOS topology, communication between hosts occurs through multiple equal-cost paths, enhancing network robustness and performance. However, efficient load balancing is essential in this topology, particularly at ToR (Top of Rack) switches, to distribute network traffic across different paths. Numerous approaches have been explored and implemented to enhance the efficiency of switches and fully utilize available links.

Load balancing schemes implemented on switches can be categorized into three granularities: flow-level, packet-level, and flowlet-level. Flow-level load balancing schemes, such as ECMP (Equal-Cost Multipath) [10], are easy to implement and have low computational overhead, but they lack flow flexibility [3]. Packet-level schemes, such as DRILL [8], offer high flexibility due to frequent rerouting, but they suffer from severe out-of-order issues. Flowlet is a granularity between flow and packet, typically characterized by a burst of packets followed by an idle gap larger than a specific threshold. Flowlet-level schemes can offer high flexibility while simultaneously preventing out-of-order issues. Thus, in environments where TCP naturally generates flowlets due to its window mechanism, flowlet-level schemes outperform packet-level and flow-level schemes. Most existing commodity switches already support flowlet load balancing.

However, flowlet-level schemes fail in RDMA networks. In a recent investigation [17], it has been observed that ECMP outperforms flowlet-level schemes in RDMA environments under some situations. RDMA, distinct from TCP, implements packet pacing as a strategy to enhance overall performance. The nature of RDMA traffic patterns is such that naturally generated flowlets are not present. In our observation, in RDMA networks, the distribution of the packet stream tends to be more uniform, with an extremely

low proportion of time gaps exceeding the flowlet threshold, which spans several Round-Trip Times (RTTs). In the absence of frequent flowlet detection, flowlet-level schemes struggle to operate effectively and may degrade into a situation resembling ECMP, but with higher overhead and suboptimal hash functions, making it challenging to reroute flows. Even if a flowlet threshold smaller than the RDMA packet time gap is set, it might not be sufficient to prevent out-of-order issues.

This observation highlights the necessity of tailored load-balancing strategies that align with the unique characteristics of RDMA traffic. Recognizing the appeal of flowlets, we propose to adapt flowlet-level schemes for RDMA to leverage their advantages. We present **HF^{2}T**, a host-based approach that elongates a minimal number of specific time gaps to proactively create rerouting opportunities, enhancing the effectiveness of flowlet load balancing in RDMA. The contributions of this paper are as follows:

- We have analyzed the traffic patterns in RDMA networks and uncovered why flowlet switching underperforms in such environments from a new perspective.
- To adapt flowlet-level load balancing for RDMA, We design **HF^{2}T**, a host-based method that selectively manipulates a small number of specific packets to promote flowlet generation and create rerouting opportunities. This method minimally impacts the existing advantages of RDMA.
- Our approach only requires modification at the end host, without the need to modify existing commodity switches that support flowlet-level load-balancing schemes, making it deployment-friendly.
- We evaluate **HF^{2}T** on NS3 [1] simulations. Our results show that **HF^{2}T** can reduce medium FCT and 99-percentile FCT by 37.32% and 28.75%, respectively, compared to the cutting-edge flowlet-level load balancing scheme.

2 RELATED WORK

Data center switch load-balancing schemes can be classified into three categories based on the size of the load-balancing unit: flow-level, packet-level, and flowlet-level.

Load-balancing schemes, like ECMP [10] and its variants (WCMP [22], Expeditus [19], TinyFlow [20], DiFS [5]), operate at the flow level, reducing packet reordering. Despite their common use in data centers for their simplicity, these schemes share the same challenge that they lack prompt rerouting during congestion, limiting responsiveness and potentially leading to suboptimal resource utilization.

Packet-level load-balancing schemes, like RPS [6] and DRILL [8], aim for precise rerouting using packets as the smallest granularity. RPS employs random forwarding, while DRILL selects the least loaded link based on local congestion information. Packet-level schemes introduce the challenge of packets out-of-order issues, which is undesirable for TCP or RDMA protocols.

Flowlet-level load-balancing schemes, such as LetFlow [18] and CONGA [4], target out-of-order issues while preserving flexibility. Using flowlet gaps, these schemes enable rerouting without introducing packet reordering. Especially effective in TCP environments given the bursty nature of TCP packets, flowlet-level approaches manage flows at a granular level, ensuring efficient handling of TCP traffic patterns.

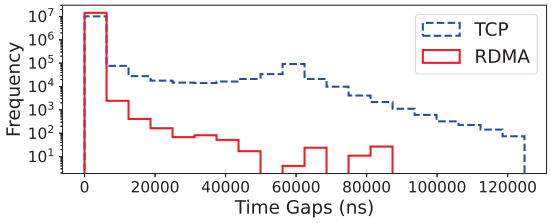


Figure 1: RDMA vs TCP in Time Gap Distribution

Table 1: Comparison of RDMA and TCP in Time Gaps

Protocol \ Time Gaps	>=RTT	>=2RTT	>=3RTT
RDMA	0.012%	0.006%	0.003%
TCP	2.376%	1.854%	1.550%

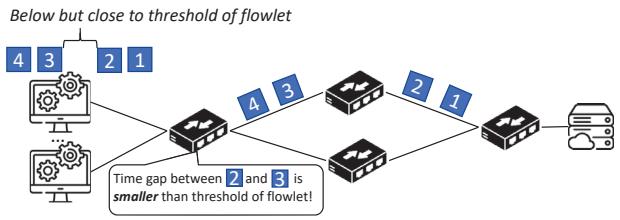
RDMA employs distinct implementation details compared to TCP, yielding diverse traffic patterns. Addressing the imperative need for enhanced load balancing in RDMA traffic, conventional flowlet or packet-level schemes may prove unsuitable. In RDMA, ensuring load balancing without reordering poses a distinctive challenge. To counter this, ConWeave [17] proposes active in-network reordering, a proactive switch-based approach. This enables load balancing in RDMA networks without introducing reordering issues.

3 TRAFFIC PATTERN IN RDMA

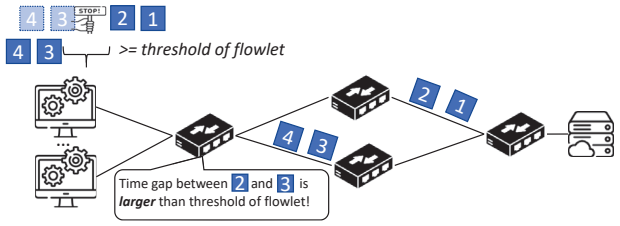
This section aims to investigate the traffic pattern in RDMA. To better introduce this topic, it is essential to clarify two key concepts. Firstly, "time gap" refers to the time difference between adjacent data packets within the same flow. Secondly, a "flowlet" is a burst of packets from the same flow, indicating a new flowlet when the time gap between adjacent packets exceeds a specific threshold [16]. This threshold is usually set to a relatively large value to prevent out-of-order issues.

Recent studies [13, 17] have shown that, under the same flowlet threshold, the size of flowlets in RDMA networks is significantly larger than those in TCP networks. This observation reveals a notable reduction in the number of flowlets in RDMA traffic compared to TCP under the same flowlet threshold. This disparity is primarily attributed to TCP's window-based packet transmission and batch optimization of ACKs [7]. However, RDMA employs hardware-based pacing, leading to more continuous traffic with shorter inter-packet time gaps. The limited occurrence of flowlets in RDMA results in a substantial loss of rerouting opportunities, rendering flowlet-level load balancing less effective in RDMA. To delve deeper into this phenomenon, considering the time gap as a new perspective, we conducted measurements of packet time gaps at the host in RDMA and TCP respectively, obtained their respective distributions, and calculated the proportion of larger time gaps relative to all time gaps.

Figure 1 illustrates a comparative distribution of packet time gaps between RDMA and TCP using a logarithmic scale on the



(a) Poor load balancing effectiveness



(b) A slight elongation of a time gap close to flowlet threshold

Figure 2: Draft of HF^{2}T

vertical axis. The graph reveals that, when contrasted with TCP, RDMA exhibits a proclivity for packet time gaps to cluster within narrower and smaller numerical ranges, consequently heightening the challenge of forming flowlets.

Table 1 provides a detailed breakdown of the proportions of time gaps greater than or equal to RTT, 2RTT, and 3RTT in both RDMA and TCP. Typically, the threshold for flowlet identification is set to a value greater than or equal to 2RTT [18, 21]. Table 1 reveals that time gaps in RDMA equal to or exceeding 2RTT constitute only 0.006%, while in TCP, the corresponding proportion exceeds RDMA by over three hundred times, reaching 1.854%. This indicates that compared to TCP, time gaps in RDMA meeting or exceeding the flowlet threshold are exceedingly rare, leading to a scarcity in the number of flowlets in RDMA. Consequently, flowlet-level load balancing schemes cannot fully leverage their robust capabilities in RDMA networks.

4 DESIGN

In the preceding section, we discussed the traffic pattern in RDMA. Our observation suggests that instances where the time gaps between packets in RDMA surpass the flowlet threshold are exceptionally rare. This rarity poses challenges for ToR switches at the initial hop in the network to effectively detect flowlets. Therefore, we initialized consideration on increasing the occurrence of time gaps between packets that exceed the flowlet threshold in the RDMA traffic pattern. The objective is to achieve this increase while enhancing the low latency and high throughput advantages of RDMA. We propose a method called **HF^{2}T**, which captures a minimal number of time gaps positioned at the tail end of the packet time gap distribution and actively elongates these specific time gaps to the flowlet threshold.

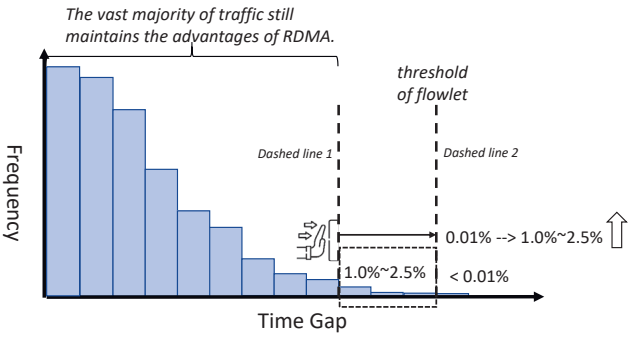


Figure 3: Delay Packets based on Time Gap Distribution

4.1 Draft: A Host-based Method

We have observed that the traffic pattern at the host in RDMA leads to suboptimal flowlet load balancing at the switch. Therefore, we aim to address this issue at its root by making slight adjustments to certain time gaps at the host, thereby increasing the number of flowlets detected by the switch.

As illustrated in Figure 2, packets 1, 2, 3, and 4 belong to the same flow. In Figure 2(a), despite a relatively large time gap between packet 2 and packet 3, it is still smaller than the flowlet threshold, preventing the ToR switch from detecting the flowlet. Consequently, packets 1 and 2, as well as packets 3 and 4 congest on the same path. In Figure 2(b), the host seizes the opportunity presented by this larger time gap, making a slight adjustment to increase the delay, just enough for the time gap to reach the flowlet threshold. As a result, the ToR switch detects the flowlet and forwards packets 3 and 4 to a different path than packets 1 and 2, achieving effective load balancing. A minor delay can create a valuable opportunity for rerouting, significantly enhancing network performance.

Why opt for implementing this task at the host instead of the switch? The choice is influenced by the challenge of managing an entry for each of the tens of thousands of flows in a switch, given the limitations in switch storage resources. Additionally, implementing it on the switch could lead to Head-of-Line Blocking, where packets from multiple flows are queued up. Delaying a packet from one flow at the forefront of the queue could potentially block packets from other flows. By implementing this resource-intensive task at the host, it can be distributed across multiple hosts. Hosts typically boast more abundant storage and computing resources compared to switches, making it more manageable to execute and optimize this task effectively.

The following subsection will provide a detailed explanation of the specific time gaps we intend to capture at the host for adjustments and elucidate their impact on the network performance enhancement. We will identify which time gaps are sufficiently large to warrant delaying them to the flowlet threshold and point out the proportion of time gaps that we aim to capture.

4.2 Elongate Gaps Based on Distribution

We will elaborate on our criteria for choosing packet time gaps in Figure 3. The proportion of time gaps exceeding the flowlet threshold under RDMA is typically below 0.01% supported by the

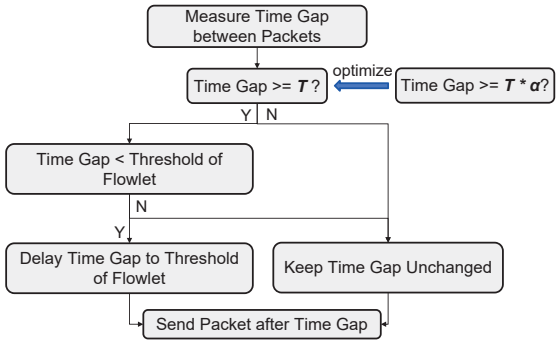


Figure 4: Process Diagram of HF^{2}T

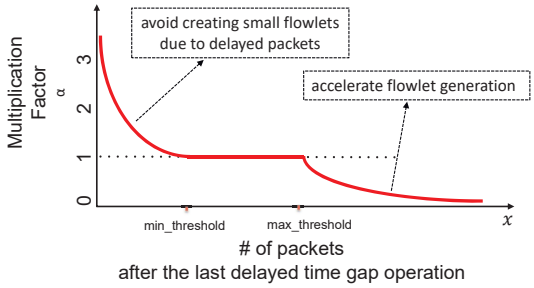


Figure 5: Function Graph of Multiplication Factor α

preceding section, as shown on the right side of Dashed line 2 in the figure. Our goal is to increase the proportion of packet time gaps that exceed the flowlet threshold while enhancing the low latency and high throughput advantages of RDMA. We focus on those time gaps that are slightly below the flowlet threshold but very close to it, such as the packet time gaps enclosed by the rectangular dashed line in Figure 3. These time gaps are already very close to the flowlet threshold, and a slight delay can enable switches to detect them as gaps for new flowlets. By delaying these time gaps to the flowlet threshold, there is a modest cost for a substantial gain in rerouting opportunities. As shown in Figure 3, we delay the time gaps between Dashed line 1 and Dashed line 2 to the flowlet threshold, thereby increasing the proportion of time gaps exceeding the flowlet threshold. At the same time, most of the time gaps on the left side of Dashed line 1 remain unchanged, maintaining the high throughput advantage of RDMA for small time gaps.

The proportion of time gaps close to the flowlet threshold that we delay should not be too high or too low. An excessively high proportion of time gaps close to the flowlet threshold can have adverse effects. Firstly, it will drastically alter the traffic pattern, leading to the loss of the efficiency advantage of RDMA transmission. Secondly, a notable delay in time gaps with a large difference from the flowlet threshold can lead to a significant increase in flow completion time(FCT). Conversely, if the proportion of time gaps close to the flowlet threshold is too low, switches will detect only a limited number of flowlets. Referring to the proportion of time gaps exceeding the flowlet threshold in TCP networks, which is mostly below 2.5%, we suggest selecting a proportion of time gaps close to the flowlet threshold between 1% and 2.5%, which represents a flexibility improvement of 100x to 250x compared to the previous proportion.

In terms of the duration of the delay for the captured time gap, we believe that delaying it to the flowlet threshold is sufficient. While link and switch queues may have an impact on the time gap, considering that data centers mostly employ Leaf-Spine or Fat-Tree topologies [2], flowlet switching typically occurs at the first-hop ToR switch. This minimizes the impact on the time gap, making delaying it to the flowlet threshold a reasonable choice.

4.3 A Comprehensive Scheme Considering Flowlet Length

In this subsection, we will integrate the content discussed in the preceding two subsections and formally introduce the logic of **HF^{2}T**. As shown in Figure 4, this is a process diagram illustrating the workflow of **HF^{2}T** implemented at the host. Firstly, the host measures the time gap between the current packet to be sent and its previous packet. Next, the host needs to determine whether the time gap is greater than or equal to T, where T is a threshold we set to evaluate whether this time gap is large enough to warrant a delay. As discussed in the preceding subsection, T should select packets located in the tail end of the time gap distribution, specifically in the range of 1% to 2.5%. Therefore, T should be set within the interval of the 97.5-percentile to 99-percentile of the time gap distribution. If the time gap is less than T, the host does not need to take any action. If the time gap is greater than T and less than the flowlet threshold, the host needs to extend this time gap to the flowlet threshold. And if the time gap itself is already greater than the flowlet threshold, the host needs no modification.

Regarding how to obtain the interval between the 97.5-percentile and 99-percentile in the distribution of packet time gaps to configure T, the host needs to undergo pre-training for a certain period. As the host accumulates a significant number of recorded time gaps, T can then be set, for example, by setting it to the 99-percentile of the time gap distribution.

Considering the impact of the flowlet length, we aim to avoid proactively generating flowlets that are too short, as this may result in a significant amount of packet reordering and a substantial increase in FCT. Simultaneously, we also want to prevent the creation of excessively long flowlets, as they can lead to suboptimal load-balancing effects. We have designed a multiplication factor α, as illustrated in Figure 5. As shown in Figure 4, in the optimized scheme, we use '$TimeGap \geq T \times \alpha$' instead of '$TimeGap \geq T$'. This multiplication factor α allows for the adjustment of the threshold for selecting time gaps, thereby avoiding excessively short or long flowlets. It's important to highlight that our approach does not prevent the natural generation of short flowlets; instead, we refrain from manually creating them. The multiplication factor α is a piecewise function, where x represents the number of packets sent by the host after the last delayed time gap operation. This variable x can be approximated as the current length of the flowlet. The formula for this piecewise function is as follows:

$$\alpha = \begin{cases} \text{min\_thr.}/x, & x \leq \text{min\_thr.} \\ 1, & \text{min\_thr.} < x < \text{max\_thr.} \\ \text{max\_thr.}/x, & x \geq \text{max\_thr.} \end{cases} \quad (1)$$

This function has two crucial parameters, min_threshold (abbreviated as min_thr.) and max_threshold (abbreviated as max_thr.). When x is less than min_thr., indicating that the current flowlet is too short, the smaller the x, the larger the α. As α rises, so does the threshold for selecting time gaps, preventing the premature creation of excessively short flowlets. When x falls between min_thr. and max_thr., α is set to 1, directly using the T value, indicating a moderate flowlet length. When x exceeds max_thr., denoting a prolonged current flowlet, α decreases as x increases. This results in a lower threshold for selecting time gaps, allowing more time gaps to be chosen for a delay to the flowlet threshold, thereby accelerating flowlet generation. Certainly, this function is a preliminary version derived from a heuristic insight, and further refinements and insights are required for enhancement.

5 EVALUATION

We utilized NS3 [1] for our preliminary simulation experiments to assess the effectiveness of our proposed **HF²T**, aiming to validate whether deploying the **HF²T** at the host could enhance the performance of flowlet-level load balancing schemes in RDMA networks.

5.1 Evaluation Setup

In preliminary simulation experiments, we employed a Fat-Tree topology [2] with k=4. This topology consists of 4 Core switches, 8 Agg switches, 8 ToR switches, and 16 servers. Each server is equipped with a single 100Gbps RNIC connected to a ToR switch. All links in the topology operate at 100Gbps with a latency of 1us. We utilized Web Search [23] and Meta Hadoop [15] as our workloads. The parameter T of **HF²T** is configured to be the 99-percentile of the packet time gap distribution. The two parameters for the multiplication factor α, min_thr. and max_thr., are set to 20 and 80, respectively. As for the congestion control protocol, we employed DCQCN [23], the standard congestion control scheme for commodity RNICs. The evaluation metric is FCT slowdown, where the actual FCT of a flow is normalized by the base FCT when the network experiences no other traffic.

Our comparisons include ECMP, LetFlow [18], LetFlow with **HF²T** lacking multiplication factor α for dynamic adjustment of T (hereinafter referred to as LetFlow w/ **HF²T** (No Dynamic T)), and LetFlow with the complete **HF²T** (hereinafter referred to as LetFlow w/ **HF²T**).

5.2 Performance of HF²T

We conducted simulations with 70% average traffic loads in the Web Search and Meta Hadoop workloads, respectively. The parameters of **HF²T** are set as mentioned in the preceding subsection. In Figure 6, we present the medium and 99-percentile FCT slowdowns under different flow sizes.

Both in Web Search and Meta Hadoop workloads, the performance of LetFlow with **HF²T** outperforms LetFlow with **HF²T** (No Dynamic T), and the latter performs better than LetFlow and ECMP. In the Web Search workload, LetFlow with **HF²T** (No Dynamic T) reduces Medium and 99-percentile FCT by 8.76% and 8.70%, respectively, compared to LetFlow. Notably, LetFlow with **HF²T** achieves a substantial reduction, with 37.32% in Medium FCT and 28.75% in 99-percentile FCT compared to LetFlow. In the Meta Hadoop workload, LetFlow with **HF²T** (No Dynamic T) sees an 8.12% reduction in Medium FCT and 4.63% in 99-percentile FCT compared to LetFlow. And LetFlow with **HF²T** achieves a significant reduction, with 22.34% in Medium FCT and 16.46% in 99-percentile FCT when contrasted with LetFlow.

The above analysis indicates that elongating a very small number of time gaps that are slightly below the flowlet threshold indeed yields rerouting benefits greater than the cost incurred by delaying transmission. Additionally, our proposed dynamic adjustment factor α for the T value can further enhance the identification of valuable time gaps.

5.3 Analysis of Internal Metrics

Table 2: Comparison of Flowlet Counts and PFC Counts

Scheme	# of flowlets	# of PFC
LetFlow	33583	3568
LetFlow w/ **HF²T**(No Dynamic T)	65865	2574
LetFlow w/ **HF²T**	138104	844

Table 2 presents statistics on the number of flowlets generated and the occurrences of Priority-based Flow Control (PFC) [12] under various schemes (ECMP excluded due to the absence of flowlets) in the Web Search workload with an average traffic load of 70%. A notable observation is that, compared to using LetFlow alone, the utilization of **HF²T** increases the number of flowlets by 311.23% and reduces the occurrences of PFC by 76.35%. This indicates that **HF²T** indeed raises the number of flowlets and creates numerous rerouting chances. Moreover, it effectively decreases the instances of PFC, as larger time gaps predict a higher likelihood of PFC occurrences on the current path. Delaying larger time gaps and redirecting traffic to alternative paths proves to be an efficient strategy for preventing PFC incidents.

6 DISCUSSION

Implementation: The logic of **HF²T** implemented at the end-host requires the host to accurately calculate time gaps between packets and actively elongate specific time gaps. The software implementation of this logic can be done by spinning a CPU core, which is needed to prevent the OS scheduler from introducing delays. Thus, the logic should be implemented on the hardware network card to meet the demand of high-frequency RDMA traffic without burning the CPUs. We plan to implement **HF²T** on SmartNICs in the future, with the NVIDIA BlueField-3 SuperNIC [14] already achieving speeds of up to 400Gbps.

Thinking about PFC: While the design rationale for **HF²T** does not directly derive from PFC, our simulation experiments reveal a noteworthy reduction in the frequency of PFC incidents within the network when employing **HF²T**. This decrease can be attributed to **HF²T**'s extension of larger time gaps and rerouting flows to

(a) Medium FCT slowdown for Web Search

(b) 99-percentile FCT slowdown for Web Search

(c) Medium FCT slowdown for Meta Hadoop

(d) 99-percentile FCT slowdown for Meta Hadoop

Figure 6: Medium and 99-percentile FCT Slowdowns for Web Search and Meta Hadoop (70% Avg. Load)

alternative paths. The prolonged time gaps indicate an increased probability of PFC. Rerouting can help alleviate this probability. For future refinements to **HF^{2}T**, considerations may include the reduction of PFC storms [9] or the prevention of PFC deadlock [11] as part of the optimization objectives.

A Comprehensive and In-depth Theoretical Model: Moving forward, it is imperative to formulate a theoretical framework to examine **HF^{2}T**'s impact on RDMA traffic comprehensively and guide the determination of threshold T and multiplication factor α systematically. This model requires a broader consideration, such as flow size, where distinct T values may be assigned for delay-sensitive small flows and throughput-sensitive large flows. And it is also necessary to derive a more rigorous formula for the multiplication factor α through modeling.

7 CONCLUSION

Flowlet-level load-balancing schemes face challenges in RDMA networks due to the absence of sufficient large time gaps in RDMA traffic. In this paper, we introduce **HF^{2}T**, a host-based method meticulously crafted to proactively identify time gaps near the flowlet threshold and elongate them to match the flowlet threshold, enhancing the chances of ToR switches detecting flowlets. A key feature of this approach is that a slight delay can yield significant benefits in terms of rerouting opportunities.

Through software simulations, we demonstrate that **HF^{2}T** can reduce medium FCT and 99-percentile FCT by 37.32% and 28.75%, respectively, compared to the exclusive deployment of LetFlow at switches. We believe that this is a work with great potential and worth further exploration. The future works will focus on the hardware implementation of **HF^{2}T** and the development of a theoretical model to thoroughly investigate its impact on RDMA networks.

8 ACKNOWLEDGMENTS

We express our sincere gratitude to the anonymous reviewers for their invaluable feedback. This work is sponsored by Key-Area Research and Development Program of Guangdong Province (2021B0101400001), National Natural Science Foundation of China (62172108), the Major Key Project of PCL, and Natural Science Foundation of Shanghai (23ZR1404900).

REFERENCES

[1] 2024. Network Simulator 3 (NS-3). https://www.nsnam.org/.
[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review* 38, 4 (2008), 63–74.
[3] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, et al. 2010. Hedera: dynamic flow scheduling for data center networks.. In *Nsdi*, Vol. 10. San Jose, USA, 89–92.
[4] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 503–514.
[5] Wenzhi Cui, Ye Yu, and Chen Qian. 2016. DiFS: Distributed Flow Scheduling for adaptive switching in FatTree data center networks. *Computer Networks* 105 (2016), 166–179.
[6] Advait Dixit, Pawan Prakash, Y Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *2013 proceedings ieee infocom*. IEEE, 2130–2138.
[7] Doug Freimuth, Elbert Hu, Jason LaVoie, Ronald Mraz, Erich Nahum, and John Tracey. 2006. *Evaluating Batching for TCP Offload.* Technical Report. Technical report, IBM, IBM TJ Watson Research Center.
[8] Soudeh Ghorbani, Zibin Yang, P Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. 2017. Drill: Micro load balancing for low-latency data center networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 225–238.
[9] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over commodity ethernet at scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 202–215.
[10] Christian Hopps. 2000. *Analysis of an equal-cost multi-path algorithm.* Technical Report.
[11] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. 2017. Tagger: Practical PFC deadlock prevention in data center networks. In *Proceedings of the 13th International Conference on emerging Networking*

EXperiments and Technologies. 451–463.

[12] IEEE. 2011. Priority based flow control. *Journal Name* (2011). IEEE 802.11Qbb Standard.

[13] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. 2018. {Multi-Path} transport for {RDMA} in datacenters. In *15th USENIX symposium on networked systems design and implementation (NSDI 18)*. 357–371.

[14] NVIDIA. 2024. NVIDIA BlueField Networking Platform. https://www.nvidia.com/en-us/networking/products/data-processing-unit/.

[15] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 123–137.

[16] Shan Sinha, Srikanth Kandula, and Dina Katabi. 2004. Harnessing TCP's burstiness with flowlet switching. In *Proc. 3rd ACM Workshop on Hot Topics in Networks (Hotnets-III)*. Citeseer.

[17] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. 2023. Network Load Balancing with In-network Reordering Support for RDMA. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 816–831.

[18] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. 2017. Let it flow: Resilient asymmetric load balancing with flowlet switching. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 407–420.

[19] Peng Wang, Hong Xu, Zhixiong Niu, Dongsu Han, and Yongqiang Xiong. 2016. Expeditus: Congestion-aware load balancing in clos data center networks. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*. 442–455.

[20] Hong Xu and Baochun Li. 2014. TinyFlow: Breaking elephants down into mice in data center networks. In *2014 IEEE 20th International Workshop on Local & Metropolitan Area Networks (LANMAN)*. IEEE, 1–6.

[21] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. 2017. Resilient datacenter load balancing in the wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 253–266.

[22] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*. 1–14.

[23] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 523–536.