PIAS: Practical Information-Agnostic Flow Scheduling for Data Center Networks

[1]Wei Bai, [1]Li Chen, [1]Kai Chen, [2]Dongsu Han, [3]Chen Tian, [1,4]Weicheng Sun
[1]The SING Lab, CSE Department, Hong Kong University of Science and Technology
[2]KAIST [3]HUST [4]SJTU
{wbaiab, lchenad, kaichen}@cse.ust.hk dongsu_han@kaist.ac.kr
tianchen@hust.edu.cn wsunad@connect.ust.hk

ABSTRACT

Many existing data center network (DCN) flow scheduling schemes minimize flow completion times (FCT) based on prior knowledge of flows and custom switch designs, making them hard to use in practice. This paper introduces, PIAS, a practical flow scheduling approach that minimizes FCT with no prior knowledge using commodity switches. At its heart, PIAS leverages multiple priority queues available in commodity switches to implement a Multiple Level Feedback Queue (MLFQ), in which a PIAS flow gradually demotes from higher-priority queues to lower-priority queues based on the bytes it has sent. In this way, short flows are prioritized over long flows, which enables PIAS to emulate Shortest Job First (SJF) scheduling without knowing the flow sizes beforehand. Our preliminary evaluation shows that PIAS significantly outperforms all existing information-agnostic solutions. It improves average FCT for short flows by up to 50% and 40% over DCTCP [3] and L2DCT [16]. Compared to an ideal information-aware DCN transport, pFabric [5], it only shows 4.9% performance degradation for short flows in a production datacenter workload.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design; Performance

Keywords

Data Center Networks; Flow Scheduling; TCP

Hotnets '14, October 27–28, 2014, Los Angeles, CA, USA.

http://dx.doi.org/10.1145/2670518.2673871.

1. INTRODUCTION

There has been a virtually unanimous consensus in the community that minimizing the flow completion times (FCT) is one of the most important goals for data center network (DCN) transport designs [3–5, 9, 13, 15, 16, 20, 22]. This is because many of today's cloud applications, such as web search, online social networking, and retail, etc. have very demanding latency requirements; even a small delay directly affects application performance and degrades user experience [3, 16].

To minimize FCT, many "ideal" solutions [5, 13, 15, 22] assume accurate flow information, such as flow sizes and deadlines. For example, PDQ, pFabric and PASE [5, 13, 15] all assume that the flow size is known a priori and attempt to approximate Shortest Job First (SJF), which is the optimal scheduling discipline for minimizing the average FCT over a single link. However, to ensure near-optimal performance, they require non-trivial modifications on applications to expose accurate flow information to the transport layer.

In another line, congestion control-based approaches, such as DCTCP, HULL and L2DCT [3, 4, 16], try to reduce FCT without relying on flow information by keeping low queue occupancy using Explicit Congestion Notification [18]. However, the performance is sub-optimal because they rely on fair bandwidth sharing with reactive and implicit rate control, which is known to be ineffective [5].

Motivated by these observations, this paper asks a fundamental question: *without a priori knowledge of accurate flow information, what is the best practical flow scheduling solution that minimizes FCT while being readily implementable in today's DCN with commodity switches?*

The question leads to three key design goals:

- **Information-agnostic:** We must not assume a priori knowledge of flow size information being available from the applications.
- **FCT minimization:** The solution must be able to enforce an optimal information-agnostic flow scheduling. It should minimize average and tail FCTs for short flows, while not adversely affecting FCTs of long flows.
- **Readily-deployable:** The solution must work with ex-

isting commodity switches in DCNs and be backward compatible with legacy TCP/IP stacks.

In this paper, we present PIAS, a practical information-agnostic flow scheduling scheme that minimizes FCT in DCNs. PIAS mimics SJF with no prior information. At its heart, PIAS leverages multiple priority queues available in existing commodity switches to implement a Multiple Level Feedback Queue (MLFQ) as shown in Figure 1. A PIAS flow in its lifetime gradually demotes from higher-priority queues to lower-priority queues based on the bytes it has sent. In this way, PIAS ensures short flows to finish in the higher-priority queues and, thus, be generally prioritized over long flows. As a result, PIAS ends up effectively emulating SJF without knowing the flow sizes beforehand.

PIAS is partially inspired by Least Attained Service (LAS) policy [17] that gives service to the job that has received the least amount of service in time-sharing systems. However, directly adopting LAS has two problems: First, enabling LAS on switches requires comparing the amount of bytes transferred for each flow, which is not supported by commodity switches. Second, pure LAS can result in starvation especially when multiple large flows share a link. PIAS's MLFQ effectively solves the first problem while mitigating the second problem. This is because large flows in PIAS will eventually sink to the lowest priority queue, and they will share the link fairly in a FIFO manner, mitigating the starvation problem.

Realizing the goals of PIAS and the benefits of MLFQ requires solving three challenges. First, how to determine the demoting threshold for each queue of MLFQ? Second, how do we ensure optimal performance under realistic workload in which flow size distribution varies across time and space (i.e., links)? Third, how to ensure that PIAS works compatibly with legacy TCP/IP stacks in production DCNs?

We address the first challenge by deriving a set of optimal demoting thresholds through solving a FCT minimization problem. We further prove that the derived threshold setting is robust to a reasonable range of traffic distributions.

A particular set of thresholds works best within a certain range of traffic distributions. Thus, PIAS adaptively adjusts the thresholds to keep up with the traffic dynamics. However, the key problem is that the mismatch between thresholds and underlying traffic is inevitable in realistic traffic traces. Once happens, short flows may be adversely affected by large ones in a queue, impacting their latency. To address this, we maintain low queue occupancy using ECN-based rate control [3, 4, 16], so that short flows always see small queues and will not be seriously delayed even if they are mistakenly placed in a queue with long flows.

Finally, we employ DCTCP for rate control. A potential problem is that many concurrent short flows can collude and starve a long flow, triggering TCP timeouts and spurious retransmissions. In response, PIAS differentiates packet starvation from real packet losses and reacts accordingly.

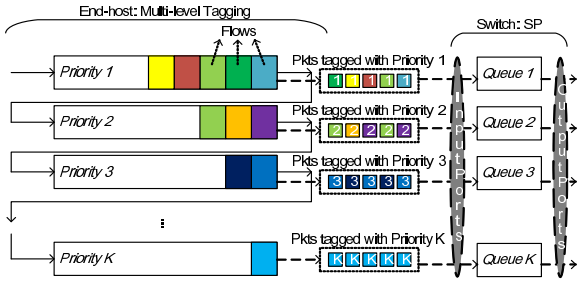


Figure 1: PIAS overview.

PIAS ensures that all its mechanisms can be implemented in a shim layer over NIC without touching the TCP stack.

Our preliminary ns-2 simulations show that PIAS outperforms all existing information-agnostic solutions in realistic DCN workloads, reducing the average FCT for short flows by up to 50% and 40% compared to DCTCP and L2DCT [3, 16] respectively. In addition, our results show that PIAS delivers comparable performance (a 4.9% gap for short flows in data mining workload) to an ideal information-aware DCN transport, pFabric [5].

2. PIAS DESIGN

At a high level, PIAS's main mechanisms include packet tagging at end hosts, priority queueing and ECN marking on switches, and rate control. Figure 1 gives the overview of PIAS.

Packet tagging at end-hosts: PIAS tags packets with priority at the end host. When a new flow starts, its packets will be tagged with the highest priority P_1. As more bytes are sent, packets of this flow will be tagged with decreasing priorities P_j. We denote the thresholds for demoting the priorities as α_j. Such packet tagging is easy to implement in a shim layer over the NIC.

A key challenge here is to determine the set of demoting threshold to minimize the average FCT. We derive the thresholds by solving an FCT minimization problem over a single link (§3). Theoretical analysis show that our threshold setting is robust to a certain range of traffic distributions defined by the load and the flow size distribution (§3). This enables us to perform packet tagging at the end hosts, while achieving good performance.

As the flow size distribution and the load can vary significantly over time, PIAS can periodically measures them in the network [1, 8, 24]and adaptively adjusts the thresholds. However, the mismatch between the threshold setting and the underlying traffic is inevitable because past measurements cannot always predict the future. Once a mismatch happens, it can significantly impact short flows' latency. Thus, mitigating the impact is required for PIAS to operate in highly dynamic datacenter networks. Our solution is to use ECN.

Priority queueing and ECN marking: The PIAS switches enable two built-in functions available in most commodity switches [15]:

- Priority scheduling: Packets are dequeued strictly based on their priorities.
- ECN marking: Packets are marked with Congestion Experienced (CE) if the instant buffer occupancy is greater than the ECN marking threshold.

With priority scheduling at switches and packet priority tagging at end hosts, PIAS performs MLFQ-based flow scheduling in the network with stateless switches. Packets with different priority tags are classified into different priority queues. When a link is idle, packets from the highest non-empty priority queue are transmitted.

Our intention to employ ECN is to mitigate the effect of the mismatch between the demoting thresholds and the traffic distribution. By maintaining low queue occupancy, flows always see small queues, and thus will not be seriously delayed even if they are placed in a queue with a long flow due to non-optimal threshold settings. In addition, ECN also prevents packet drops (e.g., incast [3, 21, 23]) which greatly affects the flow completion time in practice.

Rate control: PIAS uses DCTCP [3] for rate control. Other legacy TCP protocols can also be integrated into PIAS as long as they are ECN-enabled. A key issue is to prevent packet trapping. When packets of a large flow get trapped in a low priority queue for long time, this may trigger TCP timeouts and retransmissions. The retransmit packets would increase the buffer occupation, leading to an aggressive ECN marking or even packet drops in the worst case.

To address this, we adopt a probe based approach inspired by [5, 15]. When a TCP flow experiences a fixed number (e.g., two) of consecutive timeouts, PIAS end-hosts hold all retransmitted packets and periodically generate small probe packets with only one-byte payload for this flow. When PIAS receives ACKs for the probe packets, it indicates that the previous packets were dropped. PIAS then releases the held retransmitted packets to the network. When PIAS receives ACKs for previous packets, it indicates that these previous packets were trapped rather than dropped, and then PIAS deems the retransmission unnecessary and discards the held packets accordingly. Note that the generation of probe packets and their corresponding ACKs along with holding and releasing retransmitted TCP packets can be implemented entirely at the end-hosts within a shim layer, without modifying the TCP stack.

Discussion: The key idea of PIAS is to emulate SJF scheduling which is optimal to minimize the average FCT over a single link. An optimal algorithm that schedules flows over an entire data center fabric with multiple links is not known [5]. In this sense, similar to pFabric [5], PIAS also makes local prioritization decisions in every link. This approach in theory may lead to performance loss over an ideal fabric [15]. For example, when a flow traverses multiple hops and gets dropped at the last hop, causing bandwidth to be wasted on the upstream links that could otherwise have been used to schedule other flows.

To address this problem, we note that some existing solutions [13, 15] leverage arbitration, where a common network entity allocates rates to each flow based on global network visibility. Although arbitration can bring some benefits, it is challenging to implement and often requires non-trivial switch modifications [13] or building a complex control plane [15], which is against our design goal. Fortunately, local-decision based solutions maintain very good performance for most scenarios [5] and only experience performance loss at extremely high loads, e.g., over 90% [15]. Note most datacenter networks operate at moderate loads (e.g.,30% [6]).

3. DETERMINING THE THRESHOLDS

We describe our formulation to set the thresholds in MLFQ to minimize the average FCT over a single link. We identify this problem as a Sum-of-Linear-Ratios problem, and provide a method to derive the optimal thresholds analytically for any given load and flow size distributions. We further show the robustness of our threshold setting to traffic variations.

Problem formulation: We consider a bottleneck link and flows that use the link. We assume there are K priority queues P_j $(1{\le}j{\le}K)$ with P_1 having the highest priority. We denote the threshold for demoting to the priority queue j (from $j{-}1$) as α_{j-1} $(2{\le}j{\le}K)$. We denote the cumulative density function of flow size distribution as $F(x)$. Thus $F(x)$ is the probability that a flow size is not larger than x. Note we make no assumptions about the properties of $F(x)$.

A flow of size $x{\in}[\alpha_{j-1},\alpha_j)$ experiences delays from the highest priority queue up to the j-th priority queue. We denote T_j as the average time spent in the j-th queue. Let x^+ be the size of this flow tagged with P_j, the last priority it will experience. Thus, $x^+{=}x{-}\alpha_{max}(x)$, where $\alpha_{max}(x)$ is the largest demoting threshold less than x, and let $j_{max}(x)$ be the index of this threshold.

The average FCT for this flow can be written as the sum of the delays in each queue: $T(x){=}\sum_{l=1}^{j_{max}(x)}T_l{+}\frac{x^+}{1-\rho_{j_{max}(x)}}$. The second term is upper-bounded by $T_{j_{max}(x)}$. Thus an upperbound for $T(x)$ is $T(x){\le}\sum_{l=1}^{min(j_{max}(x)+1,K)}T_l$.

We have the following optimization problem, where we choose a optimal set of thresholds $\{\alpha_j\}$ to minimize the objective: the average FCT of flows on this bottleneck link:

$$\begin{aligned} \min_{\{\alpha_j\}} \quad & \mathcal{T}{=}\sum_{l=1}^{K}(\theta_l\sum_{m=1}^{l}T_m){=}\sum_{l=1}^{K}(T_l\sum_{m=l}^{K}\theta_m) \\ \text{subject to} \quad & \alpha_0{=}0,\alpha_K{=}\infty \\ & \alpha_{j-1}{<}\alpha_j,j{=}1,...,K \end{aligned} \tag{1}$$

Analysis: Let $\theta_j{=}F(\alpha_j){-}F(\alpha_{j-1})$ be the percentage of flows with sizes in $[\alpha_{j-1},\alpha_j)$. With a traffic load of ρ, the average time for a flow to stay in the lth queue, T_l, can be expressed as [14]: $T_l{=}\frac{\theta_l\rho}{1-\rho F(\alpha_{l-1})}$. Since $\sum_{m=1}^{l}\theta_m{=}\sum_{m=l}^{K}F(\alpha_m){-}F(\alpha_{m-1})$, we can re-express the objective as: $\mathcal{T}{=}\sum_{l=1}^{K}T_l(1{-}F(\alpha_{l-1}))$. $\sum_{l=1}^{K}\theta_m{=}1$.

Because $\sum_{l=1}^{K}\theta_m=1$, we can finally transform the problem as:

$$\max_{\{\theta_l\}} \quad \mathcal{T}''=\sum_{l=1}^{K-1}\frac{\theta_l}{1-\rho(\sum_{m=1}^{l-1}\theta_m)}+\frac{1-\sum_{m=1}^{K-1}\theta_m}{1-\rho\sum_{m=1}^{K-1}\theta_m} \qquad (2)$$

which is a Sum-of-Linear-Ratios (SoLR) problem [19], a well-known class of fractional programming problems. The only constraint is that $\theta_l \geq 0, \forall l$. We find this formulation interesting because the upperbound of average FCT is independent of the flow distribution $F(x)$, and only concerns θs, which represent the percentages of this link's traffic in different queues. Note that $F(x)$ is needed only when we calculate the thresholds, which means that we can first obtain the optimal set of θs for all links, and then derive the priority thresholds based on $F(x)$.

Solution method: Generally, SoLR problem is $\mathcal{NP}$-hard [7], and the difficulty to solve this class of problems lies in the lack of useful properties to exploit. For our problem, however, we find a set of properties that can lead to a closed-form analytical solution to Problem 2. We describe the derivation procedure as follows:

Consider the terms in the objective. Since the traffic load $\rho \leq 1$, we have $\rho(\sum_{m=1}^{l-1}\theta_m) \leq \sum_{m=1}^{l-1}\theta_m$. Also, $\theta_l+\sum_{m=1}^{l-1}\theta_m = \sum_{m=1}^{l}\theta_m \leq 1$. Thus we have:

$$\theta_l \leq \sum_{m=l}^{K}\theta_m = 1-\sum_{m=1}^{l-1}\theta_m \leq 1-\rho(\sum_{m=1}^{l-1}\theta_m) \qquad (3)$$

The property to exploit is as follows: each term in the summation, $\theta_l/(1-\rho\sum_{m=1}^{l-1}\theta_m)$, is no larger than 1, and to maximize the summation, we should make the numerator and denominator as close as possible, so that the ratio is close to 1.

Consider the first two portions, θ_1 and θ_2. By making the the numerator and denominator equal, we have:

$$\theta_2 = 1-\rho\theta_1 \qquad (4)$$

We can obtain the expression of the third portion and all the portions after it iteratively:

$$\theta_3 = 1-\rho(\theta_2+\theta_1) = 1-\rho(1-(1-\rho)\theta_1) \qquad (5)$$

By the formula $\theta_l = 1-\rho(\sum_{m=1}^{l-1}\theta_m)$, each portion θ_l can be represented by an expression of θ_1 iteratively. And by the constraint $\sum_{l=1}^{K}\theta_l=1$, we can obtain the analytical expressions of all θs, which represents the percentages of the traffic in different priority queues on the link. Thus, given traffic load θs and the flow size distribution $F(\cdot)$, it gives an optimal set of demontion thresholds in the unit of bytes.

Robustness to traffic variations: Each flow may be bottleneck at different links, each of which usually observes different flow size distributions. Since PIAS's priority tagging is distributed at the end-hosts, and end-hosts do not know the bottleneck links of their flows in advance, it is important that the threshold setting (θs) is robust to varying flow size distributions as well as varying loads (ρ). In this section, we show that 1) a wide range of load (ρ) and threshold settings (θs) leads to close-to-optimal FCT; and 2) given a reasonable threshold setting, the objective remains close-to-optimal for most distributions even under varying loads.

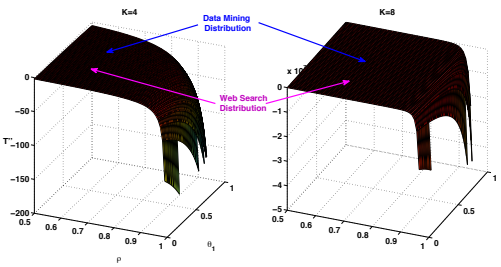


Figure 2: Objective ($\mathcal{T}''$) with respect to load (ρ) and portion of traffic in the first queue (θ_1).

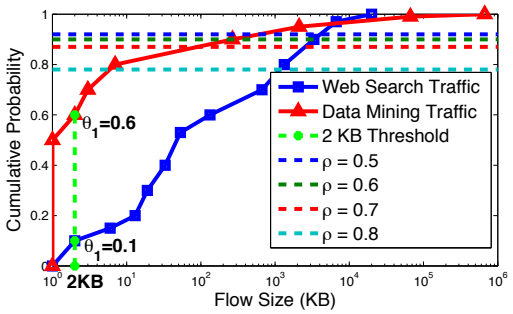


Figure 3: Flow size distributions for web search and data mining traffic. Different ρ correspond to different upperbound of θ_1. A threshold of 2 KB satisfies the optimality criterion for almost all distributions below 80% load. (K=8).

Since every portion θ_l can be represented by θ_1, the objective function in Problem 2 can be expressed using only θ_1 and ρ. We plot the objective function with respect to these two variables in Figure 2. Consider the two cases where K=4 and 8. In both cases, we can see that the range of possible (θ_1,ρ) pairs that result in close-to-optimal solutions is large. This shows that PIAS is robust to different traffic distributions. Note the algorithm we gave in §3 is only one method to obtain a close-to-optimal solution without complicated computation. Also, note that with increase in the number of queues in the switch (K), the range for optimal (θ_1,ρ) becomes larger, which means that we should use a large number queues in the switch for increased robustness.

Consider the traffic distributions in Figure 3. For the first threshold θ_1 of c KB, the corresponding percentage of traffic is $F^{-1}(c)$. If the threshold of the first queue is set to be 2 KB, it means that θ_1=60% of traffic will complete in this queue for data mining distribution [12], and θ_1=10% for web search distribution [3]. However, as shown in Figure 2, when there are 8 queues at the switch (K=8), the range for possible optimal θ_1 is large. The dashed lines in Figure 3 indicate the largest values that θ_1 can take without affecting the optimality at different traffic loads. The significance is the following: for any traffic distribution $F(\cdot)$, if $F(2KB)$ is below the dashed line for ρ=0.8, then the 2 KB threshold is optimal for $F(\cdot)$ for traffic load no more than 80%. We find this optimality criterion for 2 KB is easily satisfiable in

realistic flow size distributions because only extremely biased distributions, *e.g.*, a distribution that more than 80% of the flows are less than 2 KB, can violate it.

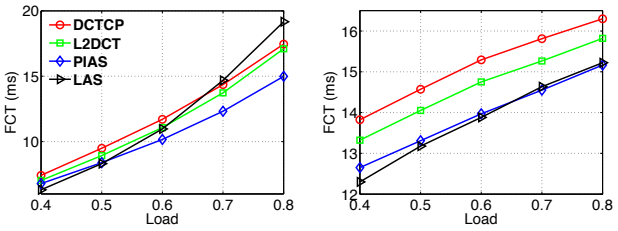
4. EVALUATION

We now evaluate the performance of PIAS using simulations [2]. We answer three key questions:

- **How does PIAS perform compared to information agnostic schemes?** PIAS outperforms DCTCP [3] and L2DCT [16], and significantly improves the average flow completion times for short flows by 50% and 40% respectively. Furthermore, PIAS is close to LAS for short flows. Also, by solving the starvation problem, it greatly outperforms LAS for long flows, reducing its average FCT by 50% in the web search workload.
- **What is the effect of number of queues to PIAS?** PIAS achieves reasonable performance even with two queues. However, using a larger number of queues improve the overall performance.
- **How does PIAS perform compared to pFabric?** Without a priori knowledge of flows, PIAS achieves the comparable performance to pFabric for short flows (within a 4.9% gap in the data mining workload).

We use a leaf-spine topology with nine leaf switches to four spine switches. Each leaf switch has 16 10Gbps downlinks (144 hosts) and four 40Gbps uplinks to the spine, forming a non-oversubscribed network. The base round-trip time across the spine (four hops) is 85.2μs. We use packet spraying [10] for load balancing. We use the web search workload [3] and the data mining workload [12], taken from production DCNs, whose flow size distributions are shown in Figure 3.

4.1 Comparison with information-agnostic schemes

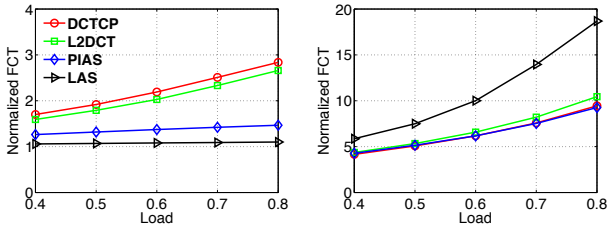


(a) Web search workload (b) Data mining workload

Figure 4: Overall average flow completion time

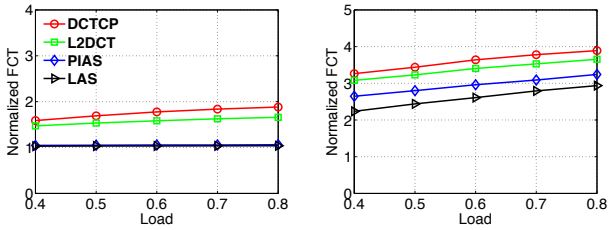
We mainly compare PIAS (using eight queues) with LAS, DCTCP [3] and L2DCT [16].

Overall performance: Figure 4 shows the overall average FCT of information-agnostic schemes under different workloads and load levels. From the figure, we see that PIAS performs well overall. First, PIAS has an obvious advantage over DCTCP and L2DCT in all the cases. Second, PIAS is comparable to LAS in the data mining workload, but outperforms LAS by 28% (at load 0.8) in the web search workload.



(a) (0,100KB]: Avg (b) (10MB,∞): Avg

Figure 5: Web search workloads: Normalized FCT



(a) (0,100KB]: Avg (b) (10MB,∞): Avg

Figure 6: Data mining workloads: Normalized FCT

This is because PIAS effectively mitigates the starvation between long flows unlike LAS. In the data mining workload, there are not so many large flows sharing the same link. As a result, LAS does not suffer from starvation as much.

Breakdown by flow size: We now breakdown the average FCT across different flow sizes, (0,100KB] and (10MB,∞) (Figure 5 and 6).[1] We normalize each flow's actual FCT to the best possible value it can achieve in the idle fabric.

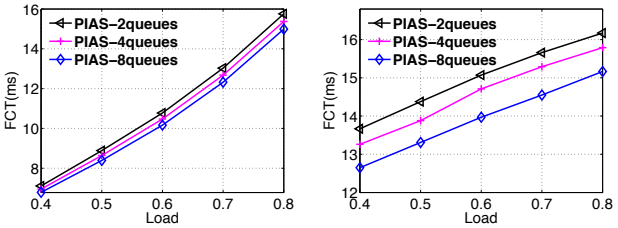
For short flows in (0,100KB], we find that PIAS significantly outperforms both DCTCP and L2DCT, improving the average FCTs by up to 50% and 40% respectively. This is because DCTCP and L2DCT use reactive rate control at the end hosts, which is not as effective as PIAS for in-network flow scheduling. We further observe that PIAS achieves similar performance as LAS for short flows. PIAS performs slightly worse than LAS in the web search workload when there is a congestion event (e.g., packet drop or ECN).

For long flows in (10MB,∞), we find that PIAS is slightly worse than LAS in the data mining workload, but performs significantly better in the web search workload (50% reduction in FCT at load 0.8). This is because, in the web search workload, it is common that multiple large flows are present in the same link. In such scenarios, LAS always stops older flows to send new flows. Since large flows usually take a very long time to complete, it causes a serious starvation problem. However, with PIAS, large flows receive their fair sharing in the lowest priority queue, which mitigates this problem. PIAS performs comparable to DCTCP under the web search workload and achieves 17% lower FCT in the data mining workload.

4.2 Effect of number of queues to PIAS

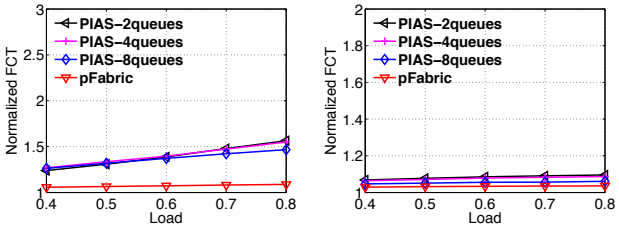
We evaluate the performance of PIAS with different num-

[1]We omit the performance in (100KB,10MB] for space limitation.



(a) Web search workload (b) Data mining workload

Figure 7: Overall average FCT with different queues



(a) Web search workload (b) Data mining workload

Figure 8: Average normalized FCT for (0,100KB]

bers of queues. Figure 7 shows the overall performance and Figure 8 shows the performance of small flows as we increase the number of queues from two to eight.

In Figure 7, we see that PIAS generally achieves better performance as we increase the number of queues. This trend is expected because with more queues, we can perform more fine-grained flow prioritization. Figure 8 reveals that for short flows the performance improvement is relatively small as we increase the number of queues (about 6% difference for both web search and data mining workloads). Actually, we find that the improvement in FCT mainly comes from the medium size flows (100KB,10MB] and large flows (10MB,∞). Due to the space limitation, we omit detailed data and figures here. For example, our results show that PIAS with eight queues reduces the average FCT by 48% for flows of size (100KB,10MB] compared to two-queue PIAS in the data mining workload.

4.3 Comparison with information-aware schemes

We compare PIAS to an ideal information-aware approach for DCN transport, pFabric [5], on small flows using the two workloads. We note that the most recent work PASE [15] can achieve better performance than pFabric under some workload (*e.g.*, very high load and single rack). But in our topology setting with realistic workloads, pFabric is better than PASE and PDQ [13]; pFabric actually achieves near-optimal performance. Thus, we directly compare PIAS with pFabric in Figure 8. In general, PIAS delivers comparable average FCT for short flows as pFabric, particularly within 4.9% in the data mining workload.

We find that the gap between PIAS and pFabric is smaller in the data mining workload than in the web search workload. This is mainly because the data mining workload is more skewed in terms of the flow size distribution than the web search workload. Around 82% of data mining flows are smaller than 100KB, while only 54% of web search flows are smaller than 100KB. For the web search workload, it is more likely that large flows will coexist with short flows in the high priority queues at the start of the flow, which in turn increases the queueing delay for short flows.

5. RELATED WORK

Many transport solutions try to minimize FCT. We classify them in to two categories: information-agnostic [3, 4, 16] and information-aware approaches [5, 13, 15].

Information-agnostic solutions [3, 4, 16] improve FCT by keeping low queue occupancy. L2DCT [16] adds bytes sent information to DCTCP [3]. HULL [4] further improves the latency of DCTCP by trading network bandwidth. In addition, RCP [11] emulates processing sharing to minimize FCT in the Internet.

Information-aware solutions [5, 13, 15] attempt to approximate the ideal, preemptive SJF scheduling. PDQ [13] employs switch arbitration and uses explicit rate control. pFabric [5] implements decentralized in-network prioritization. PASE [15] synthesizes existing solutions to provide better performance. These solutions can potentially provide superior performance but require non-trivial modifications to both end hosts and the switch hardware.

There are also some other works [9, 20, 22] focusing on meeting flow deadlines. D3 [22] achieves explicit rate control to flows while D2TCP [20] and MCP [9] add deadline-awareness to ECN-based window adjustment.

6. CONCLUSION

In this paper, we introduced PIAS, a practical information-agnostic flow scheduling to minimize FCT in DCNs. Without a priori knowledge of flow information, PIAS minimizes FCT by implementing a Multiple Level Feedback Queue (MLFQ) scheduling discipline. PIAS builds on a theoretical foundation and is readily deployable in today's DCNs with no changes to the legacy TCP stack, switch hardware, and applications. Our preliminary evaluation shows that PIAS outperforms all information-agnostic schemes and provides comparable benefits to information-aware schemes on short flows. We plan to implement a PIAS prototype and evaluate its performance with real applications.

Acknowledgements This work is supported by the Hong Kong RGC under ECS 26200014, the China 973 Program under Grant No. 2014CB340303, the ICT R&D program of MSIP/IITP, Republic of Korea [14-911-05-001], and the Basic Science Research Program of NRF funded by MSIP, Rep. of Korea (2013R1A1A1076024). We thank Mohammad Alizadeh for sharing codes of pFabric, Haitao Wu for insightful discussions, and the HotNets reviewers for their constructive comments.

7. REFERENCES

[1] "NetFlow," http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.

[2] "The Network Simulator NS-2," http://www.isi.edu/nsnam/ns/.

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

[4] 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 *NSDI 2012*.

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

[6] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *IMC 2010*.

[7] J. G. Carlsson and J. Shi, "A linear relaxation algorithm for solving the sum-of-linear-ratios problem with lower dimension," *Operations Research Letters*, vol. 41, no. 4, pp. 381–389, 2013.

[8] J. Case, M. Fedor, M. Schoffstall, and C. Davin, "A simple network management protocol (SNMP)," 1989.

[9] L. Chen, S. Hu, K. Chen, H. Wu, and D. Tsang, "Towards Minimal-Delay Deadline-Driven Data Center TCP," in *HotNets 2013*.

[10] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *INFOCOM 2013*.

[11] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, 2006.

[12] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *SIGCOMM 2009*.

[13] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," in *SIGCOMM 2012*.

[14] L. Kleinrock, "Queueing systems, volume II: computer applications," 1976.

[15] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. Liu, and F. Dogar, "Friends, not Foes - Synthesizing Existing Transport Strategies for Data Center Networks," in *SIGCOMM 2014*.

[16] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *INFOCOM 2013*.

[17] I. A. Rai, G. Urvoy-Keller, M. K. Vernon, and E. W. Biersack, "Performance Analysis of LAS-based Scheduling Disciplines in a Packet Switched Network," in *SIGMETRICS 2004*.

[18] K. Ramakrishnan, S. Floyd, D. Black *et al.*, "RFC 3168: The addition of explicit congestion notification (ECN) to IP," 2001.

[19] S. Schaible and J. Shi, "Fractional programming: the sum-of-ratios case," *Optimization Methods and Software*, vol. 18, no. 2, pp. 219–229, 2003.

[20] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *SIGCOMM 2012*.

[21] V. Vasudevan *et al.*, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *SIGCOMM 2009*.

[22] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *SIGCOMM 2011*.

[23] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast Congestion Control for TCP in data center networks," in *CoNEXT 2010*.

[24] M. Yu, A. G. Greenberg, D. A. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim, "Profiling Network Performance for Multi-tier Data Center Applications." in *NSDI 2011*.