Aemon: Information-agnostic Mix-flow Scheduling in Data Center Networks

Tao Wang
Huazhong University of Science and Technology

Hong Xu
City University of Hong Kong

Fangming Liu
Huazhong University of Science and Technology

ABSTRACT

Data center networks carry a mix of flows, some with deadlines and some without. Existing mix-flow transport designs assume prior knowledge of flow sizes, which may not hold in practice. Without such information, mix-flow scheduling becomes particularly challenging due to (1) the lack of precise rate control of deadline flows with minimal impact on non-deadline flows; (2) difficulty in assigning priority to both two types of flows.

We present Aemon, an information-agnostic mix-flow transport. Aemon relies on a new congestion control mechanism based on urgency—the ratio between the elapsed transmission time and the remaining time to deadline—to strategically modulate the sending rate of deadline flows. In addition, Aemon uses a novel two-level priority scheduling policy to differentiate mix flows. As time goes, a deadline flow's priority level increases as its urgency rises, and a non-deadline flow's priority level decreases as it sends more bytes similar to PIAS to approximate Shortest-Job-First policy. While in the same priority level, non-deadline flows take precedence to avoid possible starvation caused by aggressive deadline flows. Extensive simulations on ns-2 show that Aemon outperforms existing information-agnostic schemes and is only slightly worse than state-of-the-art Karuna.

CCS CONCEPTS

• **Networks** → **Transport protocols**;

KEYWORDS

Data center network, Flow scheduling, Transport protocol

APNet'17, Hong Kong, China
 978-1-4503-5244-4/17/08…$15.00
DOI: 10.1145/3106989.3106992

1 INTRODUCTION

Data center networks (DCNs) host tons of different services. Some (e.g. interactive social services, etc.) usually have tight latency requirements; thus minimizing flow completion time (FCT) is the main performance objective[5, 6]. Some other services (e.g. web search, online advertising, etc.) employ online data intensive applications, which generate many flows with strict deadlines [18, 19]. The results exclude flows that miss their deadlines, which leads to performance degradation and hurts user experience [8, 18, 19]. Consequently, this creates a mix-flow environment in DCNs with both deadline flows and non-deadlines ones [8]. Deadline flows need to meet their deadlines while non-deadline flows need to finish as quickly as possible.

There exists a large body of work for either improving deadline miss rate for deadline flows [14, 18, 19] or cutting FCT [5–7, 12, 15]. Yet mix-flow scheduling that jointly considers both types of flows only starts to receive attention more recently. In particular, Karuna [8] is proposed to address mix-flow scheduling using both rate control at end-hosts and priority scheduling at switches. By assuming complete prior knowledge of flows including flow sizes, Karuna assigns a deadline flow a rate equal to the remaining flow size divided by deadline, and also the highest priority to minimize deadline miss rate. This ensures minimum bandwidth is used by deadline flows, leaving more headroom for non-deadline flows to achieve shorter FCTs.

Unfortunately, in practice such flow characteristics cannot be obtained a prior in many cases (§2). For example, multi-stage jobs with pipelining keeps sending data as soon as they are generated. Stream processing applications [1, 4] also continuously generate and send data for processing. In these cases, it is hard to know the exact size of a flow at the beginning of the transmission [7, 9].

Thus we ask this question: how can we effectively schedule mix flows without prior knowledge of flow sizes? The challenge manifests in both aspects of the problem: rate control, and flow scheduling. First, without flow size, it is nearly impossible to precisely control the sending rate of deadline flows in order to minimize the impact on non-deadline flows. This implies that directly applying Karuna [8] or other flow size-driven rate control schemes like D2TCP [18] or D^3 [19] is not feasible. Second, Multi-Level Feedback Queue (MLFQ)

[10] can efficiently solve flow scheduling without size information as shown by PIAS [7], when the objective is to minimize FCT. In a mix-flow setting where we need to jointly consider both deadline miss rate and FCT, it is not clear how deadline and non-deadline flows should be prioritized against each other in order to achieve the best performance. Strictly prioritizing non-deadline flows clearly does not work since aggressive non-deadline flows can dramatically slow down the deadline flows. Strictly prioritizing deadline flows as in Karuna [8] may lead to starvation of non-deadline flows due to lack of precise rate control without knowing flow size infomation.

In this paper, we present Aemon, an information-agnostic DCN transport that balances the interests of deadline and non-deadline flows. Aemon addresses the above design challenges as follows. At the end-host, Aemon modulates the TCP congestion window based on flow urgency which does not require flow size. A flow's urgency is the ratio between the elapsed transmission time and the remaining time to its deadline. That is, a flow's urgency increases as the deadline approaches. The urgency of non-deadline flows is fixed throughout its lifetime. Upon congestion, lesser urgent deadline flows cut down their window sizes more aggressively than other flows to reduce congestion and vacate bandwidth, so that urgent deadline flows as well as non-deadline flows whose urgency is higher can still make progress.

For flow scheduling in the network, Aemon relies on a two-level priority scheduling policy. On the first level, Aemon assigns a fixed number of priorities to both types of flows. In other words, the first level differentiates flows with the same type. Deadline flows are differentiated based on their urgency, and non-deadline flows based on the number of bytes sent following Least-Attained-Service (LAS) [7, 17]. Effectively, during the lifetime of a deadline flow, its priority gradually increases as the deadline approaches to reduce deadline miss rate. On the other hand, a non-deadline flow's priority decreases throughout its lifetime in order to approximate Shortest-Job-First (SJF) policy and reduce FCT.

The second level of Aemon's priority scheduling then differentiates flows across different types. That is, within the same priority, we choose to schedule non-deadline flows before deadline flows. The rationale behind this design choice is that prioritizing deadline flows in the second level will significantly affect FCT of non-deadline flows without precise rate control using flow size. We need to protect (short) non-deadline flows from the over-aggressive deadline flows *at the same priority*. Note deadline flows still gain precedence over lower-priority flows.

To validate the effectiveness of Aemon, we conduct large-scale ns-2 [2] simulations using real-world workloads. Compared to PIAS with DCTCP which represents state-of-the-art flow size-agnostic transport, Aemon reduces deadline miss rate by up to 13.2% and average FCT for all non-deadline flows by over 45% under a production data center workload. More specifically, Aemon provides the same 99th-percentile FCT for mice flows, and reduces average FCT of medium and large flows by 46.4% and 43%, respectively, compared to PIAS+DCTCP. Aemon is at most ~19.1% worse than Karuna in terms of deadline miss rate. For non-deadline flows, Aemon reduces the 99th-percentile FCT for mice flows by ~19x and the average FCT of medium flows for 62.5%.

2 BACKGROUND AND MOTIVATION

We first motivate our work by introducing a few examples to explain why flow sizes are hard to obtain in practice. We then discuss why existing DCN transport mechanisms do not work for information-agnostic mix-flow scheduling.

2.1 Motivating Examples

Pipelining. Pipelining is widely used in multi-stage job processing where once generated, data is immediately transferred to the next stage between two successive stages, making it hard to know the flow sizes [9].

Stream Processing. Unlike batch processing frameworks such as MapReduce [11] that process data hourly or weekly, streaming processing frameworks [1, 4] focus on instantaneous data which is processed in real time as soon as they arrive.

Real-time Databases. Another example is real-time databases, such as PipelineDB [3], which are widely used in stock trading, banking, etc. Due to strong interactivity requirements, the transaction data needs to be streamed to the database from the front-end servers within stringent deadlines. Since these data is sent out as soon as they are generated, again the flow size is unknown at the beginning of transmission.

2.2 Inapplicability of Existing Work

Existing work has largely focused on either just deadline flows or just non-deadline flows. To reduce deadline miss rate, i.e. the fraction of flows that miss their deadlines, Earliest-Deadline-First (EDF) based proposals such as pFabric-EDF [6] that use deadline as priority offer the best performance in terms of flow scheduling. They cannot be directly applied in a mix-flow setting since they finish deadline flows too aggressively and hurt the FCT of non-deadline flows, as shown by Karuna [8]. Rate control schemes such as D^3 [19], D^2TCP [18], and Karuna's MCP [8] use both (remaining) flow size and deadline to modulate the congestion window in order to assign just enough bandwidth for flows to finish by the deadline. Though this is friendly to non-deadline flows, precise rate control does not work without flow size information.

To minimize FCT, many new rate control schemes (e.g. L2DCT [16]) have been proposed following [5]. pFabric [6] offers superior performance by approximating the ideal SJF through decentralized in-network prioritization. However, it assumes the complete knowledge of flow sizes. Inspired by LAS [17], information-agnostic flow scheduling proposals such as PIAS [7] use MLFQ to gradually demote the priority of a flow have been demonstrated to work well. They are however deadline-agnostic and do not work well in the mix-flow setting. Karuna [8] proposes to assign deadline flows highest priority in order to protect their bandwidth allocation. As we will show in §4.2, without flow size and precise rate control, this results in poor FCT for non-deadline flows which starve by aggressive deadline flows using excessive bandwidth.

Other transport schemes, like PDQ [14] and PASE [15], need complex mechanisms for rate arbitration, which is not readily deployable in DCN.

3 SYSTEM DESIGN

We now present the design of Aemon in this section. Aemon comprises of two main mechanisms: congestion control and in-network flow scheduling. It adopts a urgency-based congestion control protocol, UCP, which adjusts the rate of a deadline flow based on a notion of urgency that reflects how close to the deadline the flow is. In addition, Aemon employs a novel two-level priority scheduling discipline called 2LPS. In the first level, 2LPS assigns a fixed number of priorities to both deadline and non-deadline flows. Flows in a higher priority precede those in a lower priority irrespective of the type. In the second level, within the same priority, non-deadline flows take precedence over deadline flows in order to achieve the best trade-off between deadline miss rate and non-deadline flows' FCT.

3.1 UCP: Urgency-based Congestion Control

To handle deadline flows without flow size, the basic idea behind UCP is to modulate the flow's congestion window based on both the urgency of the flow and the extent of congestion in network. The congestion control loop is based on DCTCP [5]. We maintain α to express the extent of network congestion based on ECN as follows:

$$\alpha \leftarrow (1-g)\cdot\alpha + g\cdot F, \tag{1}$$

where the g is the given weight between new samples and old ones, and F the fraction of Congestion Experienced (CE) marked packets in the recent window of data.

Handling deadline flows. We first define the urgency s of a deadline flow as follows:

$$s = \frac{T_e}{T_d - T_e}, \tag{2}$$

where T_e is the elapsed time of the flow since it starts and T_d is the deadline of this flow. A flow's urgency s increases when it approaches the deadline, indicating (most likely) that it needs more bandwidth from the network. Based on the urgency, we construct a gamma-correction penalty function similar to [16, 18]:

$$p = \alpha^s. \tag{3}$$

The penalty p is always between 0 and 1 as α is between 0 and 1.

The congestion window of deadline flows is adjusted according to the following:

$$\text{cwnd} = \begin{cases} \text{cwnd}\cdot(1-p/2), & p > 0, \\ \text{cwnd}+1, & p = 0. \end{cases} \tag{4}$$

If there is no congestion in the network (i.e. $\alpha = 0$) and therefore the penalty p is 0, the window size just grows by one segment. When all on-the-fly packets are CE-marked (i.e. $\alpha = 1$) and the penalty is 1, the window size halves which is identical to TCP.

Handling non-deadline flows. For non-deadline flows, UCP modulates its congestion window in the same way as DCTCP [5], where the urgency is fixed at 1 and hence the penalty p is always equal to α.

$$\text{cwnd} = \begin{cases} \text{cwnd}\cdot(1-\alpha/2), & \alpha > 0, \\ \text{cwnd}+1, & \alpha = 0. \end{cases} \tag{5}$$

Discussion. Though related to D2TCP [18] which relies on flow size to modulate congestion window, UCP is essentially information-agnostic. The key idea of UCP is that when a deadline flow approaches its deadline and becomes more urgent, UCP avoids cutting down its congestion window too much even when the network is congested. When the deadline is still far away, however, UCP tries to back off more aggressively in order to leave more bandwidth to non-deadline flows for shorter FCTs and to deadline flows with high urgency as well. Consider the example in Figure 1. At the beginning of the lifetime of a new deadline flow, its urgency is very low, and UCP aggressively cuts down its window to prioritize non-deadline flows with more bandwidth. As time elapses and the flow's urgency rises above 1, UCP prioritizes it by decreasing the window less than standard DCTCP. In practice, we cap the urgency in a small range (§4.1), to avoid congestive collapse.

3.2 2LPS: Two-level Flow Scheduling

Without flow size information, prior work such as PIAS [7] has shown that using multiple queues in commodity switches to implement a Multi-Level Feedback Queue (MLFQ) is effective in delivering superior FCT performance. Yet PIAS is deadline-agnostic, and here Aemon must manage both deadline and non-deadline flows which is a different problem.

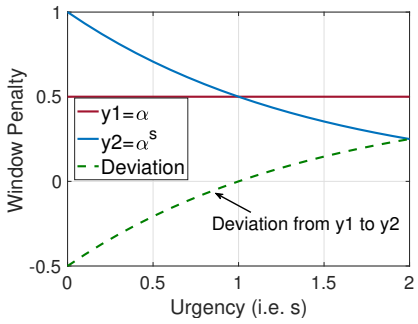


Figure 1: The penalty p for deadline flows in UCP.

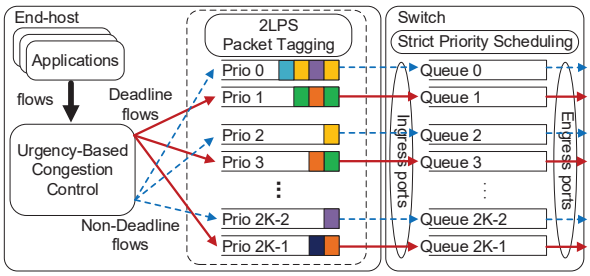


Figure 2: Two-level priority scheduling (2LPS).

A naive idea is to directly apply PIAS and also assign deadline flows priority based on the number of bytes sent. This essentially means that a deadline flow's priority decreases over time, which is problematic and can cause high deadline miss rate as we will show in §4.2.

Another strawman solution is then to prioritize one type of flows over the other, while still using PIAS within non-deadline flows. Strictly prioritizing non-deadline flows clearly does not work since aggressive non-deadline flows can dramatically slow down the deadline flows. Strictly prioritizing deadline flows as in Karuna [8] may lead to starvation of non-deadline flows and does not work either. Recall that without flow size Aemon cannot ensure deadline flows only use the minimum bandwidth. It instead simply gives more bandwidth to urgent deadline flows using UCP. Thus as a flow approaches its deadline, it is increasingly likely that it obtains a higher sending rate far greater than the minimum rate required to meet the deadline, starving the non-deadline flows with lower priority.

Thus we need a mix-flow scheduling policy that achieves two seemingly conflicting goals: (1) prioritize urgent deadline flows when they are about to miss the deadline, and (2) protect (short) non-deadline flows from the over-aggressive deadline flows. Next we discuss 2LPS, a novel two-level priority scheduling policy that also leverages MLFQ to achieve the above two goals.

First-level. The first level priority differentiates flows of the same type. For non-deadline flows, they are assigned K priorities based on the number of sent bytes just like in PIAS. A flow in its lifetime gradually demotes from highest priority to lower priority, approximating the Shortest-Job-First policy without knowing flow size.

To handle deadline flows, 2LPS also assigns K priorities. The priority is based on the urgency of the flow defined in Equation (2) instead of number of bytes sent, in order to prioritize flows that are closer to their deadline. Thus a deadline flow in its lifetime gradually promotes from the lowest priority to higher priority. This achieves the first design goal by ensuring that urgent flows are more protected, while avoiding finishing flows too early before the deadline.

An important aspect of the first-level priority scheduling is that a flow with higher priority level precedes another with lower priority level, no matter the type. For example, a non-deadline flow of priority level 0 will be scheduled before a deadline flow of priority level 1. Thus we avoid the shortfalls of strictly prioritizing one type of flows as discussed already.

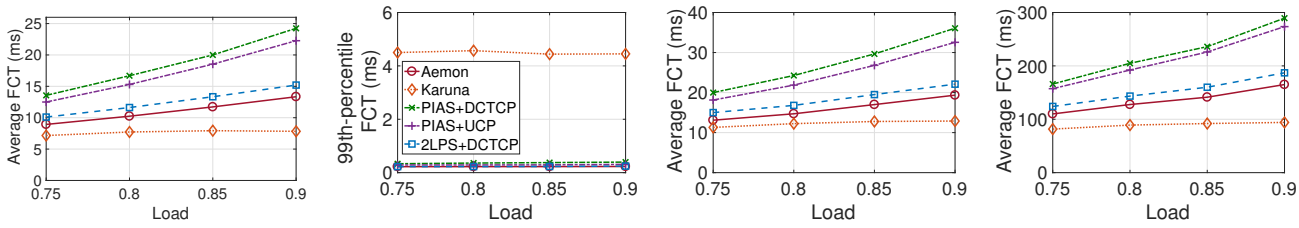
Second-level. The second level of 2LPS then differentiates flows of the same priority. Specifically, within the same priority level, we give precedence to non-deadline flows over deadline flows, in order to avoid possible starvation due to the coarse-grained rate control. This achieves the second goal of protecting non-deadline flows from aggressive use of bandwidth by deadline flows (at the same priority). Taken together, 2LPS uses in total $2K$ priority queues with K priority levels to schedule both types of flows. In the i-th level, $i \in \{0, 1, \ldots, K-1\}$, P_{2i} and P_{2i+1} serve as the priority queues for non-deadline and deadline flows, respectively.

Thresholds of Priority. 2LPS can be implemented by performing packet tagging at the end-host as shown in Figure 2. There are correspondingly two sets of demotion thresholds, $\{\beta_i\}$ and $\{\widetilde{\beta}_i\}$ for non-deadline and deadline flows, respectively. Commodity switches usually have 8 physical priority queues [7], i.e. $K = 4$. For deadline flows, we simply use 10, 1, and 0.1 as the threshold values for P_1, P_3, and P_5, respectively. For non-deadline flows we follow the method in [7] that determines the threshold values based on the flow size distribution information which can be collected over time.

4 EVALUATION

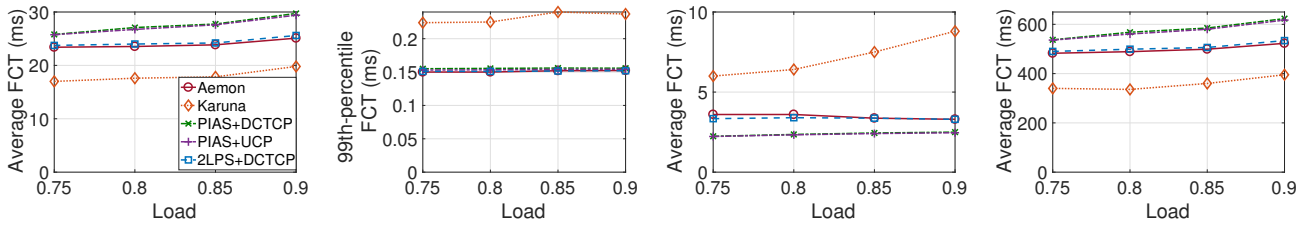
In this section, we evaluate Aemon's performance by conducting extensive packet-level simulations in ns-2 [2]. Our evaluation focuses on following key issues.

- **How Aemon achieves non-deadline flows' FCTs?** Aemon performs the best among all information-agnostic schemes. Specifically, compared to PIAS [7], Aemon reduces average FCT by ~45.1% and ~15.6% at 90% load in web search and data mining workload, respectively.



(a) (0, ∞) (b) (0, 100KB] (c) (100KB, 10MB] (d) (10MB, ∞)

Figure 3: Web search workload: Non-deadline flows' FCTs across different flow sizes.



(a) (0, ∞) (b) (0, 100KB] (c) (100KB, 10MB] (d) (10MB, ∞)

Figure 4: Data mining workload: Non-deadline flows' FCTs across different flow sizes.

- **How Aemon meets deadlines?** Aemon is at most ~19.1% worse than state-of-the-art Karuna with flow size information on deadline miss rate. Compared to PIAS, Aemon reduces deadline miss rate by up to 13.1%.
- **How UCP and 2LPS benefit on its own?** Aemon's UCP reduces the large flow's average FCT by ~11.8% and ~2% in both workloads respectively, compared to using DCTCP [5]. Further, 2LPS alone reduces average FCT by 37.3% and 13.9%, respectively, in both workloads and achieves lower deadline miss ratio compared to PIAS.

4.1 Setup

Fabric Topology. We use the spine-leaf topology. The fabric consists of 144 hosts connected with 9 ToR switches and 4 spine switches in a full mesh. Host and ToR switches are connected via 10Gbps links while ToR and spine switches via 40Gbps links, forming a non-oversubscribed network. The base RTT is ~85μs of which 80μs is spent on the hosts. Similar to [6, 7], we also use packet spraying [12] for load balancing and disable dupACKs to cope with packet reordering.

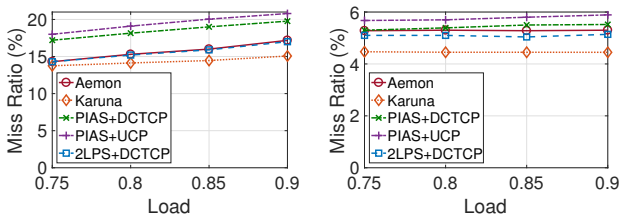
Parameter Settings. As suggested in [8], the ECN threshold is set to 65 packets for 10Gbps link and 250 packets for 40Gbps link. The buffer size of a switch port is 360KB. There are 8 priority queues (i.e. $K = 4$) on each switch and the demotion thresholds are derived accordingly as in §3.2. For UCP we cap the urgency between 0.5 and 1.5 to avoid aggressive cut-down of rate when urgency is low and avoid hogging the bandwidth when urgency is high. 2LPS still uses the original urgency value to derive priority of a deadline flow, and the thresholds are set to 10, 1, 0.1.

Benchmark Workloads. We use two empirical workload traces from production data centers that are widely used in prior work [5–8, 18]: a web search workload [5] and a data mining one [13]. Flows arrive according to a Poisson process and the source-destination pair of each flow is chosen uniformly at random. The flow arrival rate is set to achieve a desired load (i.e. Γ) in the network. Deadlines are assigned as follows, which is consistent with [8]: We control the expected load of active deadline flows (i.e. Γ^D) and record the current load of all active deadline flows (i.e. $\bar{\Gamma}^D$). For a newly generated flow, if $\bar{\Gamma}^D < \Gamma^D$, we assign a deadline to achieve Γ^D as much as possible (minimum deadline is 5ms). Otherwise, we tag this flow as a non-deadline flow. In the simulations below, we fix Γ^D to 75% of the host-to-ToR link capacity and vary Γ from 75% to 90%. For each setting, we run 100000 flows, in which deadline flows account for ~69% and ~58% under 90% load of web search and data mining workload, respectively.

Schemes Compared. We compare the following schemes: **Karuna** [8]: The state-of-the-art mix-flow scheduling transport in DCN. **PIAS+DCTCP** [7]: The baseline scheme which uses PIAS as the state-of-the-art information-agnostic scheduling policy and DCTCP as the congestion control scheme without knowing flow size. **PIAS+UCP**: To study how UCP alone benefits, we use UCP together with PIAS to replace DCTCP at end-host. **2LPS+DCTCP**: Similarly we use 2LPS with DCTCP to study how 2LPS alone benefits.

4.2 Results

FCT for Non-deadline Flows. Figures 3 and 4 plot the FCTs of non-deadline flows under different workloads and



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

Figure 5: Deadline miss ratio of different workloads.

at different load levels.[1] We also break down the FCTs across flow sizes: (1) short flows in (0, 100KB], (2) medium flows in (100KB, 10MB] and (3) large flows in (10MB, ∞).

From Figures 3(a) and 4(a), we observe that among all the information-agnostic schemes, Aemon achieves the lowest average FCT. Specifically, compared to PIAS+DCTCP, Aemon reduces ~45.1% and ~15.6% average FCT in web search (Figure 3(a)) and data mining (Figure 4(a)) workloads, respectively, and reduces ~43% and ~16% large flow average FCT as shown in Figures 3(d) and 4(d). Note that PIAS+DCTCP outperforms Aemon in terms of medium flow average FCT in Figure 4(c). The reason is that short flows take up most of the traffic in the data mining workload. Thus short flows with tight deadlines delay the processing of medium flows in Aemon, while PIAS simply priorities flows with smaller sizes.

Aemon significantly outperforms Karuna in terms of 99th-percentile FCT for short flows, providing ~94.8% and ~35.7% reduction under two workloads respectively (Figure 3(b) and Figure 4(b)). This is because Karuna always assigns the highest priority to deadline flows. As a result, non-deadline flows are delayed a little. While in Aemon, non-deadline flows are assigned relatively higher priorities and hence shorter non-deadline flows finish much faster. However, Aemon is worse than Karuna for medium and large flows. This is because the rate control in Karuna can better rein in the bandwidth used by deadline flows by exploiting the flow sizes.

Deadline miss ratio. As depicted in Figures 5(a) and 5(b), Karuna achieves the lowest deadline miss ratio among all schemes. More interestingly, without flow size information Aemon's performance is only slightly inferior to Karuna, with a difference of ~14% and ~19.1%, respectively in two workloads. Compared to PIAS+DCTCP, Aemon achieves lower deadline miss ratios, with a reduction of 13.1% and 4% in both workloads, respectively. This demonstrates the effectiveness of Aemon in ensuring that most deadlines are met even without complete information.

[1]For Karuna, a small number of flows experience many timeouts under the web search workload. Under this workload, we only show the FCTs of flows with no timeouts.

We now independently assess the two key components of Aemon, namely, UCP and 2LPS.

How UCP benefits? In terms of FCT, from Figures 3 and 4 we observe that PIAS+UCP outperforms PIAS+DCTCP by small margins. However, in terms of deadline miss ratio as depicted in Figures 5(a) and 5(b), PIAS+UCP are slightly worse than PIAS+DCTCP. The same observation holds between Aemon and 2LPS+DCTCP. The reason of this trade-off is that Aemon aggressively cuts down deadline flows' congestion window when its urgency is low (§3.1), to enable non-deadline flows to finish faster. Thus, FCT of non-deadline flows benefits from UCP's design.

How 2LPS benefits? As shown in Figures 3 and 4, compared to PIAS+DCTCP, 2LPS+DCTCP reduces 37.3% and 13.9% average FCT of all non-deadline flows, and large flows FCTs by 35.4% and 14.3%, in web search and data mining, respectively. The same holds between Aemon and PIAS+UCP. Moreover, 2LPS also lowers the deadline miss ratio compared to PIAS in both workloads. This demonstrates 2LPS' effectiveness of achieving lower FCTs and deadline miss ratio at the same time.

5 CONCLUSION AND DISCUSSION

In this paper, we presented Aemon, an information-agnostic transport design that addresses the challenges of mix-flow scheduling without flow sizes known as a prior. Aemon's design consists of urgency-based congestion control (UCP) at end-host and a novel in-network two-level priority scheduling (2LPS), which aims at achieving best trade-off between deadline miss rate and non-deadline flows' FCT. Preliminary evaluation shows that Aemon significantly reduces FCT of non-deadline flows and achieves lower deadline miss ratio compared to existing solutions.

Future work: We plan to explore a few directions in the future. (1) We observe from the evaluation that the benefit of Aemon mainly comes from 2LPS. We thus believe there is room to improve the design of UCP and reduce both FCT and deadline miss rate at the same time. (2) We plan to further investigate the relation between the urgency thresholds and the performance of 2LPS, and find the optimal setting. (3) 2LPS provides us a way to differentiate both types of flows and avoid starving any one. We decide to explore other alternatives to determine the relative priorities of both types of flows. (4) We intend to implement an Aemon prototype and evaluate its effectiveness through testbed experiments.

ACKNOLEDGEMENTS

We thank the anonymous reviewers for their insightful feedback. This work was supported in part by the National 973 Basic Research Program under grant 2014CB347800, NSFC under grant 61370232, Hong Kong RGC GRF-11202315 and CRF-C7036-15G, and by CityU SRG grant 7004677.

REFERENCES

[1] *Microsoft Azure Stream Analytics.* https://azure.microsoft.com/en-us/services/stream-analytics/.
[2] *The Network Simulator 2.* http://www.isi.edu/nsnam/ns/.
[3] *The PipelineDB.* https://www.pipelinedb.com/.
[4] *Stream Analytix.* http://streamanalytix.com/.
[5] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). In *Proc. of ACM SIGCOMM.*
[6] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pFabric: Minimal Near-optimal Datacenter Transport. In *Proc. of ACM SIGCOMM.*
[7] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang. 2015. Information-Agnostic Flow Scheduling for Commodity Data Centers. In *Proc. of USENIX NSDI.*
[8] Li Chen, Kai Chen, Wei Bai, and Mohammad Alizadeh. 2016. Scheduling Mix-flows in Commodity Datacenters with Karuna. In *Proc. of ACM SIGCOMM.*
[9] Mosharaf Chowdhury and Ion Stoica. 2015. Efficient Coflow Scheduling Without Prior Knowledge. In *Proc. of ACM SIGCOMM.*
[10] Fernando J. Corbató, Marjorie Merwin-Daggett, and Robert C. Daley. 1962. An Experimental Time-sharing System. In *Proc. of ACM Spring Joint Computer Conference.*
[11] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of USENIX OSDI.*
[12] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella. 2013. On the impact of packet spraying in data center networks. In *Proc. of IEEE INFOCOM.*
[13] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. In *Proc. of ACM SIGCOMM.*
[14] Chi-Yao Hong, Matthew Caesar, and P. Brighten Godfrey. 2012. Finishing Flows Quickly with Preemptive Scheduling. In *Proc. of ACM SIGCOMM.*
[15] Ali Munir, Ghufran Baig, Syed M. Irteza, Ihsan A. Qazi, Alex X. Liu, and Fahad R. Dogar. 2014. Friends, Not Foes: Synthesizing Existing Transport Strategies for Data Center Networks. In *Proc. of ACM SIGCOMM.*
[16] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. 2013. Minimizing flow completion times in data centers. In *Proc. of IEEE INFOCOM.*
[17] Idris A. Rai, Guillaume Urvoy-Keller, Mary K. Vernon, and Ernst W. Biersack. 2004. Performance Analysis of LAS-based Scheduling Disciplines in a Packet Switched Network. In *Proc. of ACM SIGMETRICS.*
[18] Balajee Vamanan, Jahangir Hasan, and T.N. Vijaykumar. 2012. Deadline-aware Datacenter TCP (D2TCP). In *Proc. of ACM SIGCOMM.*
[19] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. 2011. Better Never Than Late: Meeting Deadlines in Datacenter Networks. In *Proc. of ACM SIGCOMM.*