# Per-Flow Fairness in the Datacenter Network

YiXi Gong, James W. Roberts, Dario Rossi
Telecom ParisTech

***Abstract*—Datacenter network (DCN) design has been actively researched for over a decade. Solutions proposed range from end-to-end transport protocol redesign to more intricate, monolithic and cross-layer architectures. Despite this intense activity, to date we remark the absence of DCN proposals based on simple fair scheduling strategies. In this paper, we evaluate the effectiveness of FQ-CoDel in the DCN environment. Our results show, (i) that average throughput is greater than that attained with DCN tailored protocols like DCTCP, and (ii) the completion time of short flows is close to that of state-of-art DCN proposals like pFabric. Good enough performance and striking simplicity make FQ-CoDel a serious contender in the DCN arena.**

## I. Introduction

In the last decade, datacenter networks (DCNs) have increasingly been built with relatively inexpensive off-the-shelf devices. DCNs are frequently assumed to be highly specialized environments, owned by a single entity that has full control of both the network architecture and the workload. DCN transport research has consequently explored a larger design space than that of the Internet leading to designs that generally do not focus on a single layer of the protocol stack but are more typically cross-layer. The proposed transport solutions may for instance integrate application information, use explicit or implicit rate control, incorporate routing and load balancing or rely on an omniscient oracle.

Freedom in the DCN design space translates into a relatively crowded landscape of proposals, each of which is typically designed and tweaked with a specialized application scenario in mind. DCN proposals are further entangled as their design is tailored for very specific workloads, with diverse application patterns including for instance, query-response [6], [8], map-reduce jobs [12], [13], or packet-size access to CloudRAM [7]. Rarely, if ever, is a DCN design tested with a workload other than that for which the system was explicitly designed.

Beyond any reasonable doubt, the single-tenant assumption will be severely challenged in the near future. Increased user reliance on Cloud applications will require DCNs to evolve toward multi-tenant systems with a significantly more heterogeneous set of applications and workloads. For instance, DCN owners could handle several types of VM-based or container-based applications in their infrastructure as they enter new lines of business, or as they rent resources to third-parties. The DCN workload will thus evolve beyond a typical mixture of short transactions and fat elastic transfers, as frequently considered today, to include a significant fraction of rate-limited flows with strict latency requirements as produced by gaming applications [1], instant voice translation [29] and augmented reality services, for example. The exact flow mix will also be highly variable in time, depending both on the degree to which applications rely on offloading to the Cloud and on the accessibility of the Cloud that varies due to device capabilities, battery life, connectivity opportunity, etc. [9].

As DCN resources are increasingly shared among multiple stakeholders, we must question the appropriateness of some frequently made assumptions. How can one rely on all end-systems implementing a common, tailor-made transport protocol like DCTCP [6] when end-systems are virtual machines under tenant control? How can one rely on applications truthfully indicating the size of their flows to enable shortest flow first scheduling as in pFabric [8] when a tenant can gain better throughput by simply slicing a long flow into many small pieces? Expected future DCN usage clearly puts these assumptions in doubt and leads us to question the expected benefits of fragile DCN designs that rely on them.

Following the above discussion, we observe an obvious yet surprising omission in the explored DCN landscape, namely a scheduling mechanism providing fairness among flows, coupled with Active Queue Management (AQM) to upper-bound delay. Such a scheduler is FQ-CoDel that has recently gained prominence in another network application, namely fighting bufferbloat notably in home routers [18]. Its aim of keeping throughput high while controlling queueing delay and its generality with respect to the traffic mix clearly make it an excellent candidate for DCN operation. Our objective in this paper is to compare its performance to that realized by alternative state-of-the-art DCN proposals. We find notably that, under FQ-CoDel:

- throughput of plain TCP is better than that of a tailored end-to-end solution like DCTCP [6],
- short flows incur a delay comparable to what they incur under pFabric [8].

We believe its "good enough" and future-proof performance, coupled with a striking deployment simplicity, make FQ-CoDel a particularly appealing solution for multi-tenant DCNs. In the reminder of the paper, after overviewing related work (Sec. II), we describe our proposal (Sec. III), present methodology and the results of our comparison (Sec. IV), and then conclude (Sec. V).

## II. Background

In Tab. I we report a non-exhaustive view of DCN research to illustrate several important trends. We follow the

TABLE I
TAXONOMY OF RECENT DATA CENTER NETWORK DESIGNS.

| Proposal | Yr | Primary metric | Information | Rate Control† | Routing Load balancing | Scheduling | Layers‡ L | N | T | A | O |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **DCTCP** [6] | 10 | Latency | | Imp+ECN | | FS | | | • | | |
| **Hedera** [4] | 10 | Bisection bw | switch buffer | | Yes | | | | | | • |
| **D3** [39] | 11 | Throughput | deadline, size | Exp (D) | ECMP VLB | greedy | | | • | | • |
| **MPTCP** [32] | 11 | Throughput | | MP-enabled | ECMP | | | | • | | |
| **Orchestra** [12] | 11 | Transfer-CT | everything | Exp (C) | Yes | FIFO, FS, Priority | | | | | • |
| **HULL** [7] | 12 | Throughput | | DCTCP+ECN | ECMP | | | | • | | |
| **DeTail** [41] | 12 | Tail FCT | priority | TCP +ECN | packet-based | | • | • | • | • | |
| **PDQ** [19] | 12 | FCT | deadline, size | Exp (D) | ECMP | EDF/SJF | | | • | • | |
| **pFabric** [8] | 13 | FCT | priority | Imp | RPS | Priority | | | • | • | |
| **RepFlow** [40] | 14 | Short flow FCT | size | | ECMP | | | | • | • | |
| **Baraat** [15] | 14 | Task-CT | priority | Exp (D) | | FIFO-LM | | | • | • | |
| **PASE** [25] | 14 | FCT | size, max rate | Exp+ECN (D) | | SJF | | | • | • | |
| **Varys** [13] | 14 | Coflow-CT | size | Exp (C) | | SEBF + MADD | | | • | • | • |
| **CONGA** [5] | 14 | FCT | | | flowlet-based | | | • | | | |
| **Fastpass** [31] | 14 | Fairness/FCT | size | Exp (C) | packet-based | maximum matching | | | • | • | • |
| **FlowBender** [21] | 14 | Latency | | | flow-based | | | • | • | | |
| **PIAS** [10] | 15 | FCT | | DCTCP+ECN | | SJF | | | • | | |
| **RAPIER** [35] | 15 | Coflow-CT | size | Exp (C) | coflow-based | MRTF | | • | • | • | |
| **pHost** [17] | 15 | FCT Tail-CT | | Exp (D) | packet-spraying | RTS-mechanism | | | | • | |
| **NUMFabric** [26] | 16 | FCT | flow rates | Exp (D) | | WFQ+ XWI | • | • | • | | |
| **ExpressPass** [11] | 17 | FCT | | Exp (D) | symmetric | Credit rates | | • | • | | |
| **Flowtune** [30] | 17 | FCT | flowlet epochs | Exp (C) | | central allocator | | • | • | | • |
| **LetFlow** [36] | 17 | FCT | | Imp (D) | flowlet-based | FIFO | | • | • | | |

† Rate/congestion control: (Exp)licit vs (Imp)licit; (D)istributed vs (C)entralized
‡ Layers: (L)ink, (N)etwork, (T)ransport, (A)pplication, (O)racle

common assumption that DCNs have the leaf- spine topology (e.g., [5], [8], [10], [35]) derived from the seminal FatTree proposal [3]

DCN research was initially confined to a single layer of the protocol stack. For instance, starting with the pioneering work of DCTCP [6], the community recognized that specific designs were needed to enable TCP to cope with the high bandwidth and low delay properties of DCNs. This led to the design of new protocols exemplified by HULL [7] that, to the best of our knowledge, have not yet been deployed. It is noteworthy that optimal parameter tuning for deployed TCP variants like DCTCP remains the subject of active research [20].

Alternative proposals for an improved network fabric include Hedera [4] and Orchestra [12] that rely on a centralized oracle to deal with routing, load balancing, congestion avoidance and fault tolerance. Starting with pFabric [8], which has become the de-facto standard for comparing new DCN proposals, a number of papers have promoted a cross-layer design. These DCN designs jointly impact multiple aspects including explicit [19], [25], [30], [35], [39] or implicit [8], [41] congestion and flow control at end-systems, flow scheduling [8], [10], [13], [15], [19], [25], [31], [35], [39], load balancing [5], [8], [21], [31], [35], [36], [41] and oracles [13], [30], [31]. Recent exceptions to the above trend are represented by pHost [17] and ExpressPass [11] that introduce a credit-based transport mechanism.

Application-layer information has been progresively integrated into decision logic (as either a priority index [8], [41], or a flow deadline [19], [39] or size [19], [39], [40]). This tendency was recently exacerbated by moving from network-oriented to service-oriented metrics: e.g., Varys [13] and RAPIER [35] use co-flow completion time, and Baraat [15] uses task completion time. Mechanisms proposed in [13], [15], [25], [30], [31], [35] all employ explicit rate control, with either distributed or centralized arbitration. Some recent proposals like NUMFabric [26] and Flowtune [30] would apply utility maximizing rate controls, rapidly computed by a centralized controller made aware dynamically of the precise population of flows in progress.

This rapid survey reveals the wide range of research on DCN architectures and it is all the more surprising that a technique as simple and potentially effective as FQ-CoDel, to our knowledge, remains unexplored.

## III. FAIRNESS IN THE DCN

We argue that the advantages of decoupling rate control from scheduling, put forward in the pFabric proposal [8], would similarly be obtained by implementing per-flow fair scheduling on bottleneck links.

### A. *Be fair to flows!*

*Fair queuing*. Starting with the original proposal of Nagle [27], per-flow fair scheduling has often been advocated as a means to make Internet bandwidth sharing robust and

more efficient. However, the need has never been considered sufficiently compelling to warrant widespread implementation. An exception is the recent work on fighting bufferbloat, notably on home routers [38] where the preferred solution has been to perform fair queuing on the user access line, in association with the CoDel queue management algorithm [18], [28]. Per-flow scheduling on the user access line ensures low latency for packets of delay-sensitive flows like VoIP while allowing bulk data transfers to fully utilize residual bandwidth. Such scheduling is generally unnecessary on Internet links beyond the access line since these are rarely a bottleneck.This is not the case in datacenters, however, where an upstream or downstream link between a server and its top-of-rack switch can be saturated by a single flow. On the other hand, the large bisection bandwidth of DCNs coupled with efficient load balancing over multiple paths (e.g. using LetFlow [36]) tends to avoid congestion in the interconnect. We therefore follow [8] in supposing scheduling is applied only on the ToR–server links, in both directions.

*Priority fair queuing*. The DRR-based "FlowQueue Controlled Delay" (FQ-CoDel) scheduler [18], originally proposed for fighting bufferbloat, does in fact more than just fair scheduling: it incorporates a priority mechanism to further reduce the packet latency of low rate flows, which would also be very useful in DCNs. A fair queuing scheduler maintains a list of flows that currently have backlogged packets. It can therefore recognize packets that come from flows that are not currently backlogged. Such flows will include single packet queries as well as any flows emitting packets at a rate less than the current fair rate. In FQ-CoDel, these packets are dequeued with priority, minimizing their latency without undue impact on the throughput of backlogged flows. This mechanism had previously been proposed as a means to realize implicit service differentiation [24].

*Controlling queue delay*. In FQ-CoDel, packets can be dropped on applying the CoDel [18], [28] AQM algorithm to each flow queue. Packets are also dropped from the flow with the biggest backlog when the shared buffer would otherwise overflow. The latter, longest queue drop policy [34] would, as in [8], avoid the coupling between rate control and scheduling that is implicit in CoDel. A further alternative would be to use per-flow backlogs as the drop criterion [2], [33]. However, in the present work we retain the full implementation of FQ-CoDel as currently available in the Linux kernel.

### B. Suitability for DCN

*Flow identity*. In our evaluation we identify flows by the usual IPv4 5-tuple. However, it may be more appropriate in a DCN to use other criteria like the origin and destination servers or virtual machines, or to identify co-flows belonging to a single task. Note that the proposal is to apply lightweight, egalitarian fair sharing as opposed to weighted fair sharing and therefore requires no additional state beyond flow ID.

*Quantum size*. The default DRR quantum size in FQ-CoDel is one 1500 byte MTU. This implies the packets of any flow emitting less than 1500 bytes in a DRR cycle are handled with priority. In the datacenter environment, it may make more sense to increase the quantum size, to ensure all packets of a short query are handled with priority, for instance. It is interesting to note that the Baraat [15] scheduler has similar behavior to FQ-CoDel with a particular choice of quantum. Baraat handles flows in FIFO order until they are observed to have emitted more than a given number of bytes $\theta$. They are then required to fairly share the link bandwidth under the end-to-end control of RCP [16]. The result of applying FQ-CoDel with a quantum equal to $\theta$ would be broadly similar.

*Rate-limited flows*. The workloads considered in pFabric [8] and Baraat [15] do not include flows whose rate is intrinsically limited. Such flows exist in any datacenter supporting streaming applications or gaming, for example, and would benefit from the low latency provided naturally by FQ-CoDel. For example, packets of a 10 Mbps flow on a 10 Gbps link would not suffer significant delay until the number of concurrent active flows exceeds 1000.

*Incast*. Imposing fair sharing has a similar impact on incast as DCTCP [6] since fair sharing between the fanned-in flows is guaranteed. The fair rate during incast would likely still be high enough that packets of any streaming flow sharing the bottleneck link would be handled with priority.

*Scalablity*. The complexity of fair queuing depends on the number of *active* flows, i.e., flows that currently have one or more buffered packets. It was shown in [22] for wide area networks with Poisson flow arrivals that this number is typically less than 100 even though the number of flows *in progress* attains hundreds of thousands. In the datacenter, the stochastic demand model may be different but the number of flows is similarly limited especially on the considered ToR–server links. A recent paper shows how fair queuing can be realized in a reconfigurable switch [33] while our own work on software routers [2] shows how upstream bandwidth might be fairly shared an applying a fair dropping algorithm in the server.

## IV. Evaluation

We describe the scenarios used to evaluate the performance of FQ-CoDel in the datacenter before discussing calibration of protocol parameters, and reporting *ns2* simulation results.

### A. Methodology

*Terms of comparison*. We compare FQ-CoDel against two representative alternative DCN designs: (i) DCTCP [6], a distributed end-to-end mechanism that exposes a general transport service through an L4 abstraction, and (ii) pFabric [8], a clean-slate cross-layer design that aims to optimize

TABLE II
PFABRIC TCP TUNING

| | param | pFabric | (default) | note |
|---|---|---|---|---|
| **rtx**† | minrto_ | 45$\mu$s | (200ms) | |
| | maxrto_ | 2s | (60s) | |
| | rtxcur_init_ | 45$\mu$s | (3s) | initial rto |
| | tcpTick_ | 1$\mu$s | (10ms) | clock granularity |
| **rx**‡ | interval_ | 6$\mu$s | (100ms) | delayed ack |
| | window_ | $10^6$ | (20) | rx window |
| **cc**⋆ | windowInit_ | 12 | (2) | initial cwnd |
| | maxcwnd_ | queue-1 | ($\infty$) | max cwnd |
| | windowOption_ | 0: basic | (1: std) | cong. avoid. |

†**Retransmission,** ‡**Receiver,** ⋆**Congestion control**

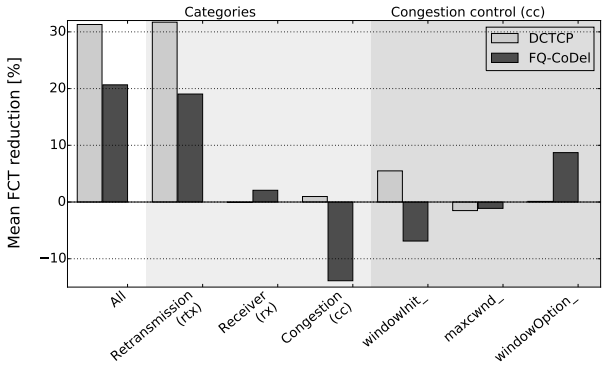


Fig. 1. Applying pFabric TCP tuning to DCTCP and FQ-CoDel: overall, per category, and breakdown for the cc category impact w.r.t. default TCP settings.

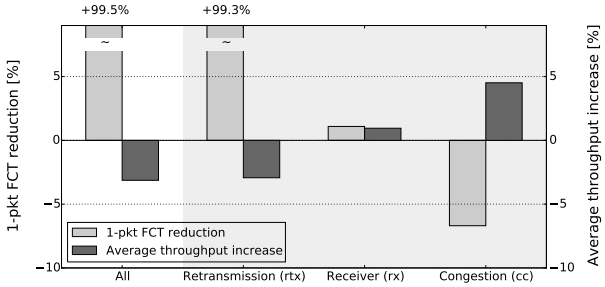


Fig. 2. Breakdown of pFabric TCP parameters impact on DCTCP: per-category and per-flow size account.

flow completion time performance. The first is actually implemented in production datacenters [20], while the second represents an ideal that may only be attained in a particular protected datacenter environment where end-systems are compliant. The code of both designs is available in *ns2*.

*Network and workload.* We adopt a downscaled version of the pFabric network scenario (32 vs 144 hosts [8], interconnected by a leaf-spine topology with the same delay characteristics). We adopt the pFabric "data mining" workload, presented in Fig. 4b in [8]. Flows arrive according to a Poisson process and have a size drawn independently from an empirical distribution, where half of the flows are single packet while 95% of bytes are in flows larger than 35 MB.

Note that this workload does not include any rate limited flows as arise in streaming and gaming applications, for instance. To account for the growing impact of rate limited flows in DCNs, we tweak the data mining scenario by controlling the percentage of 1-packet flows. In the DCN setting, in addition to query traffic, such flows are an approximate representation of bandwidth-limited flows (streaming, games,...) whose packets arrive widely spaced compared to the time scale of queue dynamics. We let the overall volume of 1-packet flows vary between 0.01% and 10% of the overall offered traffic.

*Performance measures.* We compare the performance of the considered DCN designs through two measures: the flow completion time (FCT) of 1-packet flows and the mean throughput of all flows defined as the ratio of mean size in bits to to mean FCT in seconds. The 1-packet flow completion time is a measure of latency for both single packet queries and packets of rate limited flows.

### B. Calibration

Since we consider the pFabric scenario (i.e., workload, topology, capacities, etc.), we retain the transport protocol settings of [8] and make necessary adjustments for DCTCP and FQ-CoDel to make the performance comparison as fair as possible.

*Local AQM tuning.* For DCTCP, we resort to standard Drop-Tail FIFO, and experiment with two buffer sizes (namely, 100 and 1000 packets). FQ-CoDel uses the stochastic fair queuing implementation of DRR and relies essentially on three parameters: the number of FQ hash buckets (we stick with the default value of 1024), the CoDel target delay (default 5ms) and the inference interval (default 100ms). The default CoDel settings were meant to counter bufferbloat in the access network, with timescales correlated to human perception that are thus orders of magnitude larger than what is reasonable for the DCN environment. Considering that the RTT propagation delay between any two hosts in our DCN scenario is 12$\mu s$ and an MTU packet transmission time is 1.2$\mu s$, we downscale by a factor of 1000x both the target delay (about 4 packets per bucket) and the inference interval. While these settings work well in practice, a more careful tuning along the lines of [14] might further improve performance.

*End-to-end TCP tuning.* To perform a fair comparison it is necessary to specialize transport protocol parameters to the DCN environment [20]. Tab. II contrasts the pFabric settings with default TCP values. The DCN environment clearly requires an increase of timestamp precision [37], a significant reduction of time-related parameters (such as delayed ack and retransmission timer [37]), and an increase of window-related parameters. Overall, the pFabric settings

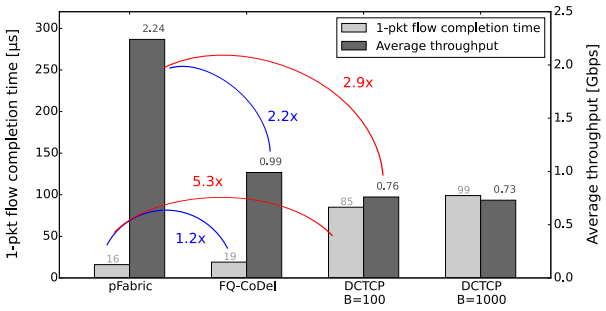


Fig. 3. Performance at a glance. Original pFabric "data mining" scenario, load 0.6.

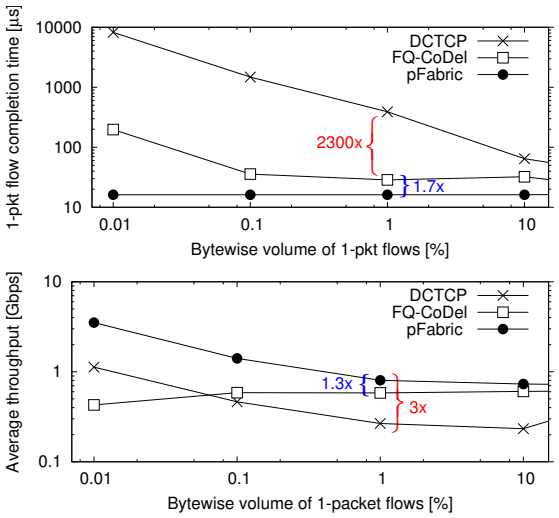


Fig. 4. Impact of 1-packet flow intensity: Tweaked pFabric scenario, load 0.6.

reduce the mean FCT by *a factor of more than 2* compared to that provided by pFabric used with vanilla TCP settings.

To verify that pFabric settings do not play against DCTCP and FQ-CoDel, we activate features progressively and measure differences in latency (i.e., single packet FCT) and throughput. Results are shown in Fig. 1 showing that activating all parameters yields a reduction in mean FCT of 30% for DCTCP and 20% for FQ-CoDel. The breakdown per rtx/rx/cc categories of Tab. II shows that time-related parameters play a paramount role, as expected. On the other hand, we see that congestion-related parameters have a limited impact on DCTCP and even a negative impact on FQ-CoDel. It follows that, by enabling pFabric parameters without any further tuning, our results are slightly more favorable for pFabric, and provide a conservative estimate of relative FQ-CoDel performance.

Fig. 2 further breaks down the rtx/rx/cc parameter impacts on the FCT and throughput metrics for DCTCP. Here again it can be seen that, while the FCT of 1-packet flows benefits from the combined settings, the throughput of long DCTCP flows suffers slightly with the bulk adoption of pFabric settings. On the other hand, selective parameter adoption is hardly possible since each impacts performance metrics differently (e.g., cc parameters have a negative impact for short flows and a positive impact for long flows).

### C. Performance comparison

*Original data-mining scenario.* To show differences at a glance, we start from the original pFabric scenario where 1-packet flows represent a significant fraction of the flow volume but account for only a negligible portion of the byte-wise traffic volume. Average throughput and completion time of 1-packet flows is reported in Fig. 3 showing that pFabric indeed exhibits outstanding performance for both metrics. It also shows, however, that FQ-CoDel comes second: FCT for FQ-CoDel is very close to that of pFabric; throughput of long flows is significantly smaller than with pFabric but still higher than with DCTCP.

*Tweaked data-mining scenario.* Robust DCN designs should maintain their performance across scenarios. Simply increasing the proportion of traffic generated by 1-packet flows constitutes an insightful sensitivity analysis. For technical reasons tied to the simulation duration, we cap the maximum flow size to $10^5$ packets. This tweak is favorable to pFabric as the gap between pFabric and FQ-CoDel for 0.01% in Fig. 4 is larger than in Fig. 3. However, it can be easily seen that the difference reduces significantly for increasing intensity of 1-packet flows (i.e., as the proportion of low rate flows increases). While the difference between DCTCP and pFabric always remains significant, the performance of FQ-CoDel approaches that of pFabric.

## V. A FAIR STATEMENT

We have argued that multi-tenant DCNs cannot rely on specialized protocols and mechanisms that assume single ownership and end-system compliance. It is necessary rather to implement general, well-understood mechanisms provided as a network service that require as few assumptions about DC workload as possible.

Specifically, we have proposed that fair scheduling coupled with a mechanism to explicitly controll the queue size constitutes an appealing traffic management solution for multi-tenant DCNs. In the light of our results, we believe it is fair to say FQ-CoDel achieves "good enough" performance while being "simple enough" for rapid deployment.

It is surprising that this simple yet effective approach has so far been neglected in DCN research and our main aim in this paper has been to raise awareness of this omission. Clearly, much remains to be done. For instance, FQ-CoDel should be compared against DCTCP in a real deployment to prove that benefits are immediately and painlessly achievable in today DCNs.

While FQ-CoDel is already available in the current Linux kernel release, it is not directly applicable in DCN ToR

switches. However, the recent paper [33] shows how the latest configurable switches can be used to realize per-flow fair bandwidth sharing. The proposed mechanism could be readily extended to accurately approximate priority fair queuing [23]. Instead of using FQ-CoDel to control sharing in the server–TpR link, we currently envisage the use of a fair dropping algorithm as discussed in [2].

It remains a significant challenge to design and perform the experimental campaign that is necessary to confirm the expected advantages of priority fair queueing under realistic conditions. More broadly, it is also necessary to gain a better understanding of how it would perform in combination with additional key DCN developments (like co-flow identification and efficient load balancing techniques).

## ACKNOWLEDGEMENTS

This work has been carried out at LINCS (http://www.lincs.fr) and benefited from support of NewNet@Paris, Ciscos Chair "NETWORKS FOR THE FUTURE" at Telecom ParisTech (https://newnet.telecom-paristech.fr).

## REFERENCES

[1] http://www.gartner.com/newsroom/id/2614915.

[2] V. Addanki, L. Linguaglossa, et al. Controlling software router resource sharing by fair packet dropping. In *IFIP Networking 2018*. 2018.

[3] M. Al-Fares, A. Loukissas, et al. A scalable, commodity data center network architecture. In *ACM SIGCOMM*. 2008.

[4] M. Al-Fares, S. Radhakrishnan, et al. Hedera: Dynamic flow scheduling for data center networks. In *USENIX NSDI*. 2010.

[5] M. Alizadeh, T. Edsall, et al. Conga: Distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM*. 2014.

[6] M. Alizadeh, A. Greenberg, et al. Data center TCP (DCTCP). In *ACM SIGCOMM*. 2010.

[7] M. Alizadeh, A. Kabbani, et al. Less is more: Trading a little bandwidth for ultra-low latency in the data center. In *USENIX NSDI*. 2012.

[8] M. Alizadeh, S. Yang, et al. pfabric: Minimal near-optimal datacenter transport. In *ACM SIGCOMM*. 2013.

[9] V. Bahl. Cloud 2020: The Emergence of Micro Datacenters (cloudlets) for Mobile Computing. Microsoft Devices and Networking Summit, 2015.

[10] W. Bai, L. Chen, et al. Information-agnostic flow scheduling for commodity data centers. In *NSDI'15*, pages 455–468. 2015.

[11] I. Cho, K. Jang, et al. Credit-scheduled delay-bounded congestion control for datacenters. In *SIGCOMM '17*, pages 239–252. 2017.

[12] M. Chowdhury, M. Zaharia, et al. Managing data transfers in computer clusters with Orchestra. In *ACM SIGCOMM*. 2011.

[13] M. Chowdhury, Y. Zhong, et al. Efficient coflow scheduling with Varys. In *ACM SIGCOMM*. 2014.

[14] M. Christiansen, K. Jeffay, et al. Tuning RED for web traffic. In *ACM SIGCOMM*. 2000.

[15] F. R. Dogar, T. Karagiannis, et al. Decentralized task-aware scheduling for data center networks. In *ACM SIGCOMM*. 2014.

[16] N. Dukkipati. *Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly*. Ph.D. thesis, Stanford University, '08.

[17] P. X. Gao, A. Narayan, et al. pHost: Distributed near-optimal datacenter transport over commodity network fabric. In *ACM CoNEXT*. 2015.

[18] T. Hoeiland-Joergensen, P. McKenney, et al. The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. RFC 8290, 2018.

[19] C.-Y. Hong, M. Caesar, et al. Finishing flows quickly with preemptive scheduling. In *ACM SIGCOMM*. 2012.

[20] G. Judd. Attaining the promise and avoiding the pitfalls of tcp in the datacenter. In *NSDI*. 2015.

[21] A. Kabbani, B. Vamanan, et al. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In *ACM CoNEXT*. 2014.

[22] A. Kortebi, L. Muscariello, et al. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *ACM SIGMETRICS*. 2005.

[23] A. Kortebi, S. Oueslati, et al. Cross-protect: implicit service differentiation and admission control. In *2004 Workshop on High Performance Switching and Routing, 2004. HPSR.*, pages 56–60. 2004.

[24] A. Kortebi, S. Oueslati, et al. Implicit service differentiation using deficit round robin. In *Proceedings of ITC 19*. 2005.

[25] A. Munir, G. Baig, et al. Friends, not foes: Synthesizing existing transport strategies for data center networks. In *ACM SIGCOMM*. 2014.

[26] K. Nagaraj, D. Bharadia, et al. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *ACM SIGCOMM*. 2016.

[27] J. Nagle. Congestion Control in IP/TCP Internetworks. RFC 896, 1984.

[28] K. Nichols and V. Jacobson. Controlling queue delay. *Commun. ACM*, 55(7):42, 2012.

[29] D. Patterson. The trouble with multi-core. *Spectrum, IEEE*, (7), 2010.

[30] J. Perry, H. Balakrishnan, et al. Flowtune: Flowlet control for datacenter networks. In *NSDI'17*, pages 421–435. 2017.

[31] J. Perry, A. Ousterhout, et al. Fastpass: A centralized "zero-queue" datacenter network. In *ACM SIGCOMM*. 2014.

[32] C. Raiciu, S. Barre, et al. Improving datacenter performance and robustness with multipath TCP. In *ACM SIGCOMM*. 2011.

[33] N. K. Sharma, M. Liu, et al. Approximating fair queueing on reconfigurable switches. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 1–16. 2018.

[34] B. Suter, T. V. Lakshman, et al. Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing. *IEEE JSAC*, pages 1159–1169, 2006.

[35] Z. Uestc, K. Chen, et al. RAPIER : Integrating Routing and Scheduling for Coflow-aware Data Center Networks. *IEEE INFOCOM*, 2015.

[36] E. Vanini, R. Pan, et al. Let it flow: Resilient asymmetric load balancing with flowlet switching. In *NSDI 17*, pages 407–420. 2017.

[37] V. Vasudevan, A. Phanishayee, et al. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *ACM SIGCOMM*. 2009.

[38] G. White. Active queue management in DOCSIS 3.x cable modems. Technical report, CableLabs, 2014.

[39] C. Wilson, H. Ballani, et al. Better never than late: Meeting deadlines in datacenter networks. In *ACM SIGCOMM*. 2011.

[40] H. Xu and B. Li. Repflow: Minimizing flow completion times with replicated flows in data centers. In *IEEE INFOCOM*. 2014.

[41] D. Zats, T. Das, et al. Detail: Reducing the flow completion time tail in datacenter networks. In *ACM SIGCOMM*. 2012.