Fairness in the Data Center

YiXi Gong
Telecom ParisTech
yixi.gong@enst.fr

James Roberts
LINCS
james.walter.roberts@gmail.com

Dario Rossi
Telecom ParisTech and LINCS
dario.rossi@enst.fr

ABSTRACT

Data center 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 short 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.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Network**]: Network Protocols

Keywords

Data Center Network, Design, Performance

1. INTRODUCTION

In the last decade, data center 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 over both the network architecture and its 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. 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 and workload scenario in mind. Considered applications include, for instance, query-response [5, 7], map-reduce jobs [10, 11], or packet-size access in a CloudRAM [6]. Rarely, if ever, is a DCN design tested with a workload other than the one the system was explicitly designed for.

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 [26] 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 (e.g., streaming gaming meta-data for local rendering or streaming Cloud-rendered videos) and on the accessibility of the Cloud that varies due to device capabilities, battery life, connectivity opportunity, etc. [8].

As DCN resources are increasingly shared among multiple stakeholders, it is necessary to question the appropriateness of frequently made assumptions. How can one rely on all end-systems implementing a common, taylor-made transport protocol like DCTCP [5] 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 [7], when a tenant can gain better throughput by simply slicing a long flow into many small pieces? Future DCN usage clearly threatens these assumptions and is likely to erode the benefits of fragile DCN designs that rely upon them.

Given these premises, we remark 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. We believe an obvious candidate is FQ-CoDel [25, 16] that has recently been developed in another context with, however, aims that largely coincide with the requirements of the DCN. Our preliminary comparison with state of the art proposals, demonstrates that FQ-CoDel:

- enables throughput with plain TCP that is greater than that obtained with a tailored end-to-end solution like DCTCP [5]
- limits the latency of very short flows to values close to those obtained using pFabric [7]

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 the related effort (Sec. 2), we describe our proposal (Sec. 3) and present

Table 1: Taxonomy of recent data center network design.

Proposal	Yr	Primary metric	Information	Rate Control[†]	Routing	Scheduling	Layers[‡] L	N	T	A	/O
DCTCP [5]	10	Latency		Imp+ECN		FS			✓		
Hedera [3]	10	Bisection bw	switch buffer		Yes						✓
D3 [33]	11	Throughput	deadline, size	Exp (D)	ECMP VLB	greedy			✓		✓
MPTCP [28]	11	Throughput		MP-enabled	ECMP				✓		
Orchestra [10]	11	Transfer-CT	everything	Exp (C)	Yes	FIFO, FS, Priority					✓
HULL [6]	12	Throughput		DCTCP+ECN	ECMP				✓		
DeTail [35]	12	Tail FCT	priority	TCP +ECN	packet-based		✓	✓	✓	✓	
PDQ [17]	12	FCT	deadline, size	Exp (D)	ECMP	EDF/SJF			✓	✓	
pFabric [7]	13	FCT	priority	Imp	RPS	Priority			✓	✓	
RepFlow [34]	14	Short flow FCT	size		ECMP				✓	✓	
Baraat [13]	14	Task-CT	priority	Exp (D)		FIFO-LM			✓	✓	
PASE [22]	14	FCT	size, max rate	Exp+ECN (D)		SJF			✓	✓	
Varys [11]	14	Coflow-CT	size	Exp (C)		SEBF + MADD			✓	✓	✓
CONGA [4]	14	FCT			flowlet-based			✓			
Fastpass [27]	14	Fairness/FCT	size	Exp (C)	packet-based	maximum matching			✓	✓	✓
FlowBender [19]	14	Latency			flow-based			✓	✓		
PIAS [9]	14	FCT		DCTCP+ECN		SJF			✓		
RAPIER [30]	15	Coflow-CT	size	Exp (C)	coflow-based	MRTF		✓	✓	✓	
pHosts [15]	15	FCT Tail-CT		Exp (D)	packet-spraying	RTS-mechanism				✓	
NUMFabric [23]	16	FCT		Exp (D)		WFQ + xWI	✓	✓	✓		

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

simulation results of our comparison (Sec. 4), based on which we conclude the paper (Sec. 5).

2. BACKGROUND

We report an incomplete yet instructive view of DCN research in Tab. 1, illustrating several important trends. We follow the common assumption that DCNs have the leaf-spine topology (e.g., [7, 4, 9, 30]) derived from the seminal FatTree proposal [2].

DCN research was initially confined to a single layer of the protocol stack. For instance, starting with pioneering work on DCTCP [5], the community recognized that specific designs were needed for TCP to cope with the high bandwidth and low delay properties of DCNs. This led to the design of new protocols such as HULL [6]. Note that state of the art tuning of DTCP variants in DCN is still a subject of active research [18].

Alternative proposals for an improved network fabric include Hedera [3] and Orchestra [10] that rely on a centralized oracle to deal with routing, load balancing, congestion avoidance and fault tolerance. Starting with pFabric [7], which has become the de facto standard for comparing the performance of new DCN proposals, a number of papers have promoted a cross-layer design. These DCN designs jointly impact explicit [33, 17, 22, 30, 23] or implicit [35, 7] end-system congestion and flow control, flow scheduling [33, 17, 7, 13, 22, 11, 27, 9, 30], routing [35, 7, 4, 27, 19, 30] and oracles [11, 27]. Recent exceptions to the above trend is represented by pHost [15] (that introduces a mechanism for requesting/granting transmission tokens and operates solely at transport level) and RepFlow [34] (that opportunistically replicates packets of short flows, and thus operates at a higher load).

Application-layer information has been gradually integrated into decision logic (either explicitly as a priority level [35, 7], or implicitly via a flow deadline [33, 17] or flow size [33, 17, 34]). This trend was recently exacerbated by moving from network-oriented to service-oriented metrics, e.g., Varys [11] and RAPIER [30] use co-flow completion time while Baraat [13] uses task completion time. In contrast to the implicit rate control of pFabric, [13, 22, 11, 27, 30] all employ explicit rate control with either a distributed or a centralized arbitrator. Recent proposals such as NUMFabric [23] still employ explicit rate control, though coupled with a Weighted Fair Queueing (WFQ) approach, whose weights are dynamically set via a distributed eXplicit Weight Inference (xWI) algorithm. While NUMFabric is flexible in the bandwidth allocation schemes it can enforce (e.g., minimize FCT, achieve α-fairness or enforce bandwidth functions) it however requires modification a the host (Swift control, xWI) and switches (xWI price computation) and protocols (the message passing algorithm requires 5 additional header fields).

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.

3. FAIRNESS IN THE DCN

We argue that the advantages of decoupling rate control from scheduling, put forward in the pFabric proposal [7], would similarly derive from implementing per-flow fair scheduling on bottleneck links.

3.1 Be fair to flows!

Fair queuing. Starting with the original proposal of Nagle [24], per-flow fair scheduling has often been advocated as a means to make Internet bandwidth sharing more robust and efficient. However, the need has never been considered sufficiently compelling to warrant widespread implementation. An exception is the recent work on bufferbloat, notably on home routers [32], where the preferred solution has been to perform fair queuing on the user access line, in association with the CoDel queue management algorithm [25, 16].

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. Scheduling is generally unnecessary on Internet links beyond the access line since these are rarely a bottleneck: flow rates are limited by the access bottleneck and the combined rate of flows in progress on a network link is typically less than its capacity. This is not the case in data centers, however, where single flows can readily saturate an upstream or downstream link between a server and its top-of-rack switch.

Priority fair queuing. The DRR-based "FlowQueue Controlled Delay" (FQ-CoDel) scheduler [16], 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. 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. In the DCN, such flows will include single packet queries as well as any flows emitting packets at a rate less than the current fair rate. FQ-CoDel dequeues these packets with priority, minimizing their latency without undue impact on the throughput of backlogged flows. This can be done without keeping per-flow state, and it has been shown to be highly scalable in [20]

Controlling queue delay. In FQ-CoDel, packets can be dropped on applying the CoDel [25] AQM algorithm to each flow queue. Packets are also dropped from the flow with the biggest backlog when the shared buffer would otherwise overflow. Longest queue drop might in fact be a sufficient AQM (see [29, 21], for instance) though it is simpler to adopt FQ-CoDel since this is already implemented in the Linux kernel.

3.2 Suitability for DCN

Flow identity. In our evaluation we identify flows by the usual IPv4 5-tuple. However, it may in practice 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. The latter would incur additional complexity to determine the weight from local state or a header field. Simple fairness is arguably sufficient for a DCN: it ensures low latency for short and low rate flows and largely decouples bandwidth sharing from the implemented transport protocol.

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 data center 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 [13] 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 θ. They are then required to fairly share the link bandwidth under the end-to-end control of RCP [14]. The result of applying FQ-CoDel with a quantum equal to θ would be broadly similar.

Rate-limited flows. The workloads considered in pFabric [7] and Baraat [13] do not include flows whose rate is intrinsically limited. Such flows exist in any data center supporting streaming applications or gaming, for instance, 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 flows exceeds 1000.

Incast. Imposing fair sharing has a similar impact on incast as DCTCP [5] 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 or gaming flow sharing the bottleneck link would be handled with priority.

Stability. As in the evaluation of pFabric [7], we envisage a simple stochastic demand model where flows arrive according to a Poisson process. For such workloads fair sharing is known to be stable as long as the overall offered load is less than 1 on every link. On the other hand, as noted in [7], "*size-based traffic prioritization may reduce the stability region of the network*" with respect to the maximum capacity attained by fair sharing. This phenomenon impacts both pFabric and FQ-CoDel. It remains to more thoroughly evaluate the capacity reduction under a relevant range of workloads. It was not significant for the workloads considered here or in [7].

4. EVALUATION

We describe the scenarios used to evaluate the performance of FQ-CoDel in the data center (Sec.4.1) before discussing calibration of protocol parameters (Sec.4.2) and reporting our results (Sec.4.3).

4.1 Methodology

Terms of comparison. We compare FQ-CoDel against two representative alternative DCN designs: (i) DCTCP [5], a distributed end-to-end design that exposes a general transport service through an L4 abstraction, and (ii) pFabric [7], a clean-slate cross-layer design that optimizes flow completion time performance. The first is already implemented in production data centers [18], while the second represents an ideal that may only be attained in a particular protected data center environment where end-systems are compliant. The code of both designs is available in *ns2* and has been used in our evaluation.

Network and workload. We adopt a downscaled version of the pFabric network scenario of [7] with 32 instead of 144 hosts interconnected by a Leaf-Spine topology with the same delay characteristics. We adopt the pFabric "data mining" workload, presented in Fig. 4b in [7]. Flows arrive according to a Poisson process and have a size drawn independently from an empirical distribution in which half 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 such flows in the DCN, we tweak the data mining scenario by controlling

Table 2: pFabric TCP tuning

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

[†]**Retransmission,** [‡]**Flow control,** [⋆]**Congestion control**

the percentage of single packet flows. We argue that this is a sufficiently accurate representation of the traffic of flows whose rate is just a small fraction of the link rate (e.g., a 10 Mbps flow on a 10 Gbps link). Packet latency for such flows in FQ-CoDel is determined by the random arrival process of packets with similar priority and this is locally Poisson, as when modelled as a process of single packet flows. We let the overall volume of single packet flow vary between 0.01% and 10% of the overall offered traffic and proportionally reduce the occurrence of all the other flows sizes.

Performance measures. We compare the performance of the considered DCN designs through the realized flow completion time (FTC). For single packet flows this coincides with the packet latency. For long flows we actually report the mean flow throughput defined as the ratio of size in bits to FCT in seconds.

4.2 Calibration

Since we consider the pFabric scenario (i.e., workload, topology, capacities, etc.), we retain the transport protocol settings of [7] and make appropriate adjustments for DCTCP and FQ-CoDel to make the performance comparison as fair as possible.

Local AQM tuning. For DCTCP, queue management is standard DropTail FIFO and we experiment with two buffer sizes, 100 and 1000 packets. FQ-CoDel uses a stochastic fair queuing implementation of deficit round robin (DRR) and relies essentially on three parameters: the number of FQ hash buckets (we retain 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 where timescales are orders of magnitude larger than what is reasonable for the DCN environment. Considering that the RTT propagation delay between any two host in our DCN scenario is 12 μs and packet transmission time is 1.2 μs, we downscale CoDel settings by a factor of 1000. While these settings work well in practice, a more careful tuning might improve performance further [12]. We do not use packet spraying with FQ-CoDel so that our results may be considered conservative.

End-to-end TCP tuning. To perform a fair comparison it is necessary to specialize transport protocol parameters to the DCN environment [18]. Tab. 2 contrasts the pFabric settings with default TCP values. The DCN environment clearly requires an increase in timestamp precision [31], significantly reduced time-related parameters (such as delayed

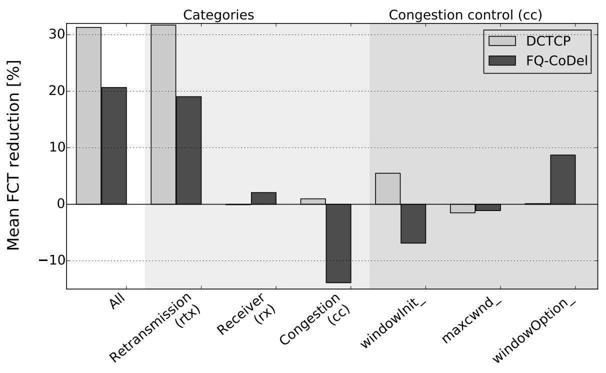


Figure 1: Impact of pFabric TCP tuning on DCTCP and FQ-CoDel: comparison with default settings - overall, per category, and per cc sub-category.

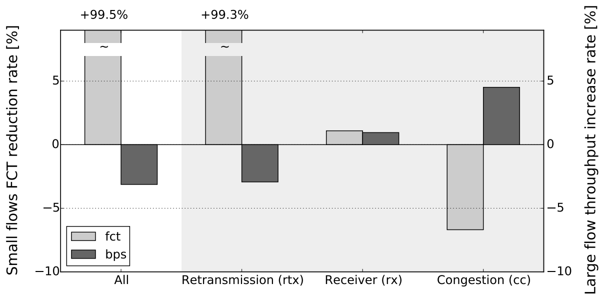


Figure 2: Flow size breakdown of pFabric TCP tuning impact on DCTCP (FCT of single packet flows, throughput of long flows).

ack and retransmission timers [31]), and increased window-related parameters. Overall, the pFabric settings reduce the mean FCT by *a factor of more than 2* compared to that provided by pFabric used with standard TCP settings.

To verify that pFabric settings do not play against DCTCP and FQ-CoDel, we activate features progressively and evaluate incremental differences in latency (i.e., single packet FCT) and throughput. Results presented in Fig. 1 reveal that activating all parameter updates yields roughly a 30% reduction in mean FCT for DCTCP and 20% for FQ-CoDel. The breakdown per rtx/tx/cc categories of Tab. 2 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 FQ-CoDel performance.

Fig. 2 further breaks down the impact of rtx/tx/cc parameters on latency and throughput metrics for DCTCP. Here again it can be seen that, while the single packet flow FCT benefits from the combined settings, the throughput of long DCTCP flows suffers slightly.Selective parameter tuning would be hard, however, since performance metrics are impacted differently (e.g., cc parameters have a negative impact for short flows and a positive impact for long flows).

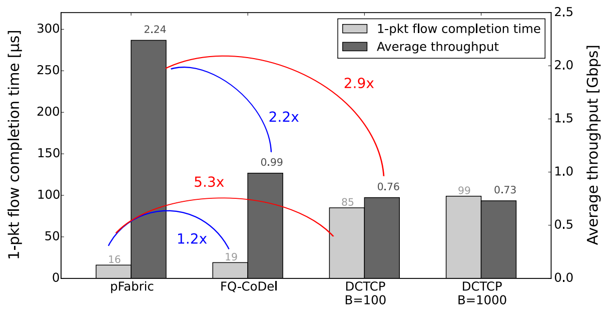


Figure 3: Comparative performance with original pFabric "data mining" scenario, load 0.6.

4.3 Performance comparison

Original data-mining scenario. We first consider the original pFabric scenario where single packet flows represent a significant fraction of flows but only account for a negligible portion of the traffic volume in bytes. Average throughput and single packet flow completion time are reported in Fig. 3. Results confirm that pFabric exhibits outstanding performance for both metrics. However, they also show that FQ-CoDel comes second. Specifically, single packet flow latency for FQ-CoDel is very close to that of pFabric while throughput for long flows is higher than that of DCTCP though still significantly lower than pFabric. Fairness thus appears as a potentially interesting alternative combining simple implementation with performance that may be qualifies as "good enough".

Tweaked data-mining scenario. Robust DCN designs should maintain their performance across scenarios. We perform a very simple yet insightful sensitivity analysis by increasing the volume of byte-wise traffic generated by single packet flows (a similar approach was adopted in [15]). To limit the simulation duration required for reliable statistics, we cap the maximum flow size at 10^5 packets[1]. It is clear from Fig. 4 that the performance penalty of FQ-CoDel with respect to pFabric is reduced significantly as the relative intensity of single packet flows (or rate limited flows) increases. On the other hand, the large gap between pFabric and DCTCP performance is not reduced.

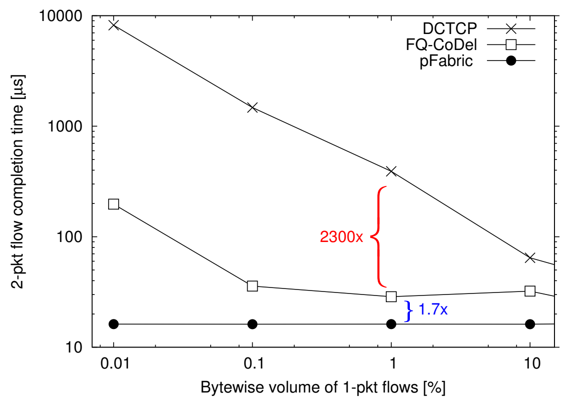
5. A FAIR STATEMENT

This paper addresses multi-tenant DCN: contrarily to the specialization trend, with designs that need to tightly control many aspects of the DCN at several layers of the stack, we argue that DCN design would benefit from an increased generalization: by this, we mean reliance on well understood mechanisms, provided as network services, that require as few as possible assumptions on the DCN workload.

Specifically, we have argued that fair scheduling coupled with a mechanism to explicitly controlling the queue size constitutes an appealing traffic management solution for multi-tenant DCN. 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. Hopefully, since FQ-CoDel implementations are readily available in the Linux kernel, raising awareness of its expected benefits can possibly lead to its deployment in DCN, and especially to comparison with already deployed techniques such as DCTCP, whose performance however fall well below that of FQ-CoDel.

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, however it remains a significant challenge to design and perform the experimental campaign that is necessary to confirm the expected advantages under realistic conditions. More broadly, it is also interesting to better understand how FQ-CoDel would be combined with further key DCN developments (like co-flow identification and the use of packet spraying).

[1]This tweak is in fact favorable to pFabric as the gap between pFabric and FQ-CoDel for 0.01% in Fig. 4 is larger than in Fig. 3.



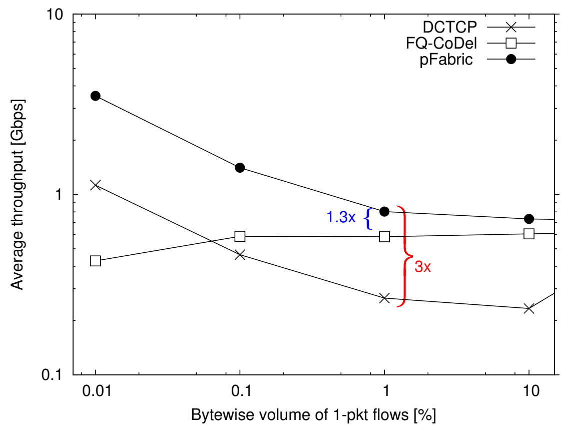


Figure 4: Impact of single packet flow intensity: tweaked pFabric scenario, load 0.6.

Acknowledgements

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

6. REFERENCES

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

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM*, 2008.

[3] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *USENIX NSDI*, 2010.

[4] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese. Conga: Distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM*, 2014.

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). In *ACM SIGCOMM*, 2010.

[6] 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 *USENIX NSDI*, 2012.

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

[8] V. Bahl. Cloud 2020: The Emergence of Micro Datacenters (cloudlets) for Mobile Computing, 2015.

[9] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and W. Sun. Pias: Practical information-agnostic flow scheduling for data center networks. In *ACM HotNets*, 2014.

[10] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing data transfers in computer clusters with orchestra. In *ACM SIGCOMM*, 2011.

[11] M. Chowdhury, Y. Zhong, and I. Stoica. Efficient coflow scheduling with varys. In *ACM SIGCOMM*, 2014.

[12] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. Tuning red for web traffic. In *ACM SIGCOMM*, 2000.

[13] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron. Decentralized task-aware scheduling for data center networks. In *ACM SIGCOMM*, 2014.

[14] N. Dukkipati. *Rate Control Protocol (Rcp): Congestion Control to Make Flows Complete Quickly.* PhD thesis, '08.

[15] P. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker. phost: Distributed near-optimal datacenter transport over commodity network fabric. In *ACM CoNEXT*, 2015.

[16] T. Hoeiland-Joergensen, D. Taht, J. Gettys, and E. Dumazet. FlowQueue-Codel. Internet-Draft, `https://www.ietf.org/archive/id/draft-hoeiland-joergensen-aqm-fq-codel-01.txt`, 2014.

[17] C.-Y. Hong, M. Caesar, and P. B. Godfrey. Finishing flows quickly with preemptive scheduling. In *ACM SIGCOMM*, 2012.

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

[19] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In *ACM CoNEXT*, 2014.

[20] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *ACM SIGMETRICS*, 2005.

[21] A. Kortebi, S. Oueslati, and J. Roberts. Implicit service differentiation using deficit round robin. In *ITC 19*, 2005.

[22] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, and F. R. Dogar. Friends, not foes: Synthesizing existing transport strategies for data center networks. In *ACM SIGCOMM*, 2014.

[23] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *ACM SIGCOMM*, 2016.

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

[25] K. Nichols and V. Jacobson. Controlling queue delay. *Commun. ACM*, 55(7):42–50, July 2012.

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

[27] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal. Fastpass: A centralized "zero-queue" datacenter network. In *ACM SIGCOMM*, 2014.

[28] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath tcp. In *ACM SIGCOMM*, 2011.

[29] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhury. Buffer management schemes for supporting tcp in gigabit routers with per-flow queueing. *IEEE JSAC*, pages 1159–1169, 2006.

[30] Z. Uestc, K. Chen, W. Bai, M. Y. Usc, C. T. Hust, Y. G. Huawei, Y. Z. Nudt, D. L. Tsinghua, and S. W. Uestc. RAPIER : Integrating Routing and Scheduling for Coflow-aware Data Center Networks. *IEEE INFOCOM*, 2015.

[31] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *ACM SIGCOMM*, 2009.

[32] G. White. Active queue management in docsis 3.x cable modems. Technical report, CableLabs, May 2014.

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

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

[35] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz. Detail: Reducing the flow completion time tail in datacenter networks. In *ACM SIGCOMM*, 2012.