NDP: Rethinking Datacenter Networks and Stacks Two Years After

Costin Raiciu
University Politehnica of Bucharest, Romania
costin.raiciu@cs.pub.ro

Gianni Antichi
Queen Mary University of London, UK
g.antichi@qmul.ac.uk

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

CCS CONCEPTS

• **Networks** → **Network protocols**; **Data center networks**;

KEYWORDS

Datacenters; Network Stacks; Transport Protocols

1 DATACENTER NETWORKING STATUS-QUO

The key goals of datacenter networks are to simultaneously provide wire-level latency for RPC-style applications and high-throughput for network-bound applications such as storage. Folded Clos networks [1, 11] are used in datacenters worldwide; such networks use many cheap commodity switches to provide the illusion of a big non-blocking switch to all hosts in a datacenter, offering many paths between any two pairs of hosts.

Efficiently utilizing datacenter networks is challenging. The standard solution is to place each TCP connection onto a quasi-random path chosen from the ones available by using Equal Cost Multipath routing, but this results in flow collisions which can half throughput for long running connections in the worst case [2]. A long line of research addresses this problem of giving high throughput to network-bound traffic: either by using software-defined networking [2, 8], multipath transport [19], modifying switches to track flows and reroute them [3], changing flow ID at the endhosts when performance is poor [15], or, more radically load balancing every packet independently [9].

Few, if any of, these works were adopted in practice, for multiple reasons: a) the problem is less pressing when the host links are slower than switch-to-switch links; b) network-bound applications are rare and have less stringent constraints, and c) they only tackle large flows and largely ignore the numerous short flows.

Reducing latency for short flows is another, largely parallel area of research that was sparked by the discovery of TCP incast [6]: time synchronization of many flows arriving at the same shallow switch buffer leads to repeated timeouts at the endhosts, severely inflating flow completion times. A good solution to incast is DCTCP [4], a TCP congestion control algorithm which relies on aggressive ECN marking at the switches and gently reduces its sending rate on overload instead of halving it. This, coupled with shared switch buffers and a low ECN marking threshold enable DCTCP to cope well with incast and keep queue utilization low in steady state.

Short flows are affected by more pathologies than incast. In particular, when competing with long flows that fill buffers, the latency of short flows increases dramatically due to buffering even when there are no losses. As TCP variants are allergic to loss, large buffers are the norm, so this pathology is omnipresent.

To attack this problem, more radical approaches to ensuring short flow latency beyond incast is to implement strict flow prioritization in the network [5, 14, 20] or use a global scheduler to schedule packets [18] or flowlets [17]. Host based approaches to ensuring low latency include pHost [10] and Homa [16]. While we only mentioned a few here, many other solutions were proposed; unfortunately, few, if any, were deployed in production networks.

Discussions with engineers working in large datacenter networks point to the state of the art relying flow-based ECMP coupled with prioritization in the switches for known latency sensitive traffic (e.g. search), together with application-based solutions that break large flows into chunks which are sent as individual TCP connections. The only research outcome that has been adopted widely is DCTCP, as it alleviates incast and is relatively easy to deploy (simple switch configuration changes coupled with host kernel patches for congestion control).

More recently, RDMA has been proposed and reportedly deployed in production by Microsoft in its datacenters for storage traffic. The key advantage of RDMA is offloading most of the transport stack to the NIC, thus relieving CPU load. The downside is the different API and its reliance on priority-flow control which introduces many performance and availability issues [12].

2 NDP

NDP is a datacenter network architecture and stack that was awarded best paper at Sigcomm 2017; it is also one of the first works that explicitly aims at providing both low latency and high throughput simultaneously. The key differentiator of NDP to prior work is its "clean-slate" approach: assuming you could change both the networks and the endpoints, what would the resulting datacenter architecture be? We took this approach consciously, without worrying about the deployability of NDP, aiming instead to ask what are the right mechanisms to achieve both low latency and high throughput at the same time.

NDP adopts the following three design decisions to provide both high throughput and low latency:

- It runs aggressively small buffers to keep latency low, typically eight to ten packets per port. This means that short flows cannot by delayed by more than said packets at any network hop (assuming their packets arrive at the destination), in comparison to hundreds of packets per port today.

- The sender starts at line rate, assuming the network core can cope with the demand, to provide the lowest possible latency for short and long flows, instead of doing slow start.
- Packets are scattered across all available paths instead of per-flow ECMP.

Taken together, these mechanisms are a recipe for disaster: aggressive starts, scattering and small buffers leads to frequent congestion and packet drops, which makes the job hard for transport protocols to discover which packets got lost and should be resent; the default behavior for TCP in such cases is to fall-back to a retransmit timeout, which kills performance.

That is why NDP also *relies on switch support*. For each port, NDP switches maintain two queues: a lower priority queue for data packets and a higher priority queue for control packets. All data packets go into the lower priority queue as long as it has space; when it fills up, however, the incoming packet's payload is removed and the header is placed in the high priority queue.

Once a packet is trimmed to a header, it always will be forwarded using the high priority queues, thus providing fast notification to the receiver that the packet was lost. Upon receiving a header, the receiver generates a NACK that is sent via high priority queues to the sender, who can send the retransmission before the low priority queue has a chance to drain.

In a 4-to-1 incast scenario, roughly three quarters of the packets will be turned into headers in the first round trip time. If the receiver allows the senders to send upon receiving a NACK, trimming will continue for as long as the incast persists. To avoid this issue, NDP decouples packet delivery notifications (ACKs and NACKs) from packet clocking by having receivers send explicit PULL packets which allow the sender to send one packet.

PULL packets allow the receiver to throttle the incoming traffic after the first RTT to exactly the link rate. To this end, NDP receivers maintain a PULL queue which is increased whenever a new packet or header arrives, and drained at fixed rate (e.g. one pull packet every 1.2us for 10Gbps links with 1500B packets).

The Sigcomm paper showed simulation and testbed deployment results that show NDP can achieve near-optimal short flow-completion rates and more than 95% of the theoretical maximum throughput for long flows [13].

2.1 Making NDP Real

There are two key challenges in transitioning NDP from a research prototype into a deployable solution. First, switches must be changed to support packet trimming. In principle, this change is simple as the switches need not maintain any per-flow state, and can operate on each packet independently. The rise of P4 and programmable switches gave us hope that such functionality could be readily supported, and our paper included a P4 implementation of NDP support. However, at the time we wrote the paper Tofino switches were not available for testing, and there was no guarantee that our code could actually run on a Tofino target. That is why we ran all our experiments on the NetFPGA implementation of the NDP switch developed by Andrew Moore's team at the University of Cambridge.

The second challenge is implementing pull pacing which used by the NDP endhost stack to ensure that packets arrive at the correct rate. In short, pull pacing needs to place a PULL packet on the wire every 1.2us (for 10Gbps links), 300ns for 40Gbps links and every 120ns for 100Gbps links assuming 1500B MTU.

The prototype implementation we used for our evaluation in the Sigcomm paper (open source on GitHub) provided accurate timing at 10Gbps by spinning a CPU core just for pull pacing; spinning is needed because sleeps involve the OS scheduler and result in large delay variations. A production version of NDP cannot burn server CPUs for this task, as these are used for application workloads. The obvious solution is to implement NDP on a smartNIC, either on a SoC-based one (e.g. Broadcom Stingray PS225) or an FPGA-based one (e.g. Intel N3000).

3 DEPLOYMENT PROSPECTS

To understand the impact of NDP two years after its publication we estimate the potential for deployment in this section and the research impact in the next one.

We, like all other systems researchers, strive to create systems that are used in practice. Our previous experience with Multipath TCP shows that the pain of getting MPTCP standardized at the IETF and creating and maintaining a stable Linux kernel implementation was worth the effort, as MPTCP is now actively used widely.

Our experience at getting things deployed in datacenters, however, is not as good. NDP's presentation at Sigcomm 2017, implemented and presented by Mark Handley using the Unity game engine, caused quite a stir. Riding on this wave, we approached people from the big three cloud providers asking whether they would be interested in trying NDP in their datacenters. The answers were underwhelming, mentioning lack of switch support as a show-stopper. Instead, it was suggested we should find a solution that does not require switch support. Homa [16], published at Sigcomm 2018 does exactly this, while sacrificing some performance in corner cases; it remains to be seen if it will be adopted in practice. This was not the first time we had tried to get something deployed in datacenters and failed, so the answers convinced us that this was an uphill battle we had little chance of winning. Like the rest of our peers in the community, we moved to other areas where our research could have an impact.

In the meantime, however, something surprising happened: out of the blue, we were approached by three switch vendors that were willing to implement NDP in their switches. We began collaborating with them, and one has a working prototype that we are now testing. A smartNIC implementation of the NDP endhost stack is also ongoing, giving us hope that a prototype deployment of NDP in close-to-production networks is possible in the near future.

Whether NDP will make its way into the big three clouds or whether it will be adopted for smaller, latency sensitive enterprise datacenters, it is too soon to tell. At this point, there is however a good chance that NDP may be used in practice after all.

4 RESEARCH IMPACT

The problem of efficiently utilizing datacenter networks has been in the research spotlight for ten years now, with thousands of papers published on this topic (the Hedera paper that opened the field has 1300+ citations, same for DCTCP).

NDP appears to be in a batch of papers that effectively close this field, (along with Homa at Sigcomm 2018 and ExpressPass at Sigcomm 2017 [7]): the number of papers addressing this problem has dropped constantly (13 papers in 2017, 5 in 2018 and none at NSDI 2019).

NDP has currently 67 citations on Google Scholar, ranking fourth out of 36 papers in Sigcomm 2017 after QUIC and Pensieve (200 citations each) and SilkRoad (Facebook's load balancer, with 70 citations). The NDP paper has been downloaded 7300 times from the ACM digital library, coming second to Google's QUIC paper (9000 downloads) in the Sigcomm 2017 batch (average number of downloads is 2400).

The high number of downloads hints that NDP has generated interest, but this interest has not translated to citations at the same rate as it did for QUIC; this may be linked to the slowing amount of research done on the problem.

The real measure of impact, however, is whether NDP will be deployed and used in practive. It is too soon to tell whether this will be the case, but there are some promising signs.

ACKNOWLEDGMENTS

We conclude this short restrospective of the NDP work by pointing out that in our experience, networking research impact is possible in practice but it takes long term projects and large teams to achieve.

While this editorial has been written by a subset of the NDP authors, we would like to point out that NDP was a collaborative project: the design and simulations were done by Mark Handley (UCL) and Costin Raiciu (UPB), the main software implementation was done by Alexandru Agache (UPB), the P4 implementation by Andrei Voinescu (UPB) and the NetFPGA-based switch implementation by Gianni Antichi, Marcin Wojchik and Andrew Moore (University of Cambridge).

Our reference to the Multipath TCP is again a collective one, with Mark Handley (UCL) and Costin Raiciu (UPB) contributing the protocol design, initial implementation and first phase of standardization and Olivier's Bonaventure team at Universite Catholique de Louvain being the main contributors the mature Linux kernel implementation which is now used widely. Many other people have contributed to MPTCP including Alan Ford (Pexip) leading the lengthy standardization work, Phil Eardley and Yoshifumi Yoshida leading the MPTCP working group, Marcelo Bagnulo (UC3M) working on the security part, to name just a few.

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A Scalable, Commodity Data Center Network Architecture. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. 2010. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Networked Systems Design and Implementation (NSDI)*. USENIX Association.

[3] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: Distributed Congestion-aware Load Balancing for Datacenters. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[4] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[5] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pFabric: Minimal Near-optimal Datacenter Transport. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[6] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. Understanding TCP Incast Throughput Collapse in Datacenter Networks. In *Workshop on Research on Enterprise Networking (WREN)*. ACM.

[7] Inho Cho, Keon Jang, and Dongsu Han. 2017. Credit-Scheduled Delay-Bounded Congestion Control for Datacenters. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[8] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. 2011. DevoFlow: Scaling Flow Management for High-performance Networks. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[9] Advait Dixit, Pawan Prokash, Charlie Y. Hu, and Ramona R Kompella. 2013. On the Impact of Packet Spraying in Data Center Networks. In *International Conference on Computer Communications (INFOCOM)*. IEEE.

[10] Peter X. Gao, Akshay Narayan, Gautam Kumar, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. 2015. pHost: Distributed Near-optimal Datacenter Transport over Commodity Network Fabric. In *Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. ACM.

[11] 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 *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[12] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over Commodity Ethernet at Scale. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[13] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting Datacenter Networks and Stacks for Low Latency and High Performance. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[14] Chi-Yao Hong, Matthew Caesar, and P. Brighten Godfrey. 2012. Finishing Flows Quickly with Preemptive Scheduling. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[15] Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene. 2014. FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks. In *Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. ACM.

[16] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A Receiver-driven Low-latency Transport Protocol Using Network Priorities. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[17] Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. 2017. Flowtune: Flowlet Control for Datacenter Networks. In *Networked Systems Design and Implementation (NSDI)*. USENIX Association.

[18] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: A Centralized "Zero-queue" Datacenter Network. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[19] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2010. Improving Datacenter Performance and Robustness with Multipath TCP. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.

[20] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. 2011. Better Never Than Late: Meeting Deadlines in Datacenter Networks. In *Special Interest Group on Data Communication (SIGCOMM)*. ACM.