Fast Congestion Control in RDMA-based Datacenter Networks

Jaichen Xue
Purdue University

Muhammad Usama Chaudhry
University of Illinois at Chicago

Balajee Vamanan
University of Illinois at Chicago

T. N. Vijaykumar
Purdue University

Mithuna Thottethodi
Purdue University

CCS CONCEPTS

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

KEYWORDS

Datacenters; RDMA; Congestion Control

1 MOTIVATION

Many modern, interactive datacenter applications have tight latency requirements due to stringent service-level agreements (e.g., under 200 ms for *Web Search*). TCP-based datacenter networks significantly lengthen the application latency. Remote Direct Memory Access (RDMA) substantially reduces latencies compared to TCP by bypassing the operating system via hardware support at the network interface (e.g., RDMA over InfiniBand and RDMA over Converged Ethernet (RoCE) can cut TCP's latency by 10x [8]). As such, RDMA may soon replace TCP in datacenters.

Employing RDMA in datacenters, however, poses a challenge. RDMA provides hop-by-hop flow control and rate-based end-to-end congestion control [4]. However, RDMA's congestion control is suboptimal for the well-known datacenter congestion problem, called *incast*, where multiple flows collide at a switch causing queuing delays and long latency tails [1] despite good network design [7]. Though such congestion affects only a small fraction of the flows (e.g., 0.1%), datacenter applications' unique characteristics imply that the average latency is worsened. For example, because Web Search aggregates replies from thousands of nodes, the 99.9^{th} percentile reply latency affects the average response time; or alternatively, dropping the slowest replies worsens the response quality. In TCP, incasts cause delays due to packet drops and re-transmissions [1]. Though the lossless RDMA does not incur packet drops, incast-induced queuing delays lengthen RDMA's latency tail [10].

InfiniBand uses Early Congestion Notification (ECN) marks to infer imminent congestion and cuts back the sending rates [4]. While DCQCN proposes a similar scheme for RoCE, TIMELY [9] uses round-trip times (RTT) measurements, instead of ECN marks, for rate control in user-level TCP. Unfortunately, because ECN marks and RTT measurements need many round-trips to converge to the appropriate sending rates (e.g., 50 RTTs in TIMELY), the schemes are too slow for the applications' predominantly short flows each of which lasts only a handful of round-trips. During convergence, the schemes also lose throughput due to over- and under-shooting the sending rates.

2 OUR PROPOSAL

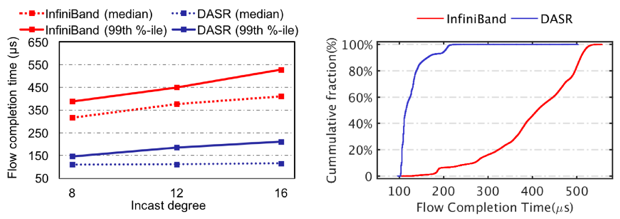
To speed up convergence, we leverage the result in several papers and reports from large datacenter operators such as Facebook, Google and Microsoft [6]: even under typical oversubscription most congestion in datacenter networks occurs at the network edge (i.e., at the link from top-of-rack (ToR) switch to the receiver) as opposed to within the network. Our simulations confirm this result which is due to high-bandwidth network core [7] and incast at the receiver. We make the key observation that while general congestion is complex and may require iterative convergence, the simpler and common case of receiver congestion can be addressed quicker via specialization. *Without isolating this case, previous schemes apply their iterative throttling to the general case. Instead, our proposal, called* Blitz, *employs a divide-and-specialize approach to isolate receiver congestion and significantly speeds up the convergence.* Blitz sub-divides the remaining case of in-network congestion into the simpler spatially-localized case and the harder spatially-dispersed case. For the former where the network capacity is not under pressure (e.g., due to imperfect ECMP hashing), Blitz avoids

SIGCOMM Posters and Demos '18, August 20–25, 2018, Budapest, Hungary
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5915-3/18/08…$15.00
https://doi.org/10.1145/3234200.3234216

throttling which is unnecessary. For the latter where the network capacity is under pressure (e.g., due to dynamic network load spikes), Blitz falls back on DCQCN's throttling which may be unavoidable.Load balancing can alleviate localized in-network congestion but not receiver congestion, and usually reorders packets which is not supported by RDMA.

To address receiver congestion, we make the key observation that unlike in a wide-area setting, datacenter applications are co-operative where a receiver of n senders can direct each sender to cut its rate by a factor of n, This mechanism, called *direct apportioning of sending rates (DASR)*, ensures that the critical, short flows get their fair share of (instantaneous) throughput without being swamped by the background, long flows. When a sender completes, the (instantaneous) sending rate is adjusted as per the new sender count. Because DASR piggybacks the count in the receiver's acknowledgments to the senders, DASR achieves accurate and *one-RTT* convergence of sending rates without any repeated adjustments, unlike previous schemes. Specifically, (1) RCP [3] proposes to apportion the rates among the senders, but employs slow, iterative convergence *at the switches because RCP targets general congestion without isolating receiver congestion.* (2) EyeQ [5] highlights edge congestion but applies RCP's iterative convergence, which takes 25-30 RTTs, without specializing for edge congestion. (3) NUMFabric achieves more flexible and faster bandwidth allocation than TCP but still employs iterative convergence (e.g., 31 RTTs). And, (4) while ExpressPass and NDP target general congestion via receiver-based congestion control,neither scheme isolates receiver congestion.NDP fundamentally relies on (a) packet spraying, which reorders packets, to reduce congestion and (b) packet trimming, which removes payloads, to unclog congestion notification to the receiver. Neither of these mechanisms is supported by RDMA which has no software stack like TCP. Without these mechanisms, NDP would see more congestion and slower feedback. DASR's faster convergence reduces latency tail (critical flows quickly get their share) and improves throughput (fewer adjustments).

To address spatially-localized, in-network congestion, Blitz simply deflects the affected packets under the premise that an alternate path is faster than being queued up in the shortest path. To avoid livelock, Blitz allows only a few deflections for a packet after which the packet is not deflected even at a congested switch. Blitz avoids deadlocks via a widely-used virtual-channel-based scheme [2]. Because RDMA does not support packet reordering, Blitz provides hardware support in the switch to keep a flow's packets in order. While deflection is well known, our contribution is *in-order flow deflection (IOFD)* unlike previous load-balancing schemes including DIBS. As a congestion response, deflection is much lighter-weight and quicker (well under one RTT) than rate-cutting using iterative convergence and does not affect the sending



(a) Median and 99th %-ile flow completion time (b) CDF of flow completion times

Figure 1: Testbed flow completion latency

rates. For spatially-dispersed in-network congestion, which is uncommon, Blitz falls back to DCQCN's heavy-weight rate modulation. By filtering out receiver congestion and localized in-network congestion, Blitz cuts the number of ECN marks, which trigger DCQCN fall-backs, by 4x for typical workloads.

3 PRELIMINARY EVALUATION

Our testbed consists of 20 nodes, each consisting of four eight-core AMD Opteron 6320 CPUs running at 2.8 GHz and 256 GB of memory, which connect to a 36-port *Mellanox SX6025 InfiniBand* switch using Mellanox ConnectX-3 Pro HCA. The switch provides bidirectional bandwidth of 56 Gbps per port. All the nodes run RHEL6.7 (kernel version 2.6.32) and Mellanox OFED 3.3-1.0.4.

We compare the completion times of short, incast flows and throughput of long, background flows of InfiniBand and DASR. We initiate short 256-KB incasts from a group of servers every 100 ms to an *aggregator* server. Meanwhile, we send continuous background traffic from another server to the aggregator. We introduce random jitter of 0-100 μs among the incast senders in each round. While InfiniBand uses its congestion control, we implement DASR's rate control by staggering the messages in time at the application layer.

Figure 1 shows the median and tail (99th percentile) flow completion times of DASR and InfiniBand (Y-axis), for varying incast degrees (X-axis). As expected, higher incast degrees lead to longer flow completion times and even longer tails. DASR reduces the medians and tails by 2.5 - 3.3x. DASR's reductions in the tails are close to those in the medians because the tails are only about 1.2x longer than the medians in InfiniBand due to our testbed's (small) scale. As the tails grow at datacenter scales (e.g., 5-10x of the median), DASR would achieve greater tail reductions. Figure 1(b) shows the flow completion time distributions of InfiniBand and DASR for the incast degree of 16. As compared to InfiniBand, DASR reduces the spread and shifts the curve to the left. Both DASR and InfiniBand achieve similar throughput (within 0.5%) for long flows (not shown).

REFERENCES

[1] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari

Sridharan. 2010. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 conference (SIGCOMM '10)*. ACM, New York, NY, USA, 63–74. https://doi.org/10.1145/1851182.1851192

[2] José Duato. 1993. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Trans. Parallel Distrib. Syst.* 4, 12 (Dec. 1993), 1320–1331. https://doi.org/10.1109/71.250114

[3] Nandita Dukkipati, Masayoshi Kobayashi, Rui Zhang-Shen, and Nick McKeown. 2005. Processor Sharing Flows in the Internet. In *Proceedings of the 13th International Conference on Quality of Service (IWQoS'05)*. Springer-Verlag, 271–285.

[4] E.G. Gran, M. Eimot, S.-A. Reinemo, T. Skeie, O. Lysne, L.P. Huse, and G. Shainer. 2010. First experiences with congestion control in InfiniBand hardware. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. 1–12. https://doi.org/10.1109/IPDPS.2010.5470419

[5] Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Changhoon Kim, and Albert Greenberg. 2013. EyeQ: Practical Network Performance Isolation at the Edge. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (nsdi'13)*. 297–312.

[6] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The Nature of Data Center Traffic: Measurements & Analysis. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC '09)*. ACM, 202–208.

[7] Charles E. Leiserson. 1985. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* 34, 10 (Oct. 1985), 892–901. http://dl.acm.org/citation.cfm?id=4492.4495

[8] Jiuxing Liu, Jiesheng Wu, and Dhabaleswar K. Panda. 2004. High Performance RDMA-based MPI Implementation over infiniBand. *Int. J. Parallel Program.* 32, 3 (June 2004), 167–198. https://doi.org/10.1023/B:IJPP.0000029272.69895.c1

[9] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 537–550. https://doi.org/10.1145/2785956.2787510

[10] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, 523–536. https://doi.org/10.1145/2785956.2787484