Research Statement

Radhika Mittal
University of California at Berkeley

With most of the day-to-day applications running over computer networks, there is now an increased emphasis on network performance. Network performance not only affects user satisfaction, but also has a huge impact on the revenues generated by online service providers such as Google, Microsoft and Facebook [1]. While the end-to-end principle [16] was a central pillar guiding early network design, this increased emphasis on performance has led to proposals for adding more sophisticated features in the hardware equipment from which network infrastructure is built. Such features include complicated scheduling policies [3, 5, 18], explicit rate signalling [6, 9], push-back mechanism to achieve losslessness [2], among others. While these features promise improved performance they come at a high cost. Many of these features need specialized network switching hardware, requiring frequent reconfigurations, upgrades and sometimes replacements in order to meet varying performance goals in varying scenarios. As a result, these proposals make networks more difficult to develop, deploy and manage.

I am interested in finding ways to meet the required performance objectives while reducing the need for specialized in-network support: *Can a network be built on general infrastructure yet meet a diverse range of performance goals?* I have tackled this question using the following two approaches: (1) can we achieve good performance using the network infrastructure deployed commonly today?, and (2) can we design a new *universal* network hardware that would obviate the need for newer ones? For the former, I revisited the need for specialized in-network support for loss recovery and congestion control algorithms (which determine the rate at which endhosts transmit data). I looked at the latter in the context of scheduling and queue management algorithms implemented in the network switches (which determine the order in which packets are transmitted and the packet dropping policy of the switch respectively). I will now elaborate on both of these approaches one by one.

Good Performance Using Common Network Infrastructure

I have designed schemes to get good performance using common network infrastructure, both in the context of wide-area and datacenter networks.

Wide Area. For wide-area networks, I designed a new congestion control algorithm called RC3 [12], that makes use of coarse-grained priority scheduling (present in commonly available switches) to fully utilize the available network capacity. RC3 provides strong gains over traditional TCP, and performs better than schemes that require explicit bandwidth signaling support from the switches [6, 9].

Datacenter. My current project looks at the network support needed for running RDMA (Remote Direct Memory Access) in datacenters. In recent years, the usage of RDMA in datacenter networks has increased significantly, due to its ability to maintain ultra-low endhost latency along with minimal CPU utilization. RDMA over Converged Ethernet (RoCE) has emerged as the canonical method for deploying RDMA in Ethernet-based datacenters [7, 20]. Early experience revealed that RoCE NICs only achieve good end-to-end performance when run over a lossless network, so operators have devoted significant effort in configuring Ethernet's Priority Flow Control (PFC) mechanism to achieve negligible packet loss [2]. With PFC, a switch sends a pause (or X-OFF) message to the upstream entity, when the queue exceeds a certain configured threshold. When the queue drains below the threshold, an X-ON message is sent to resume transmission. The combination of RoCE and PFC has enabled a wave of datacenter RDMA deployments, thereby validating the feasibility of large-scale RDMA deployments over Ethernet.

However, the current solution is not without problems. In particular, PFC adds complexity to network management, it is hard to understand, and can lead to significant problems such as head-of-the-line blocking, congestion spreading, and occasional deadlocks [7,8,17,19,20]. During my internship at Google, I developed a delay-based congestion control algorithm for RDMA-based communication called Timely [11], which reduced the number of pause frames triggered (by reducing the queuing delay in the network), but could not fully eradicate the need for PFC.

At this point, the networking community faces a critical choice. It could attempt to fix the problems with PFC which, judging by the recent literature [7,8,17], is the path most are pursuing. However, this path would require both new insights and changes to the core of most datacenter networks. Rather than proceeding along this path, in this work, we took a step back and asked whether PFC is fundamentally required for deploying RDMA over Ethernet, or is its use merely an artifact of the current RoCE NIC design.

RoCE is not the only approach to deploying RDMA over Ethernet. iWARP [14] was designed to support RDMA over a generic (non-lossless) network. It reimplements the entire TCP protocol in the hardware NIC along with multiple other layers that it needs to maintain the TCP stream semantics. As a result, iWARP NICs are considered to be significantly more complex than RoCE, making RoCE the more popular choice. Our approach to improving RoCE is to adopt the architectural approach of iWARP (having the NIC handle some basic congestion control and loss recovery), but the implementation approach of RoCE (adding only a minimal set of additional features to current RoCE NICs rather than putting an entire set of protocols in hardware).

We, thus, propose a new design called IRN (for Improved RoCE NIC) that makes two key changes to current RoCE NICs (i) more efficient loss recovery, and (ii) basic end-to-end flow control that bounds the number of in-flight packets by the bandwidth-delay product of the network. We show, via extensive simulations, that IRN does not require PFC, yet performs better than current RoCE NICs with PFC. While our design requires extensions to the RDMA packet format, we show that it is relatively easy to implement in hardware by synthesizing IRN's packet processing logic to target an FPGA device supported by Mellanox Innova Flex 4 NICs and by comparing IRN's memory requirements to those of current RoCE NICs. Inspired by our results, Mellanox is considering implementing a version of IRN in their next release.

Universal Network Hardware

There is a large and active research literature on novel packet scheduling algorithms, from simple schemes such as priority scheduling [15], to more complicated mechanisms for achieving fairness [5,13,18], to schemes that help reduce tail latency [4] or flow completion time [3], and this short list barely scratches the surface of past and current work. In this work, rather than adding to this impressive collection of algorithms, we instead studied if there is a single *universal packet scheduling algorithm* (hereafter, UPS) that could obviate the need for new ones. We can define a universal packet scheduling algorithm in two ways, depending on our viewpoint on the problem.

Theoretical Viewpoint. From a theoretical perspective, we call a packet scheduling algorithm *universal* if it can replay any *schedule* (the set of times at which packets arrive to and exit from the network) produced by any other scheduling algorithm. We proved while there is no UPS, the classical Least Slack Time First (LSTF) [10] comes as close as any scheduling algorithm to achieving universality. We then showed, via simulation, how LSTF can closely approximate the schedules of many scheduling algorithms. Thus, while not a perfect UPS in terms of replayability, LSTF comes very close to functioning as one.

In LSTF, each packet carries its slack value in the packet header, which is initialized at the network ingress and indicates the maximum possible queuing delay that the packet is willing to tolerate in the network. The switch always picks the packet with the least slack time for scheduling, and additionally updates the slack of the packet by subtracting the duration of time for which the packet has waited in the router queue.

In order to replay a schedule, LSTF uses the exit time of each packet from the original schedule to initialize its slack value. The theoretical viewpoint is, therefore, not of practical interest, since such schedules are not typically known in advance, but it offers a theoretically rigorous definition of universality that helps illuminate its fundamental limits (i.e., which scheduling algorithms have the flexibility to serve as a UPS, and why).

Practical Viewpoint. From a more practical perspective, we say a packet scheduling algorithm is universal if it can achieve different desired performance objectives. In particular, we require that the UPS should match the performance of the best known scheduling algorithm for a given performance objective. For this we came up with different heuristics for header (or slack) initialization to achieve different performance objectives and showed (via simulation) that LSTF is comparable to the state of the art for each of them. We investigated LSTF's ability to minimize average flow completion times, minimize tail latencies, and achieve per-flow fairness. We also looked at how network feedback for active queue management (AQM) can be incorporated using LSTF. Rather than augmenting the basic LSTF logic (which is restricted to packet scheduling) with a queue management algorithm, we showed that LSTF can, instead, be used to implement AQM at the edge of the network.

This work, therefore, implies that we do not need to support many different scheduling and queue management algorithms in the switch hardware to meet different objectives, and can instead, just use LSTF with different header (or slack) initialization.

References

[1] The Cost of Latency. `http://perspectives.mvdirona.com/2009/10/the-cost-of-latency/`.

[2] IEEE. 802.11Qbb. Priority based flow control, 2011.

[3] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pFabric: Minimal Near-optimal Datacenter Transport. In *Proc. ACM SIGCOMM*, 2013.

[4] D. D. Clark, S. Shenker, and L. Zhang. Supporting Real-time Applications in an Integrated Services Packet Network: Architecture and Mechanism. *ACM SIGCOMM Computer Communication Review*, 1992.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *ACM SIGCOMM Computer Communication Review*, 1989.

[6] N. Dukkipati and N. McKeown. Why Flow-Completion Time is the Right Metric for Congestion Control. *ACM SIGCOMM Computer Communication Review*, 2006.

[7] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. Rdma over commodity ethernet at scale. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 202–215. ACM, 2016.

[8] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen. Deadlocks in datacenter networks: Why do they form, and how to avoid them. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, pages 92–98, New York, NY, USA, 2016. ACM.

[9] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *Proc. ACM SIGCOMM*, 2002.

[10] J. Y.-T. Leung. A new algorithm for scheduling periodic, real-time tasks. *Algorithmica*, 1989.

[11] R. Mittal, V. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats. TIMELY: RTT-based Congestion Control for the Datacenter. In *Proc. ACM SIGCOMM*, 2015.

[12] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Recursively Cautious Congestion Control. In *Proc. USENIX NSDI*, 2014.

[13] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-node Case. *IEEE/ACM Trans. Netw.*, 1993.

[14] R. Recio, B. Metzler, P. Culley, J. Hilland, and D. Garcia. A Remote Direct Memory Access Protocol Specification. RFC 5040, 2007.

[15] S. Blake and D. Black and M. Carlson and E. Davies and Z. Wang and W. Weiss. An Architecture for Differentiated Services. RFC 2475, 1998.

[16] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 1984.

[17] A. Shpiner, E. Zahavi, V. Zdornov, T. Anker, and M. Kadosh. Unlocking credit loop deadlocks. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 85–91. ACM, 2016.

[18] M. Shreedhar and G. Varghese. Efficient Fair Queueing Using Deficit Round Robin. *ACM SIGCOMM Computer Communication Review*, 1995.

[19] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter. Practical DCB for improved data center networks. In *Proc. IEEE Infocom*, 2014.

[20] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion Control for Large-Scale RDMA Deployments. In *ACM SIGCOMM*, 2015.