I am interested in research in network systems targeting direct benefits to application performance, especially in datacenters. My research focus and approach has evolved as I journeyed from my PhD work to research labs (MSR, Intel Labs), and then to industry (Google). My approach to solving problems has been to use theoretical analysis to develop a deeper understanding of the problem, and then address the underlying cause by leveraging software packet processing. This allows my approach to be deployed without requiring changes to the underlying network.

During my PhD, I challenged conventional wisdom that software packet processing is slow, I contributed to PacketShader and SSLShader which demonstrated that software can process tens of gigabits of traffic, thus enabling the widespread use of software packet processing. During my postdoc at MSR, I shifted my focus to datacenter networks, in particular, network performance problems arises from interferences between applications sharing the network. I contributed to building abstractions and mechanisms for providing network performance guarantees for applications running in multi-tenant datacenters, by applying network calculus theory to prove guarantees and making use of advances software packet processing to demonstrate solutions. Afterwards, I wanted to ground my research on problems from the real world and joined a team in Google with the track record of innovation in datacenter networks. Experience in operational datacenter networks have gained me more insights into fundamentals of network congestions and requirements for deployment in the running datacenters. In ExpressPass, we identify the fundamental limitations on reactive congestion control and propose a solution that works with existing switching chips and scales to datacenters.

I discuss network isolation and congestion control problems in datacenters next, then I discuss software packet processing, and finally I describe my future plans for research.

1 Network performance isolation in the cloud datacenters

One key promise of multi-tenant datacenters is increasing resource efficiency by exploiting statistical multiplexing. Network performance (both bandwidth and latency) is a critical factor in determining the performance of applications. However, when run in multi-tenant datacenters these applications share the network with other applications and tenants and experience unpredictable network performance due to interference. This unpredictability hinders cloud adoption. For example, when Netflix moved from their own datacenters to Amazon Web Services, they had to refactor applications to account for higher variance in latency. My research looked into abstractions for network performance requirements for cloud users to reason about their application performance and mechanisms for predictable network performance to meet the requirements.

Two of my research projects have focused on providing predictable network performance. The first, Silo [5], builds on theoretical results from network calculus to provide latency guarantees, and the second, Hadrian [1], identifies a new hierarchical hose model and bandwidth allocation policy to provide bandwidth guarantees for inter-tenant traffic.

1.1 How to provide latency guarantee in multi-tenant datacenters?

Many cloud applications can benefit from guaranteed latency for their network messages. However, implementing such guarantees is hard. In Silo [5], we identified three key requirements for providing predictable latencies: guaranteed network bandwidth, guaranteed packet delay and burst allowance. Among these three requirements, guaranteed packet delay is particularly difficult to achieve since it requires precise control over interactions among independent flows at all network switches.

To address these issues, we developed Silo [5], a network sharing solution for multi-tenant datacenters that provides bandwidth and latency guarantees. The key intuition to provide latency guarantees comes from network calculus, a theoretical model that provides a framework to reason about the worst case queuing delay. It allows us to determine the exact rate each VMs to achieve latency guarantee for all VMs in the network. Anoth key challenge is to shape the traffic precisely to conform to the allocation. To guarantee queuing at the packet-level granularity in the network, each host must control packet departure time at microsecond level accuracy. We implemented fine-grained packet timing with a software pacer that can control inter-packet gap at few ten nanoseconds granularity without hardware support. Silo demonstrates that applications that are throughput sensitive (i.e., require high bandwidth) and ones that are latency sensitive (i.e., require low delay through the network) can co-exist in multi-tenant environments without any application level changes.

1.2 How to provide bandwidth guarantee in the presence of inter-tenant traffic?

While the problem of providing bandwidth guarantees in multi-tenant datacenters is well studied, prior results have built on the assumption that all traffic from a tenant's VM goes to other VMs belonging to the same tenant, i.e., all traffic is intra-tenant. However, this assumption does not hold in general, and as a result, intra-tenant bandwidth guarantees are not sufficient in today's datacenters. We analyzed the traffic in several Microsoft datacenters, and found that inter-tenant traffic can amount up to 35% of the total datacenter traffic [1]. Such inter-tenant traffic makes the network-sharing problem challenging both quantitatively and qualitatively. First, offering minimum bandwidth guarantees for inter-tenant traffic is harder as the set of VMs that can communicate with each other is significantly larger. Second, since each tenant independently pays for traffic, which tenant's bandwidth allocation should be used when allocating capacity for inter-tenant traffic?

To tackle this problem, we developed a hierarchical host model where each tenant can define their intra-tenant bandwidth requirements and inter-tenant bandwidth requirements separately. We develop the bandwidth allocation policy called "upper bound proportionality," where the maximum bandwidth each VM can achieve is proportional to its payment, and we prove that this can provide minimum bandwidth guarantees. Upper bound proportionality provides strategy proofness by not allowing a malicious VM to gain more bandwidth by communicating with more VMs. In turn, it allows us to bound the maximum required bandwidth to meet the guarantee on any link regardless of its communication pattern.

2 Congestion control in datacenter networks

Datacenter networks have characteristics that are different from the Internet, which makes congestion control challenging. They have very low latency up to a few tens of microseconds at the physical layer and very high bandwidth going up to 100Gbps or more. Such environments coupled with Remote Direct Memory Access (RDMA) provide new opportunities for distributed applications, such as running complex graph algorithms in memory across thousands of servers, and large scale in-memory databases. However, realizing these new opportunities present new challenges for congestion control to keep the latency low while still providing high bandwidth. DCTCP pioneered this space by exploiting existing ECN (Explicit Congestion Notification) to keep the queuing delay to few tens of packets.

Since datacenters are operated by a single entity, there is opportunity to use new congestion control algorithms without the burden of compatibility with traditional TCP. My research explores ways to leverage this opportunity, by looking at different ways to detect and prevent congestion, and support extremely latency sensitive applications and bandwidth hungry applications at the sametime.

2.1 Is delay a good signal for congestion control in datacenters?

The idea of using delay for congestion control is not new and has been extensively studied in the past in the wide-area networks. However, the datacenter environment is very different, posing unique requirements that are difficult to address. Datacenters have much higher bandwidth ranging up to 100Gbps at the server, and very low latency as low as few tens of microseconds in the network. This makes it difficult to measure the queuing delay of individual packets for a number of reasons: (i) I/O batching at the end host, which is essential for high throughput, introduces large measurement error. (ii) Measuring queuing delay requires high precision because a single packet introduces less than a microsecond of queuing delay in 40Gbps networks. As a result, it is commonly believed that latency measurements in datacenter networks are noisy, and do not serve as a reliable indicator for congestion control.

In DX [7], we use a combination of software packet processing on DPDK and H/W timestamping to achieve sub-microsecond accuracy that can measure even a single packet queuing. We found that even H/W timestamping can be inaccurate due to I/O batching, and solved this with software calibration. We then developed a congestion control algorithm to take advantage of accurate queuing delay measurements. For a given amount of queuing in the network, we mathematically solved the exact amount of slowdown needed to drain the queue without underutilization. This allows us to achieve a more optimal balance between queuing and utilization when compared to DCTCP or HULL.

2.2 Delay bounded congestion control for datacenters

Reactive congestion control algorithms, which adjust packet rates in response to congestion observed by previous packets, are prone to unbounded queuing in the network. When there are many flows competing in the network, even if each flow sends packets at its fair-share rate, all packets from all flows can still arrive at a switch at the same time resulting in queue build up. Prior work on ensuring queue bounds can be classified into two categories: credit-based flow control schemes used in high-speed interconnects, such as Infiniband and PCIe, which require switch support and have scalability issues since they require pair-wise virtual channels to be set up between all pairs of hosts; and approaches, such as FastPass, which rely on a centralized controller to schedule every single packet to achieve zero-queuing, where the use of a centralized controller limits scalability.

In ExpressPass [2], we ask the question: how can we achieve the benefit of credit-based flow control in switched Ethernet networks at the scale of datacenters? Our approach uses credit packets to control the rate and schedule the arrival of data packets. Receivers send credit packets to senders, switches then rate-limit the credit packets on each link, and determine the available bandwidth for data packets flowing in the reverse direction. By throttling credit packets in the network, the system proactively prevents congestion before data packets are transmitted. It solves the incast problem because the throttling of credit packets at the bottleneck link naturally schedules the arrival of data packets in the reverse path at packet granularity. In addition to bounded queuing, it opens up a new space for optimizing congestion control. Loss of credit packets in the network is cheaper than the loss of data packets, and it allows flows to ramp up much more aggressively without increasing delay or causing loss for other flows in the network. Reactive congestion control is juggling three dimensions: convergence, queuing, and utilization, while credit based congestion control naturally regulates queuing which allows broader range of feedback control schemes.

3 Other research

Software routers and middleboxes running on commodity hardware are increasingly prevalent in both datacenters (where they connect virtual machines that comprise the cloud), and in wide-area networks as a part of network function virtualization (NFV) deployments. Enabling this use case has required developing techniques that enable high performance packet processing on commodity hardware. I contributed to PacketShader [3] which built user space packet processing that achieves tens of millions packets per second on commodity CPUs. Its approach has been widely adopted through DPDK, NetMap, PFRing, etc. Even with high packet I/O rate, network processing can still be too compute and memory-access intensive to be cost effective. We researched methods to use GPUs to make software routers even more cost-effective and deliver high-performance for complex processing such as OpenFlow, IPSec, and SSL [3, 4] at a fraction of the cost of specialized accelerators. In our follow-up work, we researched methods of dynamically balancing loads between GPUs and CPUs to maximize the throughput across varying workloads [6].

Building network functions from the scratch is difficult and time consuming. I have contributed to research that builds frameworks to ease developing network software by providing reusable, modular building blocks and applying programming language advances. NetBricks [8] explored using a high-level programming language, Rust, for building network functions, and disproved conventional wisdom that high-level programming languages are too slow for network processing. I also contributed to BESS (a.k.a SoftNIC), a modular software switch for NFV applications.

4 Future plans

Datacenter present opportunities for innovation in networking, and there are many challenges ahead to support higher bandwidth, lower latency and better efficiency. The problems I am interested in solving in the future include:

Offload network processing to hardware: While software provides great flexibility for implementing new network mechanisms, it is fundamentally limited in terms of performance and efficiency. I believe hardware acceleration is going to be crucial for efficiently implementing these mechanisms in networks with line rates

beyond 100 Gbps. This is particularly important in cloud as savings on CPUs directly translates to revenue by allowing more VMs to sell. In fact, the research community and industry are rapidly moving towards more flexible hardware designs. Researchers are designing new abstractions and programming models for congestion control to allow offloading congestion control to hardware. Microsoft Azure has deployed FPGA-based SmartNICs in their datacenters. Rather than trying to make hardware more flexible and complex, can we instead provide simpler and more generic primitives, such as enforcement of packet departure timing? Or is flexibility of FPGAs or network processors necessary for datacenter operators to evolve quickly? If so, what functionalities should go into hardware (or firmware) and what should remain in software?

Network performance isolation: Today's clouds provide limited network isolation and do not offer differentiated network performance products or guarantees in a shared datacenter. There exists a rich body of research on mechanisms to enable rich policies. One of spectrum is decentralized vs centralized? FastPass proposes a centralized packet scheduling which is difficult to scale. ElasticSwitch, Seawall, EyeQ propose completely decentralized solutions which are limited in form of policy it can support. Can we have a completely decentralized solution that can support rich policy? If not, what is the right separation between a centralized controller and distributed algorithms for performance isolation?

References

[1] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. O'Shea. Chatty Tenants and the Cloud Network Sharing Problem. In *NSDI*, 2013.

[2] I. Cho, K. Jang, and D. Han. Credit-Scheduled Delay-Bounded Congestion Control for Datacenters. In *SIGCOMM*, 2017.

[3] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: A GPU-accelerated Software Router. In *SIGCOMM*, 2010.

[4] K. Jang, S. Han, S. Han, S. Moon, and K. Park. SSLShader: Cheap SSL Acceleration with Commodity Processors. In *NSDI*, 2011.

[5] K. Jang, J. Sherry, H. Ballani, and T. Moncaster. Silo: Predictable Message Latency in the Cloud. In *SIGCOMM*, 2015.

[6] J. Kim, K. Jang, K. Lee, S. Ma, J. Shim, and S. Moon. NBA (Network Balancing Act): A High-performance Packet Processing Framework for Heterogeneous Processors. In *EuroSys*, 2015.

[7] C. Lee, C. Park, K. Jang, S. Moon, and D. Han. Accurate Latency-based Congestion Feedback for Datacenters. In *ATC*, 2015.

[8] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker. NetBricks: Taking the V out of NFV. In *OSDI*, 2016.