Masoud Moshref Javadi - Research Statement

I am a systems researcher in the field of computer networking. I propose efficient **algorithms** (approximation, greedy, distributed, scheduling), define **abstractions** (hide complexities, are general, expose tradeoffs) and develop **systems** (fast, distributed, at switches and end-hosts) to solve real-world networking problems. In particular, I am interested in designing and implementing systems for operators to manage networks more efficiently. My research is motivated by the fact that we need to achieve high levels of performance and availability in networks with huge scale, but our solutions for managing networks are primitive. Thus, I build **scalable, timely and accurate network management systems**.

New networks especially datacenter networks have tight requirements in scale, timeliness and accuracy. Datacenter networks have huge scale: They connect hundreds of thousands of servers inside a datacenter using thousands of network switches that must work in coordination. The bandwidth demand on these networks is in the millions of Gbps and is doubling every 12 months. Thousands of users concurrently use the network for applications with diverse requirements. Networks must work accurately: An inaccurate network control can cause congestion and packet losses that damage the performance of prevalent short connections. A challenge is that a device failure is a common event because such networks are usually built upon commodity devices. Networks must react fast: The end-to-end latency requirement is in microseconds. Traffic patterns among servers are unpredictable and change rapidly in tens of milliseconds.

However, the management tools for operators are too limited. Network operators are involved in every aspect of datacenters from design and deployment to operation, from fault detection to application performance optimization to security. However, they have a limited view of network events and limited tools to control the response to the events. Inaccurate measurement tools prevent operators from knowing where bandwidth bottlenecks are and why network packets are delayed. Slow measurement tools increase the delay of detecting a failure or an attack to minutes. Even after detecting the events, the control system that reacts to the events must scale to thousands of switches and hundreds of application requirements. Slow control systems cannot react in milliseconds to be effective for the variable network traffic patterns and to hide the effect of failures and resource congestions from user applications. Moreover, current inaccurate control systems let errors and human mistakes go through and cause inaccessible services for hours even in big companies like Google and Time Warner cable.

Dissertation Work

Software-Defined Networking (SDN), a new trend in networking, distinguishes two layers in networks: the control plane running at a logically centralized server that enforces high-level policies by making routing decisions and the data plane at switches that applies the decisions in forwarding traffic. The centralized controller receives measurements from switches and sends routing decisions to them using a standard protocol (e.g., OpenFlow). SDN allows users to define their own virtual networks on top of the shared physical network and to measure and control them through the applications running on the controller. The operators must decide where and how to run those applications to scale to thousands of users on top of thousands of switches with limited resources without compromising accuracy or timeliness.

Network management involves both measurement and control since controlling a network is not possible without observing the events in the network. In my dissertation, I built both accurate and timely measurement systems and timely and scalable network control systems. I worked on systems that run on hardware switches and end-hosts. On hardware switches, I defined general abstractions for operators that free them from the complexities imposed by hardware resource limitations. On end-hosts, I studied the effect of computer architecture on packet processing and used the result to improve measurement and control systems.

Accurate, yet resource efficient, measurement: Measurement tasks require significant bandwidth, memory and processing resources, and the resources dedicated to these tasks affect the accuracy of the eventual measurement. However, the resources are limited, and datacenters must support a variety of concurrent measurement tasks. Thus, it is important to design a measurement system that can support many tasks and keep all of them accurate on a network with limited resources.

Measurement tasks can be implemented using different primitives with different resource accuracy tradeoffs. I qualitatively and quantitatively explored the tradeoff space of resource usage versus accuracy for three different primitives [6]: (a) Flow counters monitor traffic in hardware switches using expensive and power hungry TCAM (ternary content-addressable memory) and are available in commodity switches. (b) Hash-based counters can express many more measurement task types with higher accuracy and use cheap SRAM memory, but are not available yet. (c) Arbitrary program fragments are more expressive, but they are only possible at end-hosts and have complex resource-accuracy tradeoffs. Focusing on flow counters and hash-based counters, I noticed that although the accuracy of a measurement task is a function of its allocated memory on each switch, this function changes with traffic properties, which forces operators to provision for the worst case.

I developed DREAM [3] for flow counters and SCREAM [2] for hash-based counters to provide operators with the abstraction of guaranteed measurement accuracy that hides resource limits from operators. The insight is to dynamically adjust resources devoted to each measurement task and multiplex TCAM and SRAM entries temporally and spatially among them to support more accurate tasks on limited resources. The key idea is an estimated accuracy feedback from each task that enables iterative allocations. I proposed new algorithms to solve three challenges: (a) Network-wide measurement tasks that can correctly merge measurement results from multiple switches with a variable amount of resources. (b) Online accuracy estimation algorithms for each type of task that probabilistically analyse their output without knowing the ground-truth. (c) A scalable resource allocation algorithm that converges fast and is stable.

I built a prototype of DREAM on OpenFlow switches with three network-wide measurement task types (heavy hitter, hierarchical heavy hitter and change detection), and I showed that DREAM can support 30% more concurrent tasks with up to 80% more accurate measurements than fixed allocation. For SCREAM, I have implemented heavy hitter, hierarchical heavy hitter and super source detection task types. Simulations on real-world traces show that SCREAM can support 2x more tasks with higher accuracy than the state-of-the-art static allocation and the same number of tasks with comparable accuracy as an oracle that is aware of future task resource requirements.

Scalable, timely and accurate measurement: With growing concerns of the cost, management difficulty and expressiveness of hardware network switches, there is a new trend of moving measurement and other network functions to software switches at end-hosts. I implemented a subset of

measurement algorithms in software to re-evaluate their accuracy and performance for traffic traces with different properties [1]. I observed that modern multicore computer architectures have significantly increased their cache efficiency and cache size to the extent that it can fit the working set of many measurement tasks with a usually skewed access pattern. As a result, complex algorithms that trade off memory for CPU and access many memory entries to compress the measurement data structure are harmful to packet processing performance. Then I developed an expressive scalable measurement system on servers, Trumpet [8], that monitors every packet in 10G links with small CPU overhead and reports events in less than 10ms even in the presence of an attack. Trumpet is an event monitoring system in which users define network-wide events, and a centralized controller installs triggers at end-hosts, where triggers run arbitrary codes to test for local conditions that may signal the network-wide events. The controller aggregates these signals and determines if the network-wide event indeed occurred.

Scalable control: In SDN, applying many high-level policies such as access control requires many fine-grained rules at switches, but switches have limited rule capacity. This complicates the operator's job as she needs to worry about the constraints on switches. I leveraged the opportunity that there can be different places, on or off the shortest path of flows, to apply rules if we accept some bandwidth overhead and proposed vCRIB [5,7] to provide operators with the abstraction of a scalable rule storage. vCRIB automatically places rules on hardware switches and end-hosts with enough resources and minimizes the bandwidth overhead. I solved three challenges in its design: 1) Separating overlapping rules may change their semantics, so vCRIB partitions overlapping rules to decouple them. 2) vCRIB must pack partitions on switches considering switch resources. I solved this as a new bin-packing problem by a novel approximation algorithm with a proved bound. I modeled the resource usage of rule processing at end-hosts and generalized the solution to both hardware switches and end-hosts. 3) Traffic patterns change over time. vCRIB minimizes traffic overhead using an online greedy algorithm that adaptively changes the location of partitions in the face of traffic changes and VM migration. I demonstrate that vCRIB can find feasible rule placements with less than 10% traffic overhead when traffic-optimal rule placement is infeasible.

Timely control: Current SDN interface, OpenFlow, requires the centralized controller to be involved actively in any stateful decision even though the event and action happen on the same switch. This adds 10s of ms delay on packet processing and huge computation overhead on the controller, which makes it hard for operators to implement middlebox functionalities in SDN. I proposed a new control primitive in SDN, flow-level state machines, that enables the controller to proactively program switches to run dynamic actions based on local information without involving the controller. I developed FAST [4], the controller and the switch architecture using components already available in commodity switches to support the new primitive. This motivated a collaboration with Tsinghua University on HiPPA [9] project that dynamically chains state machines in hardware and software in order to improve the performance of software-based middleboxes and the flexibility of hardware-based ones.

Future Work

My dissertation work focuses more on measurement, but once a controller can observe network events in a scalable, timely and accurate fashion, it can achieve a lot using that information. In the future, I will focus on how far we can push these attributes in network control and what new services they will make possible. My approaches are (a) Defining new primitives that allow the

right delegation of network control functionalities among end-hosts, hardware switches and the controller. (b) Exploring the tradeoff between scalability, timeliness, accuracy and other aspects such as quality of service (QoS), privacy and power efficiency. (c) Defining the right abstraction based on the tradeoff that hides the complexities inside the network from operators and developing systems that implement the abstraction in an efficient and reliable way.

Ensuring network isolation and fairness: One of the main concerns of businesses for moving services to public clouds is how collocating applications of different businesses on the same set of servers and the same network affects their performance. Different applications require different classes of service from a network. Different classes should not affect each other, and flows in the same class should receive fair service with minimum overhead. Current solutions are not scalable and timely enough: First, the scale of the problem is large as there are millions of flows per second in a datacenter, and a flow may compete for resources with other flows at multiple places around the network. However, CPU cores at servers and traffic shapers (queues) at switches are limited. Second, contention can happen in very small time-scales. My observation is that there are different places in a datacenter that can apply fairness and isolation with different tradeoffs and improve scalability, e.g., hypervisor and NIC at servers and traffic shapers at switches inside the network. I will explore the tradeoff of accuracy, timeliness and resource usage among these options. In addition, a timely measurement system can help timely reaction to flow contentions. I will design a scalable and fast coordinating system that responds to flow interactions in small time-scales and provides the abstraction of an isolated fair network to operators.

Scalable middlebox control: The number of middleboxes (usually stateful devices that inspect and manipulate traffic instead of just forwarding it, e.g., intrusion detection) and their management overhead is comparable to network switches. Today, datacenters use expensive hardware to go through traffic in line rate for tasks such as attack detection and load balancing. This solution is not scalable to the fast increasing traffic inside datacenters where different tenants use each other's services. On the other hand, software middleboxes (network function virtualization) impose overhead on CPUs at servers, network bandwidth and packet latency. To make a scalable solution, part of the computation inside middleboxes can be delegated to network measurement in order to filter their input traffic, reduce their overhead and guide their scale. For example, I want to explore how network measurement can dynamically select the traffic that must go through the middleboxes, what is the right primitive to let middleboxes delegate their computations to measurement elements, and how to attribute the resource usage of a middlebox to traffic properties in order to scale out/up middlebox resources efficiently.

Accurate network control by packet-level network-wide validation & diagnosis: Network operators change network configuration frequently because of device faults/upgrades, new applications and variable traffic patterns. However, there are simply too many places that can induce error: the translation of policies to switch configurations; interaction of different configurations (sometimes at different switches); saving the configurations at network switches; and hardware faults at switches. Such errors may only affect a subset of packets but still damage the performance of applications. Unfortunately, they are not detectable by traditional measurement tools (e.g., SNMP) or current static configuration checkers. The only way to make sure network control is accurate is through validation. I plan to develop a system to automatically translate control policies and network-wide invariants to the right packet-level measurement tasks and validate them. In addition, I will explore the solutions to diagnose the root-cause of a validation failure.

Controlling heterogeneous networks: In the near future, new networking technologies such as Intelligent NICs with TCAM, optical switches with fast configuration capabilities and wireless communication among racks will be deployed in production datacenters and will co-exist with current technologies. However, still there is no complete control system to help operators decide when and how to use such technologies. For example, I want to explore what types of applications benefit from intelligent NICs and how to dynamically push network functionalities to them without involving the operator. Moreover, I am interested in detecting traffic demands online and automatically scheduling them on Ethernet, optical networks and wireless networks to improve quality of service (QoS).

The marriage of measurement primitives: The work on measurement is not finished. Although, in my previous work, I explored the tradeoffs for different measurement primitives, I have never examined how to combine different primitives to leverage their strength points to cover each other weaknesses. For example, while sketches may find heavy hitters fast, their output always has some random errors. In contrast, flow-counters always provide exact values but must iterate to find a heavy hitter. In applications such as accounting where we need exact counters for heavy users, these two primitives can collaborate to detect heavy hitters fast with exact numbers. Similarly, expressive code-fragments at end-hosts can guide expensive measurements inside the network. There are other interesting cases for combining sampling with flow-counters and so on.

References

1. Omid Alipourfard, **Masoud Moshref**, Minlan Yu, "Re-evaluating Measurement Algorithms in Software", HotNets, Philadelphia, PA, 2015
2. **Masoud Moshref**, Minlan Yu, Ramesh Govindan, Amin Vahdat, "SCREAM: Sketch Resource Allocation for Software-defined Measurement", CoNEXT, Heidelberg, Germany, 2015
3. **Masoud Moshref**, Minlan Yu, Ramesh Govindan, Amin Vahdat, "DREAM: Dynamic Resource Allocation for Software-defined Measurement", SIGCOMM, Chicago, 2014
4. **Masoud Moshref**, Apoorv Bhargava, Adhip Gupta, Minlan Yu, Ramesh Govindan, "Flow-level State Transition as a New Switch Primitive for SDN", HotSDN, Chicago, 2014
5. **Masoud Moshref**, Minlan Yu, Abhishek Sharma, Ramesh Govindan, "Scalable Rule Management for Data Centers", NSDI, Lombard, 2013
6. **Masoud Moshref**, Minlan Yu, Ramesh Govindan, "Resource/Accuracy Tradeoffs in Software-Defined Measurement", HotSDN, Hong Kong, 2013
7. **Masoud Moshref**, Minlan Yu, Abhishek Sharma, Ramesh Govindan, "vCRIB: Virtualized Rule Management in the Cloud", HotCloud, Boston, 2012
8. **Masoud Moshref**, Minlan Yu, Ramesh Govindan, Amin Vahdat, "Trumpet: Timely and Precise Triggers in Data Centers", submitted to NSDI, 2016
9. Shuyong Zhu, Jun Bi, Chen Sun, Haoxian Chen, Zhilong Zheng,Hongxin Hu, Minlan Yu, **Masoud Moshref**, Chenghui Wu, Cheng Zhang, "HiPPA: a High-Performance and Programmable Architecture for Network Function Virtualization", submitted to NSDI, 2016