Software Defined Networks and Gossip Protocols

Robert Soulé
University of Lugano

Ken Birman
Cornell University

Nate Foster
Cornell University

The performance of data-center applications are critically dependent on the underlying network. However, given the complexities associated with management, networks today typically provide little more than best-effort packet delivery between hosts. The emergence of software-defined networking (SDN) has created an opportunity to build more dynamic networks that can be tailored precisely to the needs of applications. Unfortunately, existing solutions for monitoring within SDNs suffer from several short-comings. Either they are inaccurate (due to eventual consistency of architecture [5]), imprecise (due to limitations of current hardware [5]), or too costly to be practical at scale (due to reliance on switch forwarding rules and centralization [1]). We argue that *gossip protocols* offer an ideal alternative for SDN monitoring, due to their scalability and resiliency.

Merlin. In an SDN-enabled network, a *control* program reacts to network events, and updates forwarding rules on switches to manage packets. Building on this interface, our work on Merlin [3] is novel among network programming languages in that it determines allocations of limited network-wide resources such as bandwidth and paths.

We have used Merlin to improve the latency of Hadoop jobs running in the presence of UDP background traffic, or prioritize classes of traffic used for state-machine replication in fault-tolerant services [4]. These experiments demonstrate that an SDN framework, with the correct information as input, can provide automated network management customized to the needs of resident distributed applications.

While the Merlin compiler generates static network configurations, Merlin uses a small, runtime component to allow for dynamic adaptation. Merlin's language-based approach allows this adaptation to happen safely, by providing policy language constructs that can be automatically verified. Implicit in the design of this runtime component (and SDN networks in general) is the notion that network events are generated in response to the situational status culled from a wide range of sources, including: (i) *network state* (e.g., packet and drop rates, error counters); (ii) *application state* (e.g., job priorities, security credentials); (iii) *user state* (e.g., user preferences for a particular network); and (iv) *hardware state* (e.g., device names, hardware types, physical location).

Much of this information must be created and updated dynamically. However, existing SDN frameworks have largely ignored the crucial *monitoring* component that aggregates network and application state, and sends the events to the controller. A complete system would have a closed loop, continuously monitoring applications and the network, building situational status, then adjusting SDN policies to optimize the use of resources.

MiCA. Gossip protocols are an ideal choice for implementing a wide range monitoring tasks. With a gossip protocol, each node exchanges information with a randomly selected peer at periodic intervals. Because it is based on periodic peer-to-peer communication, gossip's network load tends to be well-behaved, scaling linearly with system size and not prone to reactive feedback. Moreover, because peers are selected randomly, no single node is indispensable, so tools built on gossip are extremely tolerant to disruptions and able to rapidly recover from failures.

Although individual gossip protocols are typically very simple, composing multiple protocols can lead to complex interactions with unpredictable behavior. We designed the MiCA [2] framework to address this problem. MiCA allows programmers to describe gossip protocols with a small, well-defined interface, and compose the protocols with a rich collection of operators to create sophisticated protocols in a modular style. MiCA ensures that the composed protocols maintain strong (albeit probabilistic) robustness and convergence guarantees. In our evaluation of MiCA, we have built monitoring tasks that maintain a predictable performance, even when hundreds of separate instances are deployed on the same machines.

Closing the Loop. To accommodate the ever-growing demands of cloud and data center application, networks will need to become more flexible and dynamic. As networks continue to grow in complexity, it will become increasingly difficult for network operators to provide this flexibility without the support of proper tools and infrastructure.

Merlin and MiCA, deployed together, provide both the *control* and *monitoring* components necessary to automatically adapt the network to the needs of the applications. Because both systems use a language-based approach, they have rigorous semantics that can be formally defined. Moreover, they provide predictable operational behavior. Together, they allow for the rigorous expression of algorithms for monitoring or managing SDN networks.

Acknowledgments. This work was supported, in part, by a grant from the DARPA MRC program.

References

[1] L. Jose, M. Yu, and J. Rexford. Online measurement of large traffic aggregates on commodity switches. In *HotICE*, 2011.

[2] L. Princehouse, R. Chenchu, Z. Jiang, K. Birman, N. Foster, and R. Soulé. MiCA: A Compositional Architecture for Gossip Protocols. In *ECOOP*, pages 644–669, July 2014.

[3] R. Soulé, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster. Managing the Network with Merlin. In *HotNets*, Nov. 2013.

[4] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, and N. Foster. Merlin: A Language for Provisioning Network Resources. In *CoNext*, Dec. 2014. To appear.

[5] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with opensketch. In *NSDI*, pages 29–42, 2013.