Datacenter Fault Discovery with SDNs

October 2014 Status Report

Ken Yocum and Alex C. Snoeren

1 Personnel

To date, the project has provided funding for one Ph.D. student, Kevin Webb. Kevin graduated in the summer of 2013 and is now an Assistant Professor of Computer Science at Swarthmore College in Pennsylvania. Two additional Ph.D. students, Arjun Roy and Danny Huang, have contributed to the project while being funded from other sources. The work described in this status update has been carried out by Arjun Roy.

2 Overview

We have developed a model for writing, deploying, and managing "applications" that control the underlying network hardware used by tenants multiplexed across a shared cloud infrastructure [2]. Sometimes referred to as network functions virtualization (NFV), the idea is to allow tenants to selectively deploy different applications to address their particular needs. Indeed, Kevin Webb presented an overview of our architecture at the ANCS conference earlier this month [3]. Over the past year, we have turned our attention to providing one particularly important function: fault detection.

In many datacenters, a tenant's VMs are unlikely to be co-located in a single rack or even set of racks. Hence, a tenant's VMs are likely to be connected over over multiple physical paths to increase the aggregate bandwidth possible between any pair. However, multiple physical paths means more switches, interfaces and cabling, which brings with it a higher likelihood that a tenant's virtual network will experience some sort of packet forwarding errors. Particularly insidious are intermittent or partial failures, which manifest themselves only for a subset of tenants, or, worse, only sporadically. For example, the authors of NetPilot describes a case where a 1% packet corruption rate in a partially degraded link caused a $4.5\times$ increase in 99th percentile latency for applications running under the effected switch.

Prior work in the fields of fault inference across computer networks [1], and end-host driven rerouting (e.g., Routing Deflections, Path Splicing, MPTCP) can help to some extent. Fault inference can be useful when applied in a wide-area context; given the knowledge of the path taken by a flow and whether or not the flow is experiencing poor performance, we can determine what points in the network are commonly traversed by poorly performing flows, and flag them for further investigation. End-host influenced rerouting can potentially help a given flow 'route around damage' and increase its performance.

2.1 SDN-assisted Localization

Unlike the traditional routers employed in older wide-area networks, switches in today's datacenters provide APIs that can help program the forwarding behavior of a switch as well as provide insight into traffic characteristics. In addition, end hosts in the network can be controlled entirely by the datacenter operator, unlike in the wide-area case. We can thus leverage what switches know (specifically, what traffic is transiting the switch) in combination with what end hosts can tell them (whether traffic is performing well or poorly at the application level) to collaboratively determine the location of performance-degrading faults in the network.

Our approach is to develop a system where end hosts communicate their satisfaction with the network by marking each flow with whether or not the end host is 'happy' with the performance that the flow is receiving. For now, this is a coarse signal that can be triggered by latency, bandwidth, reordering, or other concerns. Switches periodically examine traffic transiting through them, and can run a variety of algorithms to flag links that might be faulty. In addition, they can reroute traffic around links that are suspected to be faulty; which may indirectly benefit even short flows that wouldn't benefit from end-host triggered rerouting.

2.2 Recent Progress

Our initial explorations employed simulating application traffic in a flow-level simulator and inducing link errors to see if faults could be detected and localized by flagging flows with significantly lower-than-average performance and correlating large amounts of underperforming flows with (presumably failing) links that they traverse through the network. We started with a hit-coverage algorithm inspired by SCORE [1] that takes a network-wide look at flow performance and attempts to explain which links are most likely to have caused any observed poor performance, and flag them as likely fault locations.

Having had some initial success with determining fault locations for a variety of synthetic traffic patterns, we are now attempting to use application-provided performance information to actively avoid faulty parts of the network—in addition to just discovering their location. This is useful for a variety of cases. First, there might be cases where pinpointing the exact location of a fault might be difficult, but determining links that are not impacted by the fault might be easy; hence we might ameliorate the impact of a hard-to-find fault. Second, even when a fault is found, it can take time to determine and implement the best fix, during which time we'd like to be able to avoid taking a performance hit due to the fault.

During the period since the last review, we have been running a variety of real applications, both on physical hardware and in Mininet-style network emulation of Openflow SDNs, to determine the answers to the following questions:

1. Does real application traffic get impacted by network faults in such a way that end hosts can note a marked decrease in performance? (It appears to.)
2. Does avoiding the link causing poor performance—at the cost of potentially overloading non-failing links—result in better performance metrics, as measured by the end-host application? (It appears to.)
3. Can information provided by the host regarding application performance enable switches to make routing decisions that optimize performance, whether or not the switch in question has a faulty link? (Initial results are promising.)

We are currently developing a proof-of-concept network controller that communicates with end hosts regarding application performance, and adjusts ECMP weights based on this data to optimize performance metrics as seen by the end host.

References

[1] KOMPELLA, R. R., YATES, J., GREENBERG, A., AND SNOEREN, A. C. IP fault localization via risk modeling. In *Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (Boston, MA, May 2005), pp. 57–70.

[2] WEBB, K., SNOEREN, A. C., AND YOCUM, K. Topology switching for data center networks. In *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (Boston, MA, Mar. 2011).

[3] WEBB, K. C., ROY, A., YOCUM, K., AND SNOEREN, A. C. Blender: Upgrading tenant-based data center networking. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)* (Marina del Rey, CA, Oct. 2014).