Active scheduling for fine-grained traffic engineering in the datacenter

Aditya Srivastava and Vishal Satish

Background

Active networking was introduced by Tennenhouse and Wetherall over 25 years ago as a fundamental shift in the way we think about networks [1]. In the active networking paradigm, packets carry code to be executed at active switches in the network; this is in contrast to how networks have primarily been organized so far, with "dumb" switches/routers enacting some static network protocol embedded in a packet header. There are a number of factors which motivated active networking in 1996, and a number of these are still relevant today [2]:

1. Introduce programmability into the network to accelerate the pace of innovation and lower the barrier to entry for research
2. Virtualize the network to raise the level of abstraction exposed to application developers
3. Offer a common architectural framework to unify control and management of middleboxes (think NFV now)

While there has been a significant amount of research into enabling active networking, the architecture never saw widespread deployment due to the lack of a killer application and because of the difficulties inherent in deployment [2]. In addition, active networking has been criticized for its lack of performance and security (arbitrary code execution in the network). Now, with the development and proliferation of programmable switches such as the Barefoot Tofino there is an avenue to addressing these concerns; these switches are able to operate at rates sustained by currently deployed commodity switches and offer a fully programmable data plane [3].

We motivate this project with a few observations. In the datacenter context, important network phenomena occur at short time scales such as queuing delays, incast problems, and non-deterministic failures [4]. For example, Zhang et al. characterized microbursts in the Facebook production datacenter network, periods of high utilization lasting less than 1ms. These microbursts have a significant effect on network performance, accounting for a majority of the congestion observed in datacenter networks, a fact which has important implications concerning the design of higher level protocols such as load balancing and congestion control [4]. These events are hard to diagnose, debug, or react to as the control path is slower than the event duration and switches carry average statistics which mask microbursts. As such, a solution that addresses these problems will need to be able to react to microbursts at network time scales.

Proposal

Programmable switches offer a paradigm on which we can start to build a solution to combat problems caused by microbursts as they have the ability to introspect packets and make forwarding decisions at network time scales. In the vein of the active networking vision, we propose the development of a minimal instruction set which is supported in programmable switches. Packets will optionally carry instructions from the supported instruction set in their headers, allowing programmable switches to conditionally route packets by interpreting packet headers. This will enable not only reaction to microbursts (the killer app) but programmable usage and tuning of the network by higher level applications.

For example, a packet could carry an instruction indicating that it should be deflected to a random output port if the queue length at the switch exceeds some value or a packet could be returned to the sender if the switch contains a certain forwarding entry. This instruction set will thus enable better congestion control, network introspection, load balancing, and debugging to name a few use cases.

We will develop this instruction set using P4 so that it is compatible with existing programmable switch technologies with the intent that software using this instruction set will be implemented at the hypervisor and will be manageable by network operators [5].

Timeline

March 1 - 14

- Build out an end to end proof of concept to show a packet header format encoding an instruction and a switch with supports interpretation of this instruction
  - e.g. return packet to sender if queue length $>$ X
- This PoC will involve learning P4, defining and implementing our headers, and running simulations to verify correctness

March 15 - 28

- Decide on a higher level application we want to implement using such an instruction set
  - e.g. congestion control at microburst time scales, load balancing, debugging
- Add instructions we need to realize this higher level goal

March 29 - April 11

- Tune and evaluate our implementation of the higher level application compared to existing implementations and hopefully show that our solution provides benefits

April 12 - 25

- Build out instructions to support more higher level applications (maybe a simple deflection based congestion control, a diagnostic tool for network operators, a monitoring tool to observe network behavior over time etc.)

References

1. D. L. Tennenhouse and D. J. Wetherall. Towards an Active Network Architecture. ACM SIGCOMM Computer Communications Review, 26(2):5–18, Apr. 1996.
2. Feamster, Nick, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks." ACM SIGCOMM Computer Communication Review 44.2 (2014): 87-98.
3. Bosshart, Pat, et al. "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN." ACM SIGCOMM Computer Communication Review 43.4 (2013): 99-110.
4. Zhang, Qiao, et al. "High-resolution measurement of data center microbursts." Proceedings of the 2017 Internet Measurement Conference. 2017.
5. Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44.3 (2014): 87-95.
6. Alizadeh, Mohammad, et al. “Data center tcp (dctcp).” Proceedings of the ACM SIGCOMM 2010 Conference. 2010.