T9: SDN and Flow Management: DevoFlow

Critique

Lee, Tae-Ho

1. Problems due to switch HW may be temporary. HW will evolve over time. Section 3.3 tries to defend against this point, but none of the argument is based on facts—they are authors’ opinions.

2. In dataCenters, it may be possible to categorise flows into the three categories. Since the authors argue that the solution can be generic, how can the categorisation be done in advance?

3. The paper uses vague languages too often (e.g., expected, guess, I believe, etc). In addition no evidence is provided for Section 4.3

van Gelder, Jasper

1. They argue not enough flows can be handled by current openflow switches and that basically the CPU is the one to blame. Which I think is an argument that can easily be taken away by putting a proper CPU in a switch I also think we already have seen switches which are capable of handling more flows. They further kill their own argument partially in 3.3.

2. I am wondering what happens to their elephant detection mechanism if you have for example a streaming API that posts news feeds. It will still qualify as elephant load after x bytes went over the line simply because the connection had been open for a certain amount of time. They could easily improve this by periodically clearing the statistics or using a different measure though.

3. One other question I had when reading this paper how often do flows change? cant you effort to be a little slow on the collection of data and path calculations because generally a group of flows come from the same source towards a certain destination. You probably backup always to the same server(s), webtraffic to your server usually come from outside of your own network.

Chothia, Zaheer

1. One of the use cases presented is flow scheduling. The paper rightly motivates that collecting the statistics required for Hedera is overly expensive, so they only route
elephant flows. The bin packing approach, however, doesn't generate routes but only rates to throttle each flow. Even this has some drawbacks: the rate assigned is based on the full port capacity, however DevoFlow only observes flows above a given threshold. Further, the algorithm assigns an equal share to all flows on a given port and makes greedy selections at each port rather than scheduling flows end-to-end.

2. Evaluation: still unclear how big the gains are for a realistic setting. Whilst there is a sizeable improvement in aggregate throughput with the shuffle use case, for the real-world workload there are no or only minimal gains. The explanation given is that ECMP is already close to optimal and the reverse-engineered model could be improved. Despite that, the benefit in reducing overhead (no. of packets to controller and flow table entries) and dramatic and prove the initial problem motivation.

3. Only half the paper is spent discussing and evaluating DevoFlow itself and it could be argued this part has insufficient detail. Despite this, the first half of the paper is spent justifying issues with OpenFlow and its limitations. I feel this alone is a valuable contribution, plus they build a watertight argument and are very thorough, for instance pointing out issues with current switch hardware but also fundamental issues inherent to OpenFlow.

== References ==
* Accompanying slides from SIGCOMM’11
* Network Traffic Characteristics of Data Centers in the Wild [IMC’10]

Birkner, Rüdiger

1. By introducing DevoFlow additional load is shifted to the switches. This shift does away with OpenFlow’s goal of having simple and future-proof devices. The paper mentions that they would like to maintain this property, but that it will definitely add some complexity of the switch. However, in the remaining part of the paper this fact is not mentioned anymore and there is no analysis of how the switches are affected by their ideas.

2. DevoFlow addresses issues of OpenFlow regarding path setup and statistics. It is desirable to reduce the load of the controller by doing certain tasks locally. But the fixes for these issues need specific modifications of the switch hardware. However it would be better to try to incorporate the ideas of DevoFlow into Protocol-Oblivious/Independent Forwarding to produce hardware that is even more flexible. The protocol-oblivious forwarding ideas came up in 2013 while this paper has been presented in 2011.
3. After an elephant flow has been identified, it is most likely assigned a new more optimal path. Moving a flow from one path to another might incur some problems as discussed in Microsoft’s SWAN paper due to timing problems or transient congestion

Defense

Miladinovic, Djordje

DevoFlow is a networking platform which targets many of the scalability issues we talked about in the previous meetings. Addressing these issues is a crucial step towards deploying SDN and using it in the settings where high performance is a must. Some of the most prominent ones are:

1. DevoFlow addresses the SDN’s intrinsic scalability problem of having an overloaded controller to switches paths. This is a serious issue of OpenFlow and can damage the performance of network. DevoFlow tries to solve these overhead by identifying the flows which need controller’s monitoring.

2. DevoFlow also pays attention and emphasizes importance of having scalable statistics mechanisms that do not cause high overheads, which could affect switch-network paths, as well as controller’s CPU. Again the solution comes from removing the burden from the controller. This is done by moving a part of networking intelligence from the controller to the switch. Many mechanisms presented are for efficient statistics collection.

3. DevoFlow solves another scalability issue: Limited number of forwarding rules in switches. For this DevoFlow uses wildcard rules of the switches, coupled with DevoFlow mechanisms that control the use of these.

All these scalability solutions are accompanied with a design that aims cost-effective hardware implementation. DevoFlow also tries to maintain all the advantages we have when using OpenFlow, but makes certain tradeoffs to adapt OpenFlow for high performance networking. DevoFlow does not change OpenFlow design radically.

Pappas, Christos

DevoFlow is a slightly extended version of OpenFlow with very simple augmentations. As mentioned in the paper, there is no breakthrough mechanism proposed, just simple extensions that allow operators to game the tradeoffs between performance and control/visibility. The main benefits of the proposal are linked with this simplicity.

1. This paper reads mostly as network management practices/solutions for high-performance OpenFlow networks. The ideas presented are linked with the
OpenFlow realization of SDN. The next versions of OpenFlow might give operators other/similar tools to realize these ideas in different ways.

2. DevoFlow is an extension to OpenFlow, and hence it is based on a well-established and commonly understood technology/protocol. Network operators wouldn't have to change their practices significantly, nor to invest in very different software/hardware to enjoy the benefits of DevoFlow.

3. Simplicity leads to a hardware implementation that is based on existing elements and hence makes a hardware solution more realistic: the extensions proposed are technically feasible in logical cost boundaries.

Shinde, Pravin

1. They aim to reduce the load on controller by using wildcard-filters to group multiple flows together, and new actions like "clone" to retain ability to collect individual flow statistics.
2. They also want to push limited decision making capabilities in switches by having load-balancing actions and alternate routes which can further reduce the load on controllers.
3. On other hand, it also makes it little more difficult to trace the exact path of the flow as load-balancing and alternate routes introduce some degree of non-transparency from controllers, based on current network situation.
4. Davoflow needs changes in openflow actions and more capability in switches which fundamentally breaks the openflow model. I am not fully convinced that these additional actions are enough and future-proof that we will not have to change the spec and hardware again.

Schmid, Stefan

The authors recognize that there is a lot of overhead in early OpenFlow systems. They identify bottlenecks in flow setup and flow statistics. To make OpenFlow applicable for data center environments they propose DevoFlow which implements the following protocol changes:

1. Rule cloning: To prevent contacting the controller to setup every new flow (and introducing delay and control plane traffic), wildcard rules are installed in the switch. To still be able to differentiate flows, the wildcard rules can be locally cloned, creating a new flow entry with the specific settings of a new flow. This enables local flow setup (without contacting the controller) but still makes it possible to collect individual statistics for each flow.

2. Local actions: A switch should still have a small set of local routing actions it can invoke without contacting the controller. Two presented examples are multipath support where the switch can select (randomly, round robin, etc) a port to forward data, and rapid re-routing where the switch has fallback ports in case a port goes down.
3. Efficient statistics: The authors also propose methods to improve statistics collection efficiency to reduce data traffic in the control path:

4. Sampling: randomly chosen packet headers are forwarded to a monitor node.
5. Triggers: reports a generated and sent when a trigger condition is met. Reports can also be aggregated to pack several into one packet.

6. Approximate counters: Maintain a view on the statistics for largest flows in a memory efficient way (Streaming algorithms).

Overall the paper is well written and the proposed methods are well founded and explained.

Yu, Xinyuan

1. OpenFlow has some shortcomings.
   a. OpenFlow is not perfect for all settings, because controller has to be on the critical path, it is difficult for the OpenFlow network to be scalable.
   b. With OpenFlow implemented, the slow control-data path adds unacceptable latency.
   c. The switched are required to have large flow table. In this way, instead of full control, the controller maintains control over only significant flows which is more effective.

2. DEVOFLOW has a novel statistics collection method which is much more efficient than OVERFLOW. It uses sampling as alternative of push-based or pull-based, which uses far less load. Besides, instead of using scheduler to collect statistics several seconds, it uses triggers and reports, which can better meet the demand of high-performance network.

3. DEVOFLOW can simplify the design of high-performance OpenFlow switches. By simplifying the design of high-performance OpenFlow switches and enabling scalable management, DEVOFLOW has a wide use case, for example mlicasts, energy-aware routing, etc.