DevoFlow: Scaling Flow Management for High-Performance Networks

Ashutosh Mittal (E-mail: amittal@kth.se)

I. Paper summary

THIS paper introduces *DevoFlow*, a modification of the OpenFlow model in which attempt is to maintain a useful amount of visibility of the central controller without imposing unnecessary costs on the network. DevoFlow is designed to allow aggressive use of wild-carded OpenFlow rules, thus reducing the number of switch-controller interactions and the number of TCAM entries. It provides new mechanisms to detect QoS-significant flows efficiently, by waiting until they actually become significant. It also introduces new mechanisms to allow switches to make local routing decisions when these do not actually require per-flow vetting by the controller. It divides microflows into three broad categories: security-sensitive flows, which must be handled centrally to maintain security properties; significant flows, which should be handled centrally to maintain global QoS and congestion properties; and normal flows, whose setup can be devolved to individual switches. Leveraging these dynamically allocated categories, it tries to preserves the benefits of OpenFlow, while devolving some decisions to the switches, thereby reducing these overheads.

II. Significant Contributions

- This paper is an attempt towards gradual integration of SDN in production networks. SDN networks up till this paper were mostly research oriented and due to the latency, they were largely not scalable. However this paper came up with the idea of centrally managing only some of the 'critical' flows, thus getting overall performance acceptable to be implemented in production networks.
- The then existing statistics mechanisms had both relatively high overhead, and in particular they did not allow the controller to request statistics only for the small fraction of elephant flows that actually matter for performance. However, DevoFlow suggests using sampling instead of push or pull based collection. Alternatively, it suggests extending OpenFlow rules to include threshold-based triggers on counters.
- Invoking the OpenFlow controller on every flow setup causes latency as well as puts too much load on the control plane. This paper suggests using Rule cloning to avoid this overhead as well optimise usage of TCAM memory.

III. Unresolved Issues

- The suggestions about devolving control to switches needs to rigorously tested. Hardware implementation is required to have an actual assessment of the efficiency of the various suggestions and novel ideas suggested in the paper.
- Hardware implementation of many of the suggestions is not possible. Paper suggests rule cloning and multipath support, these require some redesign of existing data planes and its not even clear if its possible. Otherwise, rule cloning would require a control plane operation once per flow which would severely affect performance.
- The paper revolves around finding 'elephant flows' or the critical flows and managing them through the central command. However, the mechanisms to find these flows are not conclusively specified or tested by the authors. They have just put out a few ways to do it, instead of presenting a proper algorithm for it.