Paper Review October 31th

Jiaqi Yang (jxy530)

SNAP: Stateful Network-Wide Abstractions for Packet Processing

---

What problem is the paper solving and why is it important?

The problem that this paper addresses and tries to solve is that even though emerging hardware and software switches offer much more sophisticated support for persistent state in the data plane, comparing to the early simple match-action paradigm OpenFlow 1.0 switches, managing stateful, distributed systems efficiently and correctly is still one of the most challenging programming problems in the field of Software-defined networking (SDN). Due to the emergences of new proposals for switch interfaces that expose persistent state on the data plane like those in P4 and Open vSwitch, the need of new languages and abstractions to help us manage the complexity and optimize resource utilization effectively is a significant problem that this paper solves by proposing SNAP.

What is the main idea of the paper?

The main insight of this paper is that by proposing SNAP, a new language that allows programmers to mix primitive stateful operations with pure packet processing, the authors could address and solve the problem described above. Their main ideas and contributions include 1) A stateful and compositional SDN programming language (SNAP) with persistent global arrays, a one-big-switch (OBS) programming model, and network transaction, 2) Design of algorithms for compiling SNAP programs into low-level switch mechanisms , and 3) Implementation and evaluation of SNAP and its compiler's performance and scalability using about 20 applications.

How does the paper differ from previous work?

Although there are handful of researches and developments in regard to new hardware and software switches, aiming to provide more sophisticated support for persistent state in the data plane, it is still extremely challenging and difficult to manage stateful, distributed systems. Different from most of its previous work, this paper tries to solve this problem in a much more high-level approach: by proposing SNAP, a high-level language that contains stateful programs and is written in terms of an abstract network topology comprising a one-big-switch (OBS) and its compiler that automatically converts the program language into lower-level output, retains the states, and distributes the states and outputs into the corresponding switches. In this way, the programmers will have a much easier time to manage a stateful program without thinking about how to distribute and manage individual switches.

Are there any flaws in the paper? How would you improve the paper or build on it in future work?

SNAP's current prototype does not implement any particular fault tolerance mechanism in case a switch holding a state variable fails: in this case, the state on the failed switch will be lost. This should be considered as a major flaw in the paper since the problem will directly influence the reliability and validity as a language, designed to address the problem of stateful programming. A

possible solution to this could be the application of common fault tolerance techniques in the software or hardware level. By applying the technique to the switches with state, it will efficiently avoid state loss in case of failure.