Background

Recent advances in programmable switches (e.g., Barefoot Tofino, Intel Flexpipe) and the development of the P4 programming language [1] has drawn a lot of interest of network programmers. These advances give more control to programmers and allow them to, e.g., reduce network complexity, write new network applications, and perform proactive network monitoring. This resulted in proposals like in-band network telemetry, in-network aggregation and coordination algorithms.

Figure 1: PISA pipeline with programmable packet scheduler

More specifically, the P4 language allows the programmer to describe how a switch treats a packet, resulting in a concrete forwarding behaviour for packets. A P4 program, once written, is compiled to generate a configuration (or an instruction set) to run on programmable switches (like a Java program is compiled to run on CPU(s) or an OpenCL program is compiled to run on GPU(s), etc). The programmable switch is based on the protocol-independent switch architecture (PISA), as shown in Fig. 1, which consists of a programmable parser and a programmable pipeline of match-action stages.

Conventionally, switches only implement a few fixed packet scheduling algorithms, which limits a network operator to only tweaking their parameters, and there is no way to express new scheduling algorithms after the switch has been built. However, the capabilities offered by programmable switches have triggered an opportunity (as well as interest e.g., PIFO abstraction [2]) for programmable packet scheduling. Packet scheduling determines the order and time to schedule packets for transmissions. This is important in different environments. For example, in a data center running so-called data-parallel applications, it can be more important to complete multiple flows of an application at the same time rather than greedily try to maximize mere throughput.

We believe that a service curve [3] is the right abstraction for building a programmable packet scheduler for diverse objectives of data-parallel applications in data-center networks. It can express many existing scheduling algorithms, for instance, weighted fair queueing, hierarchical fair service curve, and traffic shaping.

Thesis Goals

The goal of this thesis is to implement the service curve abstraction in the P4 language, compile it to generate configuration for different scheduling algorithms, and run on a programmable switch. This implementation then needs to be extensively tested.

Milestones

- Familiarizing with the P4 programming language.
- Design and implement service curve abstraction in P4 language.
- Realize mechanism to express one (or more) scheduling algorithms e.g., weighted fair queueing, hierarchical fair service curve or traffic shaping.

Required knowledge (or willing to learn)

- Understanding of networking concepts
- Learning P4 programming language
- Good software development skills in C/C++

References


Contact: Asif Hasnain, asif.hasnain@upb.de, Phone: 05251 / 60 1757, Room: O3.164