Event-Driven Packet Processing

Stephen Ibanez[*], Gordon Brebner[†], Gianni Antichi[#], and Nick McKeown[*]

[*]Stanford University, [†]Xilinx Labs, [#]Queen Mary University of London

ABSTRACT

The current P4 programming model allows programmers to express packet processing on a synchronous packet-by-packet basis, motivated by the goal of line-rate processing in feed-forward pipelines. But some important data-plane operations do not naturally fit into this programming model. Sometimes we want to perform periodic tasks, or update the same state variables multiple times, or base a decision on state sitting at a different pipeline stage. While a P4-programmable ASIC might contain special features to handles these tasks, such as packet generators and recirculation paths, there is currently no clean and consistent way to expose them to P4 programmers. We therefore propose a common, general way to express *event processing* in a generic P4 pipeline, beyond just packet arrival and departure events. We believe that this more general notion of event processing can be supported without sacrificing line-rate packet processing and we have developed a prototype event-driven architecture on the NetFPGA SUME platform to serve as an initial proof of concept.

1. EVENT-DRIVEN PACKET PROCESSING

Many data-plane programmers tend to think of switches and NICs as processing one packet followed by another in synchronous succession. The P4 language has embraced this line of thinking and exposes a programming model that is driven by packet arrival and departure events. This programming model is convenient because it can be efficiently mapped to a feed-forward pipeline architecture that deterministically processes packets at line rate. The P4 programming model has become widely accepted as the proper way to program the data-plane. However, it has lead many data-plane developers to believe that algorithmic state can only be updated in reaction to packet arrival or departure events. This does not necessarily reflect what the underlying hardware is capable of achieving, it is simply an artifact of the current programming model.

We observe that many data-plane applications do not efficiently map to this purely packet-driven programming model. The reason is because these data-plane applications must access and update algorithmic state in response to events other than packet arrivals and departures. In particular, we would highlight two types of data-plane applications that exhibit this behavior. There are a number of important applications that must periodically perform a task, for example, Hula that periodically generates packets with link utilization information. Today, these periodic tasks are accomplished either via the control plane or by manually configuring a packet generator. Neither of these approaches are convenient for P4 programmers to express. Another important class of data-plane applications, e.g., Snappy and NDP, are those that update algorithmic state when packets are enqueued, dequeued, and/or dropped from the buffer, for instance, to derive congestion signals.

Event Type	Description
Ingress Packet	Packet arrival
Egress Packet	Packet departure
Packet Transmission	Packet finished transmission
Recirculated Packet	Packet sent back to ingress
Buffer Enqueue	Packet enqueued in buffer
Buffer Dequeue	Packet dequeued from buffer
Buffer Overflow	Packet dropped at buffer
Buffer Underflow	Buffer becomes empty
Timer expiration	Configurable timer expires
Control Plane triggered	Control plane invoked
Link status change	Link goes down / comes up
State Condition met	User-defined condition

Table 1: Set of useful data-plane events to support in an event-driven packet processing architecture.

Many of the applications that utilize congestion signals are challenging to implement on today's P4 target devices and often use some form of recirculation. In order to facilitate the deployment of such applications, we propose to generalize the notion of packet arrival and departure events to the more general class of data-plane events. Table 1 describes a set of events that we have identified as being generally useful for implementing a wide range of data-plane algorithms. Each particular P4 target architecture would define precisely the set of events that it supports.

We believe that it is feasible to support our more general notion of event processing without sacrificing deterministic line-rate packet processing. In order to demonstrate the practicality of deploying an event-driven architecture in hardware, we developed the SUME Event Switch using the P4→NetFPGA toolchain and used this architecture to implement and evaluate a version of the FRED AQM policy.