Event-Driven Packet Processing

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

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

ABSTRACT

Currently, the processing expressed in a P4 program is triggered only upon the occurrence of a very small collection of events, i.e., packet arrivals, packet departures, and packet recirculation. This is a big limitation, largely imposed by today's hardware for programmable data-plane architectures. However, network applications are inherently event-driven in many ways: their behavior depends on circumstances that might span from buffer occupancy changes to link status changes, as well as packet-related events. In this talk, we identify a set of useful data-plane events and propose a new hardware architecture — which has been prototyped on the NetFPGA SUME platform — to support them. This allow us to demonstrate how we can achieve an unprecedented level of event-driven programmability while processing packets at line rate.

1. EVENT-DRIVEN PACKET PROCESSING

Programmable data planes have already proved to be a very powerful tool for applications such as network monitoring, congestion control, load balancing, and fast reroute. However, modern programmable data-plane architectures are limited, in particular in two key areas: deriving congestion signals and performing periodic tasks.

Common congestion signals such as queue size, queue service rate, and queueing delay are often readily available on most data-plane target devices. However, they provide no means by which to compute other congestion signals such as packet loss volume, rate of change of the queue size, timestamp of buffer overflow/underflow events, per-active-flow buffer occupancy, and others that have yet to be conceived. Ideally, a programmable data-plane architecture would enable its programs to derive these congestion signals in order to facilitate the deployment of applications that could benefit from them, such as active queue management (AQM), congestion control, load balancing, and network telemetry.

Another limitation lies in the inability to perform periodic tasks. The implementation of many data-plane algorithms would significantly benefit from the ability to periodically update algorithmic state, and others would benefit from the ability to periodically generate packets in the data plane.

These limitations arise in the case of PISA-style architectures, such as the one outlined in the Portable Switch Architecture (PSA) specification, because they only support variations of three types of events: packet arrival, packet departure, and packet recirculation, events. Data-plane algorithms, on the other hand, often need to take advantage of many other event types such as the ones described in Table 1. In addition to addressing the limitations identified, there are also events corresponding to control plane events, link status changes, and also user-defined events. In this context, we believe that an event-driven architecture supporting all of the scenarios proposed in the table will addresses many of the limitations encountered by today's data-plane application developers.

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 Event	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.

We have developed two platforms to experiment with writing event-driven programs. We integrated the P4.org bmv2 framework into the NS3 network simulator to create a P4 programmable event-driven architecture. Using this platform we prototyped numerous active queue management (AQM) policies including RED, WRED, AFD, GSP, PIE, and PI^2. In order to demonstrate the practicality of deploying an event-driven architecture in hardware, we developed the SUME Event Switch using the P4$\rightarrow$NetFPGA toolchain and used this architecture to implement and evaluate a version of the FRED AQM policy.