Event-Driven Packet Processing

Stephen Ibanez, Gordon Brebner, Gianni Antichi, Nick McKeown

May 1st 2019
Synchronous packet-by-packet processing
Limitations of P4 Programming Model

> Performing periodic tasks
  ➢ Count-Min-Sketch – periodic state reset

> Updating state multiple times / using state in a different stage
  ➢ Using congestion signals in ingress pipeline (AQM, NDP [2])

<table>
<thead>
<tr>
<th>Common Congestion Signals</th>
<th>Other Congestion Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Queue size</td>
<td>• Packet loss volume</td>
</tr>
<tr>
<td>• Queue service rate</td>
<td>• Rate of change of queue size</td>
</tr>
<tr>
<td>• Queueing delay</td>
<td>• Timestamp of buffer overflow/underflow events</td>
</tr>
<tr>
<td></td>
<td>• Per-active-flow buffer occupancy</td>
</tr>
<tr>
<td></td>
<td>• Etc…</td>
</tr>
</tbody>
</table>

> Solution: Generalize: Packet arrival/departure events ➔ data-plane events

## Data-Plane Events

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Packet &amp; Metadata Events</strong></td>
<td></td>
</tr>
<tr>
<td>Ingress Packet</td>
<td>Packet arrival</td>
</tr>
<tr>
<td>Egress Packet</td>
<td>Packet departure</td>
</tr>
<tr>
<td>Recirculated packet</td>
<td>Packet sent back to ingress</td>
</tr>
<tr>
<td>Buffer Enqueue</td>
<td>Packet enqueued in buffer</td>
</tr>
<tr>
<td>Buffer Dequeue</td>
<td>Packet dequeued from buffer</td>
</tr>
<tr>
<td>Buffer Overflow</td>
<td>Packet dropped at buffer</td>
</tr>
<tr>
<td>Buffer Underflow</td>
<td>Buffer becomes empty</td>
</tr>
<tr>
<td>Timer Event</td>
<td>Configurable timer expires</td>
</tr>
<tr>
<td><strong>Metadata Events</strong></td>
<td></td>
</tr>
<tr>
<td>Control-plane triggered</td>
<td>Control-plane triggers processing logic in data-plane</td>
</tr>
<tr>
<td>Link Status Change</td>
<td>Link goes down / comes up</td>
</tr>
<tr>
<td>Packet Transmission</td>
<td>Packet finished transmission</td>
</tr>
<tr>
<td>State Condition Met</td>
<td>User defined condition</td>
</tr>
</tbody>
</table>
Event-Driven Programming Model

Dequeue Event

Enqueue Event

Ingress Packet

Does not sacrifice line-rate packet processing
Event-Driven Programming Model

> E.g: Compute total buffer occupancy:

```c
// arch.p4
extern shared_register<T> {
    shared_register();
    void read(out T result);
    void write(in T value);
}

// my_prog.p4
shared_register<bit<32>>() bufSize_reg;

// Ingress Packet Event Logic
control Ingress(inout headers_t hdr, inout std_meta_t meta) {
    bit<32> bufSize;
    apply {
        bufSize_reg.read(bufSize);
        use bufSize to make forwarding decisions
    }
}

// Enqueue Event Logic
control Enqueue(inout enq_data_t meta) {
    bit<32> bufSize;
    apply {
        bufSize_reg.read(bufSize);
        bufSize = bufSize + meta.pkt_len;
        bufSize_reg.write(bufSize);
    }
}

// Dequeue Event Logic
control Dequeue(inout deq_data_t meta) {
    bit<32> bufSize;
    apply {
        bufSize_reg.read(bufSize);
        bufSize = bufSize - meta.pkt_len;
        bufSize_reg.write(bufSize);
    }
}

E.g: Compute total buffer occupancy:
Lower Line Rate Event Processing

- Multi-ported memory is more practical
- One port per event type that accesses state array
Higher Line Rate Event Processing

- Multi-ported memory is impractical
Higher Line Rate Event Processing

- Multi-ported memory is impractical

- Approach:
  - Multiple single ported register arrays
  - Packet event RMW operations operate on main register
  - Metadata event RMW operations are aggregated
  - Staleness of algorithmic state is bounded

Enqueue Event
- Enqueue: NULL

Dequeue Event
- Dequeue: SUB 100B from queue 0

Packet: READ queue 1

Ingress Packet Event

Dequeue Event

Enqueue Event

Traffic Manager

Enqueue Event

Dequeue Event
NetFPGA SUME Event Switch Demo

- Simple Fair-RED (FRED) AQM implementation
- Isolate TCP flow from non-adaptive UDP flow
- Computes per-active-flow queue occupancy
  - Enqueue & Dequeue Events
- Queue occupancy tracing:

![Queue occupancy tracing chart](chart.png)

- With FRED – UDP flow limited to 10KB of buffer
- One 10MB TCP Flow
  - ~0.01 sec FCT
Conclusion

> Network algorithms are event-driven, so should our data-plane architectures

End Host Protocols
(e.g. Reliable Delivery)

- Start
- Send Pkt
- Wait ACK
- ACK received
- done
- timeout
- Data sent

Control-Plane Protocols
(e.g. Routing Protocols)

- Start
- Send LSU
- IDLE
- timeout
- link status change
- IDLE
- Periodic event
- Send HELLO

> Potential to offload much more functionality to our data-planes
Questions?
Line Rate Event Processing

Idle clock cycles:
1. Workload contains large packets
2. Pipeline runs faster than line rate

Bounded staleness of the main register
SUME Event Switch on NetFPGA

- Packet Generator
- Timer Module
- Event Merger
- Arbiter
- Output Queues
- P4 Pipeline (SDNet)
- Generate packet
- Timer period
- Enq event
- Deq event
- Drop event
- Link status change event
- Timer event
- Packet event
- nf0
- nf1
- nf2
- nf3
- dma
- nf0
- nf1
- nf2
- nf3
- dma