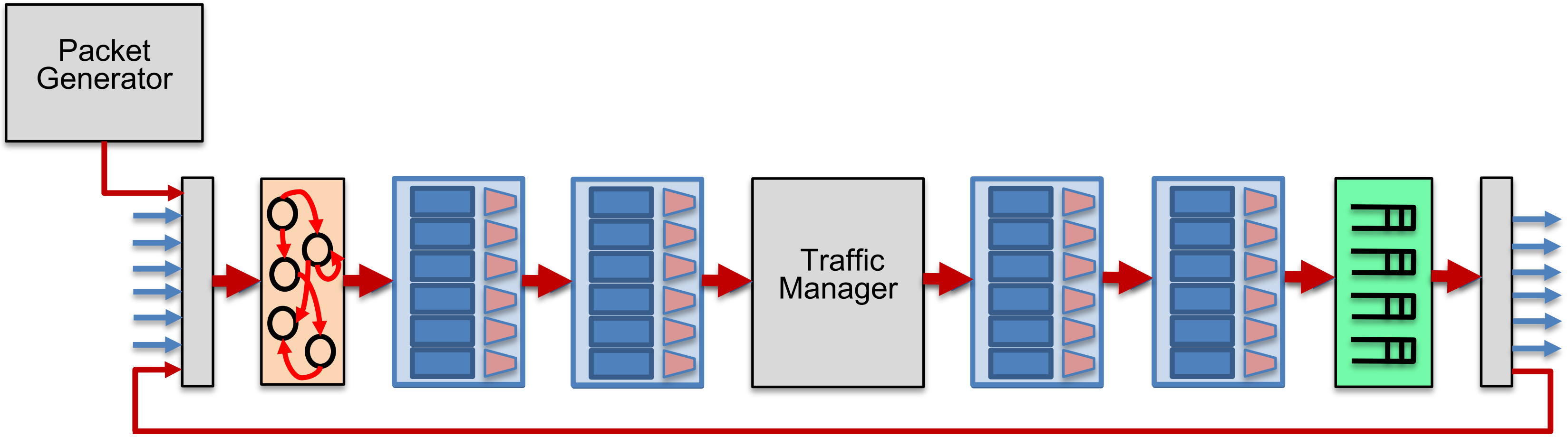
Event-Driven Packet Processing

Stephen Ibanez, Gordon Brebner, Gianni Antichi, Nick McKeown

May 1st 2019

P4 Programming Model



Synchronous packet-by-packet processing

Limitations of P4 Programming Model

> **Performing periodic tasks**
>> HULA [1] – periodic packet probes
>> 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])

Common Congestion Signals	Other Congestion Signals
• Queue size • Queue service rate • Queueing delay	• Packet loss volume • Rate of change of queue size • Timestamp of buffer overflow/underflow events • Per-active-flow buffer occupancy • Etc…

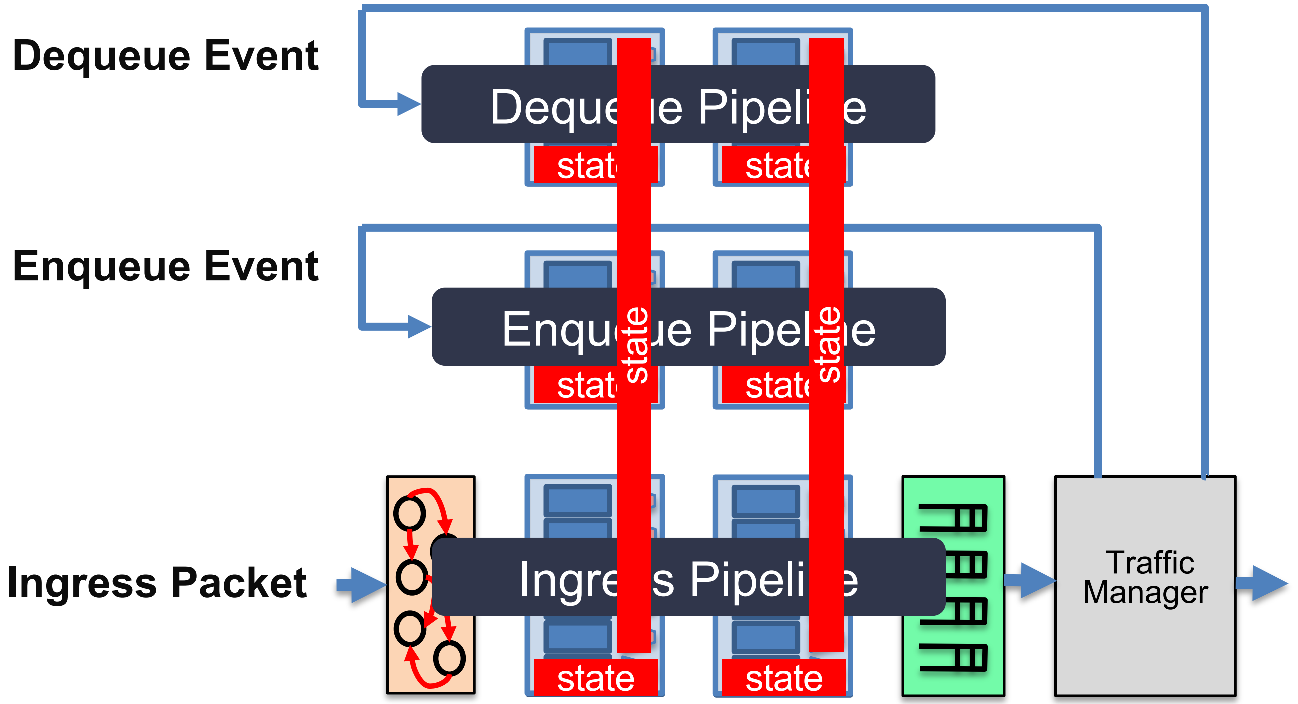
> **Solution:** Generalize: Packet arrival/departure events ➔ data-plane events

[1] Katta, Naga, et al. "Hula: Scalable load balancing using programmable data planes." *SOSR*, 2016.
[2] Handley, Mark, et al. "Re-architecting datacenter networks and stacks for low latency and high performance." *SIGCOMM*, 2017.

Data-Plane Events

	Event Type	Description
Packet & Metadata Events	**Ingress Packet**	Packet arrival
	Egress Packet	Packet departure
	Recirculated packet	Packet sent back to ingress
Metadata Events	**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 triggers processing logic in data-plane
	Link Status Change	Link goes down / comes up
	Packet Transmission	Packet finished transmission
	State Condition Met	User defined condition

Event-Driven Programming Model



Does not sacrifice line-rate packet processing

Event-Driven Programming Model

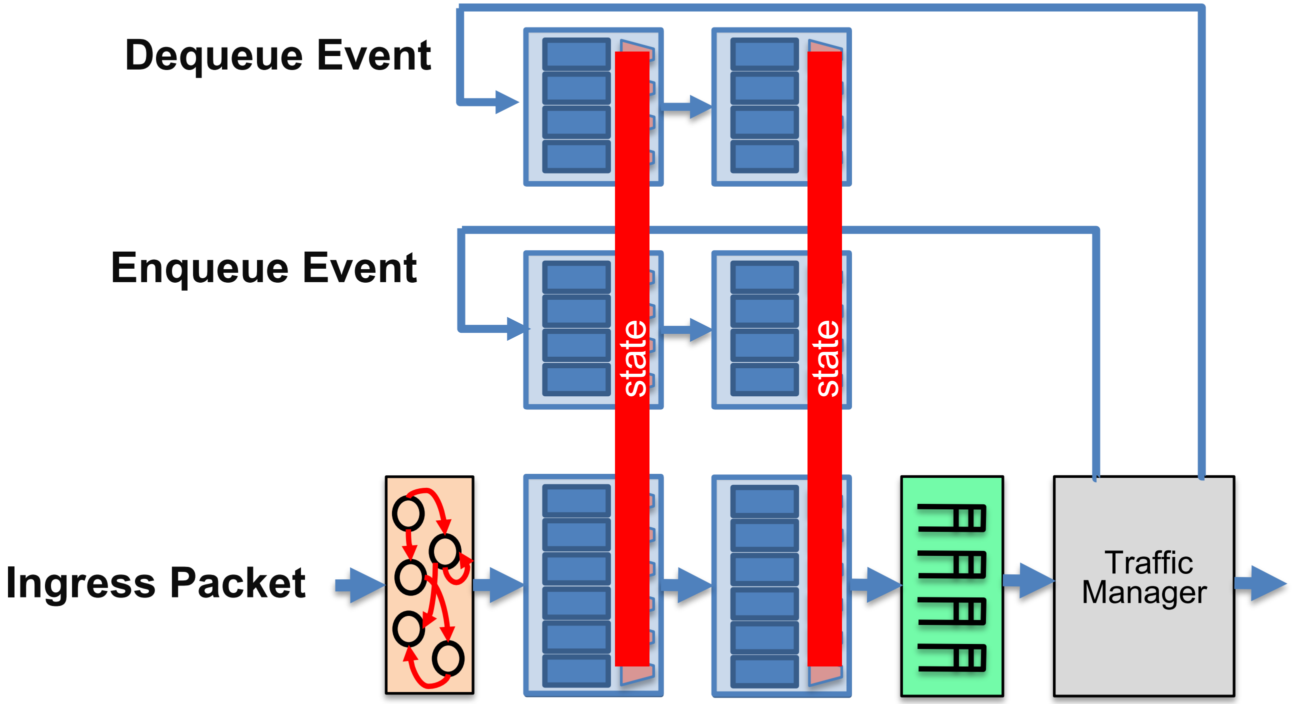
> E.g: Compute total buffer occupancy:

```
// 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);
  }
}
```

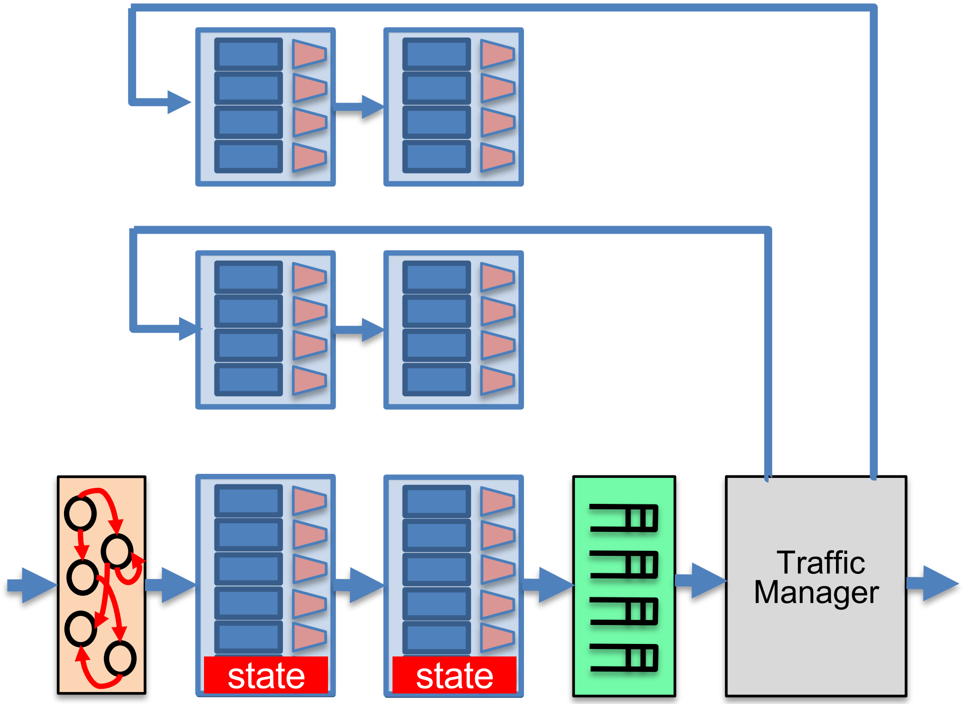
Lower Line Rate Event Processing

- Multi-ported memory is more practical
- One port per event type that accesses state array



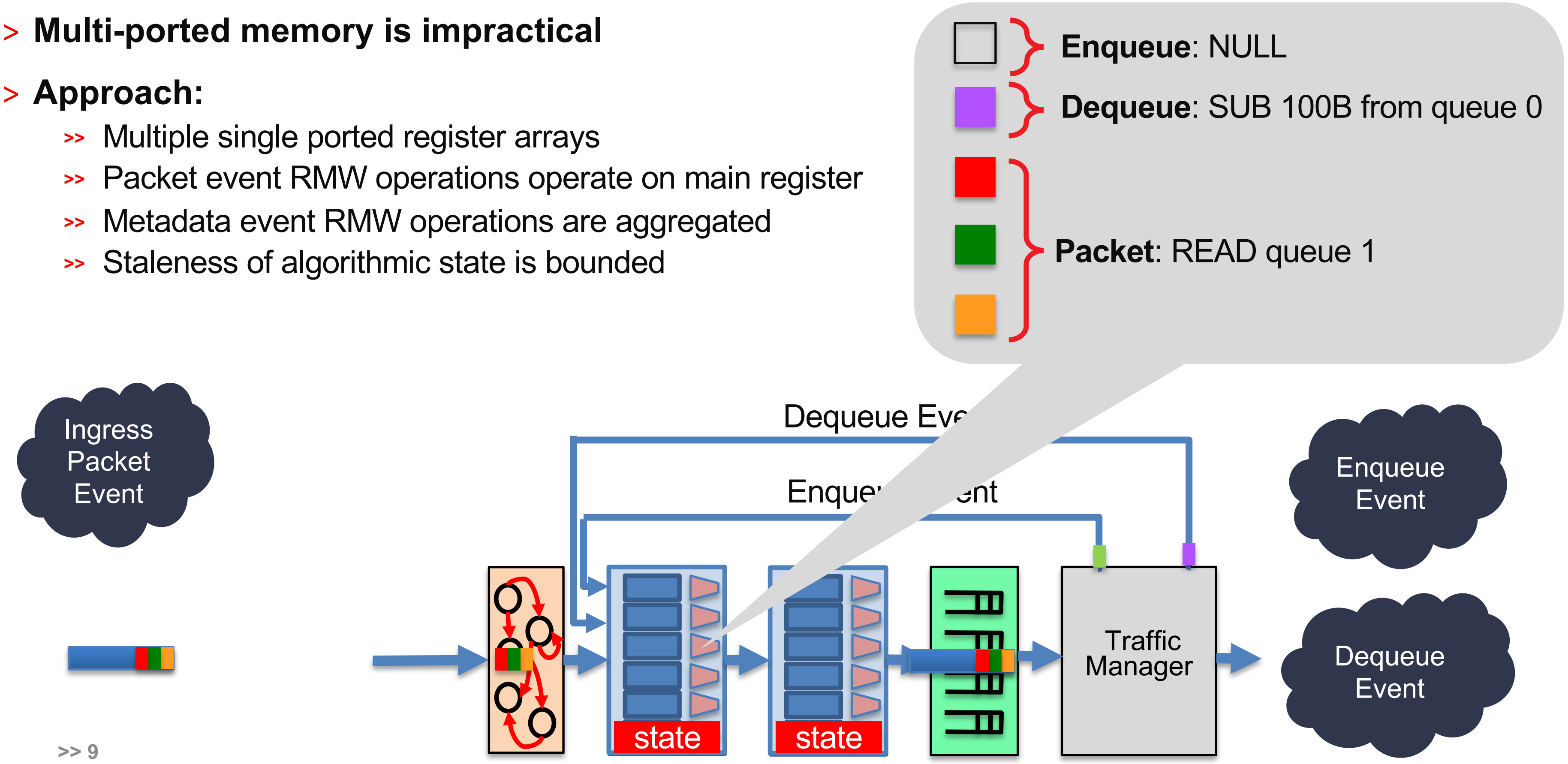
Higher Line Rate Event Processing

> **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



NetFPGA SUME Event Switch Demo

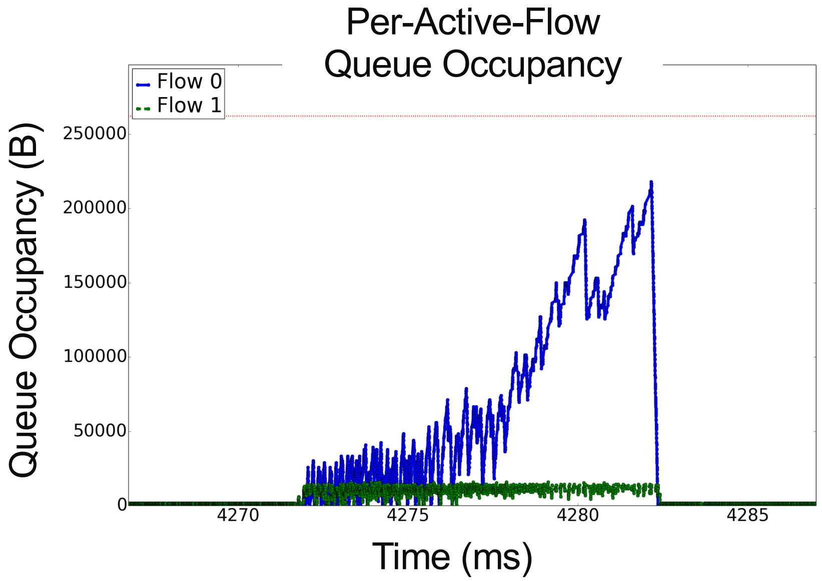
> **Simple Fair-RED (FRED) AQM implementation**

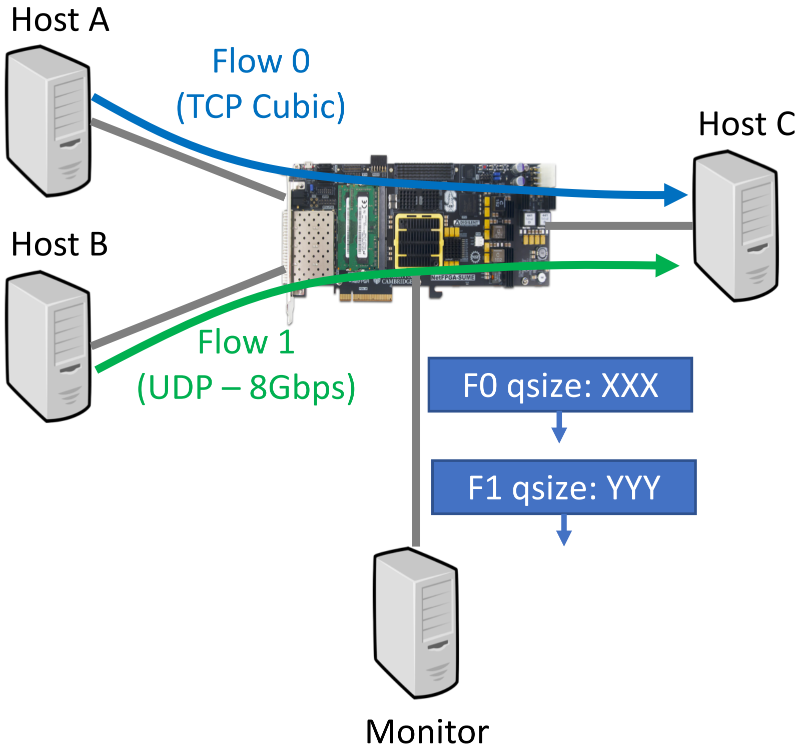
> **Isolate TCP flow from non-adaptive UDP flow**

> **Computes per-active-flow queue occupancy**

>> Enqueue & Dequeue Events

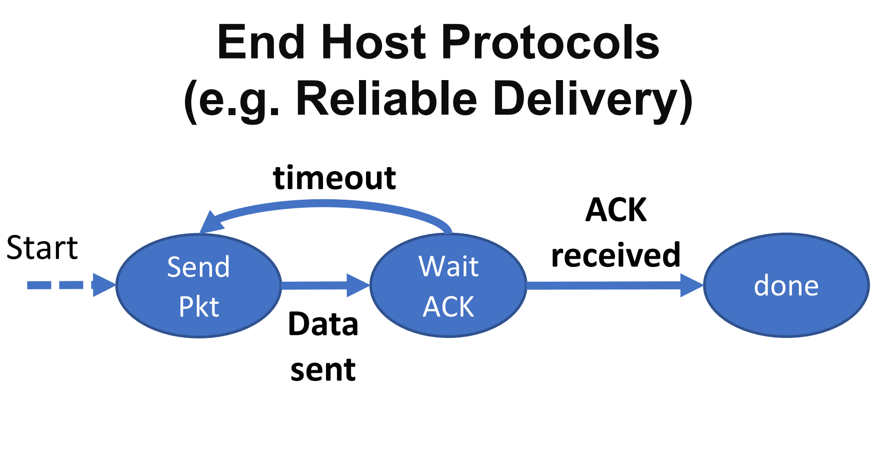
> **Queue occupancy tracing:**

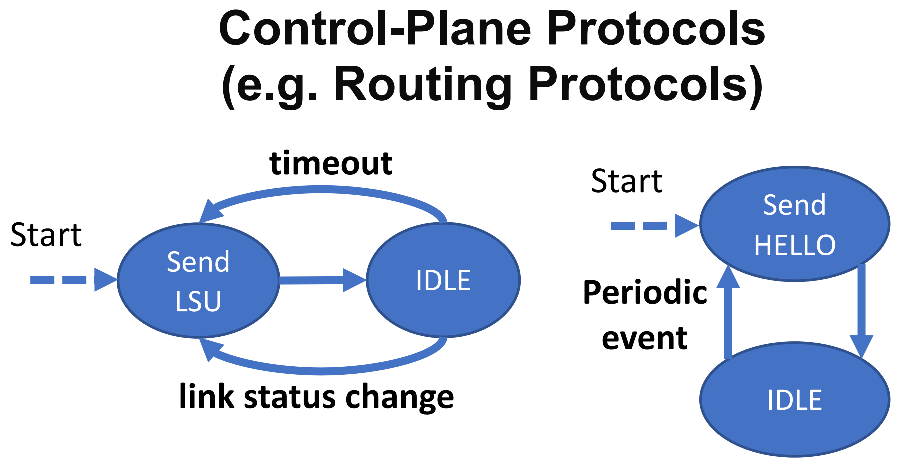




Conclusion

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

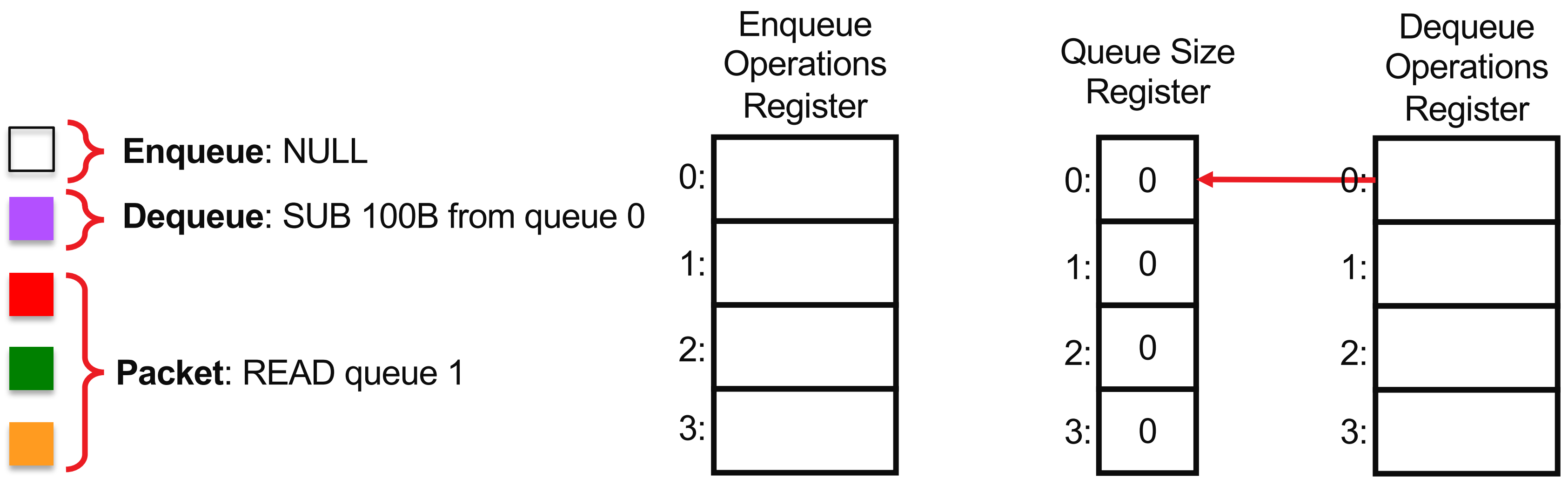




> Potential to offload much more functionality to our data-planes

Questions?

Line Rate Event Processing

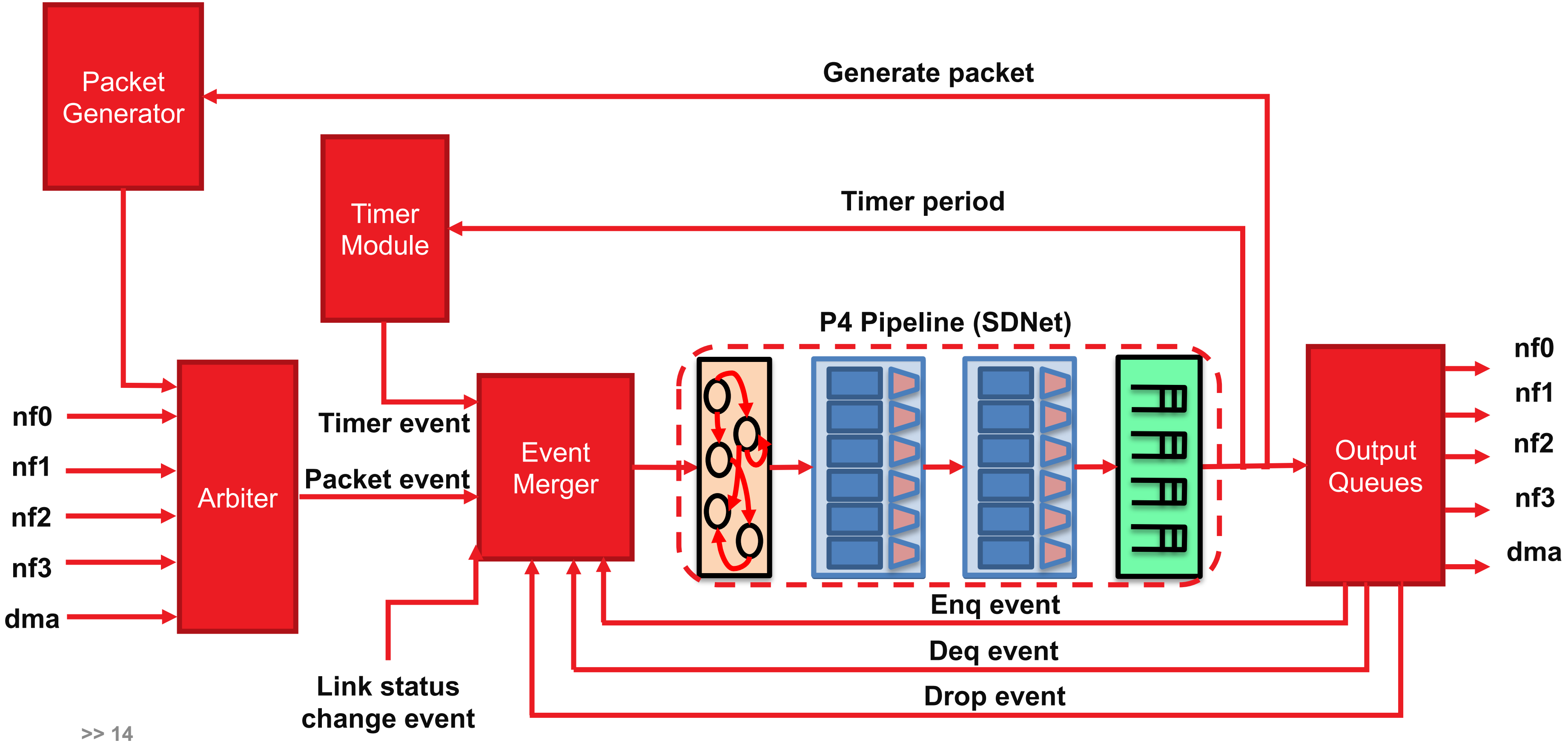


Enqueue: NULL
Dequeue: SUB 100B from queue 0
Packet: READ queue 1
Enqueue Operations Register
0:
1:
2:
3:
Queue Size Register
0: 0
1: 0
2: 0
3: 0
Dequeue Operations Register
0:
1:
2:
3:

> **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


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