end-to-end latency, and forces tight synchronization requirements between hosts and the switch. This can increase the overall traffic latency and jitter of widely used applications (i.e., VOIP, multiuser gaming etc.) and decrease the user quality of experience. As we move towards faster switching times, memory requirements diminish. Under the same configuration, a nanosecond switching time requires only kilobytes of buffering memory. This enables buffering packets directly in the ToR switch (see Figure 1, Fast Scheduling) and would remove issues relating to synchronization between the host and switch, thereby decreasing design complexity.

The scheduler is a key element that determines the performance of the data center network. With the availability of fast optical switches [3, 1] and increasing network demands, rapid scheduling is a necessity and not an option. Compared to its software counterparts, hardware based schedulers can match the speeds of fast optical switches and can be quick in responding to the dynamically varying network demand. This is inherent due to their hardware design: allowing quick demand estimation, fast schedule computation and rapid communication of computed schedules to the switch.

3. PROPOSED DESIGN

In the previous section we motivated the need for hardware based schedulers. Hardware may not be fast by default, but with proper implementation fast, high performance operation can be achieved. To this aim, we argue that the path towards the design and implementation of an optimal hardware scheduler requires a flexible framework for rapid prototyping, exploration and evaluation of novel hybrid schedulers. We aim to prototype the framework using a reconfigurable platform, NetFPGA-SUME [6]. The NetFPGA-SUME platform was designed for data center research, and enables the evaluation of new designs under real traffic workloads and with comparable performance.

We partition our design into processing logic, switching logic and scheduling logic as shown in Figure 2. The processing logic and switching logic are part of the infrastructure that is constant (yet configurable), and the users implement novel design in the scheduling logic module. Incoming packets from hosts H1, H2, .., Hn are sent to the processing logic. There, packets are classified into flows based on configurable look-up rules and places them into their respective Virtual Ouput Queue (VOQ). As the status of a VOQ changes, the subsystem generates scheduling requests and transmits packets upon receiving transmission grants from the scheduling logic. The scheduling logic processes the incoming requests, estimates the demand matrix, and runs the scheduling algorithm, generating corresponding transmission grants. Before providing a grant to the processing logic, the scheduler sends the grant matrix to the switching logic to configure the circuits in the OCS to match the grant matrix. Once the grant message is received by the processing logic, it dequeues packets from the respective VOQ and sends them to the OCS (that has already been configured according to the grant matrix) to be delivered to the respective destination. Based on the scheduling mechanism, residual traffic can be sent through the EPS. The scheme allows for multiple VOQs to be served at once, matching the port dimensions of the switching logic.

The design contains network interfaces, memory interfaces and various logical elements, omitted from the discussion for clarity. Individual partitions can be designed as separate entities and then integrated to realize a setup that emulates or uses commodity network devices. The resulting testbed enables us to explore scheduling architectures for hybrid switching, hybrid topologies for data center networks, synchronization issues, scalability and latency requirements in heterogeneous networks etc. It also allows to detect and analyse transient effects that may not be visible under simulation environments. The proposed architecture has the advantage of supporting both centralized and distributed implementations. A large testbed can be assembled, using tens of processing elements, a centralized scheduling entity and a commercial OCS. This implementation also allows to explore SDN practices over the hybrid network.

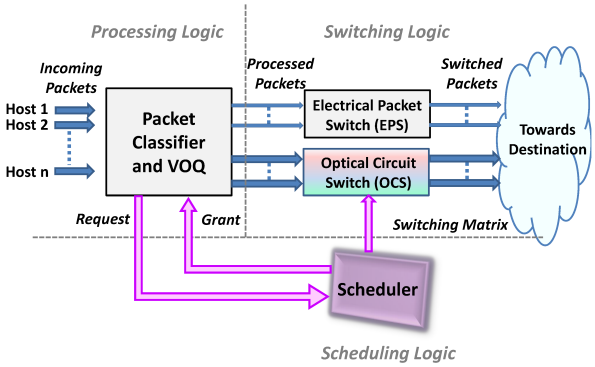

Figure 2: Proposed implementation

4. CONCLUSION

This paper motivates the need for hardware based schedulers in hybrid switches in order to meet emerging data center requirements. We have shown the main drawbacks that arise when using software based schedulers. We argue that the first step to achieve an optimal hybrid switch scheduler is to have a framework for rapid prototyping and assessment of new hardware-based scheduling algorithms. Finally, we show the architecture of the proposed framework, serving as an enabler for new scheduling algorithms.

5. ACKNOWLEDGEMENTS

This project is supported by the EPSRC INTERNET Project EP/H040536/1.

6. REFERENCES

[1] Epiphotonics. Nano-second speed plzt switch. http://www.epiphotonics.com/products3.htm. Accessed: 2015-05-08.

[2] N. Farrington et al. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*. ACM, 2010.

[3] H. Liu et al. Circuit switching under the radar with reactor. In *NSDI*. USENIX, 2014.

[4] C. Raffaelli et al. Evaluation of packet scheduling in hybrid optical/electrical switch. *Photonic Network Communications*, 2012.

[5] G. Wang et al. c-through: part-time optics in data centers. *SIGCOMM CCR*, 2010.

[6] N. Zilberman et al. NetFPGA SUME: Toward 100 Gbps as Research Commodity. *Micro*, 2014.