REACToR: A Hybrid Top-of-Rack Switch

George Porter, Alex C. Snoeren, George Papen, Geoffrey M. Voelker, Stefan Savage
University of California, San Diego Center for Networked Systems

Status Report, October 2014

1 Personnel

In addition to the PIs listed above, this project involves graduate students Lonnie Liu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Max Mellette, and Sen Zhang. The REACToR project is now an active participant in UCSD CSE's ERSP (Early Research Scholars) Program. Through ERSP, we host four undergraduate research students: Luis Sanchez, Edgar Lopez, Huayin Zhou, and Mingshan Wang. Since the last review, through NSF's REU program, we have hosted Felicia Wang, a visiting undergraduate student.

2 Research

Ever-larger data centers are powering the cloud computing revolution, but the scale of these installations is currently limited by the ability to provide sufficient internal network connectivity. Delivering scalable packet-switched interconnects that can support the continually increasing data rates required between literally hundreds of thousands of servers is an extremely challenging problem that is only getting harder. We propose to address this impending crisis by departing from today's electrically packet-switched technology and adopt the superior power and cost scaling enabled by optical circuit switching.

In this proposal, we leverage microsecond optical circuit-switch technology to develop a hybrid switching paradigm that spans the gap between traditional circuit switching and full-fledged packet switching, achieving a level of performance and scale not previously attainable. We propose a hybrid switch whose optical switching capacity is orders of magnitude larger than the electrical packet switch, yet whose performance from an end-to-end perspective is largely indistinguishable from a giant (electrical) packet switch. Fully realizing our vision, however, requires a substantial change in how data centers are networked—not just in the link technology, but encompassing the entire stack from the end-host protocol layers through the network interface card (NIC) to the top-of-rack switch (TOR) and beyond.

This project considers a holistic approach to reenvisioning the network interconnect, including modifications to the topology itself, the switches, end hosts, network interface cards, the end-system network stack, applications, and algorithm designs. Throughout this research we focus on ways to adapt existing systems to support circuit switching in the least disruptive way possible.

2.1 Recent progress

During the period since the last review, we have focused on a key challenge in building hybrid circuit/packet networks: the scheduling problem. In a traditional packet-switched network, end hosts are permitted to send packets into the network at will. In the absence of chronic congestion, temporary correlated bursts of packets from different sources are buffered in switch memory. In the case of a sustained flow between sources and a single destination, it is the responsibly of higher-layer transport protocols to reduce the sending rate to

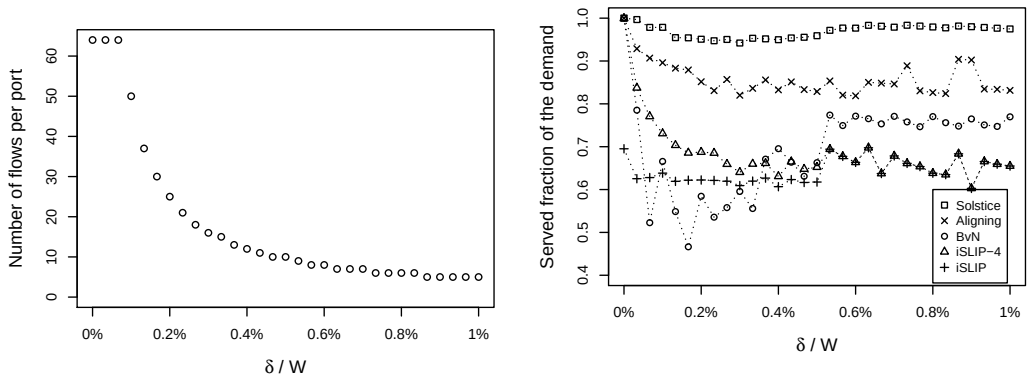

Figure 1: A comparison of the performance of various scheduling algorithms as a function of the fraction of the reconfiguration delay, δ, over the accumulation period, W. Note that the y-axis does not start at zero.

reduce congestion. In REACToR, we prevent in-network congestion (on the circuit paths) by construction, and thus it becomes necessary to carefully schedule transmissions across the source hosts to fully utilize those links. At the reconfiguration speeds of modern and impending circuit technologies, this scheduling process must be fast, scalable, and efficient. To this end, we have begun the development of *Solstice*, our approach to hybrid circuit/packet scheduling.

Solstice considers this scheduling problem for an N-port hybrid switch and places it in the context of the existing switching literature. We observe that many of the classical approaches to scheduling for switches with non-trivial reconfiguration delays divide the offered demand into two parts: an initial, heavy-weight component that is served by $O(N)$ highly utilized configurations with significant durations, and a second, residual component that is serviced by a similar number of short, under-utilized schedules. Solstice exploits the skewed nature of data center traffic patterns to create a small number of configurations with long durations that minimize the penalty for reconfiguration and leaves only a small amount of residual demand to be serviced by a low-speed (and low-cost) unconstrained packet switch.

In contrast to traditional constrained switches, hybrid architectures eschew speedup, instead preferring to use a lower-speed packet switch as a way to make up for the reconfiguration delay and any scheduling inefficiency. Because existing scheduling algorithms focus on reducing the overall reconfiguration penalty—which can be avoided entirely on the packet switch—they produce sub-optimal schedules in this setting. We motivate Solstice by considering the performance of existing algorithms in the absence of speedup—in other words, the crossbar can only forward traffic at the link rate of the input ports. To be fair, however, we ensure that the offered load can, in fact, be served without speedup; i.e., the demand matrix is not fully saturated. Yet, existing algorithms whose design is predicated on a speedup factor still service only a fraction of the demand.

For purposes of illustration, we consider in simulation a demand that consists of $N \times k$ flows, where each port has exactly k incoming and outgoing flows. Among the k flows on each port, 20% of them consume approximately 70% of the link rate, while the other flows equally share less than 30% of the link rate. In total, 98% of link is requested, which means at most 2% of the time can be spent on reconfiguring the switch (when no speedup is allowed and the demand has to be satisfied). If each reconfiguration takes δ (expressed as a function of the scheduling interval), then k can be at most $\delta/0.02$, since each new destination requires a reconfiguration—i.e., a schedule that satisfies the demand can require at most k different configurations. We can easily generate such a demand by stacking k random perfect matching of flows together.

Figure 1(a) plots the number of admissible flows, k, in this construction as a function of δ. For the purposes of this example, we limit k to at most $N = 64$, which clips the left-hand portion of the curve. To make the problem more realistic, we randomly perturb each of the computed flow sizes by $\pm 0.3\%$ of the link speed while respecting capacity limitations; the resulting random demand matrix is therefore still

satisfiable, but not as straightforward to decompose.

Figure 1(b) shows the fraction of demand serviced by several published scheduling algorithms along with Solstice, as a function of the reconfiguration time (expressed as a fraction of the accumulation time). In the figure, each mark is the mean link utilization of the algorithm serving 100 randomly generated demand matrices. As you can see, Solstice performs well over a range of possible network designs.

2.2 Ongoing work

Thus far we have built a simulator that implements Solstice. We have further developed in implementation of Solstice in the "Go" language, and integrated that codebase within the REACToR hardware prototype, based on an FPGA platform. Our ongoing research is focused in three directions. First, we are investigating the performance of Solstice across a range of data center traffic patterns. Second, we are investigating the potential of extending Solstice to support an overlay, multi-hop topology. Finally, we are starting to carry out an empirical analysis of Solstice by "closing the loop" of the REACToR prototype to include Solstice scheduling.