Switch Radix Reduction and Support for Concurrent Bidirectional Traffic in RotorNets

Yitzhak Birk
Electrical Engineering Dept.
Technion – Israel Inst. of Technology
Haifa, Israel
birk@ee.technion.ac.il

William M. Mellette
Computer Science and Engr. Dept.
University of California San Diego
San Diego, CA, USA
wmellett@ucsd.edu

Eitan Zahavi

Mellanox Technologies
Yokneam, Israel
eitan@mellanox.com

***Abstract*—Rotor Switches are optical switches that provide cyclic-shift permutations with a fixed round-robin schedule. They have been shown to offer high throughput while significantly reducing control complexity, making them a promising candidate for use in optical data center interconnects. We successfully address two critical challenges impeding the adoption of such circuit switched data center interconnects: 1) switch scalability, and 2) support for simultaneous 2-way connectivity to enable using low-latency protocols.**

Keywords— Network topology, Optical interconnections, bidirectional connectivity, radix reduction.

I. Introduction

In data centers, alongside high throughput and low latency, energy per bit and cost are of growing concern. One proposed approach is a hybrid network, whereby some of the ports of each end node, e.g., Top of Rack switch (ToR), are connected to a conventional packet-switched network, while the others are connected to an Optical Data Center Network (ODCN), a circuit-switched optical interconnect comprising electrically controlled optical switches. Latency sensitive traffic is directed to the conventional network, and the rest, most notably "Elephant" flows, is directed to the ODCN. The actual sender and recipient need not be aware of this, as they transmit (receive) conventional packets. One can view the ODCN as providing bypass jumpers between ToR switches. Inherent fault tolerance is another advantage.

One of the key challenges in building an ODCN is the need to decide which optical circuits to create or destroy at a rate that is fast enough to serve traffic that may be changing rapidly [1]. Using centralized scheduling to make these decisions, and even disseminating them to the optical switches and to the ToR switches, is extremely complex and often does not scale, especially for flows that are not gigantic.

A similar and relevant problem in the context of high-bandwidth electronic packet routers was solved by applying load balancing [2] and cyclic-shift permutations across two switching stages [3]. Subsequent work also looked at decomposing such a router into a number of separate line cards, taking into account the possibility of partial deployments or failures [4]. The RotorNet proposal [5] applied the idea of cyclic shift permutations to ODCNs, and employed multiple optical switches to offer higher total bandwidth and reduce the cycle length required to apply the full set of cyclic shifts. Thus, RotorNet provides an ODCN that avoids the need for a central circuit scheduler by employing a fixed TDMA schedule connecting all the network end-nodes to each other.

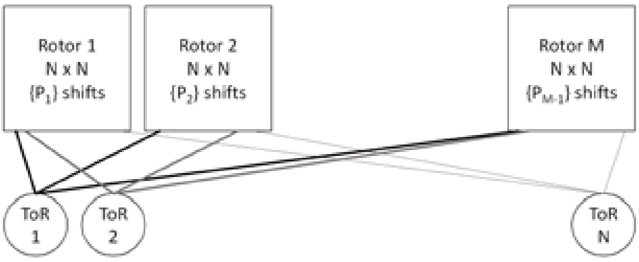

Fig. 1. A RotorNet of *N* end ports (ToRs) and *M* Rotor Switches. Each Rotor Switch applies cyclically a different set of *N*/*M* shift permutations $\{Pj\}|\ j \in [0, M)$.

The building block of RotorNet, a Rotor Switch, is an *NxN* circuit switch that is cycled among a set of cyclical shift permutations. Unlike an Optical Crossbar Switch (OCS) used by many other optical network proposals, which must enable all *N!* permutations, a Rotor Switch only needs to enable the N shift permutations.

If each ToR dedicates *M* ports to the ODCN, The RotorNet, depicted in Fig. 1, uses *M* concurrently operating Rotor Switches of size *NxN*, each providing a distinct subset of *N/M* cyclic shift permutations.

For a system of *2048* end nodes, each dedicating *128* ports to the ODCN, the RotorNet paper suggests using *M=128* Rotor Switches of *N=2048* inputs and *2048* outputs, each implementing *N/M = 16* permutations.

An M-fold reduction in the number of shift permutations that each rotor switch must implement simplifies its construction [6]. However, such a large radix switch has yet to be demonstrated, and its cost, power, maximum size, and manufacturability remain unproven. A large radix switch can easily be constructed using lower-radix ones, with $O(log_{radix})$ switches along any source-destination path, but this increases path loss, in turn requiring more active components and increasing cost. Consequently, a "few-stage" implementation of a large radix Rotor Switch from lower radix Rotor Switches would be of great benefit: it would allow the construction of larger single-stage rotors or fewer-stage ones using any given maximum component-switch radix, thereby reducing cost and enabling the market for actual deployment of the technology.

In conjunction with the fixed schedule and two-hop routing (going via an intermediate node that also acts as a buffer and relay), the RotorNet paper also suggests the optional use of link-level flow control between the connected ToR switches – such that when the destination ToR buffers are full, it may signal the sender ToR to stop its traffic. However, this feature, known as lossless network operation, adds a new requirement: the applied permutations must be

978-1-5386-9392-6/18/$31.00 ©2018 IEEE

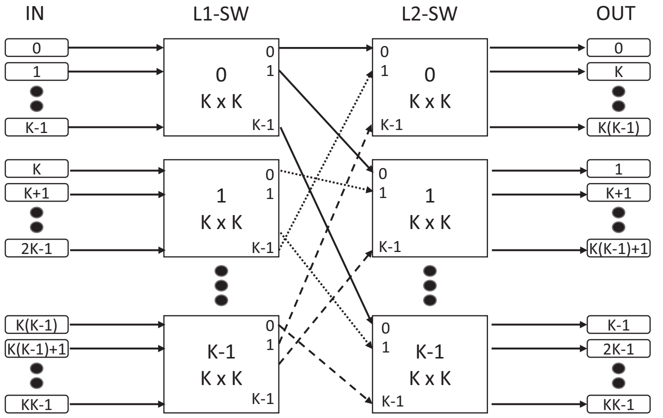

Fig. 2. Decomposition of an *NxN* Rotor switch into two layers of $\sqrt{N}x\sqrt{N}$ Rotor switches .

symmetric; i.e., paths (i, j) and (j, i) must be provided concurrently.

In this paper, we consider the same setting of *N* end-nodes, each having *M* connections to a circuit-switched optical network comprising rotor switches capable of providing cyclical shift permutations. We show one- and two-layer solutions for reducing the radix of component Rotor switches, and show how to provide bidirectional connectivity at little or no additional cost and with no other negative effects. We define $K = \sqrt{N}$, and will use the two interchangeably.

II. 2-Layer Decomposition of an *NxN* Rotor Switch into $\sqrt{N}x\sqrt{N}$ Rotors

For N sources and N destinations, a cyclic Shift permutation (denoted S_d) maps every source $src \in [0, N)$ to destination $dst = (src + d)\%N$. We now present the construction of an *NxN* rotor switch from two layers of *K* rotor switches of size *KxK*. Unless the level is mentioned, in- and out-port refers to the "external" ports of the resulting *NxN* rotor switch.

As depicted in Fig. 2, the *N* in-ports are connected to the inputs of the Layer-1 (L1) rotor switches in order. The *N* out-ports are connected to the output ports of Layer-2 (L2) switches with a stride of K. Thus, Rotor Switch out-port *m* is connected to out-port m/K of L2 switch number $m\%K$. The outputs of L1 switch m are connected, in order, to in-ports number *m* of all the L2 switches.

Consider a desired shift $0 \le d < K^2$. We express it as a 2-digit base *K* number. The L1 rotor switches all shift by LSD(*d*), and the L2 switches all shift by MSD(*d*).

Proposition 1: There is no contention. □

Proposition 2: In-port *i* is routed to out-port $(i + d)\%N$.

Proof: Note that LSD(*d*)= $d\%K$ and MSD(d)= d/K, and that in-port i is connected to input port $i\%K$ of L1 switch i/K.

The L1-SW shifts by LSD(d)= $d\%K$, so i is routed to output port $i\%K + d\%K = (i + d)\%K$ of this L1 switch. This is connected to input port i/K of L2 switch $(i + d)\%K$.

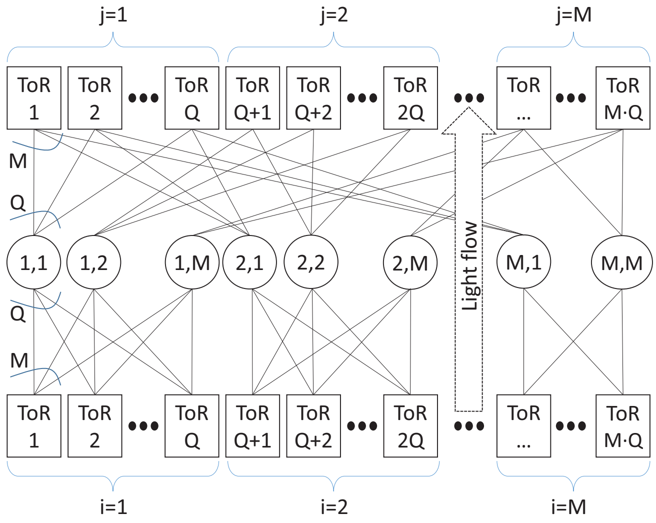

Fig. 3. A RotorNet topology based on M^2 *Q* x *Q* (*Q*=*N*/*M*) Rotor Switches.

For any $0 \le m < K$, the LSD of all destinations connected to L2 switch *m*'s output ports is *m*, so the LSD of the destination is correct. For any $0 \le m < K$ and *d*, the L2 switch input port *m* is connected to its output port $(m + d)\%K$. The L2 shift is by MSD(d)= d/K . Accordingly, port i/K is routed to output port $(i/K + d/K)\%K = ((i + d)/K)\%K$, which equals the MSD of $(i + d)$ and is connected to a destination with this MSD. Since the LSD was already shown to be correct, this completes the proof. □

III. Single-Layer RotorNet Decomposition

Consider a RotorNet that provides *M* concurrent permutations. (Each end node dedicates *M* ports to RotorNet.)

The original RotorNet employs *M* Rotor Switches, with every end node connected to all *M* switches, but with each Rotor Switch only providing a distinct subset of *Q=N/M* shift permutations, such that the *M* switches jointly implement all *N* such permutations. We now show how to attain the same functionality by a single-layer topology comprising M^2 switches of size *(N/M)x(N/M)*, each of which again provides (all) *Q=N/M* shift permutations. (Each such Rotor Switch can be further decomposed into two layers of $\sqrt{N/M}$ x $\sqrt{N/M}$ Rotor Switches per Section II).

In [7], single-hop connectivity was provided among *N* end nodes, each with *M* transmit ports and *M* receive ports, via M^2 directional *(N/M)x(N/M)* directional star couplers, enabling M^2 concurrent non-interfering transmissions. The *N* end nodes were partitioned into *M* disjoint subsets of equal cardinalities *Q=N/M*. (Without loss of generality, the sets can contain consecutive-numbered nodes.) Next, the Star Couplers were enumerated $S_{i,j}$, where *i,j=0,1,…,(M-1)*. Finally, each of the end nodes in subset *i* was connected to an input of all couplers $S_{i,*}$, and likewise each of the end nodes in subset *j* was connected to an output port of all couplers $S_{*,j}$. We now do the same, replacing the star couplers with Rotor switches that can implement all cyclic shift permutations. The resulting topology is shown in Fig. 3. The *(N/M)x(N/M)* Rotor Switches are drawn between the two rows of ToRs, which are split into inputs (bottom) and outputs (at the top) merely for clarity.

Proposition 3: The aforementioned topology provides all-to-all direct connectivity among all *N* end nodes, and each of the Rotor Switches need only implement *N/M* permutations.

Proof: $S_{i,j}$ provides connectivity from all nodes in subset *i* to all those in subset *j*. There is a switch connecting every pair of subsets, so full connectivity is provided. The total number of input and output ports is *M·N*, with each ToR connected to *M* input ports and to *M* output ports; given that a single permutation only "consumes" *N* ports, the claim holds. □

IV. Efficiently Supporting Symmetric Permutations

We define a permutation as Symmetric iff whenever it contains the connection (i,j) it also contains (j,i). The RotorNet paper does not provide a construction method for symmetric permutations, yet symmetric permutations are advantageous (if not mandatory) for three reasons: 1) traffic congestion control protocols, such as TCP, which relies on this symmetry and assumes that when traffic is sent in one direction, ACKs or CNPs (RoCE congestion notification packets) are able to traverse the network in the reverse direction; 2) the potential need to allow for link-level flow control packets (like Xon/Xoff) to be sent from the destination ToR port to the source ToR port. With this feature it is possible to avoid packet drops caused by buffer overflow, resulting in a lossless network; and 3) the RotorNet traffic load balancing algorithm relies on this symmetry for its required bidirectional communication. This algorithm improves on Valiant Load Balancing's 50% throughput by exchanging information between the source ToRs and the intermediate ToRs that are used to balance load.

The challenge addressed in this section is how to provide symmetric permutations at no or minimal hardware cost and using only unidirectional shift-permutation switches as building blocks. We first address the original RotorNet, and then proceed to our reduced-radix scheme of Section III. We note that the symmetry is guaranteed if, whenever S_d is scheduled, S_{N-d} is also scheduled, simply because $(s+N-d)\%N = (s-d)\%N$.

A. Symmetric Permutations in the original RotorNet

Proposition 4: A RotorNet comprising an even number of Rotor Switches providing distinct, equi-sized contiguous subsets of the shift permutations, can provide symmetric permutations.

Proof: For any $0<i<N$, a cyclic shift by N-i is the same as a cyclic shift by $-i$. Accordingly, for every permutation offered by the 1st switch, its inverse permutation is offered by the last one. Similarly for the 2nd one and next to last one, etc. To offer a permutation and its inverse concurrently, we simply run the schedule of the 2nd half of the switches in reverse order (pairs of "counter-rotating" Rotor Switches). □

Corollary 5: if allowable, one can swap the input and output connections of the Rotor Switches providing the larger shifts (shifts by *N/2* through *N-1*), and keep the original schedules. □

B. Symmetric Permutations in Reduced-Radix RotorNet

M^2-M switches have different sets of end nodes connected to their input and output ports, so the symmetry can only be provided at the overall network level. This is done as described in Section IV.A by arranging the switches in pairs of counter-rotating Rotor Switches. The remaining *M* switches can be duplicated at the cost of *M* extra *(M/N)x(M/N)* switches and an additional port of every ToR switch connected to RotorNet.

V. Conclusions

We presented two methods for reducing the radix of the component switches of a RotorNet with little path-loss penalty or none at all, thereby improving scalability. The two methods can be combined by decomposing each *(N/M)x(N/M)* switch into two layers of $\sqrt{N/M}\,\mathbf{x}\sqrt{N/M}$ Rotor switches. Finally, we showed how to support concurrent bidirectional traffic at little or no cost, both with the original RotorNet and with our ones. (Doing so with a single decomposed 2-layer Rotor Switch is also possible, but is beyond the scope of this paper.)

In summary, this paper significantly broadens the RotorNet design space, improving scalability, increasing flexibility of technology choices, and greatly facilitating the use of existing communication protocols, thereby facilitating adoption of this approach on a large scale.

References

[1] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," Proc. ACM Conf. on Special Interest Group on Data Commun., New York, NY, USA, 2015, pp. 123–137.

[2] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," 1981, pp. 263–277.

[3] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced Birkhoff–von Neumann switches, part I: One-stage buffering," Comput. Commun., 25(6), pp. 611–622, 2002.

[4] I. Keslassy et al., "Scaling Internet Routers Using Optics," Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun., New York, NY, USA, 2003, pp. 189–200.

[5] W. M. Mellette et al., "RotorNet: A Scalable, Low-complexity, Optical Datacenter Network," Proc. Conf. of the ACM Special Interest Group on Data Commun, New York, NY, USA, 2017, pp. 267–280.

[6] W. M. Mellette, G. M. Schuster, G. Porter, G. Papen, and J. E. Ford, "A Scalable, Partially Configurable Optical Switch for Data Center Networks," J. Light. Technol., vol. 35, no. 2, pp. 136–144, Jan. 2017.

[7] M. E. Marhic, Y. Birk, and F. A. Tobagi, "Selective broadcast interconnection: a novel scheme for fiber-optic local-area networks," Optics Letters, 10(12), pp. 629–631, Dec. 1985.