1-to-N and N-to-1 Communication for Optical Networks

Tuan Le
Department of Computer Science
University of California, Los Angeles
Los Angeles, USA
tuanle@cs.ucla.edu

Mario Gerla
Department of Computer Science
University of California, Los Angeles
Los Angeles, USA
gerla@cs.ucla.edu

ABSTRACT

Efficient 1-to-N and N-to-1 communication benefits numerous data center applications that require group communication by reducing network traffic and improving application throughput. Meanwhile, optical interconnection networks are emerging as a key enabling technology for future data center networking. Optical networks not only support higher bit-rates through optical links, but can also dynamically reconfigure the network topology and link capacities through the optical switch, providing substantial flexibility to various traffic patterns. Currently, there exists limited support for efficient 1-to-N and N-to-1 routing that leverages the advanced features of optical networks.

In this paper, we propose a set of algorithms to support 1-to-N and N-to-1 traffic flows in optical networks. We base our work on top of the Optical Switching Architecture (OSA). Through extensive analytical simulations, we show that the proposed algorithms are effective in minimizing the number of optical links used for 1-to-N traffic, while eliminating link bottlenecks for N-to-1 traffic.

Keywords
Optical networks, 1-to-N, N-to-1, routing, algorithms

1. INTRODUCTION

Optical networking technology is a promising technology for more flexibly allocating bandwidth across data centers [11, 15, 33]. Optical networks have many advantages. First, they provide higher bandwidth than traditional electrical networks. A single optical fiber can carry hundreds of gigabits per second [33]. Second, optical fibers can sustain high bit-rates over longer distances than copper cables. For 10 GigE electrical switches, the “power wall” limits copper cables to approximately 10 m [21]. Third, optical networks are more energy-efficient. Optical switches run cooler than electrical switches [12], resulting in lower heat dissipation and a lower cooling cost. For example, a Glimmerglass OCS consumes 240 mW/port, whereas a 10 GigE switch such as the 48-port Arista 7148SW [1] consumes 12.5 W per port, in addition to the 1 W of power consumed by an SFP+ transceiver [15]. Lastly and more importantly, optical networks can adapt network topology and link capacities to different traffic patterns, thus eliminating the need to provide uniformly high capacity between all servers as in static, electrical networks [8, 17, 18, 28].

However, there are disadvantages present in optical networks. First, the reconfiguration delay (during which no servers can use any optical path) is quite significant. As an example, MEMS (Micro-Electro-Mechanical Systems) optical switches, which reconfigure by physically rotating mirror arrays, have a reconfiguration time in the 1 to 10 ms range [33]. Second, optical components are more expensive than electrical components. An SFP+ 10 GigE optical transceiver can cost up to $200, compared to about $10 for a comparable copper transceiver [15]. However, there are new developments that are expected to mitigate these disadvantages in the near future. Tunable lasers can reduce the switching time to tens of nanoseconds [10]. Furthermore, volume manufacturing and emerging technologies such as silicon nanophotonics [32] may reduce the cost of optical components.

Many data center applications rely on multicast (1-to-N) communication patterns, such as publish-subscribe services for data dissemination [6], web cache updates [7], and system monitoring [24]. Furthermore, N-to-1 communication is also increasing in importance. Examples of N-to-1 applications include MapReduce [13] and Search. Whereas efficient routing algorithms for group communication in electrical networks have been thoroughly investigated [19, 22], to the best of our knowledge, no existing equivalent work leverages the reconfigurability of optical devices and the adjustable link capacity to optimize for the group communication in optical networks. Recent Data Center Networking (DCN) proposals that couple electrical network with optical network either schedule multicast packets over the pure electrical network (and thus completely ignore the optical components) [15, 33], or schedule over the optical network using shortest path routing [11] which results in high packet duplication and network congestion.

There are several challenges associated with group communication in optical networks. The first challenge is to configure the network topology and link capacity to optimize not only for group communication, but also for other kinds of traffic. The second challenge is to establish efficient routing paths that minimize the number of optical links used for 1-to-N communication and eliminate link bottlenecks for N-to-1 communication.

In this paper, we introduce a set of algorithms to address the above challenges. We evaluate these algorithms on top of the Optical Switching Architecture (OSA) [11]. First, we use the Hedera traffic demand estimation algorithm [9] to convert a TCP flow-rate matrix (among server racks) into a demand matrix. This demand matrix takes into account both the demand for group communication as well as for other kinds of traffic. Second, based on the estimated traffic
demand, since it is desired to maximize the amount of traffic between any pair of racks, we formulate the problem of localizing high-volume Top-of-Rack (ToR) connections into the maximum weighted $b$-matching problem. We leverage the fast Edmonds' algorithm [14] to obtain the solution in polynomial time. Then to make the $b$-matching graph connected, we use the edge-exchange technique [29]. We then compute efficient paths for routing. For 1-to-N routing, we build a Steiner tree using Kou, Markovsky, and Berman (KMB) algorithm [20]. For N-to-1 routing, we develop a Weighted Shortest Path Routing algorithm (WSPR) that uses pseudo-weight to spread traffic over various alternative paths, which helps to eliminate link bottlenecks. Finally, to provision the wavelengths among links to satisfy the traffic demand and the inter-rack wavelength assignment constraint, we reduce the problem to an edge-coloring problem which can be solved in polynomial time using Misra and Gries' algorithm [26].

The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 describes our system design in detail. Section 4 presents the experimental results. Section 5 concludes this paper.

2. RELATED WORK

Many optical interconnect schemes have been proposed and investigated. They fall into two categories: hybrid and all-optical schemes. c-Through is a hybrid electrical-optical network proposed by Wang et al. [33]. In c-through, the ToR switches are connected both to an electrical packet-based network (i.e. Ethernet) and to an optical circuit-based network. The architecture uses a traffic monitoring system placed in the hosts to measure the bandwidth requirement with the other hosts. These measurements are used to determine the configuration of the optical switch based on the traffic demand. After the configuration, each ToR can communicate directly to at most one other ToR. c-Through assigns ToR switches two different VLANs that logically isolate the optical network from the electrical network. If the packets are destined to a ToR switch that is connected to the source ToR through the optical circuit, the packets are assigned to the optical VLAN.

Another hybrid scheme presented by Farrington et al. is called Helios [15]. Helios uses a similar architecture to c-Through but is based on wavelength division multiplexing (WDM) links. Furthermore, unlike c-Through which performs traffic demand estimation and traffic demultiplexing in the end hosts, Helios implements these tasks on switches. Helios follows the architecture of typical 2-layer data center networks. It consists of ToR switches and core switches. ToR switches are common electrical packet switches, while core switches can be either electrical packet switches or optical circuit switches. The electrical switches are used for N-to-N communication, while the optical switches are used for high bandwidth long lived (or slowly changing) communication between the ToR switches. In short, both c-Through and Helios try to combine the best of the optical and the electrical networks.

All-optical schemes have also been proposed for data center interconnects. Ye et al. [34] present a scalable Datacenter Optical Switch (DOS) which is based on the Arrayed Waveguide Grating Router (AWGR) which allows contention resolution in the wavelength domain. The cyclic wavelength routing characteristics of the AWGR is exploited, that allows different inputs to reach the same output simultaneously.

3. SYSTEM DESIGN

In this section, we first describe a technique to estimate traffic demand at switches. Based on the estimated traffic demand, we present an algorithm to compute the optimal network topology. We then propose efficient routing schemes for 1-to-N and N-to-1 traffic. Finally, we discuss a way to provision wavelengths to satisfy network demands.

3.1 Traffic Demand Estimation

We leverage earlier work on estimating traffic demands among hosts [9] to compute the traffic demands among ToRs. The idea is to use max-min fair bandwidth allocation for TCP flows. To estimate the traffic demand, we perform repeated iterations of increasing the flow capacities from the sources and decreasing exceeded capacities at the receivers until the flow capacities converge. It is known that the estimation time complexity is linearly proportional to the number of source and destination pairs for active large flows.

As an example, consider 4 ToRs ($A$, $B$, $C$, $D$) (see Figure 2). Suppose $A$ sends 1 flow each to $B$, $C$, and $D$; $B$ sends 2 flows to $A$ and 1 flow to $C$; $C$ sends 1 flow each to $A$ and $D$; and $D$ sends 2 flows to $B$. Originally, each sender (row)
distributes the bandwidth equally among outgoing flows. In the matrix, as the aggregate demand exceeds the NIC bandwidth of receiver A (first column) \( \frac{1}{2} \times 2 + \frac{1}{2} = \frac{3}{2} \), the demand estimator decreases the exceeded capacity on the incoming flow from C to A \( \frac{3}{2} \rightarrow \frac{1}{2} \). This allows sender C to correspondingly increases the flow capacity to D from \( \frac{1}{2} \) to \( \frac{1}{2} \). The change is reflected in the first matrix of Figure 1. Similarly, in the second matrix, the capacities of the two incoming flows to B from D are reduced (from \( \frac{1}{2} \) to \( \frac{1}{2} \)) to match the receiver NIC’s aggregate capacity. The last matrix indicates the final estimated natural demands of the flows.

![Figure 1](image.png)

**Figure 1:** The iterative process of estimating demands in a network of 4 ToRs. Each matrix element denotes demand per flow as a fraction of the NIC bandwidth. Subscripts denote the number of flows from that source (rows) to destination (columns). Entries in parentheses are yet to converge. Entries in square brackets have converged.

We select this model to approximate traffic demands because of its simplicity and ability to determine the true traffic demand. It should be noted that TCP flow’s current sending rate may be misleading as it does not reflect its natural bandwidth demand due to blocking and congestion control.

### 3.2 Topology Computation

In optical networks, the topology is optimal when the system throughput is maximized. This can be achieved by greedily assigning direct links to pairs of racks with high traffic demands between them. More formally, this problem can be formulated as a maximum weighted b-matching problem [27], where b represents the number of ToRs that a ToR connects to via MEMS. As mentioned earlier, this implies that each ToR can communicate with b other ToRs simultaneously. In an undirected b-matching graph, each node has a degree equal to b. Note that the OSA architecture, upon which our routing algorithm is based, enables bidirectional transmission over a fiber by using optical circulators, thus making the graph of ToRs undirected. A maximum weighted b-matching problem is defined as a b-matching where the sum of edge values is maximal.

As an example, consider 4 ToRs \( A, B, C, D \). Figure 3 shows the traffic demand matrix among racks (which can be estimated as in Section 2.1), and the resulting maximum weighted 2-matching in which the degree of each node is 2 and the total aggregate edge weight is maximized.

![Figure 2](image.png)

**Figure 2:** The layout of ToRs and flows sent among them.

![Figure 3](image.png)

**Figure 3:** An example of maximum weighted b-matching \( b = 2 \) with 4 ToRs.

We implement maximum weighted b-matching using multiple perfect matchings (a perfect matching is a matching that matches all vertices of the graph). We use Edmonds’ algorithm [14] which can find the solution in polynomial time. The public library for Edmonds’ algorithm is available at [5].

Finally, note that the resulting b-matching graph is not necessarily connected. To achieve full connectivity, we first identify connected components in the graph using breadth-first search, which runs in linear time. Then, for different connected components, we connect them using a technique known as edge exchange [29], which selects the lowest-weight edge in each connected component and exchanges their connections. As an example, consider 8 ToRs \( A, B, C, D, E, F, G, H \). Suppose after computing the maximum weighted 2-matching, the resulting graph is a disjoint graph with two connected components as shown in Figure 4. To make the graph connected, we select two undirected edges \( C \leftrightarrow D \) and \( G \leftrightarrow H \) which have the lowest weights in their own connected components, and connect them via replacing links \( C \leftrightarrow D \) and \( G \leftrightarrow H \) with links \( C \leftrightarrow G \) and \( D \leftrightarrow H \). The resulting graph is connected and each node degree remains unchanged.

![Figure 4](image.png)

**Figure 4:** An example of edge exchange between two connected components.

### 3.3 Routing Path Computation

At this stage, the MEMS configuration is known. We proceed to compute the routing paths for 1-to-N and N-to-1
traffic. Since we limit the degree of a ToR to \( k \) (for example, \( k = 4 \)), some of the routing paths are single-hop while others will be multi-hop.

### 3.3.1 1-to-N Communication

The main objective for 1-to-N multicast routing is to send data from one or more sources to multiple destinations in a way that minimizes the usage of resources such as bandwidth, communication time and connection costs. A spanning tree has been considered one of the most efficient mechanisms for multicast routing since it minimizes the duplication of packets in the network [30]. Messages are duplicated only when the tree branches, which ensures that data communication is loop-free. An efficient multicast routing algorithm will aim to build a loop-free minimum-weight connected subgraph that includes all multicast members. This problem naturally maps to the minimal Steiner tree problem which is formally formulated as follows:

**Formulation:** Given an undirected weighted graph \( G = (V, E, w) \) and a set \( S \), where \( V \) is the set of vertices in \( G \), \( E \) is the set of edges in \( G \), \( w \) is a weight function which maps \( E \) into the set of nonnegative numbers and \( S \subseteq V \) is a subset of the vertices of \( V \), the minimal Steiner tree problem is to find a tree of \( G \) that spans \( S \) with minimal total weight on its edges.

Finding a minimal Steiner tree is an NP-Hard problem. A number of approximate and heuristic solutions have been proposed [20, 31, 25]. In our design, we choose to implement Kou, Markowsky, and Berman (KMB) algorithm [20], which is known to be a fast approximate algorithm for Steiner trees. KMB has a worst-case time complexity of \( O(|S||V|^2) \), and guarantees to output a tree that spans \( S \) with total weight no more than \( 2 \times (1 - \frac{1}{3}) \) times that of the optimal tree, where \( l \) is the number of leaves in the optimal tree. According to Paul [30], KMB tree averages a 5% higher cost than the optimal Steiner tree.

We now describe KMB algorithm:

**KMB Algorithm**

**INPUT:** An undirected weighted graph \( G = (V, E, w) \) and a set of Steiner points \( S \subseteq V \).

**OUTPUT:** A Steiner tree, \( T_H \), for \( G \) and \( S \).

1. Construct the complete undirected weighted graph \( G_1 = (V_1, E_1, w_1) \) from \( G \) and \( S \) in such a way that \( V_1 = S \) and, for every \( \{v_i, v_j\} \in E_1 \), \( w_1(\{v_i, v_j\}) \) is set equal to the total weight (cost) of the shortest path from \( v_i \) to \( v_j \) in \( G \).

2. Find the minimal spanning tree \( T_1 \) of \( G_1 \). If there are several minimal spanning trees, pick an arbitrary one.

3. Construct the subgraph \( G_S \) of \( G \) by replacing each edge in \( T_1 \) with its corresponding shortest path in \( G \). If there are several shortest paths, pick an arbitrary one.

4. Find the minimal spanning tree \( T_S \) of \( G_S \). If there are several minimal spanning trees, pick an arbitrary one.

5. Construct a Steiner tree \( T_H \) from \( T_S \) by deleting edges in \( T_S \), if necessary, so that all the leaves in \( T_H \) are Steiner points.

In step 1 and 3, we compute the shortest paths using Dijkstra’s algorithm [2]. In step 2 and 4, we compute the minimal spanning tree using Kruskal’s algorithm [4]. Note that in some cases, the Steiner tree can be obtained at the end of step 3. For multicast 1-to-N routing in the optical network, since the objective is to minimize the number of optical links used, we set the edge weights to 1. The set of Steiner points will correspond to the multicast members.

Figure 5 shows an example of building a Steiner tree using KMB algorithm. \( S = \{v_1, v_2, v_3, v_4\} \) form a multicast group.
link. We name our approach Weighted Shortest Path Routing (WSPR).

**WSPR Algorithm**

**INPUT:** An undirected weighted graph \( G = (V, E, p) \), a receiver \( V_d \in V \), and a set of senders \( S \subseteq V \).

**OUTPUT:** A set of paths for \( S \).

1. Initialize pseudo-weight \( p_i \) to 1 for all the edges in \( G \).
2. For each node \( V_s \in S \), compute the least-cost path to node \( V_d \). If there are several least-cost paths, pick an arbitrary one.
3. Order \( V_s \) by the least-cost path (the smallest total pseudo-weight path) to the destination \( V_d \) and save the result into a sorted list.
4. Sender with the least-cost path will take the preferred path first. If there are several senders with the same least-cost path, an arbitrary sender can be selected.
5. Once a path is taken, double each link pseudo-weight along the path.
6. The next sender in the sorted list recomputes the least-cost path on the modified graph. If there are several least-cost paths, pick an arbitrary one.
7. Repeat step 5 & 6 until the sorted list is empty.

In step 2 and 6, we compute the least-cost paths using Dijkstra’s algorithm [2]. Intuitively, as we continue adding the weight (cost) of a path once a node selects it, that path will become costly, causing other senders to avoid it. This naturally results in traffic flows evenly distributed among different paths and links, allowing us to achieve better load balancing in the network. As a tradeoff, the resulting paths may not be optimal in terms of hop count.

Figure 6 illustrates WSPR algorithm with a simple example. \( G = (V, E, p) \) is given in Figure 6a. Each edge weight is initialized to 1. Suppose \( v_1 \) and \( v_2 \) send a flow to \( v_7 \). Each sender independently computes the least-cost path to \( v_7 \). As shown in the figure, both the least-cost paths for \( v_1 \) and \( v_2 \) pass through edges \( e_{2,6} \) and \( e_{6,7} \). According to the WSPR algorithm, since \( v_2 \) has a lower least-cost path than \( v_1 \) (2 < 3), \( v_2 \) can take its preferred path first. Once the path is taken, the weights on the edges along the path from \( v_2 \) to \( v_7 \) are doubled as shown in Figure 6b. \( v_1 \) then recomputes the least-cost path on the modified graph. The resulting path passing through \( v_2 \) is of length 4 with a total cost of 4. On the other hand, an alternative path passing through \( v_2 \) has length 3 and a total cost of 5. Figure 6c shows the least-cost path for \( v_1 \). The weights are doubled from 1 to 2 after the path is taken.

### 3.4 Wavelength Assignment

The final step before communications between ToRs take place is to provision the wavelengths among the optical links. Given the matrix demand estimated in Section 3.1 and routes computed in Section 3.3, we can compute the capacity desired on each optical link while satisfying the wavelength contention constraint. This constraint stems from the fact that two data flows encoded over the same wavelength cannot share the same optical fiber in the same direction. In other words, a wavelength can only be assigned to a ToR at most once. This problem naturally maps to an edge-coloring problem [3] on a multigraph (a graph that is permitted to have multiple edges between two end nodes). In this graph, nodes and edges correspond to the ToRs and the wavelengths, respectively. Multiple wavelengths can be provisioned between two ToRs to satisfy high volume traffic between them. Assume that each wavelength has a unique color. Then, a feasible wavelength assignment corresponds to an assignment of colors to the edges of the multigraph so that no two adjacent edges have the same color. This is exactly the edge-coloring problem.

The edge-coloring problem is NP-Hard, which fast heuristics are capable of solving in polynomial time [16, 23, 26]. In this work, we use Misra and Gries’ algorithm [26], which is known to color any graph with \( \Delta + 1 \) colors in polynomial time, where \( \Delta \) is the maximum degree of the graph. Figure 7 shows an example of wavelength assignment on a multigraph of 4 ToRs, using 5 wavelengths (5 different colors).

**Figure 7:** An example of wavelength assignment (edge coloring).

### 4. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the two proposed routing algorithms via analytical simulations. We first describe the simulation setup, followed by the metrics used and the results.

#### 4.1 Simulation Setup

Our experiment consists of 80 racks. Each rack connects to \( k \) ports at the MEMS. We model this as an undirected
graph of 80 nodes, and each node is of degree $k$. For a given number of senders/receivers $N$ (which varies somewhere between 10 and 79), we perform 50 runs and average the results. For each run, we perform the following steps:

1. Randomly generate a 1-to-N or N-to-1 group.
2. Randomly generate a traffic demand for each node (a value in $[1,30]$).
3. Run maximum weighted $b$-matching on the traffic demands to find the optimal graph topology.
4. Run breadth-first search to find the connected components of the graph and then exchange the edges between the components (in a way that loses as little weight as possible) to make the graph connected.
5. Run 1-to-N or N-to-1 routing.

In step 2, to simulate a mice or an elephant 1-to-N multicast flow, we deliberately set the traffic demand value from a sender to N multicast members to either 1 or 30, respectively. The simulation codes are written in C++. We leverage the existing implementation of Edmonds’, Dijkstra’s, and Kruskal’s algorithm from LEMON library [5].

We conducted two sets of experiments. In the first set of experiments, we evaluate the routing performance of 1-to-N and N-to-1 using a fixed number of ports ($k = 4$) that the racks connect to at the MEMS. In the second set of experiments, we evaluate the sensitivity of the performance of routing algorithms to port count by varying the number of ports (value of $k$). Note that port usage is an important measure in designing optical switching networks. Adding more ports for the optical switch is very expensive and cost is a large consideration.

4.2 Evaluation Metrics

We compare our proposed KMB routing for 1-to-N and WSPR routing for N-to-1 against standard Shortest Path Routing (SPR). As mentioned earlier, group communication is currently not well supported in pure optical networks such as OSA. OSA, in its current state, always uses SPR routing regardless of different traffic patterns. We use the following metrics in the experiments:

- **Average number of links (hops) used.** For multicast 1-to-N traffic, this metrics translates into how many packets are duplicated in the network. A smaller number of links used indicates better bandwidth utilization due to the reduction of duplicate packets.

- **Average number of flows per link.** For N-to-1 traffic, this translates into how congested a link is. A smaller value indicates a better load balancing of traffic among different paths.

4.3 Experimental Results

In this subsection, we first present the routing performance for a fixed number of ports. We then show the sensitivity of the routing algorithms to the number of ports being used.

4.3.1 Routing Performance

1-to-N Routing: Figure 8 shows the average number of links used by KMB and SPR on mice and elephant multicast flows. Similarly, Figure 9 shows the performance gap between KMB and SPR on the two flow types. We make the following observations.

![Figure 8: Average number of links used in KMB vs in SPR.](image)

First, we find that SPR uses more links than KMB for both mice and elephant multicast flows. This is because shortest-path tree routing is inflexible; there is no coordination among multicast members to build an optimal tree (i.e., a tree that minimizes the number of edges). The path from the root to each member is fixed in a shortest-path tree, regardless of the identities and locations of other members. KMB, on the other hand, attempts to build a near optimal Steiner tree spanning members of the multicast group. In the experiment, the percentage difference in terms of the number of links used between KMB and SPR can reach up to 30%.

Second, we observe that the performance gap between KMB and SPR increases up to a multicast size of around 40 or 50 before decreasing steadily to 0. It reaches 0 when the multicast size is 80 (1-to-all communication). This is because when the multicast size is small, there are more choices for KMB over which path to take, to minimize the total length cost. However, when multicast size becomes too big, there are limited path choices. Thus, the gap between KMB and SPR decreases until it reaches 0. At that point, we are performing 1-to-all communication.
Third, we note that mice flow multicasting uses more links than elephant flow multicasting. This result is not surprising. Elephant flow causes high traffic demand between the multicast source and the multicast receivers, which in turn results in up to $k$ direct links configured between the source and the $k$ receivers. Recall that in our optical network, direct links are assigned between racks with high traffic demands. However, this is not the case for mice flows. Due to limited direct links configured for mice flows, the routing paths from the source to the multicast receivers are basically multi-hop paths. Thus, mice flow multicasting tends to use more links than elephant flow multicasting.

Furthermore, the gap between KMB and SPR is bigger for mice flows than for elephant flows (see Figure 9). As explained prior, KMB and SPR follow multi-hop routing more often with mice flows. With multi-hop routing, there are more edges involved and the shortest-path tree becomes much less optimal than the Steiner tree, resulting in bigger performance gap between KMB and SPR. In the case of elephant flows, both KMB and SPR benefit from some direct links to multicast members. Thus, the performance gap between them is smaller.

**N-to-1 Routing:** Figure 10 shows the average number of flows per link in our network. We find that as the total number of senders increase, the number of flows per link grows proportionally. However, the growth tends to be smaller in WSPR than in SPR. This shows that traffic flows from multiple senders are distributed more evenly across existing links under WSPR scheme. This is in contrast to SPR which tends to keep traffic along the same set of links, resulting in heavy link congestion.

![Figure 10: Average number of flows per link in WSPR vs SPR.](image)

Figure 11 further shows the performance gap between WSPR and SPR. The gap increases in a linear fashion as more senders are added to the network. By examining the loads of links in the network, we find that under SPR, bottlenecks appear around groups of nodes that are sending traffic in close proximity. However, these bottlenecks disappear under WSPR. Furthermore, we observe that links close to the receiver experience high bottlenecks under both routing schemes. This is expected because there are relatively few alternative paths to be taken closer to the receiver. For links that are farther away from the receiver, we see a relatively even traffic distribution among those links under WSPR. Hence, we can conclude that WSPR routing is efficient for load-balancing N-to-1 traffic.

![Figure 11: Performance gap between WSPR and SPR.](image)

**4.3.2 Sensitivity to Port Count**

**1-to-N routing:** Figure 12 and Figure 13 show the performance of KMB for multicast mice and elephant flows, respectively, as we vary the maximum number of ports $k$ that racks can connect to at the MEMS. We find that for $k$ less than 3, the performance of KMB degrades significantly, especially for small multicast groups. A large number of links are used for multicasting. This is the result of an insufficient number of direct links between the source and the multicast receivers, which in turn causes long multi-hop routing paths to deliver the multicast data. For $k \geq 3$, the routing performs better. We observe a sharp drop in the number of links used when $k = 3$. However, the perceived gain is negligible for $k > 3$. Thus, we conclude that our KMB scheme performs well when the maximum number of ports allowed is 3 or greater.

![Figure 12: KMB’s performance for multicast mice flows under different number of ports allowed.](image)

**N-to-1 routing:** Figure 14 shows the performance of WSPR as we vary the maximum number of ports allowed. We observe the same trend as in the case of 1-to-N routing. The performance increases dramatically when moving from 2 ports to 3 ports. However, beyond that, the gain is insignificant.

5. **CONCLUSION**

We have presented a set of algorithms to support efficient group communication in pure optical networks. Hedera algorithm is used to estimate traffic demands among racks.
Edmonds’ algorithm is used to obtain the optimal network topology. The routing paths for 1-to-N and N-to-1 traffic flows are computed using KMB and WSPR, respectively. Finally, the wavelengths are provisioned among links using Misra and Gries’ algorithm. We have evaluated the proposed algorithms via extensive analytical simulations. The results show that the algorithms can locate and assign direct links to pairs of racks with high bandwidth demands between them, providing full network connectivity. Moreover, the two proposed routing schemes are shown to be effective in minimizing the number of links used for 1-to-N traffic, while eliminating link bottlenecks for N-to-1 traffic.

![Figure 13: KMB’s performance for multicast elephant flows under different number of ports allowed.](image)

![Figure 14: WSPR’s performance for N-to-1 flows under different number of ports allowed.](image)

6. REFERENCES