HyperOptics: A High Throughput and Low Latency Multicast Architecture for Datacenters

Dingming Wu, Xiaoye Sun, Yiting Xia, Xin Huang, T. S. Eugene Ng
Department of Computer Science, Rice University

Abstract

Multicast has long been a performance bottleneck for data centers. Traditional solutions relying on IP multicast suffer from poor congestion control and loss recovery on the data plane, as well as slow and complex group membership and multicast tree management on the control plane. Some recent proposals have employed alternate optical circuit switched paths to enable lossless multicast and a centralized control architecture to quickly configure multicast trees. However, the high circuit reconfiguration delay of optical switches has substantially limited multicast performance.

In this paper, we propose to eliminate this reconfiguration delay by an unconventional optical multicast architecture called HyperOptics that directly interconnects top of rack switches by low cost optical splitters, thereby eliminating the need for optical switches. The ToRs are organized to form the connectivity of a regular graph. We analytically show that this architecture is scalable and efficient for multicasts. Preliminary simulations show that running multicasts on HyperOptics can on average be 2.1× faster than on an optical circuit switched network.

1 Introduction

As datacenters scale up, online services and data intensive computation jobs running on them have an increasing need for fast data replication from one source machine to multiple destination machines, or the multicast service. Apart from traditional multicast applications such as simultaneous server OS installation and upgrade [9], data chunks replication in distributed file systems [4, 13, 29] and cache consistency check on a large number of nodes [14], recent distributed machine learning models also see a huge demand for multicast services. The explosion of data allows the learning of powerful and complex models with 10^9 to 10^{12} parameters [11, 18], in which broadcasting the model parameters alone poses a challenge for the underlying network. Some learning algorithms require the processed intermediate data to be duplicated across different nodes. For example, the Latent Dirichlet Allocation algorithm for text mining needs to multicast the word distribution data in every iteration [10]. A few thousand iterations of LDA with 1 GB of data for each iteration would easily cause over 1 TB of multicast data transfer in today's datacenters. Reducing the multicast delay would significantly accelerate the machine learning jobs.

However, multicast services are still not natively supported by current datacenters. The most established solution is IP multicast which is originally designed for the Internet. Even though some efforts have been made to improve its scalability in the datacenter context [17, 19, 26], the complex dynamic multicast tree building and maintenance, the potentially high packet loss rate and costly loss recovery, and the lacking of satisfactory congestion control have caused most network operators to eschew its use.

On the other hand, as data size continues to grow, there is an increasing trend towards deploying a high bandwidth (40/100 Gbps) network core for datacenters [1]. However, high data rate transmissions are not feasible for even modest-length electrical links. For example, data transmissions on traditional twinax copper cable propagate at most 7 m at 40 Gbps due to power limitation [5]. Optical communication technologies are well suited to such high bandwidth networks. The advantages of optical devices and links, such as data rate transparency, lower power consumption, less heat dissipation, lower bit-error rate and lower cost have been noted or already exploited by the industry [3]. As datacenters gradually evolve from electrical to optical, we believe a system design that fully leverages the key physical features of optical technologies is necessary for future datacenters.

In this paper, we propose HyperOptics, a novel optical multicast architecture for datacenters. HyperOptics follows the recent efforts such as [12, 20, 23, 24, 27, 28] that augment the traditional electrical network with a high speed optical network, but HyperOptics dedicates the optical network to multicast transmissions. The existing optical network proposals usually employ an Optical Circuit Switch (OCS) to provide configurable connectivity for ToRs. The switching speed in today's large port-count OCSes is, however, orders of magnitude slower (about tens of millisecond) than packet switches. In [23], the authors propose a specific implementation of OCS that is capable of switching in microseconds, but it is unscalable to support a large port-count due to the limited number of available optical wavelengths. Also, OCSes are high cost devices. According to our recent quote from a vendor, a 192-port OCS would cost 365 K USD. All these problems of OCSes motivate us to design an optical network that gets rid of its use and directly interconnect the racks by low cost optical splitters. The design of HyperOptics is inspired by Chord's [25] way of organizing peer nodes in traditional overlay networks. Each ToR in HyperOptics can talk to multiple neighbor ToRs simultaneously via passive optical splitters, by which the ToRs form the connectivity of a regular graph.

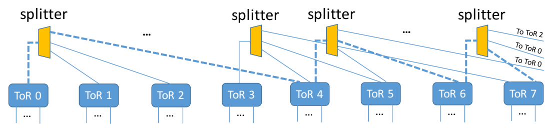
We identify two main advantages of HyperOptics over the OCS architecture. First, HyperOptics can provide high bandwidth even at the packet granularity because the slow circuit switching delay is completely eliminated. Second, unlike the existing OCS architecture, HyperOptics scales well in the number of ToRs because the constraint of the OCS port-count no longer exists in HyperOptics. We show that HyperOptics is well suited for high throughput and low latency multicast transmissions. Data from one ToR could be physically duplicated via an optical splitter to multiple ToRs at line speed. For multicasts with large group sizes, data is relayed by some intermediate ToRs. Due to the path flexibility of regular graphs, we show that the maximum path length for any multicast is bounded by $\log n$, where n is the number of ToRs. Another distinguishing property of HyperOptics is that it can support 2 simultaneously active multicasts with maximal group size. To take full advantage of the underlying optical technologies, we propose a centralized control plane that manages the routing policy and multicast scheduling. Preliminary simulations show that HyperOptics can on average be 2.1× faster than the OCS architecture for multicast services.

2 HyperOptics Architecture

We first introduce the connectivity structure of ToRs and then discuss the routing strategy under the given network architecture. Next, we analyze the multicast performance and the wiring complexity of HyperOptics. And finally, we present an overview of the system.

2.1 ToR Connectivity Design

We assume that all splitters have the same fanout k and the number of ToRs is $n = 2^k$. In our model, optical signals can only pass through the splitters in one direction, i.e., from the input port to the output ports. The ToRs are interconnected as a special k regular graph. The only difference of HyperOptics from the general k regular graph



Source ToR	0	1	2	3	4	5	6	7
Neighbors	1	2	3	4	5	6	7	0
	2	3	4	5	6	7	0	1
	4	5	6	7	0	1	2	3

Figure 1: An example of HyperOptics connectivity with splitter fanout $k = 3$. The connectivity of t_0, t_3, t_4 and t_6 are shown in the figure. All other ToRs are interconnected to their neighbors in a similar way. The table on the bottom demonstrates the connectivity of all ToRs.

is that a node (ToR) can only send the same data to its k neighbors simultaneously. This limitation comes from the fact that the splitter just passively duplicates the input signal on its output ports.

Assume that the ToRs are denoted as t_0, t_1,..., t_{n-1}. All ToRs are logically organized on a circle modulo 2^k. Each ToR t_i is connected to the input port of splitter s_i. The k output ports of s_i are connected to t_{i+2^0}, t_{i+2^1},..., $t_{i+2^{k-1}}$, respectively. Note that the gap between t_i's two consecutive neighbors increases exponentially, which is very similar to Chord [25] in organizing peer nodes in overlay networks. Since all ToRs are on a logical circle, the operations above are all modulo 2^k. For example, if $k = 3$ and $n = 8$, the third neighbor of t_4 is $t_{4+2^2} = t_0$. An example of HyperOptics with $k = 3$ is given in Fig.1. We only show the connectivity of t_0, t_3, t_4 and t_6 in the figure. The other ToRs are connected in a similar way, e.g., t_1 is connected to t_2, t_3, t_5. The full connectivity of the architecture is shown in the table on the bottom.

2.2 Routing and Relay Set Computation

Routing traffic to indirect destinations needs relays. For example, in Fig 1, a possible path shown as dashed lines from t_0 to t_7 is $t_0 - t_4 - t_6 - t_7$ where t_4 and t_6 are relays. There may exist multiple paths between each ToR pair. The relay set of a multicast is mainly determined by the routing strategy of HyperOptics. For simplicity, we propose a best-effort based routing strategy for HyperOptics. We note that our routing strategy might not be optimal and there is room for improvement. But it already provides satisfactory gains as we will show in Sec. 3.

For a single source-destination pair, best-effort routing will always designate the neighbor that is nearest to the destination as the next relay. Also, we ensure that the index of the next relay is logically smaller than the destination. Mathematically, given a destination t_j, a relay ToR t_i will specify $t_{i+2^{\lfloor log(j-i) \rfloor}}$ as the next relay. The routing algorithm will recursively compute the remaining relays as if the next relay is the source. For example, consider the traffic from t_0 to t_7 in Fig 1, the next relay for t_0 is t_4 because t_4 is one of t_0's neighbor that is nearest to t_7.

And the next relay for t_4 is t_6. Hence, the relay set for the path from t_0 to t_7 is $\{t_4, t_6\}$. Note that the next relay of t_4 is not t_0 because t_0 has logically passed the destination t_7. For multicasts, best-effort routing will compute the relay set for each individual destination and then return the union of all relay sets.

2.3 Analysis

We now analyze the multicast performance under the design of HyperOptics and compare the cost of HyperOptics with the traditional OCS networks.

Multicast hop-count: The hop-count of a multicast characterizes the minimum latency of a packet traversing from the source to the destination. The following theorem gives the worst case and average hop-count of a multicast in our architecture.

Lemma 1. *The maximum hop-count of a multicast under best-effort routing is upper-bounded by* $\log n$ *and the average hop-count is* $\frac{\log n}{2}$.

Proof. All ToRs in HyperOptics are logically equal. Without loss of generality, we consider a multicast originating from t_0. The k direct neighbors of t_0 is ToR 2^0,...,2^{k-1}, these IDs differ from 0 by only one bit. Similarly, the IDs of ToRs that are two hops away from t_0 differ by two bits. The farthest ToR differs by k bits. In best-effort routing, traversing a hop is equivalent to flipping the most significant bit of the source ToR's ID that is different with the corresponding bit of the destination's ID. Therefore, the largest hop-count is $k = \log n$. The number of ToRs that are j hops away from t_0 is $\binom{k}{j}$, $(1 \le j \le k)$. The average hop-count is $\frac{\Sigma_{j=1}^{k} j\binom{k}{j}}{\Sigma_{j=1}^{k}\binom{k}{j}} = \frac{k2^{k-1}}{2^k} = \frac{k}{2} = \frac{\log n}{2}$. □

For one hop, the signal decoding and packet processing can be done in sub-nanosecond [15]. Therefore, for a datacenter with 1 K racks, the average latency for a multicast is less than $0.5 * \log 1000 * 1\,\text{ns} \approx 5.0\,\text{ns}$. In the following, we simply assume that the multicast latency is negligible.

Simultaneously active multicasts: Each ToR in HyperOptics has k direct neighbors. In an extreme case where all group members of a multicast are the source's direct neighbors, HyperOptics could support n active multicasts simultaneously. In another extreme case where multicasts' group sizes are maximal and need the most number of relays, the number of simultaneous active multicasts would be much smaller. However, the following theorem shows that HyperOptics still has the capability of servicing multiple multicasts simultaneously in the worst case.

Lemma 2. *HyperOptics can simultaneously service two one-to-all multicasts.*

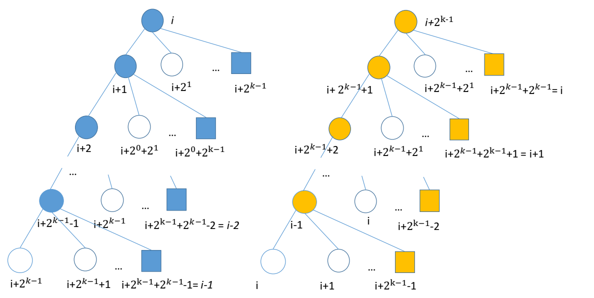


Figure 2: Two broadcast trees originating from t_i and $t_{i+2^{k-1}}$. Solid circles are relays. The union of the relays and the last neighbor of each relay, shown by squares, form a complete set of all ToRs. The two broadcasts have disjoint relay sets.

Proof. We consider two broadcast sources ToR i and ToR $i+2^{k-1}$. Under best-effort routing, we draw the two broadcast trees in Fig 2 where solid circles are relays, and squares are the last neighbor of each relay. As can be seen, the relay set and the last neighbor of each relay form a complete set of all ToRs for each broadcast. ToR i's relay set is $\{i, i+1, i+2, ...i+2^{k-1}-1\}$, while ToR $i+2^{k-1}$'s relay set is $\{i+2^{k-1}, i+2^{k-1}+1, i+2^{k-1}+2, ..., i-1\}$. The two relay sets are disjoint and therefore, both broadcasts can be active simultaneously. □

ToR port-count: In HyperOptics, each ToR is connected to the input port of a $1 \times k$ splitter. One splitter would take up $k+1$ ports across the ToRs. The average number of of occupied ports on each ToR would be $\frac{n*(k+1)}{n} = 1 + \log n$.

Cost: Even though HyperOptics does not use the OCS, it occupies more ToR ports than the OCS network. The per-port OCS cost is 1.5K USD, derived from our recent vendor quote (365K USD for a 192-port switch) but factors in a 20% discount. The per port cost of ToR and transceiver are 100 USD and 200 USD respectively, from [6, 8]. Splitters are very inexpensive at 5 USD per port [7]. For a medium size datacenter with 128 ToRs where each ToR is connected to other ToRs via 40 Gbps links, the total networking cost for HyperOptics is approximately 0.31 M USD. The total cost of the OCS network using a commercially available 192-port OCS is comparable at 0.33 M USD. For a datacenter with 256 racks, the total costs for HyperOptics and the OCS network using a 320-port OCS become 0.69 M and 0.56 M, respectively. HyperOptics is thus cost comparable with the OCS architecture under the current price of different network elements.

Wiring complexity: While the total number of fibers needed to interconnect the ToRs is $n \log n$. Many of them are short fibers that only go across a few racks. In a datacenter with 2^k racks, the k fibers from each ToR will go across 2^0, 2^1,..., 2^{k-1} racks, respectively. For instance, in a datacenter with 256 racks, only 2 fibers will go across over 50 racks for each ToR. The total number of long

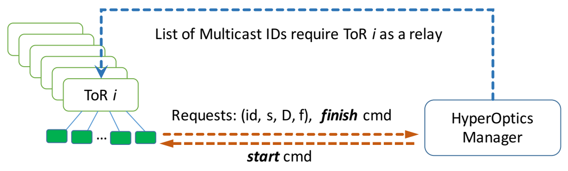

List of Multicast IDs require ToR *i* as a relay
ToR *i*
Requests: (id, s, D, f), ***finish*** cmd
HyperOptics Manager
start cmd

Figure 3: An overview of the HyperOptics system.

fibers that go across over 50 racks is $2*256=512$. For large datacenters with thousands of racks, we envisage that the ToRs are packaged into Pods. Pods can be wired in the same way as if one Pod is a single virtual ToR. This hierarchical organization of ToRs would significantly reduce the number of global fibers. A systematic study of this hierarchical design is our future work.

2.4 System Overview

Fig. 3 shows an overview of the HyperOptics architecture. Our current design of HyperOptics assumes that the network core bandwidth can be fully utilized. This assumption holds when the link bandwidth between a server and its ToR is the same as the inter-rack bandwidth. Or when a server's bandwidth is lower than the inter-rack bandwidth, multiple sources within a rack have the same destination set. The work-flow of HyperOptics is as follows. The manager first receives multicast requests from source servers. A multicast request contains the request ID, the source server, the destination servers and the flow size. The manager then computes the relay set for each request and send to each ToR i a list of IDs of multicasts that require ToR i as a relay. All multicast data packets contain a multicast ID in their headers. During the service period, when ToR i receives a packet, it will read the packet header and check whether it is a relay for the packet and relay the packet if it is. Note that this rule installation process is conducted only once before each scheduling cycle. Since relays are non-sharable resources for a multicast, multicasts that require common relays must be serviced sequentially. The HyperOptics manager will compute a schedule for all requests, which we will discuss in the next section. Every time a server finishes sending its multicast traffic, it will send a *finish* message to the manager, the manager then checks whether it is time to schedule the next batch of multicasts. If yes, then the manager will send a *start* message to the source servers of the next batch. Rules for the current scheduling cycle will be deleted on ToRs before the next cycle begins.

2.5 Multicast Scheduling

Given the input of n multicast requests, we now consider how to schedule these multicasts such that the overall delay is minimized. We formulate this problem as a max vertex coloring problem [22] where a vertex corresponds to a multicast, the edges correspond to the conflict relations among multicasts, i.e., if two multicasts have common relays, there's an edge between them. The weight of a vertex corresponds to the flow size of the multicast. Max vertex coloring has been shown to be strongly NP-hard [16]. We therefore focus on efficient heuristics. HyperOptics adopts a heuristic called Weight based First Fit (WFF) in which the vertices are first sorted in a non-increasing order of their weights. WFF then scans the vertices and assign each vertex a least-index color that is consistent with its already colored neighbors. The WFF heuristic is a specific version of the online coloring method for general graph coloring problems whose approximation ratio is analyzed in [21]. The time complexity of WFF is $\Theta(|V|^2)$.

3 Preliminary Evaluation

3.1 Experiment Set Up

In HyperOptics, the inter-rack link bandwidth is 40 Gbps. We also simulate the following two networks to compare with HyperOptics.

OCS network: Each ToR is connected to an OCS via a 40 Gbps link. The OCS has 320 ports, among which some are occupied by the ToRs, and the remaining ports are reserved for optical splitters. The number of splitters varies with the fanout of each splitter. The maximum group size achieved by cascading m $1\times k$ splitters is $k+(m-1)*(k-1)$. We assume the OCS reconfiguration delay is 25 ms according to commercially available products [2]. As is discussed in Sec. 2.3, the total cost of this network is comparable to HyperOptics.

Conceptual OCS network: We assume the Conceptual OCS has zero reconfiguration delay and sufficient port-count to support arbitrary multicast group size. The other configurations are the same as the OCS network. This network is not feasible in practice; it only serves as a comparison baseline to isolate the effect of different design components of HyperOptics.

The control plane delay consists of the scheduling algorithm computation time, the rule installation time and the control message transmission time between the manager and the servers. The computation time (measured at run-time) and the rule installation time (about 8.7 ms [28]) are one-time overheads for each scheduling cycle. The control messages between the manager and the servers can be implemented using any existing RPC solutions and its delay has been shown to be less than 2 ms [24, 28]. We assume in a scheduling cycle every rack has exactly one server that generates a multicast request (id, i, D, f) with itself being the source, D being a random subset of servers in other racks as receivers (each rack has a 50% chance of having some receivers for each source), and f being a random flow size between 10 MB and 1 GB. The number of requests is equal to the number

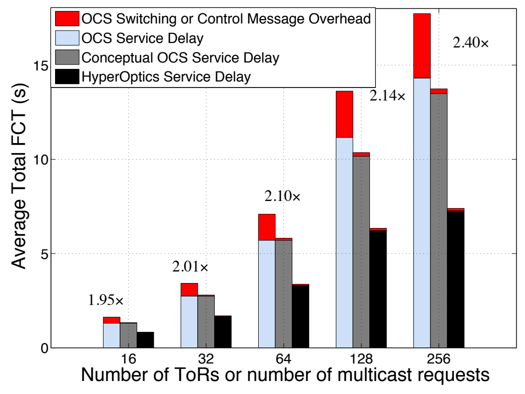


Figure 4: Comparison of the average total FCT of all three networks. The speedups of HyperOptics over the OCS network are labeled in the figure.

Fanout	2	4	6	8
FCT(s)	17.5	17.8	17.7	17.8

Table 1: Average FCT of 256 random multicasts on the OCS network under ToR size 256. The OCS uses splitters with fanout varying from 2 to 8.

of ToRs. We repeat the experiment 500 times and report the average result. This traffic pattern helps us evaluate the network core capacity of the HyperOptics architecture. Note that the group size of a multicast is constrained in the OCS networks due to the limited number of ports available for splitters. For better evaluations, we make sure that all multicast group sizes are no larger than the largest group size that the OCS network can support.

3.2 Simulation Results

We apply the WFF scheduling algorithm on both HyperOptics and the OCS network and compare the total flow completion time (FCT) of multicasts. The conflict relations of multicasts are only slightly different in the OCS network than in HyperOptics. In the OCS network, multicasts conflict when they share some destinations or there are not enough splitter resources to service them simultaneously, or one multicast's source is another multicast's destination.

3.2.1 Effect of Splitter Fanout Used in OCS

The overall multicast delay for the OCS network might vary as the splitter fanout changes. Table 1 shows the average FCT of 256 random multicasts on the OCS network with varying splitter fanout. It can be seen that the FCT remains quite constant. Intuitively, smaller/larger splitter fanout would yield better result when the multicast group size is small/large. In the following experiments, we always report the best result of the OCS network using various splitters.

3.2.2 Performance Comparison

Fig. 4 shows the average FCT for the three networks. We see that the speedup of HyperOptics is on average 2.13× over the OCS network. The speedup also increases with an increasing number of ToRs. We identify two reasons for HyperOptics' advantages. First, HyperOptics does not use the OCS, the high reconfiguration delay (occurring every time the circuits need to change) is completely eliminated. As can be seen, the overhead of OCS, mainly the OCS reconfiguration delay, is on average 24× larger than the overhead of HyperOptics, which contains only the 2 ms control message delay between the manager and the servers. Second, in the OCS network, a ToR can only receive traffic from one other ToR at a time. As a result, multicast requests that share some common destinations must be serviced sequentially. However, ToRs are interconnected in a $\log n$ regular graph in HyperOptics, each ToR can receive traffic from $\log n$ other ToRs simultaneously. We observe that the Conceptual OCS network is still 1.8× slower than HyperOptics. This fact shows that the unique connectivity structure of HyperOptics alone can lead to a significant FCT improvement.

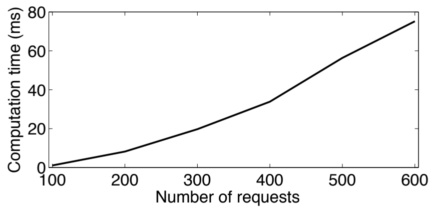


Figure 5: Computation time of HyperOptics' control plane under different number of multicast requests.

3.2.3 Computation Time of Control Algorithm

We run our C++ implementation of WFF scheduling on a 3.4 GHz, 4 GB RAM Linux machine. As is shown in Fig. 5, the time cost is less than 80 ms with 600 requests and less than 18 ms with 256 requests. In addition, this time cost is a one-time overhead for a scheduling cycle. The manager does not need to recompute the schedule in the service period. Results in Fig. 5 demonstrates that the HyperOptics manager is responsive in handling a large number of requests.

4 Conclusion

We have presented HyperOptics, a multicast architecture for datacenters. A key contribution of HyperOptics is its novel connectivity design for the ToRs that leverages the physical layer optical splitting technology. HyperOptics achieves high throughput and overcomes the high switching delay of the OCS. We show that the overall cost of HyperOptics is comparable with the OCS network, but it is on average 2.1× faster than the OCS network for multicast services. Our current routing and scheduling techniques in HyperOptics are quite basic and have much room for improvements. Our next step is to explore alternate routing and scheduling techniques to fully exploit the HyperOptics architecture.

Acknowledgement

We would like to thank the anonymous reviewers for their thoughtful feedback. This research was sponsored by the NSF under CNS-1422925, CNS-1305379 and CNS-1162270, an IBM Faculty Award, and by Microsoft Corp.

References

[1] 100 gbps for the data center. `http://www.networkcomputing.com/data-centers/100-gbps-headed-data-center/407619707`.

[2] Calient. `http://www.calient.net/products/s-journal-photonic-switch/`.

[3] Calient application note. `http://www.calient.net/resources/application-notes/`.

[4] Hadoop. `https://hadoop.apache.org`.

[5] Ieee802.3ba-2010 standard. `http://www.ieee802.org/3/ba/`. Accessed: 2016-02-01.

[6] Mellanox sx6536. `http://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=1760&idcategory=7`.

[7] Optical splitter. `http://www.fs.com/-p-11959.html?currency=USD`.

[8] Optical tranceiver. `http://shop.sfpcables.com`.

[9] Windows server. `https://technet.microsoft.com/en-us/library/hh831764.aspx`.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 2003.

[11] K. Canini, T. Chandra, E. Ie, J. McFadden, K. Goldman, M. Gunter, J. Harmsen, K. LeFevre, D. Lepikhin, T. Llinares, et al. Sibyl: A system for large scale supervised machine learning. *Machine Learning Summer School, Santa Cruz, CA*, 2012.

[12] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM*, 2010.

[13] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. *ACM SOSP*, 2003.

[14] C. Gray and D. Cheriton. Leases: An efficient fault-tolerant mechanism for distributed file cache consistency. *ACM SOSP*, 1989.

[15] G. Keeler, D. Agarwal, C. Debaes, B. E. Nelson, N. C. Helman, H. Thienpont, and D. A. Miller. Optical pump-probe measurements of the latency of silicon cmos optical interconnects. *IEEE Photonics Technology Letters*, 2002.

[16] M. Kubale. *Graph colorings*. American Mathematical Society, 2004.

[17] D. Li, Y. Li, J. Wu, S. Su, and J. Yu. Esm: efficient and scalable data center multicast routing. *IEEE/ACM Transactions on Networking (TON)*, 2012.

[18] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. *Advances in Neural Information Processing Systems*, 2014.

[19] X. Li and M. Freedman. Scaling ip multicast on datacenter topologies. *ACM Conext*, 2013.

[20] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky, et al. Scheduling techniques for hybrid circuit/packet networks. *ACM Conext*, 2015.

[21] A. Miller. Online graph colouring. *Canadian Undergraduate Mathematics Conference*, 2004.

[22] S. V. Pemmaraju and R. Raman. Approximation algorithms for the max-coloring problem. *ICALP*, 2005.

[23] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat. Integrating microsecond circuit switching into the data center. *SIGCOMM*, 2013.

[24] P. Samadi, V. Gupta, J. Xu, H. Wang, G. Zussman, and K. Bergman. Optical multicast system for data center networks. *Optics express*, 2015.

[25] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM*, 2001.

[26] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, R. Burgess, G. Chockler, H. Li, and Y. Tock. Dr. multicast: Rx for data center communication scalability. *ACM Eurosys*, 2010.

[27] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan. c-through: Part-time optics in data centers. *ACM SIGCOMM*, 2010.

[28] Y. Xia, T. S. E. Ng, and X. S. Sun. Blast: Accelerating high-performance data analytics applications by optical multicast. *INFOCOMM*, 2015.

[29] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 2010.