FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-Space Optics

Navid Hamedazimi,† Zafar Qazi,† Himanshu Gupta,† Vyas Sekar,⋆ Samir R. Das,† Jon P. Longtin,† Himanshu Shah,† and Ashish Tanwer†
†Stony Brook University ⋆Carnegie Mellon University

ABSTRACT

Conventional *static* datacenter (DC) network designs offer extreme cost vs. performance tradeoffs—simple leaf-spine networks are cost-effective but oversubscribed, while "fat tree"-like solutions offer good worst-case performance but are expensive. Recent results make a promising case for *augmenting* an oversubscribed network with reconfigurable inter-rack wireless or optical links. Inspired by the promise of reconfigurability, this paper presents *FireFly*, an inter-rack network solution that pushes DC network design to the extreme on three key fronts: (1) all links are reconfigurable; (2) all links are wireless; and (3) non top-of-rack switches are eliminated altogether. This vision, if realized, can offer significant benefits in terms of increased flexibility, reduced equipment cost, and minimal cabling complexity. In order to achieve this vision, we need to look beyond traditional RF wireless solutions due to their interference footprint which limits range and data rates. Thus, we make the case for using free-space optics (FSO). We demonstrate the viability of this architecture by (a) building a proof-of-concept prototype of a steerable small form factor FSO device using commodity components and (b) developing practical heuristics to address algorithmic and system-level challenges in network design and management.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design

Keywords

Data Centers; Free-Space Optics; Reconfigurablility

1 Introduction

A robust *data center (DC) network* must satisfy several goals: high throughput [14, 25], low equipment and management cost [14, 41], robustness to dynamic traffic patterns [16, 28, 50, 54], incremental expandability [20, 46], low cabling complexity [38], and low power and cooling costs. With respect to cost and performance, conventional designs are either (i) overprovisioned to account for worst-case traffic patterns, and thus incur high cost (e.g., fat-trees or Clos networks [14, 18, 25]), or (ii) oversubscribed (e.g., simple trees or leaf-spine architectures [1]) which incur low cost but offer poor performance due to congested links.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.

ACM 978-1-4503-2836-4/14/08 ...$15.00.
http://dx.doi.org/10.1145/2619239.2626328.

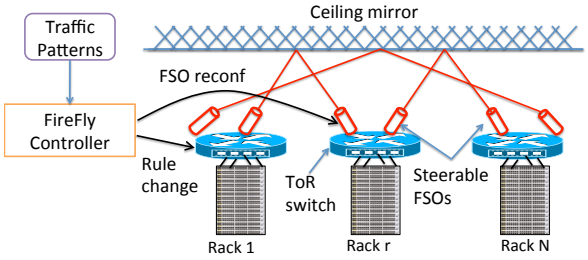


Figure 1: High-level view of the FireFly architecture. The only switches are the Top-of-Rack (ToR) switches.

Recent work suggests a promising middleground that augments an oversubscribed network with a few reconfigurable links, using either 60 Ghz RF wireless [28, 54] or optical switches [50]. Inspired by the promise of these flexible DC designs,[1] we envision a radically different DC architecture that pushes the network design to the *logical extreme* on three dimensions: (1) All inter-rack links are flexible; (2) All inter-rack links are wireless; and (3) we get rid of the *core* switching backbone.

This extreme vision, if realized, promises unprecedented qualitative and quantitative benefits for DC networks. First, it can reduce infrastructure cost without compromising on performance. Second, flexibility increases the effective operating capacity and can improve application performance by alleviating transient congestion. Third, it unburdens DC operators from dealing with cabling complexity and its attendant overheads (e.g., obstructed cooling) [38]. Fourth, it can enable DC operators to experiment with, and benefit from, new topology structures that would otherwise remain unrealizable due to cabling costs. Finally, flexibly turning links on or off can take us closer to the vision of energy proportionality (e.g., [30]).

This paper describes *FireFly*,[2] a first but significant step toward realizing this vision. Figure 1 shows a high-level overview of FireFly. Each ToR is equipped with reconfigurable wireless links which can connect to other ToR switches. However, we need to look beyond traditional radio-frequency (RF) wireless solutions (e.g., 60GHz) as their interference characteristics limit range and capacity. Thus, we envision a new use-case for *Free-Space Optical communications* (FSO) as it can offer high data rates (tens of Gbps) over long ranges using low transmission power and with zero interference [32]. The centralized FireFly controller reconfigures the topology and forwarding rules to adapt to changing traffic patterns.

While prior work made the case for using FSO links in DCs [21, 29], these fail to establish a viable hardware design and also do not address practical network design and management challenges that

[1]We use the terms *flexible* and *reconfigurable* interchangeably.
[2]FireFly stands for Free-space optical Inter-Rack nEtwork with high FLexibilitY.

arise in reconfigurable designs. Our work bridges this gap along three dimensions:

- **Practical steerable FSO devices (§3):** Commodity FSO designs are bulky, power hungry, and offer fixed point-to-point links. Our vision imposes new form-factor, cost, and steerability requirements which are fundamentally different w.r.t. traditional FSO use-cases. To this end, we establish the viability for a small-form factor FSO design using commodity optical devices. We demonstrate two promising *steering* technologies using switchable mirrors [4] and Galvo mirrors [2].
- **Network provisioning (§4):** Given the budget and physical constraints, the FireFly network hardware must be provisioned to handle unforeseen traffic patterns. We argue that flexible network designs should strive to optimize a new notion of *dynamic bisection bandwidth*. While it is hard to analytically reason about topologies that optimize this metric, we show that random regular graphs are surprisingly good in practice.
- **Network management (§5,§6):** The FireFly controller needs fast and efficient topology selection and traffic engineering algorithms to adapt to changing traffic conditions. However, state-of-art off-the-shelf solvers fail to scale beyond 32-rack DCs. Thus, we develop fast heuristics that achieve near-optimal performance (§5). In addition, the FireFly controller must ensure that performance is not adversely impacted during reconfigurations. We design simple but effective mechanisms that ensure that the network always remains connected, there are no black holes, and that the per-packet latency is bounded.

We evaluate our FSO prototype using a range of controlled lab experiments and a longitudinal study in a real DC setting. We find that the links are robust to real environmental disturbances and achieve wireline-equivalent throughput, and the steering mechanisms are fast, precise, and accurate. We evaluate the end-to-end performance of FireFly using a combination of detailed packet-level simulations [3], large scale flow-level simulations, and virtual emulation platforms [7]. We compare FireFly against state-of-art augmented designs and overprovisioned DC architectures. Overall, we find that FireFly can achieve performance close to a full bisection bandwidth network but at 40-60% of the cost.

We acknowledge that there are several operational concerns, especially related to maintenance, reliability, and reluctance of operators to adopt dynamic network designs. (We discuss some of these in §9). We also note that other cost factors may dominate deployment considerations; e.g., server vs. network vs. management costs. In spite of these valid concerns, we believe that there is value in exploring an FSO-based all-wireless design since it could lead to unprecedented benefits. In particular, pursuing such a vision could open up other avenues amenable to more incremental adoption (e.g., hybrid/augmented networks) and inspire other novel use cases for FSO in DCs.

2 Motivation and Overview

We begin with motivating key aspects of our vision: *full flexibility*, *wireless links*, and use of *free-space optics*.

2.1 Case for Full Flexibility

A key intuition behind FireFly's design is that a *fully-flexible inter-rack network can yield near-optimal bisection bandwidth even without any core (non-ToR) switches.*

To provide the basis for this intuition, we consider an abstract model of a DC network with n racks (and hence n ToR switches). We consider two abstract DC designs: (a) *FBB:* a full-bisection bandwidth network, and (b) *Flexible(f):* an architecture with only

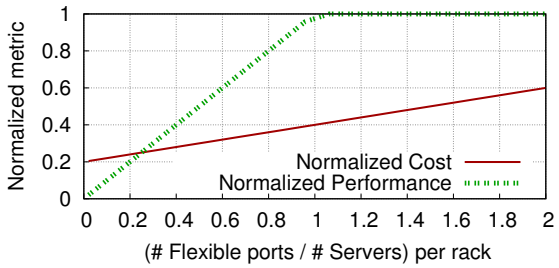


Figure 2: A fully-flexible network can offer optimal (full-bisection) performance at a fraction of the cost. Here, the *normalized* performance and costs are that of the flexible architecture w.r.t. a full-bisection bandwidth network.

ToR switches, each of which has f flexible ports that can be rewired, whenever needed, to connect to another ToR switch. (The flexible ports are in addition to the ports connected to the servers.)

The performance metric of interest here is the *evacuation time* of satisfying the demands in a given inter-rack traffic matrix. Computing evacuation time for FBB is straightforward as there is no congestion. For Flexible(f) too, we can compute evacuation time by computing a near-optimal sequence of reconfigurations (see Appendix A for details). Note that, for a given traffic matrix, Flexible(f) can reconfigure the topology multiple times; we restrict the number of reconfigurations to be at most the number of non-zero entries in the given traffic matrix.

In Figure 2, we plot the normalized performance and cost of Flexible(f) w.r.t. FBB, for varying *ratio* of number of flexible ports and number of servers on each rack. Here, we model the cost as simply a (constant) multiple of the total number of ports in the architecture. The normalized performance is the average ratio of evacuation times of the two architectures, over many random traffic matrices. The key takeaway is that the coreless fully-flexible Flexible(f) architecture yields near-optimal performance when the number of flexible ports (f) is equal to the number of servers per rack, and at this point, its cost is only 40% of FBB (assuming FBB to be a *complete* FatTree [14]). We note that this result is independent of the number of racks or the number of servers per rack.[3]

Of course, any actual realization of Flexible(f) will be less optimal because of limited flexibility, non-zero reconfiguration latency, and other system inefficiencies. We show that our instantiation of Flexible(f) via FireFly results in only a minimal degradation in this cost-performance tradeoff.

2.2 Case for Wireless via Free-Space Optics

To realize a Flexible(f)-like network, conceptually we need a "patch-panel" between racks. Of course, this is infeasible on several fronts: (1) it requires very high fanout and backplane capacity (potentially nullifying the cost benefits), (2) the cabling complexity would be high [38], and (3) it introduces a single-point of failure [16, 22]. Thus, we look toward *reconfigurable wireless* links between the ToR switches.

The seemingly natural solution is RF-based wireless (e.g., 60GHz) [28, 54]. However, since we seek to remove the wired core entirely rather than just augment it, we need a large number of high-bandwidth links to be active simultaneously. This makes the interference management problem much more challenging than prior work (e.g., using interference alignment and cancellation [24]). Alternatively, one could *eliminate* interference by enabling laser-like

[3]We can show (see Appendix A) that the normalized performance of Flexible(f) w.r.t. FBB is $\approx \min(f/l, 1)$ where l is the number of servers per rack.

directionality in RF links, but this would require antennas or antenna arrays that are a few meters wide [40].[4] Fundamentally, the hardness here is due to the large wavelengths of RF for commonly used RF bands. Finally, regulations over RF bandwidth and transmit power further limit achievable data rates.

These limitations can be circumvented if we use a different part of the EM spectrum with much smaller wavelengths. In particular, free-space optics (FSO) is a relatively established technology[5] that uses modulated visible or infrared (IR) laser beams transmitted through free space [32]. Laser beams can be very narrow, thus automatically eliminating interference and minimizing path loss. Further, optical spectrum is unregulated and has no bandwidth limitations. Thus, FSO links can easily offer Gbps–Tbps bitrates at long distances (several kms) using relatively low transmit power [17,32].

2.3 FireFly System Overview

Building on the previous insights, FireFly uses a fully-flexible inter-rack fabric enabled by wireless FSO links (Figure 1). FireFly uses traditional wires for intra-rack connections. We assume an out-of-band control network to configure the ToR switches and the FSO devices (e.g., [8]).

FSO Links. Each ToR is equipped with a number of *steerable* FSO devices. We assume that the FSO devices export APIs to the controller for reconfiguration. We exploit the space above the racks to establish an obstruction-free optical path. To ensure that the FSO devices do not obstruct each other, we use ceiling mirrors [54] as shown in Figure 1. The requirements of such mirrors are quite minimal so long as they reflect IR and many conventional mirrors work sufficiently well (§3). The ceiling mirror need not be a single piece, and small unevenness in the mirrors is unlikely to be an issue due to the misalignment tolerance of the FSO links (§3.1).

Network Provisioning. In the limit, we would like to have a very large number of FSO devices per ToR. In practice, however, there are physical and geometric constraints. For instance, FSO devices will have a finite size that constrains the number of such devices per ToR. Thus, we need to provision or *preconfigure* the network so that it is robust to future (and unforeseen) traffic patterns.

Network Management. The FireFly controller dynamically selects the *runtime topology* and configures forwarding paths, based on prevailing traffic demands and events. Following prior work, we leverage software-defined networking (SDN) capabilities for data plane reconfiguration [13, 19, 39]. Each SDN-capable ToR switch also reports observed traffic demands to the controller. FireFly can use other demand estimation algorithms; e.g., host buffer sizes [13] or new switch features [19]. Since our focus is on FireFly-specific aspects, we do not discuss these extensions.

In the following sections, we describe the design of a viable steerable FSO link, network preconfiguration, and run-time network management.

3 Practical Steerable FSO Design

In order for the FireFly vision to deployed in a DC, the FSO devices must ideally have a small form factor (e.g., so we can pack several devices on each ToR), be low-cost commodity devices (e.g., as we envision thousands of these in a DC), with low power footprint relative to switches, and be steerable to enable flexibility.

At first glance, these requirements are at odds with the trajectory of today's commercial FSO devices—they are bulky, expensive, and power-intensive [5]. The main challenge is that achieving robust links at high data-rates and long ranges (a requirement in both traditional deployments and FireFly) is hard. It has typically required powerful lasers and expensive mechanisms for dynamic alignment for outdoor use. Furthermore, conventional FSO deployments provide fixed point-to-point links and do not focus on steerability.

In this section, we demonstrate (perhaps surprisingly) that (a) it is viable to repurpose commodity DC-centric optical networking gear to establish robust and sufficiently long FSO links in a DC, and (b) we can leverage existing commodity optical technologies to steer the FSO beam with high precision and low latency.

3.1 FSO Link Engineering

As a first step, we demonstrate that it is possible to engineer an FSO optical link using commodity DC-grade optical networking equipment that can achieve high data rates, at ranges sufficient for DC-scale deployment, and with sufficient (mis)alignment tolerance. In particular, we can avoid the additional overheads in commercial FSO devices as concerns about outdoor environmental factors largely disappear in the indoor controlled DC setting. However, we do need to carefully design the *optical path* to balance the tradeoff between the laser beam divergence and misalignment tolerance, as we discuss below.

We engineer the FSO system by coupling two optical fiber end points directly with a free-space link without any opto-electric conversion thus saving on both power and cost. This Fiber–FSO–Fiber link connects to standard optical interconnect technology widely used in DCs (e.g., 10GBASE-SR). A typical example of this interface is optical SFP (small form-factor pluggable) or its variants such as SFP+.

This approach requires optical designs on both ends: (i) on the transmit side, where the fiber 'launches' the laser beam in free space, and (ii) on the receive side, where the laser beam is received into the fiber (Figure 3(a)). Normally, when the laser beam comes out of the fiber into free space it diverges with a significantly large angle. To minimize divergence, we collimate the beam using a suitably designed lens located at its focal length from the transmitting fiber endpoint.[6] A similar lens near the receiving fiber end point focuses the beam back to the fiber.

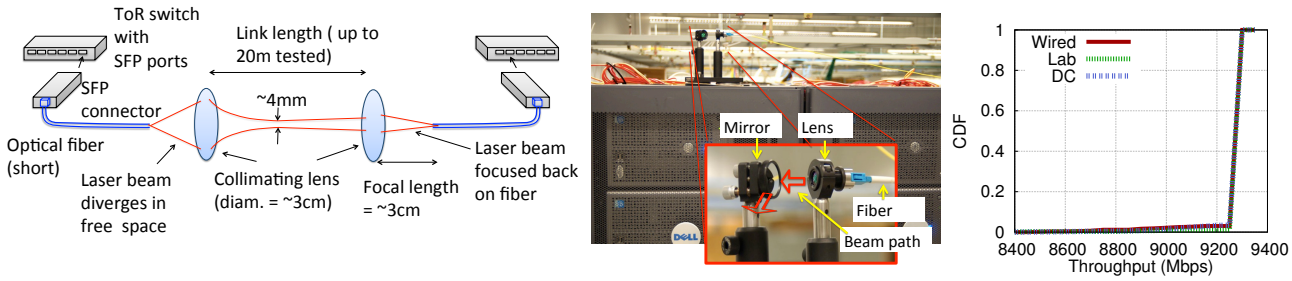
The above optical design is done carefully to ensure that the laser beam maintains a sufficient width [47] so that it can tolerate minor misalignments due to rack vibrations and other effects. However, this presents a tradeoff: wider beams can tolerate misalignments better, but suffer from a poorer power density at the receiving end. The design we develop shows that a good balance is indeed possible using optical SFPs used for long range fiber communications (e.g., 10GBASE-LR can go up to 10 km). They use highly sensitive detectors that can work with very little received power. Our current prototype has been tested for 20m. (20m is a limitation of our lab set up; the general design can extend to 100 m, as discussed in Appendix B.)

This approach satisfies all the design requirements except steering. The lens is small (about 3 cm diameter) with focal length about the same. Even considering additional hardware (e.g., mounts or adapters), the footprint of the assembly is only 5 cm across. The costs are also modest when procured in volume: $\approx$ \$50 for the lens and \$50 for the assembly. We acknowledge that there might be an additional cost of using optical SFP ($\approx$\$100), if optical wired links are not already used. Finally, there is no additional power burden

[4] In effect, we need RF beams with angular divergence $\approx$ 1 milliradian to create a small interference footprint at 50-100m range.

[5] TeraHertz wireless technology may be a good alternative, but it is much less mature at this time.

[6] For brevity, we skip the details of the optical design. We only state the basic approach and relevant tradeoffs. We refer readers to a similar approach that has been used in a different context [53].



(a) FSO link design (b) DC set up (c) TCP throughput

Figure 3: FSO link design, prototype and performance. (a) Optical path and networking gear (not drawn to scale). (b) One end point of the 10 Gbps link prototype running in a DC over $\approx$ 20m); the inset shows a zoomed in version. The FSO link connects to a similar set up on the other end. (c) Distribution of per-sec TCP throughputs on a 10 Gbps FSO link over $\approx$ 20m on optical bench (Lab) and DC racks (DC), over days of continuous runs. Throughput distribution on wired optical fiber is used for comparison.

beyond the SFP power consumption as the design does not add any new opto-electronic conversion.

Prototype. We have developed a proof-of-concept prototype following the above design. We have successfully used both 1 Gbps and 10 Gbps links with very similar optical setups. The prototype links use 1000BASE-LX (1GBASE-LR) SFP (SFP+) for the 1 Gbps (10 Gbps) case and multi-mode fibers for their larger diameter. We first use standard optical bench set up to validate the design using controlled experiments and then test the link in a production DC. Figure 3(a) shows the general setup.[7] For the current design the collimated laser beam maintains a ≈4 mm diameter after it converges. To get up to a 20 m length on a small optical bench, we use a standard technique used in optics: the beam path is reflected multiple times via mirrors. This also validates use of mirrors on the optical path and shows that optical loss due to reflections is low.

Link Performance. We test the link by running continuous TCP transfers over the FSO link for several days at a time for several selected link lengths with the set up on the optical bench. The results are very similar for different lengths. For brevity we only report the results for the longest tested case (≈20m) for the 10 Gbps link. See Figure 3(c). Note that the distribution of TCP throughputs is almost identical to that observed over regular fiber links, demonstrating no additional loss in the FSO links. To study misalignment tolerance, we shift the transmit side set up in tiny incremental steps (using a translating mount) perpendicular to the beam axis keeping the receive side fixed. We see no throughput loss until 6 mm shift, beyond which the link becomes unstable. As we will see below, this 6 mm tolerance is sufficient to handle minor misalignments due to rack vibrations and environmental issues in a DC.

To understand the link performance "in the wild," we set up the link in a production (university run) DC environment. Unlike the optical bench, this real environment has several key differences that can produce mis-alignments: (1) racks experience vibrations due to several factors (e.g., server fans, discs, HVAC and UPS units [48]) and (2) the beam could 'wander' due to fluctuating air density caused by temperature variations. We set up the FSO link with the optical components placed on top of the rack using magnetic bases. Two racks are used at ≈20 m apart. See Figure 3(b). We use mirrors on the beam path - one on each end - for ease of alignment. The reader can view these mirrors as proxies for mirrors to be used in steering (next subsection). Alignment is done manually with the help of an infra-red viewer (more on this in §9). The TCP transfer experiment is run continuously over several days as before. The statistics of per-sec TCP throughput is almost identical again to the optical bench and wired cases (Figure 3(c)). This establishes the potential of our design. The average TCP throughput is $\approx$ 9.3 Gbps — note that no commodity RF technology exists in small-form factor that can deliver such throughput at 20 m range.

The above design provides a basis for a low-cost, commoditizable, and small-form factor FSO link at high data rates over ranges sufficient for DC scale. Our experiments suggest that the link is likely robust to realistic (mis)alignment concerns due to environmental effects in DCs.

3.2 Developing Steering Mechanisms

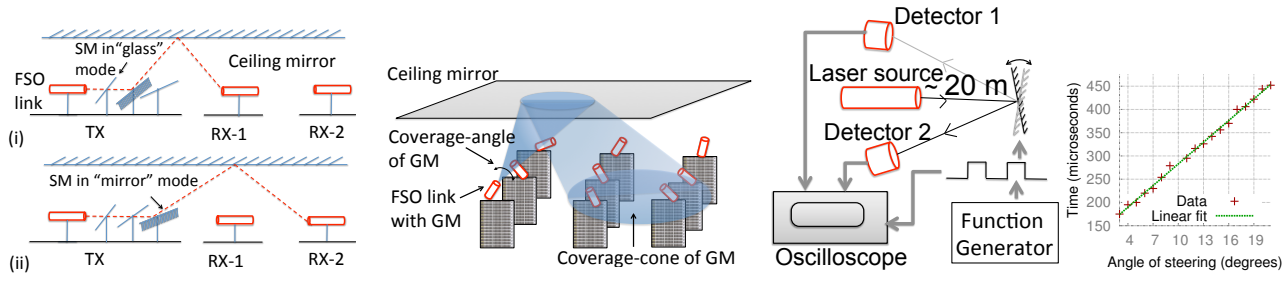
Having established the viability of a point-to-point FSO link using commodity optics, next we focus on making the beam steerable to enable flexibility. Our goal is to establish a design roadmap that is *commoditizable*. We explore two promising solutions: switchable mirrors and Galvo mirrors. We do not claim these are optimal in any sense or that these are the only alternatives. Our choice is pragmatic in that we want to establish a feasible roadmap using off-the-shelf components. (There are a variety of other beam steering approaches [36], but they are not off-the-shelf technologies.) Both solutions offer different tradeoffs w.r.t. latency, degree of flexibility, and cost/power, and at this time, no one solution is strictly better. Thus, we believe it is instructive to understand and evaluate the promise of both alternatives and the tradeoffs they offer.

Switchable Mirrors (SMs). Switchable mirrors (SM) are made from a special liquid crystal material that can be electrically controlled to rapidly switch between reflection (mirror) and transparent (glass) states at millisecond timescales [4]. While the intended use cases are different (e.g., rear-view mirrors that switch between a regular mirror and a back-up camera display), we can use them for beam steering as shown Figure 4(a).

Each FSO device has multiple SMs, with each SM aligned (during a pre-configuration step as discussed in §4) to target a point on a ceiling mirror and thus, a receiving FSO. The link is established by switching one of the SMs to the mirror state, while leaving the rest in the transparent state. (This is done at both ends, but we only show transmit side for clarity.)

SMs directly satisfy our design requirements. It can be miniaturized as it only needs to be slightly larger than the beam diameter: 1 cm^2 is sufficient. A SM of this size is expected to have low cost ($<$ \$5) at volume [34]. Power consumption is also low: only 40 mW for the stated size [4]. We have evaluated the reconfigu-

[7]Note that typical optical SFPs require two fibers and thus two optical paths for duplex communication. However, single fiber SFPs [10] that use WDM principles to multiplex both links on a single fiber are beginning to be available. For 1 Gbps experiments we use such SFPs. Due to unavailability of such devices for 10 Gbps, we use fiber for the return path for the 10 Gbps experiments; all reported performance measurements are from the FSO path.



(a) Switchable mirror operation (b) Galvo mirror operation Experimental set up Response time

(c) Galvo mirror response

Figure 4: (a) Using switchable mirrors (SMs) with FSOs: (i) 2nd SM is in mirror mode to steer the beam to RX-1; (ii) 3rd SM is in mirror mode, steering the beam to RX-2. RX-s also have aligned SMs to direct the beam to its detector (not shown for clarity). (b) Galvo mirror (GM) on an FSO device can steer the beam within its coverage-cone. (c) Evaluating Galvo mirror response.

ration latency using an off-the-shelf 12" x 15" SM [4], and it is ≈250 msec. The switching latency decreases with the decrease in the surface area, and is estimated [34] to be ≈10-20 msec latency for the 1 cm^2 SM size we envision.

Galvo Mirrors (GMs). Galvo mirrors (GMs) [2] are typically used in laser scanning applications. Here, a small mirror, few mm across, rotates (up to specific angular limits) around an axis on the plane of the mirror in response to an electrical signal. The laser beam is made to reflect from this mirror. The mirror rotation deflects the reflected beam by a specified angle depending on the signal. Using a pair of such mirrors at right angles, we can steer the beam within a desired rectangular cone. In our context, equipping an FSO device with a GM enables us to target any receiver within a pre-configured rectangular cone chosen offline. See Figure 4(b).

As proof of concept, we evaluate the response parameters of GMs using an off-the-shelf GM [11] using the setup shown in Figure 4(b). The mirror rotation is controlled programmatically changing the applied voltage. Here, two detectors receive the reflected beam from the mirror alternately as the mirror is fed by a square wave (100 Hz) from a function generator. We measure the time between the instant the voltage trigger is initiated (via the square wave generator) and the time the mirror settles to its new position. Figure 4(c) shows that the steering latency is linear w.r.t. the steering angle and $\leq$ 0.5 ms even for angles up to about $\pm 20^\circ$. We measured the pointing error to be $\leq$ 10 μrad, which translates into ≈1 mm positioning error at 100 m, which is well within the 6 mm tolerance of the FSO link.

The GM is inexpensive (≈$100) and small (few inches across). But, off-the-shelf GMs have a somewhat higher average power consumption (7 W measured) due to the use of an electro-mechanical system. That said, MEMS-based scanning mirrors that provide the same functionality as GMs are already being commoditized [6] and can reduce the power to a few milliWatts.

3.3 Design Summary

In summary, the device roadmap we outlined will have: (1) a rough form factor of 3"x6"; (2) a range of ≈100m and a misalignment tolerance of 6mm; (3) a power footprint of 3W (most of this is in SFP, assuming MEMS-based GMs); and (4) an estimated per-port cost of $300 ($100 for the SFP and $200 for the FSO+steering when produced in volume). The two steering mechanisms impose different constraints and tradeoffs for the FireFly network design (discussed in §4). In particular, having k SMs at an FSO can switch the FSO beam between a set of k arbitrarily chosen but pre-aligned receivers, while a GM on an FSO can steer the beam to any receiver within the coverage-cone that the GM has been pre-oriented to target.

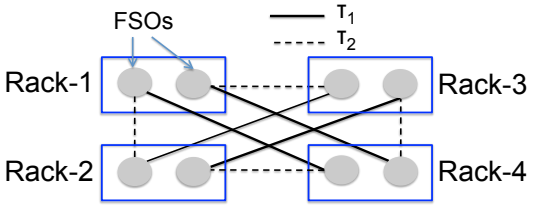


Figure 5: A PCFT with candidate links (solid and dashed). The set of solid links represents one possible realizable topology (τ_1), and the set of dashed lines represents another (τ_2).

4 Network Preconfiguration

At a high level, there are two network design problems in FireFly that occur at different timescales:

- First, we need to provision the network hardware (e.g., how many FSOs) and also pre-align/pre-orient the SMs/GM at each FSO. This is done offline in a *pre-configuration* phase.
- Second, given a pre-configured network, we need to *reconfigure* the network in near real-time to implement a *runtime* topology suited for the current traffic.

We address the first problem of preconfiguration in this section, and defer the second problem to the next section. Ideally, we want to create a dense preconfigured network by placing a large number of FSO devices on each rack, with each FSO device equipped with several SMs or high-coverage GMs. In practice, we have physical limitations, e.g., the size/cost of the FSO devices, size of SM, angle of GMs etc. Our goal is to design a high performance DC network working within these size and cost constraints.

4.1 Preliminaries and Objective

Consider a FireFly network, i.e., a set of FSOs on each rack with pre-aligned SMs or pre-oriented GMs. We can establish a *candidate (bi-directional) link* between a pair of FSOs a and b if (i) a has an SM aligned towards b and vice-versa or (ii) a is located in the coverage-cone of the GM at b and vice-versa. At any instant, only one candidate link per FSO can be an *active link*. For example, in Figure 4(a), links (TX, RX-1) and (TX, RX-2) are candidate links, and link (TX, RX-1) is active in Figure 4(a)(i) while (TX, RX-2) is active in Figure 4(a)(ii). We refer to the set of all candidate links as the *pre-configured flexible topology (PCFT)*. Given a PCFT, we refer to a set of candidate links that can be active simultaneously as a *realizable topology*. Note that the only constraint on a set of links to be active simultaneously is that each FSO has at most one active candidate link incident on it, due to lack of wireless interference. Thus, any realizable topology is a matching in the PCFT graph over FSOs (Figure 5) and vice-versa.

Metric of Goodness. If we knew the expected set of traffic demands, then we can design a customized PCFT that is optimal for this set. However, DC workloads are variable and unpredictable [25]. Thus, we want a metric to quantify the performance of a PCFT that is analogous to the traditional notion of high bisection bandwidth that captures the throughput of a network for arbitrary traffic patterns [33]. More formally, given a topology t and considering all possible partitions P of t into two equi-sized sets of racks, the *bisection bandwidth* is defined as $\min_{p \in P} \mathrm{BW}(t, p)$, where $BW(t, p)$ is the cut-size in t corresponding to p. However, bisection bandwidth only applies to a *static* topology, and is not meaningful for a flexible network. With flexible designs such as FireFly, the topology t itself can be changed on demand, which the bisection bandwidth metric fails to capture.

We introduce a new notion of *dynamic bisection bandwidth (DBW)* as the metric of goodness to evaluate a PCFT. The dynamic bisection bandwidth of a PCFT Π is defined as follows. Let T be the set of realizable topologies of a given PCFT Π. Then, the *dynamic bisection bandwidth* (DBW) for a PCFT Π is defined as: $\min_{p \in P} \max_{t \in T} \mathrm{BW}(t, p)$. Note that this reflects the ability to choose the best realizable topology t for each given partition p.

To illustrate this, consider the PCFT in Figure 5 again. If we consider τ_1 (solid lines) as a static topology, its bisection bandwidth is zero due to the partition {(2,3), (1,4)} of racks. Similarly, we can see that the bisection bandwidth of τ_2 (dashed lines) is 2. However, the DBW of the overall PCFT is 4, since τ_1 yields a bandwidth of 4 for all equi-partitions except for {(2,3), (1,4)}, for which τ_2 yields a bandwidth of 4.

Constrained Optimization. Our goal is to design a PCFT that operates within the given cost and physical constraints and optimizes the DBW. For clarity, we focus on the SM and GM problems independently and defer hybrid SM-GM combinations for future work. In each case, we solve the overall budgeted PCFT selection problem in two steps. First, we develop techniques to design a PCFT with maximum DBW for a fixed configuration (i.e., fixing #FSOs, coverage angle, and #SMs per FSO). Then, given the price/size constraints, we exhaustively search the space of *feasible* combinations of these network parameters and pick a feasible PCFT with the highest DBW. Since preconfiguration runs offline, this brute force step is reasonable.

4.2 SM-PCFT Design Problem

Problem Formulation. Given the number of racks n, number of FSOs m per rack, and the number of SMs k per FSO, the SM-PCFT problem is determine the *alignments* of each SM such that the resulting PCFT has maximum dynamic bisection bandwidth.

Said differently, we want a PCFT with maximum DBW, under the constraint that the number of candidate links at each FSO is at most k. From this view, the SM-PCFT problem falls in the class of network design problems [27], but is different from prior work due to the novel DBW objective. For instance, even the special case of $k = 1$, the SM-PCFT problem reduces to constructing an m-regular graph over n nodes with maximum (static) bisection bandwidth. Even this simple case is harder than the well-studied problem of determining an upper-bound on the bisection bandwidth of m-regular graphs of size n, for which approximate results are known only for very small values of m and n [37]

Random Graphs for SM-PCFT. One promising approach to constructing a SM-PCFT solution is to consider random regular graphs. This is based on the intuition that graphs with (static) bisection bandwidth are likely to have high DBW. (Because random graphs have near-optimal spectral gap [23], they are good "expanders" and have high static bisection bandwidth.) We can construct an n-node regular graph of degree mk, and then group the mk edges on each node into m sets of k edges each (corresponding to each of the m FSOs). For every edge connecting a pair of FSOs (a, b), we align one SM each of a and b towards each other. Because of the randomness, there is a small chance of some random instance performing poorly; thus, we generate many different solutions, and pick the one with the best DBW.[8]

4.3 GM-PCFT Design Problem

Problem Formulation. Given the DC layout, the number of racks n, number of FSOs per rack m, and uniform coverage-angle (see Figure 4(b)) of GMs, the GM-PCFT problem is to determine the *orientation* of the GM on each FSO such that the resulting PCFT has the maximum dynamic bisection bandwidth.

Note that we cannot directly use a random graph as a GM-PCFT solution, since an FSO a's neighbors in a PCFT must be colocated in a coverage-cone of the GM at a. Thus, this problem imposes certain geometric constraints. In particular, for a pair (a, b) to form a (bi-directional) candidate link in the resulting PCFT, the coverage-cone of GM at a must cover b and vice-versa. A naive approach is to iteratively pick a pair of FSOs (a, b) at a time and orient their GMs towards each other. However, this approach may create only one candidate link per FSO/GM, and hence, could result in a sparse PCFT with poor DBW.

Block-based Heuristic. To address the shortcomings of the above strawman approach, we use a "block"-based approach. The intuition here is to create a random graph at a coarser *block* granularity, where each block is a group of nearby FSOs that fall within a GM's coverage cone.

The approach runs in m iterations, and in each iteration we fix the orientation of the GM on the i^{th} FSO of each rack, as described below. (The numbering of FSOs is arbitrary; we just need some ordering.) In each iteration, we randomly partition the set of racks into disjoint *blocks*. The only requirement here is that each block of racks is colocated and small enough to be covered by a GM (when oriented appropriately) on any FSO in the DC. That is, for each block B and FSO $a \notin B$, there exists an orientation of GM at a such that all racks in B fall within its coverage cone. At first glance, this partitioning requirement may seem complex, but we observe that a simple grid-based partitioning scheme is sufficient in practice. Next, we create a random block-level matching M_i over the blocks. Now, for each edge $(B_1, B_2) \in M_i$, we orient the GM on each i-FSO in each rack within block B_1 (correspondingly B_2) towards B_2 (B_1). By construction, the partitioning algorithm guarantees that a GM on any i-FSO in B_1 can cover (with some orientation) all i-FSOs on racks in B_2.

The partitioning in each iteration i can be different. In particular, we can create random partitioning schemes: starting from the basic grid, we can do a random offset to create a new partitioning scheme. Finally, as in the case of SM-PCFT, we generate many randomized GM-PCFT solutions, and pick the best.

5 Real-time Reconfiguration

We consider two types of reconfigurations in FireFly: (1) *periodically* optimizing the network based on estimated demands; and (2) *triggered* by certain network events (e.g., planned migrations or elephant flows).

[8] One subtle issue is even computing DBW is hard. To estimate the DBW for a given random instance, we extend the Kernighan-Lin [33] heuristic for estimating the bisection bandwidth. Our experiments suggest this is within 5-7% of the true DBW. Due to space constraints, we do not discuss the DBW estimation in depth.

$$\max \sum_{i,j} T_{i,j},\ subject\ to: \quad (1)$$

$$\forall b: \sum_{a\ s.t\ (a,b)\in\kappa} l_{a,b} \leq 1;\ \forall a: \sum_{b\ s.t.\ (a,b)\in\kappa} l_{a,b} \leq 1 \quad (2)$$

$$\forall a,b: \sum_{i,j} f_{a,b}^{i,j} \leq l_{ab} \times C \times E \quad (3)$$

$$\forall i,j,k: \sum_{a} \sum_{b\in FSOs(k)} f_{a,b}^{i,j} = \sum_{b\in FSOs(k)} \sum_{d} f_{b,d}^{i,j} \quad (4)$$

$$\forall i,j: \sum_{a\in FSOs(i)} \sum_{b} f_{a,b}^{i,j} = \sum_{a} \sum_{b\in FSOs(j)} f_{a,b}^{i,j} = T_{i,j} \quad (5)$$

$$\forall i,j: T_{i,j} \leq D_{i,j} \quad (6)$$

$$\forall (a,b)\in\kappa: l_{a,b}\in\{0,1\};\ \forall i,j,a,b: f_{a,b}^{i,j} \geq 0 \quad (7)$$

Figure 6: ILP formulation for periodic reconfiguration.

5.1 Periodic Reconfiguration

Given a PCFT and the prevailing traffic load, the *periodic reconfiguration problem* is to optimally select a runtime (i.e., some realizable) topology and set up routes for the current traffic flows.

This constrained optimization is captured by the integer linear program (ILP) shown in Figure 6.[9] Let κ be the set of candidate links in the PCFT, C be the (uniform) link capacity, E be the given epoch size (say a few seconds), and $D_{i,j}$ be the estimated traffic demand (volume) between a pair of racks (i,j). This demand can be obtained by using the measurement counters from the SDN switches from previous epoch(s). We use the subscripts a, b, c, d to refer to FSOs, and i,j,k to refer to racks, and $FSOs(k)$ to denote the set of FSOs on the top of rack k.

There are two key sets of control variables: (i) The binary variable $l_{a,b}$ models *topology selection* and is 1 iff a candidate link (a,b) is chosen to be active; and (ii) $f_{a,b}^{i,j}$ captures the *traffic engineering* (TE) strategy in terms of the volume of inter-rack traffic between i and j routed over the link (a,b). Let $T_{i,j}$ be the total traffic volume satisfied for the flow (i,j).

For clarity, we consider a simple objective function that maximizes the total demand satisfied across all rack pairs as shown in Eq (1). Eq (2) ensures that each FSO can have at most 1 active link. Eq (3) ensures that the total flow on each link (on average) does not exceed the capacity. Eq (4) are flow conservation constraints for each flow (i,j) and a rack k. Eq (5) captures the volume of the demand satisfied using a constraint over the ingress and egress racks. Eq (6) ensures that the volume satisfied for each rack pair is at most the demand. Finally, we have bounds on the control variables.

Unfortunately, even state-of-art solvers like `Gurobi` or `CPLEX` take several hours to solve this ILP (§8.4). Hence, we follow a heuristic strategy and decouple the optimization into two stages. First, we solve the "integer" problem to select the active links. Then, given this runtime topology, we compute the flow routes.

Greedy Matching for Topology Selection. Recall from §4 that a realizable topology is essentially a matching over FSOs in the PCFT graph. Thus, a simple starting point is to select the maximum-weighted matching, where each candidate link (a,b) is weighted by the inter-rack traffic demand $D_{i,j}$ between the racks i and j. In effect, this maximizes the total demand that can be served using direct links. However, this can be very inefficient if the given PCFT does not have direct links between racks with high traffic demands.

[9]This problem is harder than the optimization problem considered in §2.1, since we were assuming arbitrary flexibility.

The high-level idea behind our heuristic is to extend the traditional Blossom algorithm for computing the maximum matching to incorporate multi-hop traffic. Recall that the Blossom algorithm improves the matching by computing an alternating path at each stage. We define a new utility function that captures multi-hop traffic and then pick the path with the highest benefit. Specifically, we use the intuition that shorter inter-rack paths imply lower resource usage and higher network throughput [46]. Thus, we define the *benefit* of an alternating path L as the decrease in the weighted-sum of inter-rack distances if L were used to modify the current matching. Formally, given a current matching τ, the benefit of an alternating path L that would modify the matching from τ to τ', is the total reduction in the network footprint: $\sum_{ij} D_{i,j}(h_{i,j} - h'_{i,j})$, where $h_{i,j}$ and $h'_{i,j}$ are the inter-rack distances between racks (i,j) in τ and τ' respectively (when seen as graphs over racks).

We run this extended Blossom algorithm until there is no alternating path that can improve the network footprint and then output the final topology at this stage.

Flow Routing. Given a specific runtime topology (i.e., values of $l_{a,b}$), the residual TE problem is theoretically solvable in polynomial time as a multi-commodity flow (MCF) problem. However, even this takes hundreds of seconds for 256/512 racks §8.4), which is not acceptable as a near real-time solution. To address this, we use a greedy algorithm to compute the values of these flow variables. Essentially, we extend the traditional augmenting-path approach for max-flow algorithms and greedily pick an augmenting path for a currently unsatisfied commodity. We run the algorithm until no more augmenting paths can be picked; i.e., the network is saturated. From this solution, we use the "path stripping" idea [43] to convert the values of the $f_{a,b}^{i,j}$ variables into end-to-end paths.

5.2 Triggered Reconfigurations

In addition to periodically reconfiguring the network, FireFly can also run localized reconfigurations triggered by traffic events. Such reconfigurations may be frequent but likely require minimal topology and flow-route changes. We currently support two types of triggers. First, if we detect elephant flows that have sent more than 10 MB of aggregate data [19], then we activate links to create a shorter or less-congested path for this flow. Second, if traffic between a particular pair of racks exceeds some configurable threshold, then we create a direct link between them, if this does not require deactivating recently activated or high utilization links.

6 Correctness During Reconfigurations

A reconfiguration in FireFly entails: (i) addition and/or deletion of (candidate) links from the given runtime topology, and (ii) corresponding changes to the network forwarding tables (NFTs). This flux raises natural concerns about correctness and performance during reconfigurations. Our goal is to ensure that: (i) network remains connected at all times, (ii) there are no black holes (e.g., all forwarding table entries refer to available/usable links), and (iii) packet latency remains bounded (and thus, delivery is guaranteed).

The main challenges in designing a correct data plane strategy stem from two factors: (i) Activation or deactivation of candidate links incur a non-zero latency (few msecs); and (ii) We may need to execute reconfigurations concurrently if the triggers occur frequently (e.g., for every elephant flow arrival). At a high level, these are related to the problem of consistent updates [35, 44]. The key difference is that we can engineer simpler requirement-specific solutions rather than use more general-purpose solutions proposed in prior work.

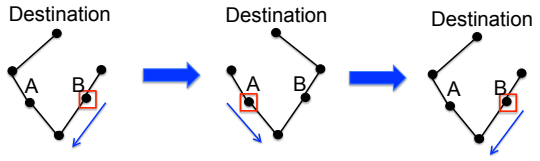


Figure 7: Packet (in-flight location shown by a square) continues to "swing" from B to A and back, due to a rapid sequence of reconfigurations.

6.1 Handling sequential reconfigurations

We begin by focusing on correctness when we execute reconfigurations serially and defer concurrent execution to the next subsection.

6.1.1 Avoiding Black Holes

To see why "black holes" may arise, consider a reconfiguration that changes the network's runtime topology by steering FSOs a and b towards each other, and in the process activating the link (a, b) and deactivating some link (a, c). Suppose the NFTs change from $\mathcal{F}$ to $\mathcal{F}'$. Now, there is a period of time (when GM/SMs at a is changing state) during which neither (a, b) nor (a, c) is available. During this period, irrespective of when the NFTs get updated (say, even atomically) from $\mathcal{F}$ to $\mathcal{F}'$, some entries in the NFTs may refer to either (a, b) or (a, c), inducing black holes in the network.

Our Solution. To avoid black holes, we split a reconfiguration into multiple steps such that: (i) link deletion is reflected in the NFTs *before* their deactivation is initiated, and (ii) link addition is reflected only *after* the activation is complete. Thus, a reconfiguration that involves deactivation (activation) of a set of links $\bigtriangledown$ ($\bigtriangleup$) is translated to the following sequence of steps:

S1: Update the NFTs to reflect deletion of $\bigtriangledown$.
S2: Deactivate $\bigtriangledown$ and activate $\bigtriangleup$.
S3: Update the NFTs to reflect addition of links $\bigtriangleup$.

One additional invariant we maintain is that every switch has a default low priority rule at all times to reach every destination rack via *some* active outgoing link. We do so to explicitly ensure that packets can reach their destination, possibly on sub-optimal paths, as long as the network is connected (see below).

6.1.2 Maintaining Connectivity

To ensure network connectivity at all times, we simply reject reconfigurations that might result in a disconnected network in step **S1** above. That is, we add a step S0 before the three steps above.

S0: Reject the reconfiguration, if deletion of links $\bigtriangledown$ disconnects the network.

To reduce the chance of such rejections, we also extend our reconfiguration algorithms to retain a connected subnetwork from the prior topology. The high-level idea here is to construct a rack-level spanning tree using the current graph, and explicitly remove these links/FSOs from consideration during the greedy matching step.

6.1.3 Bounded Packet Latency

If reconfigurations occur at a very high frequency, then we may see unbounded packet latency. Figure 7 shows a simple example where a packet can never reach its destination because the links/routes are being reconfigured quite rapidly.

Our Solution. The example also suggests a natural strategy to avoid such cases—we can delay or reject reconfigurations to allow the in-flight packets to use one of the intermediate topologies to reach its destination. We introduce a small delay of x units between two consecutive NFTs-updates, where x is the maximum packet latency in a fixed realizable topology. This ensures that each packet "sees" at most two configurations during its entire flight. This, bounds the packet latency by $(2x + z)$ where z is the total NFTs-update time.

6.2 Handling Concurrent Reconfigurations

Computing and executing a reconfiguration can take a few tens of msecs in a large DC. To achieve good performance, we may need to reconfigure the network frequently; e.g. for every elephant flow arrival in the network, which may happen every msec or less. Thus, we need mechanisms that allow reconfigurations to be executed concurrently. We could batch reconfigurations, but that merely delays the problem rather than fundamentally solving it because a batch may not finish before the next set of reconfigurations arrive.

We observe that to handle concurrent reconfigurations, we need to extend the approach from §6.1 to handle two concerns.

- *Connectivity:* One concern is that each reconfiguration in isolation may not disconnect the network but combining them might. Thus, to ensure network connectivity, the controller maintains an atomic global topology variable $\mathcal{G}$, and uses this variable to accept/reject in step S0. ($\mathcal{G}$ is also updated by accepted reconfigurations in S1 and S3.)
- *Conflicting reconfigurations:* In step S0, we also reject any reconfiguration that "conflicts" (in terms of link activations or deactivations) with already-accepted but yet-unfinished reconfigurations. That is, we follow a non pre-emptive strategy of allowing outstanding reconfigurations to complete.

We note that no other changes are required to §6.1 to handle concurrency. Black holes are still avoided since only non-conflicting reconfigurations are executed concurrently and packet latency is bounded since a minimum time-interval already precludes concurrent processing of different NFTs-updates.

6.3 Overall Scheme

Based on the previous building blocks, our overall scheme is as follows. Each reconfiguration ρ that deletes and adds a set of links $\bigtriangledown$ and $\bigtriangleup$, is translated into the following four steps. Here, $\mathcal{G}$ is as described in §6.2.

C0: Accept ρ if (i) deletion of links $\bigtriangledown$ does not disconnect $\mathcal{G}$, and (ii) ρ doesn't conflict with any unfinished reconfigurations.
C1: Update $\mathcal{G}$ and NFTs to reflect deletion of $\bigtriangledown$.
C2: Deactivate $\bigtriangledown$ and activate $\bigtriangleup$.
C3: Update $\mathcal{G}$ and NFTs to reflect addition of links $\bigtriangleup$.

In addition, as discussed in §6.1, we ensure (a) default path rules, and (b) a minimum time-interval (= maximum packet latency) units between consecutive NFTs-updates.

We can analytically prove that the above overall scheme ensures that (i) there are no black holes, (ii) network remains connected, and (iii) packet latency is bounded by $(2x + z)$, where z is the NFTs-update time. This claim holds irrespective of how the NFTs are updated across the network; i.e., we do not require atomic updates. See Appendix C for the proof.

7 FireFly Controller Implementation

We implement the FireFly controller as modules atop the POX controller. We chose POX/OpenFlow primarily for ease of prototyping. We use custom C++ modules for the PCFT generation and reconfiguration optimization algorithms. For reconfiguration, we implement heuristics to "dampen" reconfigurations by checking if there is a significant (e.g., more than 10%) improvement in the objective function from Figure 6. We use a simple translation logic to convert the output of the optimization solver to OpenFlow rules where the

"flow" variables are mapped to prefix-range based forwarding entries [51]. For elephant flows, we set up exact match flow rules [13]. We use existing OpenFlow measurement capabilities to estimate the inter-rack demands and use the observed traffic from the previous epoch as the input to the controller. Our prototype does not implement elephant flow detection [13, 19]; we currently assume this information is available out of band.

8 Evaluation

We established the performance of individual steerable FSO links in §3. In this section, we focus on:

1. Performance w.r.t. other DC architectures (§8.1);
2. Impact on performance during reconfigurations (§8.2);
3. Optimality of the preconfiguration algorithms (§8.3);
4. Optimality and scalability of reconfiguration (§8.4);
5. Sensitivity analysis w.r.t. degree of flexibility and reconfiguration latency (§8.5); and
6. Cost comparison w.r.t. prior DC architectures (§8.6).

For (1), we use a combination of detailed packet-level simulations using `htsim` and augment it with larger-scale flow-level simulations using a custom flow-level simulator. For (2), we use a detailed system-level emulation using MiniNet. For (3) and (4), we use offline trace-driven evaluations. For (5), we use the flow-level simulation. Finally, for (6) we use public cost estimates and projections from §3.

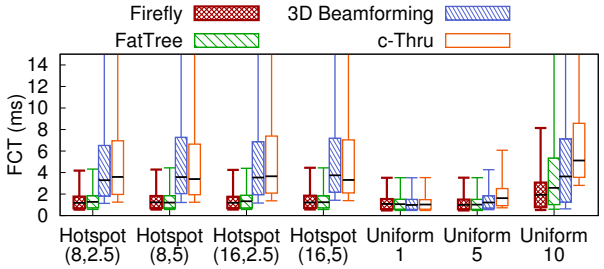
8.1 System Performance

Setup and Workloads. We consider three classes of architectures: (1) FireFly (both SM and GM) with 10Gbps links, (2) (wired) 10Gbps full-bisection bandwidth networks such as FatTree [14], and (3) augmented architectures such as c-Through [50] and 3D-Beamforming (3DB) [54] with a 5Gbps (i.e., 1:2 oversubscribed) core.[10] (We do not compare Flyways [28] since it is subsumed by 3D-Beamforming.) By default, FireFly has 48 FSOs per rack with each equipped with 10 SMs; we assume a rack size of $4' \times 2'$, which is sufficient to easily hold 48 FSO devices (§3.3). We also evaluated Jellyfish [46], but do not show this for ease of visualization since the result was close to FatTree ($\approx$ 10% lower). We assume an overall reconfiguration latency of 20 msecs for FireFly, and conservatively use zero latency for c-Through/3DB. We use ECMP routing for FatTree and backbone cores of 3dB and c-Through, and route the "overflow" traffic to their augmented links [54]. Finally, for FireFly, we use the routing strategies described in §5.

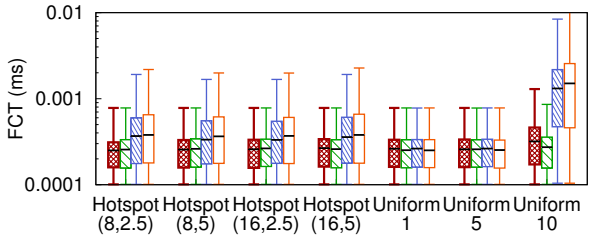
Following prior work, we use synthetic traffic models based on DC measurements [25,54]. As a baseline, we consider a `Uniform` model where flows between pairs of racks arrive independently with a Poisson arrival-rate λ/s, with an empirically-derived flow size distribution [25]. We use λ as the knob to tune the level of network saturation. Based on prior observations, we also consider the `Hotspot` model [25], where in addition to the `Uniform` baseline, a subset of rack pairs have a higher arrival rate λ_2 and a fixed large flow size of 128MB [54]. For `Uniform` loads, we use the label `Uniform X` where X is average load per server (in Gbps) by choosing suitable λ. For `Hotspot` loads, we use the label `Hotspot(Y,X)` where Y is the % of racks that are hotspots and X is the *additional* average load on each hotspot server; all `Hotspot` workloads use `Uniform 5` as the background traffic.

Performance Comparison. There are two key metrics here: (1) the average throughput per server, and (2) flow completion time

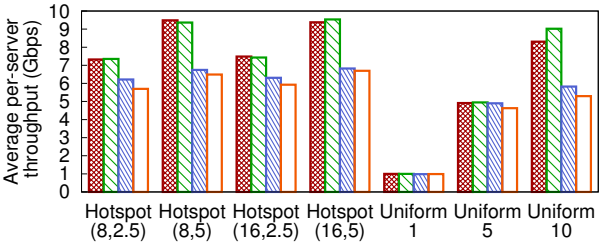
[10]While a lower oversubscription would improve c-Through's performance, the cost increases almost proportionally—eventually becoming equal to FatTree at 1:1 oversubscription.



(a) Flow completion for long flows



(b) Flow completion for short flows



(c) Average throughput per server (Gbps)

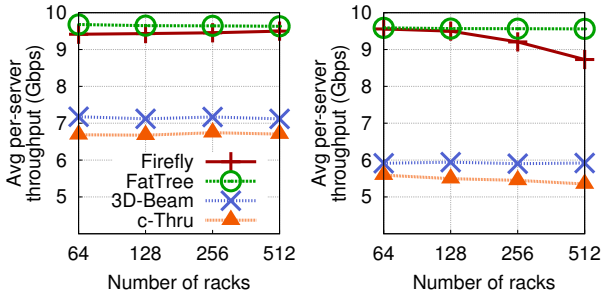
Figure 8: Flow completion times (FCT) and average throughput per-server using the `htsim` simulator on a 64-node topology for different workloads

(FCT). For ease of visualization, we do not show error bars over multiple runs, since the results were consistent across multiple runs. We also do not show FireFly-GM (with 40° coverage-angle GMs) results, since they are similar to the default FireFly-SM.

As a starting point, we use `htsim` for a detailed packet-level simulation. We extended `htsim` to support short flows, arbitrary traffic matrices, and route reconfigurations. Due to scaling limitations of `htsim`, even on a high-end server (2.6GHz, 64 GB RAM), we could only scale to a 64-rack DC at our 10 Gbps workloads. Figure 8(a) and 8(b) show a box-and-whiskers plot (showing maximum, minimum, median, 25%iles, and 75%iles) of the FCT for long/short flows respectively for a 30 secs run. The result shows that FireFly's performance is close to the full-bisection bandwidth network in both cases. c-Through and 3DB do not perform as well because their augmented network is not sufficient to compensate for the oversubscription. Thus, their tail performance for long flows suffers. We also see that the FCT for short flows is similar across FireFly and FatTree.

Figure 8(c) shows the effective average per-server throughput in the 64-rack setup for different workloads. For the `Uniform` the average is over all servers whereas for `Hotspot` the average is over the hotspot servers. In short, we see that FireFly's performance is close to the full-bisection bandwidth network and roughly 1.5$\times$ better than the augmented architectures.

To scale to larger DCs, we use a custom flow-level simulator. We do so after confirming that these simulations roughly match the packet-level simulations. In general, the flow-level simula-



(a) Hotspot(16,5) (b) Uniform10

Figure 9: Scalability evaluation using a flow-level simulator

tions overestimates the throughput 5-7% *for all* architectures since it does not model packet-level effects. Since our goal is to compare the relative performance of these architectures, these simulations are still instructive. Figure 9 shows that the earlier performance results continue to hold for the most saturating workloads even at larger scales. The only drop is at 512 racks for `Uniform10`; here the number of SMs/FSO is slightly sub-optimal as the number of racks grows. We revisit this in §8.5.

We also measured the packet latency (number of hops) statistics and found that the average latency were 3.91 (FireFly), 4.81 (FatTree), and 3.9 (3dB, c-Through), while the maximum was 5 for FireFly and 6 for the rest.

8.2 Performance during Flux

Because packet- or flow-level simulations do not give us a detailed replay of the events at the FireFly controller and in the network, we use `Mininet` for this evaluation [7]. Due to scaling limitations, we scale down the DC size to 32 racks and the link rates to 10 Mbps, and correspondingly scale the workload down. Since our goal is to understand the *relative* impact of reconfigurations w.r.t. the steady state behavior, we believe this setup is representative. In particular, note that the relative "penalty" of the reconfiguration latency remains the same since we also scale down the workload with the link rates.

For the following result, we consider a `HotSpot` workload, with seven distinct reconfigurations as elephant flows arrive. We poll the virtual switches to obtain link utilization and loss rates and use a per-rack-pair ping script to measure inter-rack latency. We divide these measurements into two logical bins: (a) *During reconfigurations* and (b) *Steady state* (i.e., no active reconfiguration). Figure 10 shows the distribution link utilization, loss rate, and inter-rack latency for each bin. While there is a small increase in the tail values, the overall distributions are very close. This suggests that the impact on the network during reconfigurations is quite small and that our mechanisms from §6 work as expected.

8.3 Preconfiguration Efficiency

#Racks	Normalized DBW w.r.t. upper bound	
	SM-PCFT	GM-PCFT
64	0.96	0.84
128	0.93	0.84
256	0.91	0.85
512	0.94	0.88

Table 1: Efficiency of the PCFT algorithms

As before, we use 48 FSOs per rack and 10 SMs per FSO for SM-PCFT, and assume GMs with an coverage-angle of 40° for GM-PCFT. We generate $\approx 15n$ (n is the number of racks) random instances and pick the best. We normalize the estimated DBW w.r.t an upper bound of $\frac{nm}{2}$.[11] Table 1 shows that the SM-PCFT and GM-PCFT solutions achieve $\geq 91\%$ and $\geq 84\%$ of the upper bound across different DC sizes. The lower performance of GM-PCFT is likely because of less randomness due to a block-level construction. (This does not however impact the performance of the runtime topology in practice for the workloads we consider.)

We also evaluate an incremental expansion scenario where we want to retain most existing PCFT as we add new racks similar to Jellyfish [46]. We find that incrementally constructed PCFTs perform nearly identical w.r.t. a PCFT computed from scratch (not shown). We posit that this stems from the incremental expandability of random graphs [46].

8.4 Reconfiguration Efficiency

# Racks	Time (ms)			Optimality Gap
	Full-LP	Greedy-LP	FireFly two-step	(%)
32	138	110	27	2.8%
128	4945	208	54	2.6%
256	1.7×10^6	3.3×10^4	60	1.3%
512	6.4×10^8	1.9×10^7	68	2.8%

Table 2: Scalability and optimality of the FireFly reconfiguration algorithm.

Table 2 shows the computation time and optimality gap of the FireFly two-step heuristic from §5.1. We consider two points of comparison: (a) *Full-LP*, a LP relaxation of Figure 6, which also yields an upper-bound on the optimal, and (b) *Greedy-LP* which uses greedy topology selection but solves the flow routing LP using `Gurobi`. Our approach is several orders of magnitude faster—Full-LP and Greedy-LP simply do not scale for ≥ 32 racks. This is crucial as the FireFly controller may need to periodically reoptimize the network every few seconds. Moreover, the (upper bound) on the optimality gap is $\leq 2.8\%$. Finally, we note that triggered reconfigurations (§5.2) incur only 5-10 msec (not shown). Most of the time is actually spent in route computation, which can be run in parallel to allow a high rate of concurrent reconfigurations.

8.5 Sensitivity Analysis

	# FSOs per rack			
	36	40	44	48
# SMs per FSO = 10	5.14	6.07	7.47	8.83
# SMs per FSO = 13	6.21	7.27	8.20	9.17
# SMs per FSO = 15	6.45	7.54	8.31	9.25
GM-based network	6.87	7.84	8.66	9.24

Table 3: Average throughput per-server on `Uniform10` for varying network parameters.

Given that we are considering a new technology, we evaluate the sensitivity w.r.t. key parameters: number of FSOs, number of SMs, and the reconfiguration latency. Table 3 shows the average per-server throughput on a range of FireFly instantiations. As before, we use the configuration of 512 racks with 48 servers each. As expected from the insights from §2.1, the performance decreases almost linearly with decrease in the number of FSOs (i.e., flexible ports) per rack. The performance degradation in GM-based networks is comparatively better. Note that the networks with fewer FSOs also have almost-proportionally lower cost. However, increasing the number of SMs per FSO does counter the degradation to some extent, due to increase in flexibility. If FSO size is the limiting concern, one way to get higher performance is to simply

[11] Any equi-sized partition of n racks with m FSOs can have at most $nm/2$ active links (one per FSO) in a "cut".

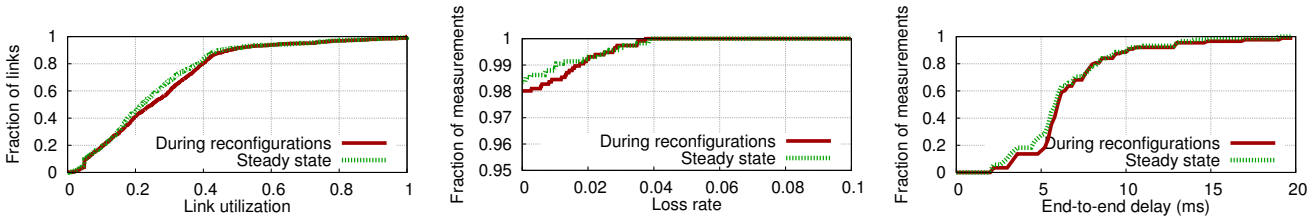


Figure 10: Comparing network performance during reconfigurations and in steady state

	Equip		Cable		Power		Total	
	Fiber	Cu	Fiber	Cu	Fiber	Cu	Fiber	Cu
FatTree	22	15	8	2	1	2	31	20
3DB	13	9	5	2	1	1	19	12
cThru	13	9	5	2	1	1	19	12
FireFly	12		0		1		13	

Table 4: Cost (in USD millions) of equipment, power, and cabling assuming 512 racks with 48 servers/rack. Since these are estimates, we round to the nearest million.

decrease the number of servers per rack; i.e., increase number of racks for a fixed total number of servers (not shown).

Finally, with respect to reconfiguration latency, we observe that varying the latency from 10msec to 50 msec has minimal (< 5%) impact on FireFly's performance (not shown). This is a positive result that we can achieve pretty good performance even with unoptimized steering delays and network update times.

8.6 Cost Comparison

As an instructive case, we consider a DC with 512 racks and 48 servers/rack to compute the equipment, power, and cabling costs of different architectures in Table 4. We consider both copper- and fiber-based realizations for the wired architectures.

We conservatively estimate the "bulk" price of 10GbE network switches to be $100 per port [12, 52] and that of 10GbE SFPs at $50 each; these estimates are roughly 50% of their respective retail costs. Thus, fiber-based architectures (including FireFly) incur a cost of $150 per-port, while copper-based architecture incur a cost of only $100 per-port. FireFly uses a 96-port (10G) ToR switch on each rack with 48 FSOs, the full-bisection FatTree needs 1536 96-port (10G) switches, while the 1:2 oversubscribed cores of c-Through/3DB use roughly half the ports of FatTree. FireFly has an additional cost for FSO devices (on *half* the ports), which we estimate to be $200 per device including SMs or a GM (§3). For 3DB, we assume there are 8 60 GHz radios per rack with each assembly costing $100. For c-Through, we conservatively assume the 512-port optical switch to be $0.5M. We assume ceiling mirrors in FSO and 3DB have negligible cost. For cabling, we use an average cost of $1 and $3 per meter for copper and optical-fiber respectively, and use an average per-link length of 30m [41]. We estimate the 5-yr energy cost using a rate of 6.5cents/KWHr, and per-port power consumption of 3W (fiber) and 6W (copper). We ignore the negligible energy cost of SMs, 60GHz radios, and the optical switches.

Overall, the total cost of FireFly is 40-60% lower than FatTree and is comparable (or better) than the augmented architectures. Note that the augmented architectures have worse performance compared to FireFly, and copper wires have length limitations.

9 Discussion

Given our radical departure from conventional DC designs, we discuss a sampling of potential operational concerns and possible mechanisms to address some of these issues. We acknowledge that there might be other unforeseen concerns that require further investigation over pilot long-term deployments.

Pre- and re-alignment. We will need external tools for pre-aligning SMs/GMs. Fortunately, as this will be done infrequently, this mechanism does not need the stringent cost/size requirements and we can repurpose existing mechanical assemblies. While our design tolerates minor misalignment (§3), long term operation may need occasional alignment corrections. Here, we can use the feedback from the digital optical monitoring support available on optical SFPs; GMs can directly use such feedback, but SMs may need additional micro-positioning mechanisms (e.g., piezoelectrics).

Operational Concerns. Some recurrent concerns we have heard include dust settling on optics, light scattering effects, reliability of mechanical parts, and human factors (e.g., need for protective eyewear for operators). We believe that these are not problematic. Specifically, dust/scattering may reduce the signal but our design has sufficiently high link margins (15dB), and the devices can be easily engineered to minimize dust (e.g., non-interfering lids or blowing filtered air periodically). While we studied an electro-mechanical GM here, future MEMS-based scanning mirrors [6] are expected to be very robust. Finally, the lasers we use are infra-red and very low power which are not harmful to human eye.

Beyond 10 Gbps. Current long-range connector standards for 40/100 Gbps (e.g., 40 or 100GBASE-LR4) use WDM to multiplex lower rate channels on the same fiber, one in each direction. However, just like the 10 GbE standard that we have used, there are still two optical paths (with two fibers) for duplex links. Single-fiber solutions (as we have used for 1 GbE [10]) are not commodity yet at these speeds as the market is still nascent. We expect, however, this to change in future. Otherwise, we will need two optical paths or custom single-path solutions.

10 Related Work

Static Wired Topologies. Early DCs used tree-like structures, which have poor performance due to *oversubscription*. This motivated designs providing full bisection bandwidth [14, 18, 46], which are overprovisioned to handle worst-case patterns. In addition to high cost, such networks are not incrementally expandable [46]. In contrast, FireFly is flexible, eliminates cabling costs, and amenable to incremental expansion. Other efforts proposed architectures where servers act as relay nodes (e.g., [26]). However, they are not cost competitive [41] and raise concerns about isolation and CPU usage.

Optical Architectures. High traffic volumes coupled with the power use of copper-based Ethernet, has motivated the use of optical links. Early works such as c-Through [50] and Helios [22] suggested hybrid electric/optical switch architectures, while recent efforts consider all-optical designs [16,42]. The use of free-space optics avoids the cabling complexity that such optical designs incur. Furthermore, by using multiple FSOs per rack, FireFly can create richer topologies (at the rack level) than simple matchings [16,22,42,50]. Moreover, FireFly doesn't need optical switching, thus eliminating concerns about cost/scalability. Finally, optical switching can disconnect substantial portions of the optical network during reconfiguration. While FireFly also has transient link-off periods, these are localized—which enables us to avoid black holes and disconnections using simpler data plane strategies (§6).

Wireless in DCs. The FireFly vision is inspired by Flyways [28] and 3D-Beamforming [54]. However, RF wireless technology suffers from high interference and range limitations and limits performance. The use of free-space optics in FireFly eliminates interference concerns. Shin et al. consider a static all-wireless (not only inter-rack) DC architecture using 60 Ghz links [45]. However, this requires restructuring DC layout and has poor bisection bandwidth due to interference.

Consistency during Reconfigurations. Recent work identified the issue of consistency during network updates [35, 44]. FireFly introduces unique challenges because the topology changes as well. While these techniques can also apply to FireFly, they are more heavyweight for the specific properties (no black holes, connectivity, and bounded packet latency) than the domain-specific solutions we engineer. Other work minimizes congestion during updates [31]. While FireFly's mechanisms do not explicitly address congestion, our results (§8.2) suggest that this impact is quite small.

11 Conclusions

In this work, we explore the vision of a fully-flexible, all-wireless, coreless DC network architecture. We identified free-space optics as a key enabler for this vision and addressed practical hardware design, network provisioning, network management, and algorithmic challenges to demonstrate the viability of realizing this vision in practice. Our work is by no means the final word. There remains significant room for improvement in various aspects of our design, viz., cost and form-factor of hardware elements, algorithmic techniques for network provisioning and management, which should further improve the cost-performance tradeoff.

Acknowledgments

We thank Yuvraj Agarwal, Daniel Halperin, and our shepherd Jitendra Padhye for their feedback, and Hanfei Chen for GM measurements. This research was supported in part by NSF grants CNS-1117719 and CNS-0831791.

12 References

[1] A Simpler Data Center Fabric Emerges . http://tinyurl.com/kaxpotw.

[2] Galvo mirrors. http://www.thorlabs.us/NewGroupPage9.cfm?ObjectGroup_ID=3770.

[3] htsim simulator. http://nrg.cs.ucl.ac.uk/mptcp/implementation.html.

[4] Kent optronics, inc. http://kentoptronics.com/switchable.html.

[5] Lightpointe flightstrata g optical gigabit link. http://tinyurl.com/k86o2vh.

[6] Mems scanning mirror. http://www.lemoptix.com/technology/mems-scanning-mirrors.

[7] Mininet. http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet.

[8] OpenGear out of band management. http://tinyurl.com/n773hg3.

[9] Renka, single mode fiber specifications. http://www.renka.com/products/SmartLITE%20Cable/DEC00%20SM%20OFC%20RENKA.pdf.

[10] Single-fiber sfp. http://www.championone.net/products/transceivers/sfp/single-fiber-single-wavelength/.

[11] Xinyu laser products. http://www.xinyulaser.com/index.asp.

[12] 10GBASE-T vs. GbE cost comparison. *Emulex white paper*, 2012. Available at http://www.emulex.com/artifacts/cdc1a1d3-5d2d-4ac5-9ed8-5cc4a72bd561/elx_sb_all_10gbaset_cost_comparison.pdf.

[13] M. Al-Fares et al. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, 2010.

[14] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM*, 2008.

[15] S. Bloom, E. Korevaar, J. Schuster, and H. Willebrand. Understanding the performance of free-space optics [invited]. *Journal of optical Networking*, 2(6):178–200, 2003.

[16] K. Chen et al. OSA: An optical switching architecture for data center networks with unprecedented flexibility. In *NSDI*, 2012.

[17] E. Ciaramella et al. 1.28-Tb/s (32×40 Gb/s) free-space optical WDM transmission system. *IEEE Photonics Technology Letters*, 21(16), 2009.

[18] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32, 1953.

[19] A. Curtis et al. DevoFlow: Scaling flow management for high-performance networks. In *ACM SIGCOMM*, 2011.

[20] A. Curtis, S. Keshav, and A. Lopez-Ortiz. LEGUP: Using heterogeneity to reduce the cost of data center network upgrades. In *CoNEXT*, 2010.

[21] H. L. Davidson et al. Data center with free-space optical communications. US Patent 8,301,028, 2012.

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

[23] J. Friedman. On the second eigenvalue and random walks in random d-regular graphs. *Combinatorica*, 11(4), 1991.

[24] S. Gollakota, S. D. Perli, and D. Katabi. Interference alignment and cancellation. In *ACM SIGCOMM*, 2009.

[25] A. Greenberg et al. VL2: A scalable and flexible data center network. In *ACM SIGCOMM*, 2009.

[26] C. Guo et al. BCube: A high performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM*, 2009.

[27] A. Gupta and J. Konemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16, 2011.

[28] D. Halperin et al. Augmenting data center networks with multi-gigabit wireless links. In *ACM SIGCOMM*, 2011.

[29] N. Hamedazimi, H. Gupta, V. Sekar, and S. Das. Patch panels in the sky: A case for free-space optics in data centers. In *ACM HotNets*, 2013.

[30] B. Heller et al. ElasticTree: Saving energy in data center networks. In *NSDI*, 2010.

[31] C.-Y. Hong et al. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM*, 2013.

[32] D. Kedar and S. Arnon. Urban optical wireless communication networks: The main challenges and possible solutions. *IEEE Communications Magazine*, 2004.

[33] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell Systems Technical Journal*, 49(2), 1970.

[34] L. Li. CEO, KentOptronics. Personal communication.

[35] R. Mahajan and R. Wattenhofer. On consistent updates in software defined networks (extended version). In *ACM HotNets*, 2013.

[36] P. F. McManamon et al. A review of phased array steering for narrow-band electrooptical systems. *Proceedings of the IEEE*, 2009.

[37] B. Monien and R. Preis. Upper bounds on the bisection width of 3- and 4-regular graphs. *Journal of Discrete Algorithms*, 4, 2006.

[38] J. Mudigonda, P. Yalagandula, and J. C. Mogul. Taming the flying cable monster: A topology design and optimization framework for data-center networks. In *USENIX ATC*, 2011.

[39] N. McKeown et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM CCR*, 38(2), 2008.

[40] S. Orfanidis. Electromagnetic waves and antennas; Chapter 15, 19. http://www.ece.rutgers.edu/~orfanidi/ewa/.

[41] L. Popa et al. A cost comparison of datacenter network architectures. In *CoNEXT*, 2010.

[42] G. Porter et al. Integrating microsecond circuit switching into the data center. In *ACM SIGCOMM*, 2013.

[43] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4), 1987.

[44] M. Reitblatt et al. Abstractions for network update. In *ACM SIGCOMM*, 2012.

[45] J. Shin, E. G. Sirer, H. Weatherspoon, and D. Kirovski. On the feasibility of completely wireless datacenters. In *ANCS*, 2012.

[46] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *NSDI*, 2012.

[47] O. Svelto. *Principles of Lasers*. Plenum Press, New York, Fourth edition, 1998.

[48] J. Turner. Effects of data center vibration on compute system performance. In *SustainIT*, 2010.

[49] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz. (in Russian)*, 3:25–30, 1965.

[50] G. Wang et al. c-Through: Part-time optics in data centers. In *ACM SIGCOMM*, 2010.

[51] R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In *Hot-ICE*, 2011.

[52] Y. Yang, S. Goswami, and C. Hansen. 10GBASE-T ecosystem is ready for broad adoption. *Commscope/Intel/Cisco White Paper*, 2012. Available at http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/COM_WP_10GBASE_T_Ecosystem_US4.pdf.

[53] K. Yoshida, K. Tanaka, T. Tsujimura, and Y. Azuma. Assisted focus adjustment for free space optics system coupling single-mode optical fibers. *IEEE Trans. on Industrial Electronics*, 60(11), 2013.
[54] X. Zhou et al. Mirror mirror on the ceiling: Flexible wireless links for data centers. In *ACM SIGCOMM*, 2012.

APPENDIX

A Evacuation Time of Flexible(f)

Here, we formally prove the claim mentioned in §2.1, with regards to the total evacuation time of the Flexible(f) architecture.

THEOREM 1. Consider an inter-rack traffic matrix D, wherein each entry D_{ij} represents the total traffic demand from source-rack i to the destination rack j. Assume that each unit of data takes a unit time to be evacuated over any link.

For $f \leq l$, the total evacuation time for the Flexible(f) architecture is at most

$$\frac{l}{f}M + \frac{1}{f}\max_{ij}(D_{ij} + D_{ji})$$

units, where M is the evacuation time for FBB (full-bisection network) and l is the number of machines per rack. For $f > l$, the evacuation time is the same as that for $f = l$. The above quantity is $\approx \frac{l}{f}M$ units with high probability for a uniformly random traffic matrix and large number of racks.

Finally, the number of reconfigurations needed to achieve the above evacuation time is at most the number of non-zero entries in the given traffic matrix.

PROOF: We start with considering the simple case of $f = 1$. Consider an undirected multigraph G over the set of racks, where the *number* of edges between a rack i and j is $D_{ij} + D_{ji}$. Each undirected edge in G represents a unit-demand between the pair of racks. Now, any matching in G represents a set of packets that can be evacuated in a unit time since: (i) a matching in G represents a valid "wiring" of the Flexible(f) architecture, and (ii) each link (i, j) can evacuate one unit of data from i to j or from j to i in a unit time. Thus, the total time to evacuate the given traffic demand matrix in Flexible(1) is equal to the minimum number of matchings the given graph G's edges can be partitioned into (or equivalently, the minimum number to colors needed to color the edges of G such that no two edges incident on a node have the same color). Now, using the classical result from Vizing [49], G can be decomposed into

$$\max_{ij}(\Delta_i + \mu_{ij})$$

matchings, where $\Delta_i (= \sum_k (D_{ik} + D_{ki}))$ is the degree of a node i and $\mu_{ij} = (D_{ij} + D_{ji})$ is the number of edges between the nodes i and j in G.

The extension to a general f is straightforward. Essentially, when $1 \leq f \leq l$, the Flexible(f) architecture can evacuate f matchings of G in one unit time. Thus, the evacuation time for a general $f \leq l$ is at most:

$$\frac{1}{f}\max_{ij}(\Delta_i + \mu_{ij}).$$

Now, since the total evacuation time of FBB is exactly $\frac{1}{l}\max_i \Delta_i$ as it can send or receive at most l units of data from a rack in a unit time, the total evacuation time of Flexible(f) is at most:

$$\frac{l}{f}M + \frac{1}{f}\max_{ij}\mu_i j = \frac{l}{f}M + \frac{1}{f}\max_{ij}(D_{ij} + D_{ji}).$$

Since Ml is expected to be much larger than $\max_{ij}(D_{ij} + D_{ji})$ for a uniformly-random traffic matrix and large number of racks, the above is $\approx M(l/f)$.

Finally, to bound the number of reconfiguration, we can show that the number of *different* matchings in the partitioning of G above can be bounded by the number of non-zero entries in the traffic matrix. We skip the details here. ■

B FSO Link for Long Distances

Here, we briefly corroborate the claim that our FSO link design should work for distances up to 100m and more. Recall from Figure 3 the basic schematic of our FSO link design. Here, the light beam emanating from an optical fiber is first collimated at the transmitter using the collimation lens, and then focused into an optical fiber at the receiver using another lens. We argue below that, with our design, the loss of power due to beam divergence and air-attenuation is expected to be minimal for longer distances (say, up to 100m), and hence, the FSO link should work at such distances.

- We note that the collimation lens in our design has been chosen appropriately to ensure the beam remains collimated for a distance of up to 100m with a width of roughly 4mm. Since the diameter of our receiving lens is greater than 25mm, a 4mm wide beam is fully "gathered" by the receiving lens. Thus, we can easily ignore the power loss resulting from beam divergence.
- Now, we show that the signal attenuation due to scattering and reflection in the air is also negligible. Note that, while attenuation of an infra-red laser beam can be high in fog or rain, it is minimal in clear whether. In particular, [15] estimates the value of attenuation-coefficient for a beam of 1310nm wavelength that we use in our design to be 3.5dBm/km. This suggests a power loss of 0.35dBm over a 100m link, which is much lower than the power-losstolerance of the detectors at SFPs. For comparison, a typical commercial optical cable [9] has an attenuation of 0.38dBm/km which results in a power loss of 3.8dBm over 10km, the range of 10GBASE-LR SFP's used in our system. Further, we note that the above attenuation-coefficient is for outdoor environments, and the indoor attenuation-coefficient (which is more relevant in our context) is much lower.

C Correctness of Data Translation Scheme

Here, we formally prove that our data translation scheme of §6.3 ensures the desired properties. We start with an observation.

OBSERVATION 1. At any instant, the set of edges in $\mathcal{G}$ is a subset of the actual set of active links in the networks.[12] □

THEOREM 2. The overall scheme of §6.3 ensures that (a) there are no black holes (i.e., all rules in NFTs refer to active links), (b) the network remains connected, and (c) the packet latency is bounded by $(2x + z)$, where x is the maximum packet latency in a fixed topology and z is the NFTs-update time. The above claim holds irrespective of how the NFTs are updated across the network; in particular, we do not require atomic updates.

PROOF: We start with stating a couple of implicit assumptions made in our translation scheme: (i) In steps C1 and C3, the update to $\mathcal{G}$ occurs before the NFTs-updates, and (ii) the NFTs-updates (in C1 or C3) from different concurrent reconfigurations finish in the same order as the corresponding updates to $\mathcal{G}$.[13]

[12]This is true because we only allow non-conflicting reconfigurations to execute concurrently; we skip the tedious details.

[13]This is reasonable since the NFTs-changes are sent to the network switches in the form of "deltas," and hence are received (and thus, finished) at each switch in a deterministic order.

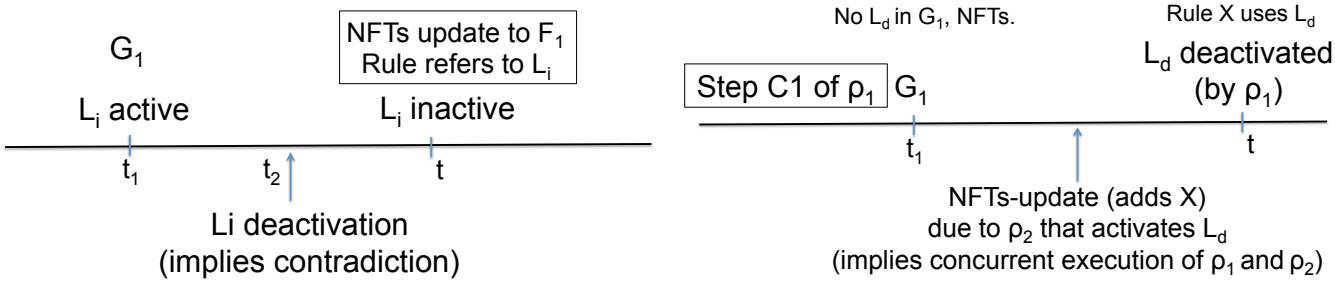


Figure 11: Correctness proof for avoidance of black holes.

Avoidance of Black Holes. Consider the first instant t when a black hole is created in the network. There are only two possible events that can induce a black hole, viz., (i) A new forwarding rule that refers to an inactive link is added, or (ii) A link that is being used in a forwarding rule is deactivated. We consider each of these cases below.

1. Lets say a rule that refers to an inactive link l_i is added at time t due to the update of NFTs to $\mathcal{F}_1$, as part of a reconfiguration ρ_1. Let the corresponding update of $\mathcal{G}$ variable to $\mathcal{G}_1$ to have occurred at time t_1 ($< t$). Now, $\mathcal{G}_1$ must include the link l_i (since $\mathcal{F}_1$, which is based on $\mathcal{G}_1$, has a rule that refers to it). By Observation 1, l_i is active at t_1. Thus, l_i must have been deactivated at time t_2 between t_1 and t (by some reconfiguration, say ρ_2). See Figure 11(a). This is impossible because it requires the corresponding step C1 of ρ_2 to have occurred before t_2, which yields one of the following two cases each of which leads to a contradiction.
   - If update to $\mathcal{G}$ in step C1 of ρ_2 occurs in $[t_1, t_2]$, then the entire step C1 of ρ_2 occurs in $[t_1, t_2]$ which implies that updates to $\mathcal{G}$ (by ρ_1 and ρ_2) occur in reverse order of corresponding NFTs-updates (a contradiction).
   - If update to $\mathcal{G}$ in step C1 of ρ_2 occurs before t_1, then the two conflicting reconfigurations (viz., ρ_2 and another[14] that adds l_i to $\mathcal{G}$ before t_1) have executed concurrently (a contradiction).
2. Lets say, at time t, a link l_d, that is being used in an existing forwarding-rule X, is deactivated. Let the deactivation be due to a reconfiguration ρ_1, and the corresponding step C1 to have finished at time t_1 ($< t$) and have updated the variable $\mathcal{G}$ to $\mathcal{G}_1$ and the NFTs to $\mathcal{F}_1$. Thus, $\mathcal{G}_1$ must not contain l_d and $\mathcal{F}_1$ must not refer to l_d. Thus, there must have been an NFTs-update between t_1 and t that added the rule X, and this must be due to a reconfiguration ρ_2 that activates l_d. See Figure 11(b). Now, ρ_1 and ρ_2 are obviously conflicting, but have executed concurrently. This is contradictory to our scheme.

We note that the above argument does not assume a minimum time-interval between updates, and hence, avoidance of black holes is guaranteed even if NFTs-updates are allowed to execute concurrently.

Network Connectivity. Note that $\mathcal{G}$ is guaranteed to be connected at all times. Thus, by Observation 1, the underlying network topology must be connected at all times.

[14]There much be such a reconfiguration, since ρ_2 removes l_i from $\mathcal{G}$ before t_1, and at t_1, $\mathcal{G}$(=$\mathcal{G}_1$) includes l_i.

Bounded Packet Latency. The minimum time-interval of x between two updates, where x is the maximum packet latency in a fixed topology, ensures that each packet encounters at most two NFTs during its flight. Thus, the packet latency is bounded by $2x + z$. ■