1 Introduction and Vision

GREEN color is to denote stuff that can/should be gotten rid of.

Data centers (DCs) are a critical piece of today's computing infrastructure that drive key networked applications []. The key benefits offered by these large-scale consolidated computing power are: the economies of scale, reduced or amortized management costs, better utilization of hardware via statistical multiplexing, and the ability to dynamically scale applications in response to changing workload patterns. The key component that drives the success of such large-scale datacenters is an efficient, robust, and cost-effective *network fabric*. In particular, the data center network designs must satisfy several potentially conflicting goals: high performance (e.g., high throughput and low latency) [2, 4]; low equipment and management cost [2, 14]; robustness to extremely dynamic traffic patterns [5, 6, 8, 15]; incremental expandability to add new servers or racks [3, 16]; and a host of other practical concerns including cabling complexity, power, and cooling [9, 11, 13]. The traditional data center architectures have been fixed (wired) in their design. Hence, due to the highly variable and unpredictable nature of data center workloads [4], these traditional fixed designs either (a) offer poor performance at low cost (e.g., simple trees or leaf/spine architectures), or (b) are over-provisioned (e.g., fat-tree architectures) to account for the *worst-case* communication requirements; over-provisioning results in higher cost and maintenance of networking architecture. Some recent works augment the simple wired infrastructure with additional inter-rack wireless [5, 8] or optical links [6] to impart certain flexibility to the network architectures and thus help alleviate congested hotspots. However, the overall vision of a fully wireless and a highly flexibible data center network architecture with very favorable cost-performance tradeoff remains far from realized. The goal of our research proposal is to realize this vision.

Our Vision. We take the flexibility offered by wireless inter-rack links to the logical extreme and propose to design an *all-wireless inter-rack data center fabric*. Our vision, if realized, would provide unprecedented degrees of flexibility for data centers. That is, it will allow operators to dynamically reconfigure the *entire* network topology to adapt to changing traffic demands. Moreover, a wireless architecture eliminates maintenance and operational costs due to cabling complexities [**?**], and more importantly, can facilitate topology structures that would otherwise remain "paper designs" due to the perceived cabling complexity.

To realize our vision of an *all-wireless* inter-rack data center fabric, we look beyond the traditional radio-frequency (RF) based (e.g., 60GHz) solutions and explore a somewhat non-standard "wireless" technology, namely *Free-Space Optics* (FSO). FSO uses visible or infra-red lasers to implement point-to-point data links, and offers very high data rates ($>$1 Gbps) at long ranges, even with minimal transmission power. Most importantly, unlike the RF wireless technologies, FSO communication links have a minimal wireless-interference footprint (due to their low wavelength) and their usage is not constrained by federal regulations. While FSO is an enabling technology, there are various scientific challenges that need to be addressed before we can realize a viable fully-flexible DC network fabric. Our research proposal plans to address these challenges and build an FSO-based DC network prototype. We start with motivating our vision, and presenting our overall network architecture and its key challenges in the following section.

Our research consists of three computer scientists with complementary networking and theoretical expertise and a mechanical engineer with expertise in laser-based optical measurement techniques, giving us a complete spectrum of expertise needed for various scientific aspects of our research proposal.

2 Motivation, Overall Architecture, and Key Challenges

Motivation for Flexibile Architectures. Our vision of a "fully flexible" architecture for data centers is motivated by the following key observations that we made through analysis of available real data center workloads [].

Figure 1: (a) Cumulative fraction of underutlized links for the fixed architecture with XX switches. (b) Number of switches used in D-flexible architectures, for varying D, for uniform performance (note that the fixed architecture corresponds to D=0). (c) Performance of k-ToROnlyFlexible architectures for varying k.

1. *Under-Utilization of Network Resources in Fixed Architectures.* When we simulated the available DC workloads over a fixed DC architecture viz., JellyFish [16], we observed that, at any point of time, a large fraction of the links/wires in the network were severely under-utilized (i.e., total traffic carried by them was only a small fraction of their total capacity).

2. *Flexibility Improves Resource Utilization and Reduces Required Infrastructure.* If we allow "reconnection" of some number of ports of each switche based on the future-epoch's traffic, then we observe that we can get same performance as the fixed architecture with a fewer number of switches. Moreover, the number of switches in the architecture decreases with the increase in available flexibility (i.e., number of ports in each switch that can be reconnected, every epoch).

Traffic Analysis Methodology. We consider a data center of 20 racks, where each rack has k machines. We use 1Gbps $2k$-port switches, as in FatTree architectures. The ToR (top of rack) switches use k ports for the machines, and the remaining k ports for inter-switch connections. The non-ToR switches use all their ports for inter-switch connections. Our fixed architecure for (a) is based on a random graph (of inter-switch connections) over XX number of switches, and delivers a performance of ZZZ flow-completion time. We generate D-flexible architecture as follows: We allow D ports of each ToR switch and $2D$ ports of each non-ToR switch to be "reconnected" at each epoch; the interconnections between remaining ports are random but *fixed*. Thus, higher the value of D, more flexible is the architecture. We also consider m-ToROnlyFlexible architectures, wherein we use only m-port ToR switches (i.e., no non-ToR switches) and $m - k$ ports of these switches are reconnected at every epoch while the remaining k ports are used for machines on the rack.

Results. Figure **??**(a) shows that a large fraction of links are underutlized in the fixed architecture. Figure (b) demonstrate that imparting flexibility to the architecture can greatly reduce the network infrastructure (switches), and the reduction in switches increases with the increase in flexibility. Figure (c) shows a full-flexible architecture based on only ToR switches (like ours) can match the performance of the fixed architecture, if a large enough ports are used on the ToR switch.

2.1 Proposed Flexible Wireless Architecture

One way to achieve a flexible architecture is to perform "re-wiring" of the ToR switches on the fly using highly directed, but steerable wireless links. Directivity is needed to reduce/eliminate interference between neighbouring links. Directivity in turn necessitates steering. We envision that these wireless links will

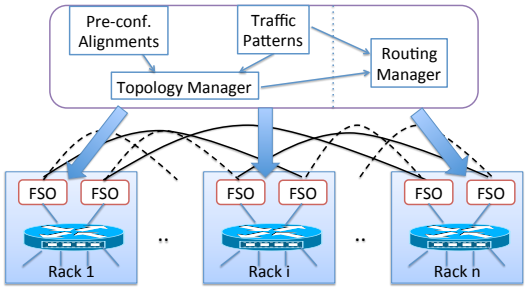

Figure 2: System overview: The Topology Manager decides the set of links to activate and the Routing Manager sets up routes for end-to-end flows. At any instant, only one candidate link per FSO is active (solid lines).

connect the ToR switches to form a fully flexible inter-rack fabric.[1] See Figure 2. The wireless transceivers are to be placed on top of each rack. We also envision a (centralized) *Topology Manager* that dynamically reconfigures the inter-rack topology and a *Routing Manager* acts in concert with the Topology Manager to setup the routing table entries for each ToR switch to route flows between racks. This latter ability is needed as the inter-rack topology may not always yield a completely connected graph.

Choice of Wireless Technology. A normal inclination will be to use high-speed RF links with steerable directional antennas to implement the inter-rack network. However, RF links produce a large interference footprint due to a wide beamwidth, which would ultimately limit the number of concurrently active wireless links. The interference footprint remains large even with 3D beamforming technique [8]. Furthermore, beam-steering (needing to impart flexibility) technologies for RF links are either slow and inaccurate [8] or present even wider beams [**?**]. Finally, at 60 GHz data rates fall off rapidly with distance [8]; higher transmit power can increase range, but it also yields higher interference, more energy usage, and is ultimately limited by regulations. Thus, the RF technology would be limited utility in realizing a *fully*-flexible (i.e., an *all*-wireless) inter-rack network.

Using Free Space Optics. To overcome the limitations of RF, we pursue a unique angle – use of free-space optics (FSO) to interconnect the ToR switches. FSO communication [10] uses modulated visible or infrared (IR) laser beams in the free space to implement a communication link. Unlike traditional optical networks, the laser beam in FSO is not enclosed in a glass fiber, but transmitted through the air. There are two main benefits of FSO compared to traditional RF technologies that make it a promising candidate for data centers:

(a) *Low Noise and Interference.* Unlike RF links, FSO links do not suffer from multipath fading or EM noises []. Moreover, FSO's beam divergence is many orders of magnitude narrower than RF (in the order of milliradians or less (1 milliradian = 0.0573°). This reduces 'interference footprint' to a negligible level. Thus, FSO communications from multiple senders do not interfere, unless they are aligned to the same receiver (which is easy to avoid).

(b) *High Data Rates over Long Ranges:* Optical communications inherently provide significantly higher data rates than any existing RF technology owing to the use of much higher frequency and absence of regulatory restrictions [10]. Coupled with much lower attenuation of power over distance, FSO links are able to offer data rates in the Gbps to Tbps regime at long distances (several kms) even with modest transmit power (watts) [10]. Commercially available FSO devices already offer data rates of 2.5 Gbps [1],

[1]Note that we are not proposing a fully wireless data center as in [7]; our focus is on the 'inter-rack' fabric only. Thus, the conventional wired rack architecture including the ToR switches remain.

and demonstration systems even report data rates in the order of Tbps [12] – both for long distance (in the order of km) links.

Key Challenges in FSO-based Inter-Rack Network. While use of FSO brings in tremendous benefits in terms of data rates and interference immunity, its use also exposes significant design challenges:

(a) *General Feasibility:* FSO communications traditionally have been targeted as an RF replacement for long-distance communications, both terrestrial and also extra-terrestrial []. Most research thus has been directed to overcome challenges for these use cases only. The commodity designs top off within a few Gbps for FSOs, while we certainly should strive for 10-100Gbps links given the timeline for this project. We also must make the design small, energy-efficient and use commodity components to make the approach cost-effective. The cost issue is important as data center network designs are fundamentally directed by cost-performance tradeoffs [] and not simply performance alone. The following should go into the next section, since its not a challenge but a solution perhaps. For cost-effectiveness, we base our design using commodity components. Instead of designing independent laser sources and modulators, our approach uses the optical SFP (small form-factor pluggable) module as the FSO transceiver that is a standard-compliant device for optical networks. This allows us to work with small form factor and be able to ride the technology curve with the available data rates. But this also brings in the challenge of effectively launching the laser beam into the air instead of a fiber that it is designed for.

(b) *Alignment and Steering:* Given the very narrow beamwidth of lasers, precise alignment of the sender and receiver pair is a significant issue and here lasers are much less forgiving than RF. While this is a serious challenge in outdoors and long distances – the conventional use-cases for FSO, it is less so in the data center context (more on this in Section**??**). In addition to alignment, the FSO beam must be 'steered' at fast time scales (10s of milliseconds or less) to switch between multiple receivers in order implement varying topologies. While phased array-based beam steering for RF is a well-established technology and have been routinely used in commercially available systems, such techniques – while available [] – are less mature for lasers. In line with our focus on cost-effectiveness, we thus investigate two techniques that are feasible with commodity components. One uses switchable mirrors for steering and separate electro-mechanical actuators for alignment and the other uses Galvano mirrors for both alignment and steering.

The following section elaborates on these challenges and outlines the key components of the design. It also presents our proof-of-concept experiments demonstrating the general feasibility.

3 Building an FSO Device

4 Topology Design

Our vision (see Figure 2) is a data center network where the ToR switches are interconnected using FSO devices. Note that we are not proposing a fully wireless data center [7]; our focus is on the 'inter-rack' fabric. The FSO transceivers are placed on top of each rack. These devices are aligned to connect, after reflection from the ceiling mirror, to devices on other racks. (FSO links are duplex by construction).

As mentioned in the previous section, we will explore use of two different mechanisms to steer the FSO devices, viz., Switchable mirrors (SMs) and Galvano Motors (GMs). Each FSO is equipped with either multiple (usually, 5-20) SMs or a single GM. Each SM is pre-aligned to target a specific FSO on a specific rack in the system, and each GM is pre-oriented to target a pre-determined geographic area in the data center. In real-time, based on the current traffic, each SM in the system is chosen to be in the mirror or transparent state (with only one SM per FSO being in the mirror state), and each GM is aligned to target a specific FSO in its pre-determine geographic area.

Ideally, we would like as many FSO transceivers on each rack, with each FSO equipped with as many SMs or a GM that can cover a large geographic area, and "reconfigure" the topology to any desired topology in real-time with zero delay. However, in practice, there will be many constraints. First, the FSO size would limit the number of FSO devices that can be placed on a rack. Second, the limited number of SMs, limited area covered by GMs, and the non-zero steering latency would make reconfiguration possibilities limited.

In the above setting, there are two natural design questions that need to be solved: (i) Determination of the pre-alignments of the SMs and GMs, and (ii) Real-time choice of SM-states and GM-orientations, in response to changing traffic. We consider each of these problems in the following two subsections.

4.1 Pre-Configuration Topology

In this section, we consider the problem of determination of the pre-configured topology. For clarity of understanding, we start with limiting our attention to the data center architecture **consisting of only SMs** (i.e., no GMs). We would consider more general architectures later. Also, we consider an illustrative data center setting consisting of exactly 512 racks, each containing 48 machines/servers. The top of the rack has m FSOs. Each rack has a ToR switch containing (48 + m) ports, with 48 of them connected to the 48 machines on the rack and m of them connected to the m FSOs. Each FSO is equipped with k SMs.

Objective Functions. We consider two different objective functions to evaluate a pre-configured topology: (a) total evacuation time for a uniform traffic model (for a given re-configuration algorithm and latency), or (b) dynamic bisection bandwidth (as defined later). Each of these two objective functions can also be considered in conjunction with a given non-uniform traffic model.

Pre-Configuration Topology Problem. The pre-configuration problem can be formally defined as follows. Given a certain setting (number of racks, number of machines per rack, number of FSOs per rack, steering capability and latency at each FSO), the problem is to determine the intial configuration (pre-alignment of each SM or orientation of each GM) so as to optimize one of the above three performance metrics, viz., minimize uniform-traffic evacuation model or dynamic bisection bandwidth (perhaps, under the constraint of maximum diameter).

Uniform-Traffic Evacuation Model. In the uniform-traffic evacuation model, we assume that each rack has a constant and large amount of traffic for each other rack, and the objective is to evacuate the entire traffic in minimum amount of time (while allowing ourselves the flexibility to change our flexible network as desired, as long as the latency of reconfiguration is taken into consideration). If we consider an extreme case of zero (or sufficiently low) reconfiguration-latency, then it is reasonable to consider regular random graphs as a plausible solution which should perform well. Random graphs have well-known advantages compared to other strucutred topologies. E.g., recent work [16] shows that random regular graphs provide bandwidth and latency comparable to structured topologies. Furthermore, a random graph is naturally amenable to incremental expandability. In our setting of k SMs per rack and m FSOs per rack, we can create a k-regular random graph over the FSOs (or a km-regular graph over the racks) by aligning the SMs appropriately. In our preliminary work [**?**], we demonstrated near-optimal performance with a random pre-configuration topology for the setting of 512 racks, 16 FSOs per rack, and 5 SMs per FSO.

If km (degree per rack) is low relative to n (# racks), then a random graph may not have good connectivity. This concern might become relevant in the regimes we are considering – km is a few tens, and n may be a few hundred. Thus, in such case, we consider an alternative topology wherein we use some FSOs (using one of its SM) to construct a "baseline" topology that guarantees connectivity properties, and align the remaining FSOs/SMs randomly. We believe that a hypercube is a suitable candidate for this baseline topology for three reasons: (1) it uses a small number of links and leaves many candidate links for the random links; (2) it has a small diameter ($\log n$ for n racks); and (3) it has a reasonable bisection bandwidth ($n/2$ over n racks). Furthermore, the performance of a hypercube can be improved by adding *diagonal* edges which

connect each node to its "complement"; these "short-cuts" halve the diameter (proof omitted). We also conjecture (based on empirical findings) that the diagonals also improve (roughly double) the (static) bisection bandwidth. Finally, note that there are different types of short-cuts/diagonals that can be considered for hypercubes, each with different trade-offs. The above idea can be used in two different ways: (i) the hypercube links are kept static (i.e., the state of the SMs used for the hypercube links are never changed), and (ii) the hypercube links are dynamic. The first strategy ensures that there is always a connected backbone for the network, while the second yields higher performance by utilizing more flexility. In our preliminary work [**?**], we observed that the in our settings (512 racks, 16 FSOs/rack, 5 SMs/FSO) the second strategy yielded similar performance as random graphs, without any connecitivity problems.

Our intuition for exploring the above two pre-configured topologies in our settings was to minimize the average hop-count between a pair of racks. In fact, with certain constraints on the uniformity of the graph and the all-pairs path, we conjecture that average hop-count is directly correlated to the evaluation time objective value. We will formally prove this result, and based on this intuition, we will consider other pre-configuration topologies that can yield better performance for given settings.

- Is there a non-traffic *average* "goodness"?
- Can we do better? Other structured k-regular graphs? Results regarding bounds.

Role of Reconfiguration Latency. Finally, we note that the latency plays in important role in the evaluation of the evacuation time objective. E.g., if the reconfiguration latency is zero, then all the "candidate" links of the network can be effectively considered as always-active links. However, if the reconfiguration latency is very high, then the evacuation time depends on a *single* configuration of the network. It would be interesting to determine how the reconfiguration latency affects the choice of pre-configuration latency, when trying to minimize the total evacuation time.

- If latency is zero: try a tree; if diameter problem is OK, then hypercube+diagonals?
- If very high latency: ...
- somewhere in between: Quantatively? 20msec translates to what? Depends on statistics about elephanat flows.

Rethinking Performance Metrics: Dynamic Bisection Bandwidth. Traditional metrics such as bisection bandwidth and diameter largely reflect a *static* perspective of the topology. For the types of flexible networks as described above, we need to rethink these metrics. In particular, we need a notion of *dynamic* bisection bandwidth (DBW) based on the fact that, in a flexible network such as ours, different topologies can be used for different communication requirements. Formally, dynamic bisection bandwidth can be defined as the minimum (across all possible equal partitions of the graph) configurable-bandwidth of the graph, where the configurable-bandwidth for a given partition is the maximum bandwidth possible across all possible configurations of the given flexible network. The above definitions implicitly assume that we can reconfigure the topology as and when needed based on the communication needs. However, if the reconfiguration is done very infrequently (due to high latency, or other design constraints), then the above definition should be changed appropriately (e.g., the dynamic bisection bandwidth should be defined as the minimum bisection bandwidth across all possible configuration).

Optimizing Dynamic Bisection Bandwidth. We now consider the problem of determine the pre-configuration topology with the aim of maximing the dynamic bisection bandwidth as defined above. Consider data center with 512 racks with 48 machines each. Let us assume the ToR switch to be a 10GigE switch, with each link (FSO to FSO, machine to ToR) having a maximum capacity of 10Gpbs. Since each machine can send/receive

at most 10Gbps traffic, the maximum desired bisection bandwidth for this DC is D = 10(512)(48)/2 Gpbs = 256(48) links of 10Gpbs each. Let us see how we can achieve this much (dynamic) bisection bandwidth using FSOs and SMs. We consider two extremes, viz., when each FSO has only one SM (i.e., an inflexible network), and when an FSO can have as many SMs as needed. When each FSO has only one SM, it is easy to see that equipping each rack with 256+48 FSOs achieves an inter-rack bisection bandwidth of D Gpbs. Furthermore, note that an inter-rack bisection bandwidth of a certain amount also implies an inter-machine bisection bandwidth of the same amount (and vice-versa), due to the 10GigE ToR switch connecting the machines on a rack. It is not clear what is the minimum number of FSOs required to achieve the bisection bandwidth of D Gpbs. Considering the other extreme, where an FSO can have an unlimited number of SMs, we see that we need at least 48 FSOs (with 256 SMs each) to achieve a *dynamic* bisection bandwidth of D Gbps. In this case, the number of FSOs required is minimum possible, but the number of SMs per FSO could be possibly further reduced. Above, we have illustrated the spectrum of possibilities by consider two ends of the spectrum. In general, we want to address two problems: (i) For a certain desired dynamic bisection bandwidth, what is the minimum number of FSOs required per rack if each can be equipped with a certain given number of SMs. Similarly, what is the minimum number of SMs required per FSO for a given number of FSOs per rack, to achieve a certain given dynamic bisection bandwidth? (ii) Given a certain number of FSOs per rack and number of SMs per FSO, what is the maximum achievable dynamic bisection bandwidth? Note that the above problems are quite different from the well-known problem of *determining* bisection bandwidth of a given graph, which is known to NP-hard []. In the above problems, we are instead interested in "constructing" a *flexible* graph with a desired (dynamic) bisection bandwidth. How about results on the regular graphs? In our research, we would address the above problems.

Incorporating Traffic Statistics. The above discussion relating to the two objective functions implicitly assumed lack of any traffic information. However, if we have some statistics available on the expected workload, then the pre-configuration can be tailored to the available traffic knowledge. Note that since the pre-configuration can even be done on a regular but infrequent basis, it is desirable to be able to pre-configure based on expected traffic. One simple form of traffic statisfics could be in the form of a weighted complete graph over the racks, with the weight signifying the traffic workload between that pair of racks. Then, one way to solve the pre-configuration problem could be to extract the maximum weighted km-regular subgraph from the weighted complete graph representing the given traffic matrix. To the best of our knowledge, this problem has been been addressed before in the theory literature. Thus, we would first attempt to prove the hardness of this problem, and develop approximation algorithms. There is a very simple ILP formulation for the problem, and hence, can be solved optimally using CPLEX for small instances. For larger instances, we can develop a greedy heuristic that interatively extracts links based on certain criteria. In our research, we would explore the above avenues. There are two additional constraints that can be incorporated in the above subgraph extraction problem: (a) First, the km SMs on each rack are actually in the form of k SMs on each of the m FSOs, implying that the km links at any rack would be organized in m sets of k links each with only one link from each set being active at any time. This fact and the given switching latency should both be taken into consideration when extracting the subgraph. (b) Second, the links from the complete graph that are not chosen in the extracted subgraph, should also have a low hop-count in the extracted subgraph. Incorporating the above constaints in the ILP formulation doesn't seem straightforward, but the greedy algorithm can be easily modified to handle the above constaints. In our research, we would address the above, and also consider the problem in the context of more sophisticiated available traffic statistics [].

Galvo Motors (GMs), and GMs with SMs. In the discussion till now, we have considered the FSO-based architecture involving only SMs. As noted in previous section, we also plan to explore the steering mechanism based on GMs, wherein each FSO is equipped with a single GM which is oriented towards a pre-configured geographic area A in the data center, and within A, the GM can be steered in real-time to target any specific FSO. The pre-configuration problem in this context involves determining the initial

orientations of the GMs. Use of GMs, instead of or in addition to SMs, adds a few additional challenges to the pre-configuration problem. Firstly, it's not clear if one can realize a truly random regular graph using GMs. If we orient the GMs randomly, that only results in *sets* of edges being distributed randomly. Secondly, the angular contraint of GM's steerability makes it less amenable to achieving a high dynamic bisecion bandwidth. Lastly, the traffic-aware pre-configuration problem in the context of GMs can't be easily expressed as a combinatorial graph problem or an ILP, and may require geometric solutions. In our research, we would extend our SM-only based solutions to the GM-based architectures. In addition, we would also explore architectures involving a mix of GMs and SMs (i.e., having some of the FSOs equipped with GMs, while the others equipped with SMs), and consider pre-configuration topology problem in this context. One of the additional challenges in this setting would be to also decide how many FSOs on each rack should be equipped with SMs and how many with GMs. This question may need to be answered within the constraint of available cost/budget. It would be interesting to understand the cost-performance tradeoffs that GMs or GMs in combination with SMs offer relative to the SM-only architecture. Note that in architectures involving GMs, the steering latency may depend on the "angular distance," which may have a bearing on the eventual performance and hence, the design choices.

Other Architectureal Ideas. We can augment our architecture (based on ToRs, FSOs, SMs, and GMs) in a few other ways to enhance the performance of our system. In particular, we would like to explore two possibilities:

- In our architecture, the only switches used are the ToR switches that have some of the ports connected to the rack machines and the remaining ports connected to the FSOs on the rack. However, adding additional "steiner/hub" switches (as in all the traditional DC designs) all of whose ports are connected to the FSOs can also be used to add more flexibility to the design. The challenges here are: (a) Find appropriate physical space to place such switches and associated FSOs, and (b) design appropriate topology to determine the connections that maximize performance.
- Even though, an FSO link is capable of transmitting a much higher bandwidth, in our design, the rate on each FSO link is limited by the rate of the ToR switch ports. One way to allow higher bandwidth on an FSO link is to use a WSS (wavelength selective switch) in between the ToR switch ports and the FSOs, as in OSA [], allowing multiple ports to "feed" into a single FSO link. The WSS can be configured in real-time to allow a configurable number of ports to feed into a single FSO – yielding another dimension of flexibility to our design, viz., links of variable bandwidth which is configurable in real-time.

Budget Based Optimization Given budget and/or equipment: pick Setting to Maximize Performance, either one or both topologies. Similar Vein as REWIRE. Best we can hope for are some other structured graph, that have good performance.

Computing DBW and Dynamic Diameter. To consider the above metrics in our design objectives, we need to first be able to efficiently *compute* these measures for a given network. Computation of dynamic diameter

for a flexible network is quite straight-forward: the dynamic diameter of a flexible network is actually just equal to the diameter of the graph with *all* the candidate links. This is true since the minimum distance across all realizations is simply the distance between the given nodes in the graph with all candidate links. However, computation of dynamic bisection bandwidth is non-trivial. Firstly, note that even the computation of traditional/static bisection bandwidth is known to be NP-complete, with the best approximation factor being polylogarithmic []. It is clear that the DBW problem is more general, and hence is also NP-complete. Since the approximation algorithm is unacceptably complicated and approximate, we would try to generalize the well-known heuristics (SA and KI) known for the static version to the dynamic BW. The heuristics are known to perform wonderfully well for a wide range of random graphs. We will try to design heuristic to achieve similar performance. Moreover, our graphs are regular graphs – so, the heuristic could be specialized. This will be explored in our research.

Reconfiguration Algorithm.

We envision a (centralized) *Topology Manager* that dynamically does the above-said reconfiguration in real-time, and the *Routing Manager* acts in concert with the Topology Manager to setup routing table entries for each ToR switch to route flows between racks.

- . Given the setting, what algorithm?
- . Should it be considered in conjunction with pre-configure?
- . Elephant flows? Continuous function?
- . If average hop is low, then what we have may be best. If average hop is high, then our method may not make sense.

5 Topology and Traffic Management

The previous sections described the two key building blocks for LightSpeed: cost-effective steerable free-space technologies and effective topology design mechanisms given cost, physical, and geometric constraints in practical datacenters. In this section, we focus on the design of a *datacenter management layer* that integrates these building blocks into a practical near real-time reconfigurable data center architecture.

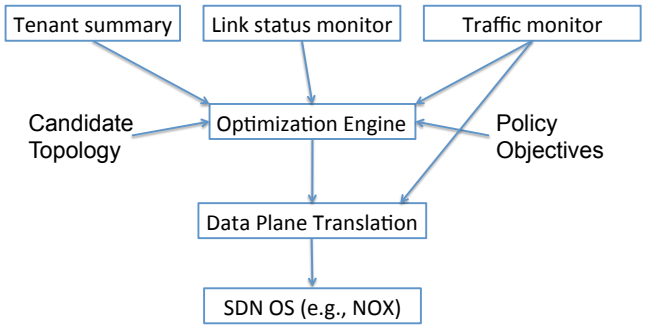

Figure 3: Overview of the LightSpeed management layer

In designing the LightSpeed management layer, we build on and extend principles from several classes of traditional cloud and network management literature including traffic engineering [], software-defined networking [], fast routing recovery [], and managing network updates []. The key difference from this prior work arises on two key dimensions. First, most of these prior efforts typically assume the topology "as a given"; i.e., the topology is typically pre-specified. With the vision of LightSpeed, this assumption or constraint no longer applies. This gives rise to unexplored research opportunities for rethinking traditional traffic engineering and routing approaches in conjunction with *topology engineering*. Second, there are unique physical and timing constraints imposed by the specific technologies available for free-space optics and the specific pre-configured topologies. This gives rise to unique *correctness* challenges with respect to ensuring guarantees on network reachability and performance even when the topology itself may constantly be in flux.

Overview: For completeness, we describe the high-level roles of the different components of this management module shown in Figure 3. We discuss the design and implementation challenges involved in each component in subsequent subsections.

- **Optimization Engine:** At the core, the management layer needs to solve a hard online optimization problem that takes in as input the offered traffic workload, the current status of the network (e.g., active links and link status), onstraints on the available network links (e.g., as defined by the candidate preconfigured topologies from Section **??**), and policy objectives (e.g., SLAs on throughput, loss, or latency reqirements for different tenants). Based on these considerations it produces a *topology and and traffic engineering strategy* that optimally (or near-optimally) meets the provider-defined policy objectives.
- **Data plane translation:** The strategy determined by the optimization engine will ultimately need to be translated into a practical data plane realization. Here, we leverage recent advances in software-defined networking as a means to efficiently and effectively realize the intended routing and traffic engineering strategy. That being said, this translation layer needs to implement suitable mechanisms to ensure *correct* and *consistent* operations in the face of potentially frequent network reconfigurations.
- **Monitoring Engine:** As seen earlier, a key input to the optimization is the information about the "health" and "status" of the datacenter network. This status information includes two main components: (1) link-level information about the reliability of individual inter-FSO links and (2) measurements of observed traffic patterns at suitable granularities such as an inter-rack *traffic matrix* as well as views of "elephant" flows []. The management layer needs a scalable real-time monitoring engine that can provide this information.
- **Interfaces to users and applications:** Finally, we envision an API layer that the datacenter management layer exposes to its tenants to allow them to better leverage the new benefits of reconfigurability as well as enabling them to provide hints about the workload patterns and application-level strategies (e.g., are they using multipath TCP) that can inform the strategies used in the optimization and dataplane translation engines.

Our specific research agenda will focus on the first two aspects. For the monitoring engine, we plan to leverage past work on scalable traffic matrix inference for large-scale datacenters and more recent techniques for adaptively changing SDN-based monitoring strategies [**?**, **?**, **?**]. Similarly, for APIs to applications and end-host stacks, we will use prior work on suitable abstractions for applications to expose their traffic patterns [**?**, **?**] and exploiting path diversity in datacenters [**?**].

For the sake of clarity, we begin by assuming that this management layer has an out-of-band, but lightweight control network that can be used for configuration dissemination and data collection. We believe this is typical of many network settings today. Before, we conclude this section we also discuss a roadmap for relaxing this assumption.

5.1 Fast reconfiguration and traffic engineering

We begin by discussing the requirements of the Optimization Engine in LightSpeed management module.

5.1.1 Problem Context

At a high-level, we need to solve a a joint topology design and traffic engineering problem in contrast to prior work in network management that considers traffic engineering in isolation on fixed topologies. To explain the optimization problem that we need to solve and highlight why this problem is hard, we begin with an abstract integer linear program formulation in a static setting.

We consider a setting with N $Rack$s each equipped with M FSO devices. We use the index i to refer to a specific $Rack$ and j to refer to a specific FSO device. Thus, the subscript i, j will refer to a specific Rack-FSO combination. Let k denote a pair of racks and let us assume that we know some estimated traffic

matrix for every pair of racks denoted by T_k for the volume of traffic from k. Suppose the bandwidth of each inter-FSO link (if enabled) is C.

We need two make two key control decisions. First, we need to decide which pair of Rack-FSO links that need to be connected. Let $d_{(i,j \to i',j')}$ be a binary variable that takes the value 1 if we choose to add a link between $(i, j \to i', j')$. As discussed in Section **??**, not all links may be viable given the physical and geometry constraints. We use $CandidateLinks_i, j$ to denote the set of possible links as defined by the pre-configured topology chosen from Section **??**. Second, we need to decide an appropriate routing/traffic engineering strategy that uses the available topology to maximize some high-level objective (e.g., aggregate throughput, per-customer SLAs, latency etc). Let $f_{k,(i,j \to i',j')}$ denote the volume of the inter-rack traffic on k that flows on the edge $(i, j \to i', j')$. Given these control variables, we can model the problem as a Integer Linear Program (not shown) that combines the "edge selection" along with traditional max-flow like constraints for the routing aspects.

5.1.2 Challenges

First, this problem is both theoretically and practically intractable. Our experiments with state-of-art ILP solvers such as CPLEX shows that solving this problem even on a 20-node instance can take as much as XXX hours! Consequently, we need to look *approximation algorithms* that can work well in practice and can respond to network changes at the millisecond timescale. Second, in this abstract formulation, we consider a simple global objective to minimize the largest link utilization in the network. In practice, we also need to take into account more complex goals including metrics such as fairness across different $k$flows, the per-packet latency, as well as any offered SLAs that the cloud provider has in place with specific tenants.

Thus, our two main research challenges in the design of the optimization engine are:

Task 1: *We will develop fast, scalable optimal or near-optimal algorithms for dynamic topology reconfiguration and traffic engineering that can respond to traffic and routing dynamics at the scale of a few milliseconds. We will systematically investigate tradeoffs between multiple objective criterion for different traffic classes using real-world datacenter traces and application requirements.*

5.1.3 Proposed Approach

As discussed above, even the simplified optimization problem is hard. Thus, we need to "relax" our optimality requirements (hopefully within a small margin) to provide the necessary near real-time responsiveness. To this end, we propose to explore a three-pronged approach that have natural synergies and complement each other:

- *Approximation algorithms:* Our first plan of attack is to design practical and scalable approximation algorithms for the optimization problem. Specifically, we will design algorithms with provable approximation bounds that are near-optimal in practice and yet really fast to run. There are two natural classes of approaches that can apply to problems of this nature. The first is "greedy" solutions especially if the specific objective functions are submodular []. For instance, we can scope the problem to only look at the traffic matrix entries with the highest demands or alternatively constrain the number of links that will be reconfigured. Second, we will explore "randomized rounding" strategies that solve a relaxed linear program (i.e., converting the discrete $d_{(i,j \to i',j')}$ into fractionals) and then selectively choosing to "round" some of these fractions to 1.
- *Triggered updates:* In general, running a pure traffic engineering or routing optimization for a given topology will be much faster than the joint optimization; e.g., these are amenable to fast max-flow style solvers []. One natural heuristic here is to see if the new traffic demands can be adequately satisfied

simply by re-engineering the routing mechanism. That is, we do not reconfigure the topology but simply route the demands differently. We might be sacrificing performance here and thus one option is to check if a pure traffic engineering solution is sufficiently close to the optimal that could have been achieved if we ran the full optimization. At first glance, this seems like a chicken-or-egg problem because we need to run the ILP to determine what the optimal might have been! Fortunately, we do not need to run the ILP – we can simply use the solution of the relaxed LP version as a (possibly loose) bound[2] and compare the solution provided by the traffic engineering optimization to this bound.

- *Exploit real-world structure:* Our third plan of attack is to exploit the natural structure of the real-life workloads to inform these approximation algorithms and triggered update heuristics. In particular, data center workloads are known to have a skewed pattern of flow size distribution, with a small number of "elephant" flows contributing to a significant share of the total traffic volume []. Moreoever, these elephant flows are typically long-lived [] and thus amenable to coarser time-scale optimizations. Thus, one potentially promising approach is to exploit this insight and focus the reconfiguration logic on the elephants and use a traditional traffic engineering approach for the remaining traffic.

[VS: Add online + local optimization also]

5.2 Efficient and consistent data plane strategies

Task 2: *We will design and implement efficient data-plane implementations leveraging new software-defined networking techniques that can gaurantee "consistent" route reconfiguration algorithms as well as local recovery mechanisms that can guarantee packet reachability and/or path consistency properties*

5.2.1 Problem context

Next, we focus on the task of translating the solution provided by optimization engine into a practical data plane forwarding strategy. Here, we plan to leverage recent advances in software-defined networking (SDN) as an enabler to rapidly implement the topology and routing reconfiguration strategies []. This is in sharp contrast to traditional traffic engineering approaches that rely on slightly ad hoc mechanisms for tweaking routing weights or relying on local strategies like ECMP that may have undesirable convergence properties or may be subptimal in terms of the traffic engineering goals []. While SDN is indeed an enabler in that it provides cleaner management abstractions as well as open interfaces that can programmatically configure the routing behaviors of SDN-capable switches (e.g., via APIs such as OpenFlow []), we are envisioning a new use-case involving *topology engineering* that gives rise to unique efficiency, correctness, and consistency challenges that we plan to address as part of the proposed research. We begin with some brief background on SDN before highlighting the unique problems relevant to LightSpeed. (Readers familiar with SDN can skip to the challenges.)

SDN background: SDN is characterized by three key ideas: (1) decoupling the "control plane" of the network from the "data plane" (e.g., avoiding embedding the traffic engineering or access control policy into the topology or routing algorithms); (2) a logically centralized management layer that operates in the principle of "direct control"; and (3) the use of open APIs to configure the behaviors of data plane elements such as routers and switches. The de-facto standard for SDN today is embodied in the design of the OpenFlow protocol []. Switches supporting OpenFlow export a simple data plane abstraction to the controller – matching on specific "flow" header fields and taking specific actions (e.g., forward, drop, count). This "flowtable" or the set of actions can be programmatically configured by a SDN controller via the OpenFlow

[2] In case of maximization problems, the LP will be a upperbound and in case of minimization problems the LP solution will be a lower bound.

API. In this respect, the LightSpeed management layer mirrors the SDN philosophy – a logically centralized management module that is reconfiguring the topology and routing in response to workload changes. The goal of our data plane translation layer is to translate the optimization solution into a suitable set of forwarding rules compatible with the OpenFlow API. Thus, SDN is an obvious enabler for LightSpeed– it is quite challenging (and possibly infeasible) to realize such the fine-grained topology and traffic engineering we required on top of traditional routing protocols.

5.2.2 Challenges

LightSpeed introduces new aspects that are outside the scope of existing SDN solutions [] on two fronts. First, the network topology is constantly in a state of flux and will be actively modified by the controller. In contrast, most SDN work treats the topology as a "read-only" object that only changes on coarse timescales. Second, the nature of the FSO-based link-layer technologies in LightSpeed may introduce very fine timescale link unreliability issues that warrant mechanisms that need *local recovery* mechanisms that possibly cannot involve the controller.

To see why dynamic topology changes might cause problems, consider the scenario where LightSpeed has decided to reconfigure the network topology. In this scenario, the forwarding tables of the switches may have an inconsistent view of the topology. Our goal is to ensure that during this intermediate period when the topology is being changed, packets are not dropped due to forwarding inconsistencies, continue to receive reasonable performance, and we do not introduce adverse effects due to forwarding loops! At first glance, this is similar to recent work on *consistent updates* in SDN that provides a per-packet consistency property that guarantees that when the network is updated a packet is either processed with the previous or the current configuration but never by both [**?**, **?**]. While this is indeed a useful theoretical framework and abstraction, it has three key limitations in our context. First, it implicitly assumes that the network topology has not changed between the two update states to translate the per-packet consistency guarantee into a reachability guarantee. In LightSpeed, the topology might itself be changed during the update. Second, it does not provide guarantees on the bandwidth utilization or the throughput of the flows during the transition period, which in the case of a large datacenter can be on the order of **XXX** milliseconds [**?**]. Finally, as the authors themselves acknowledge it effectively requires double the "rule space" on the network switches to hold both sets of forwarding rules.

The second problem is that even with a static topology state, links may be transiently unavailable because of the "micro alignment" that each FSO device may need in order to re-establish connectivity. Now, the timescales of such micro-alignment may be much smaller than the time needed to communicate with an SDN controller and waiting for rule updates [**?**]. In fact, it may be counterproductive to report such transient link failures to the controller as it may cause needless topology reconfigurations. In addition to local detection and recovery mechanisms at the link layer as discussed in Section **??**, we also need corresponding network layer techniques to utilize this information.

[VS: how can we implement local recovery in sdn – look at the ddc paper?]

5.2.3 Proposed approach

While the general problem of reasoning about resource utilization in a consistent update model is quite hard [], we can employ domain- and problem-specific heuristics here since we are not trying to solve the more general update problem. To this end, we will investigate a combination of the following approaches that tackle two aspects of the problem *connectivity* and *performance*:

- *Guarantee reachability:* The first and perhaps simplest idea is to always guarantee end-to-end reachability even in the presence of topology reconfigurations. For instance, we can use a subset of the $k$links as part of the global optimization to be "always on" or statically aligned to create a connected graph

(i.e., a spanning tree) to ensure that all pairs of racks can always reach other. Of course, by itself it does not guarantee data plane reachability or resource guarantees which brings us to the next two strategies. Future SDN roadmaps have provisions for local recovery mechanisms analogous to similar schemes in the MPLS and SONET literature []. We will explore the available alternatives for our prototype implementation. In the absence of such features, we will explore the possibility of running local "lightweight" SDN controller on every rack that can quickly react to these changes but relies on the global controller for longer-timescale reconfigurations [**?**].

- *Prefer incremental reconfigurations:* As discussed earlier, existing consitent update abstractions do not provide mechanisms to reason about the resource usage. In order to minimize the impact that the reconfiguration can have on the overall network congestion, we will also explicitly incorporate some additional checks to ensure that "elephant" flows are minimally impacted. For instance, one way to ensure this is to add additional objective criteria that penalize reconfigurations that cause a signficant disruption for the current elephant flows/demands. Again, we believe that the enhanced flexibility that LightSpeed offers can be an enabler here – by providing more degrees of topological freedom it can help minimize disruptions for existing connections.

5.3 In-band control

Task 3: *We will design protocols and mechanisms that will provide robust in-band control and data collection even in the phase of dynamic reconfigurations.*

[VS: this might be relevant .. but need to think more]

6 Education and Broader Impact

7 Results From Prior NSF Support

Samir R. Das and **Himanshu Gupta** are PI/Co-PIs on the following recently concluded/ongoing NSF awards: i) 'A Market-Driven Approach to Dynamic Spectrum Sharing' (2008-13, $406,000), and ii) 'Understanding Traffic Dynamics in Cellular Data Networks and Applications to Resource Management,' (2011-14, $320,425). These projects focus on developing market-driven algorithms and systems for dynamic spectrum access systems (first) and understanding spatio-temporal traffic dynamics in cellular data networks via analysis of network traces and using them for spectrum/energy management applications (second). Over 15 papers were co-authored by the PIs related to these awards and 6 PhD students received direct support. The PIs gave several public lectures based on the results. [SD: if we have space we may also mention the sensor grants.].

Vyas Sekar is a PI on two recently awarded NSF grants "Enabling Flexible Middlebox Processing in the Cloud" and "Rethinking Security in the Era of Cloud Computing" starting in Sep 2013. The research proposed therein focuses largely on "middlebox" functionality such as IDS, firewall, and proxies and does not focus on the datacenter topology and routing aspects. These projects have just commenced and there are no outputs at this time. As such the proposed research in these projects does not overlap with the management layer/SDN approaches proposed here.

References

[1] `http://www.fsona.com/product.php?sec=2500e`.

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

[3] A. Curtis et al. Legup: using heterogeneity to reduce the cost of data center network upgrades. In *Co-NEXT*, 2010.

[4] A. Greenberg et al. Vl2: a scalable and flexible data center network. In *SIGCOMM*, 2009.

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

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

[7] J. Shin et al. On the feasibility of completely wireless datacenters. In *ANCS*, 2012.

[8] X. Zhou et al. Mirror mirror on the ceiling: flexible wireless links for data centers. In *SIGCOMM*, 2012.

[9] Nathan Farrington. Optics in data center network architecture. `http://nathanfarrington.com/papers/dissertation.pdf`.

[10] D. Kedar and S. Arnon. Urban optical wireless communication networks: the main challenges and possible solutions. *IEEE Communications Magazine*, 42(5), 2004.

[11] Jayaram Mudigonda, Praveen Yalagandula, and Jeffrey C. Mogul. Taming the Flying Cable Monster: A Topology Design and Optimization Framework for Data-Center Networks. In *Proc. USENIX ATC*, 2011.

[12] Luka Mustafa and Benn Thomsen. Reintroducing free-space optical technology to community wireless networks. In *Proc. 19th Americas Conference on Information Systems, Chicago, August, 2013.*, 2013.

[13] Radhika Niranjan Mysore et al. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In *Proc. ACM SIGCOMM*, 2009.

[14] L. Popa. A cost comparison of datacenter network architectures. In *Co-NEXT*, 2010.

[15] Ankit Singla et al. Proteus: a topology malleable data center network. In *HotNets*, 2010.

[16] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In *NSDI*, 2012.