RAIL: A Case for Redundant Arrays of Inexpensive Links in Data Center Networks

Danyang Zhuo[†] Monia Ghobadi Ratul Mahajan Amar Phanishayee
Xuan Kelvin Zou Hang Guan[*] Arvind Krishnamurthy[†] Thomas Anderson[†]

Microsoft Research [*] Columbia University [†] University of Washington

Abstract— While there are many proposals for new data center network (DCN) architectures to help reduce cost, little attention has been paid to physical links that carry packets. We study over 300K optical links across many production DCNs and find that these links are operating quite conservatively relative to the requirements in the IEEE standards. We experiment with various optical transceivers, a key contributor to DCN cost, and confirm that today's transceivers are over-engineered, which drives up their cost. Motivated by this observation, to reduce DCN costs, we explore using commodity transceivers beyond their currently specified limit. We show that today's transceivers' reach can be "stretched" 1.6 to 4 times their specification. With stretching, the performance of 1–5% of the DCN paths can fall below the IEEE standard. We develop RAIL, a system to ensure that in such a network, applications only use paths that meet their performance needs. Our proposal can reduce the network cost by up to 10% for 10Gbps networks and 44% for 40Gbps networks, without affecting the applications' performance.

1 Introduction

The relentless demand for capacity in data center networks (DCNs) makes their cost a primary metric of interest. Network cost optimization must consider all main contributors, including switches, links, and power. Several innovations in DCN architecture [18, 35, 55, 21, 24, 45, 37] have transformed the industry and reduced the cost of building and operating DCNs.

Yet the physical layer of DCNs (i.e., links) has received little attention, and it is being engineered using conservative practices that lead to a high cost. In modern DCNs, all inter-switch links tend to be optical, and the design objective is that *every* link should have a bit error rate (BER) lower than 10^{-12}.[1] To meet this objective, when manufacturers rate transceivers—devices that convert signals between electrical and optical domains—for a certain link length (e.g., 300m), they assume worst case operating conditions (e.g., temperature, signal attenuation due to connectors). These worst-case conditions are rare, and consequently, the vast majority of the links are significantly over-engineered—the optical signal quality is much higher than what is needed to support 10^{-12} BER.

To confirm and quantify physical layer over-engineering in today's DCNs, we conduct what to our knowledge is the first large-scale study of operational optical links. We analyze over 300K links across more than 20 data centers of a large cloud provider over a period of 10 months. We find a remarkably conservative state of affairs—99.9% of the links have incoming optical signal quality that is higher than the minimum threshold for 10^{-12} BER, while the median is 6 times higher!

This over-engineering is expensive—transceivers account for 48-72% of total DCN cost depending on link speed and link length distribution (§7)—and reducing it can lower network costs. While there are multiple ways to do so, we explore an approach that does not require any changes to existing hardware. Inspired by the approach of running data centers at higher temperatures than manufacturers' specifications for hardware [29], we suggest DCN operators use transceivers for link lengths that are greater than manufacturers' specifications. This approach can lower costs because transceivers with shorter length specification tend to be cheaper; thus, cheaper transceivers can be used for many links that ostensibly need more expensive ones today.

However, because of transceiver "stretching," some links in the DCN may have BER higher than 10^{-12}. Traffic on such *gray* links will experience higher loss rates because more packets will have bit errors (which switches will drop). But the application and path diversity of modern DCNs suggest the overall impact on application performance is likely to be negligible. Many applications do not need BER of 10^{-12} and can be carried over gray links without hurting their performance. Moreover, since the network has many paths between pairs of top-of-rack switches, applications requiring high reliability can be well supported by being routed through low-loss paths.

Two questions remain about our approach: *i*) how much can cost be reduced by "stretching" transceivers beyond their specifications? And *ii*) how can we ensure applications only use paths that meet their loss tolerance? To answer the first question, we perform extensive simulations and experiments with transceivers under realistic conditions. We show that depending on the technology (10 or 40

[1] The BER requirement of 10^{-12} was standardized by the IEEE in 2000 [11] and is likely a holdover from the telecom world. In reality, few DCN applications today need that level of BER. RDMA is perhaps the most BER-sensitive application today, and a BER of 10^{-10} suffices for it [60]. However, for the purposes of this paper, we assume some (future) applications will need 10^{-12} BER.

Gbps) and the desired upper bound on the fraction of gray paths (1% or 5%), current transceivers can be stretched from 1.6 to 4 times their specified length. At these design points, for a fat tree topology, and depending on the link length distribution, the cost of the network can be lowered by up to 10% for 10Gbps and 44% for 40Gbps networks.

We answer the second question by designing RAIL. It is a practical system that builds multiple virtual topologies on the same physical topology, where the class of the topology offers a bound on the maximum packet error rate (i.e., grayness) on any path in it. The first-class topology does not have any gray paths; hence, it offers the same path packet error rate guarantee as current DCN designs. Other classes increasingly use more gray links. Each application uses the virtual topology that meets its needs. Thus, loss-tolerant applications use virtual topologies that may have more gray paths. To support applications, such as large transfers that are otherwise loss-tolerant but suffer when the transport protocol (e.g., TCP) is sensitive to losses, RAIL uses a transparent coding-based error correction scheme. We develop an efficient algorithm to compute virtual topologies that leverages the hierarchical topological structure of DCNs (e.g., Clos). RAIL is easily deployable as it requires no changes to the switch or transceiver hardware.

We evaluate RAIL using simulations-based analysis and a testbed. Even at the maximum stretched reach level we consider, we find 95% of all paths are as reliable as today. Furthermore, RAIL protects loss-sensitive applications from gray paths.

2 Optical Links' Performance in the Wild

In today's DCNs, switches are arranged in a multi-tier topology such as a fat tree or folded Clos [55, 18]. While the switches are electrical, the links between them are optical because optical links are more cost-efficient at higher bandwidths and distances.

An optical link has transceivers at each end connected via a fiber cable. Transceivers plug into switches to convert electrical signals nto optical signals and vice versa. Figure 1 shows a transceiver and its internal components. It has a transmit and a receive pipeline, each attached to an independent fiber. The transmit side has a laser that emits light and a modulator that modulates it according to the electrical input from the switch. The receive pipeline decodes incoming optical signals using a photodetector and an amplifier.

The performance of an optical link is quantified using BER, the probability of a single bit being corrupted at the receiver. Corruption happens when the optical power gap between bits 0 and 1 is too small to be reliably differentiated at the receiver. The gap can be low because of poor signal characteristics at the transmitter (e.g., a poor-quality laser that spreads light over a wide spectrum) or because of attenuation and dispersion as light travels through fiber. Attenuation refers to a reduction in average

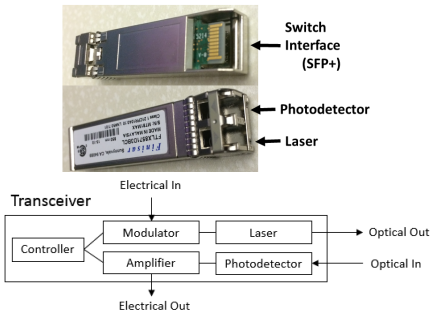


Figure 1: *The schematic of an optical transceiver.*

signal power, e.g., caused by the optical connectors attaching fibers to transceivers. Dispersion is the distortion of the shape of the signal.

To simplify manufacturing and deployment, IEEE standardizes transceiver characteristics. Apart from wavelength (e.g., 850, 1310, 1510 nm) and transmission speed (e.g., 10, 40, 100 Gbps), an important characteristic of each standard is the maximum reach at which a transceiver is guaranteed to deliver a BER of 10^{-12} or better.

Two main categories of optical technologies are used in DCNs today: multi-mode and single-mode. Multi-mode transceivers are cheaper because they have relaxed constraints on laser spectral width and coupling of laser with fiber. But they also have a shorter reach because they suffer from modal dispersion, i.e., signal distortion due to differing path lengths taken by light waves in the fiber.

2.1 Data Set

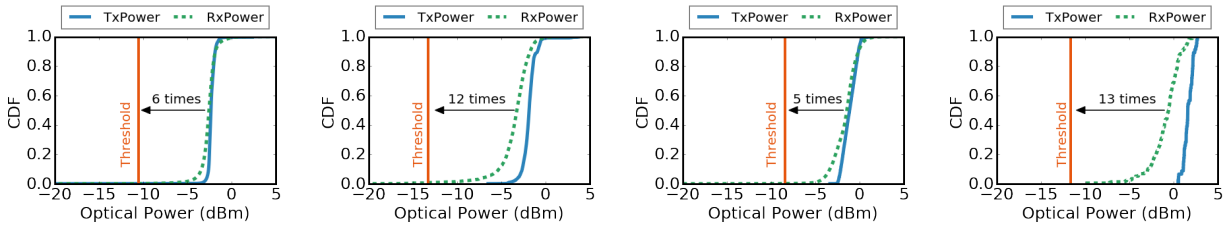
To shed light on the performance of the optical layer in today's DCNs, we build a monitoring system that uses SNMP optical MIB [12] to poll optical performance metrics from transceivers every five minutes. Our system runs in multiple production data centers and collects transceivers' transmit power (TxPower), receive power (RxPower), temperature, transmit bias current, and supply voltage. The TxPower and RxPower values reported by the transceivers are average (across bits 0 and 1) signal power values.

The data we report in this paper were collected over ten months from Nov 2015 to Aug 2016. They cover over 300K links (600K transceivers), with a mix of multi- and single- mode transceivers with 10, 40 and 100Gbps speeds.

2.2 Optical Power Levels Are Too Good

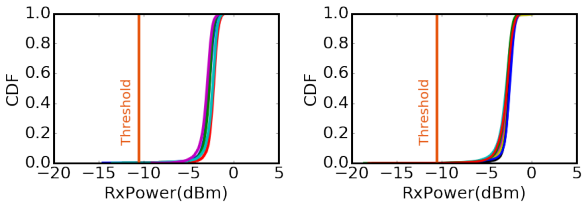
We begin our analysis by studying optical transmit and receive power levels. RxPower is a key indicator of optical layer performance. Modulo significant dispersion, it directly determines BER.[2] To keep BER under 10^{-12}, RxPower should be above the receiver sensitivity required by the IEEE standard.

[2]Most transceivers today do not report BER. Switches report packet error rate, but this measures the combined impact of errors from all sources, including electrical components.



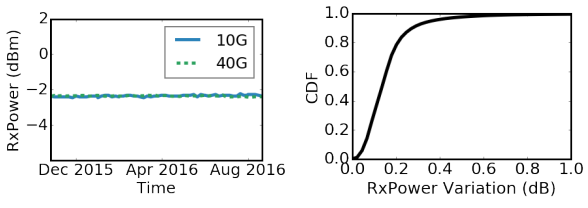
(a) 10Gbps multi-mode (b) 10Gbps single-mode (c) 40Gbps multi-mode (d) 100Gbps single-mode

Figure 2: *CDF of TxPower and RxPower for different speeds and modes. The vertical lines are the RxPower threshold needed to keep BER under* 10^{-12}. *Almost all links across all technologies have RxPower above the threshold.*



(a) *Different manufacturers* (b) *Different data centers*

Figure 3: *CDF of RxPower of 10Gbps multi-mode transceivers across five transceiver manufacturers and ten different data centers. Over-engineering is not limited to one manufacturer or data center.*



(a) *RxPower vs. time.* (b) *(max - min) RxPower*

Figure 4: *RxPower of individual transceivers has small variations over time. 10Gbps and 40Gbps lines are overlapping in the left graph.*

Degree of power over-engineering. Figure 2 shows the CDF of average TxPower and RxPower of all transceivers separated by their speed and mode (10Gbps multi- and single- mode, 40Gbps multi-mode and 100Gbps single-mode). The vertical line in each plot is the receive power threshold to keep BER under 10^{-12}. We see that across the board nearly all links (99.9%) have RxPower above the threshold. In fact, 99% of transceivers are at least 5 dB (i.e., 3 times) above it. Thus, when we use the IEEE standard as our basis of comparison, we see the vast majority of links are greatly over-engineered from an optical power perspective.

Over-engineering across transceiver models. We find this degree of over-engineering is consistent across transceiver manufacturers. Our data include over 50 transceiver models across various vendors. Figure 3a plots the CDF of RxPower of 10Gbps multi-mode transceivers across five transceiver manufacturers with more than 1000 transceivers inside one data center. For confidentiality, we do not report the name of our manufacturers. All five manufacturers exhibit similar RxPower distributions, and over-engineering is not tied to one manufacturer.

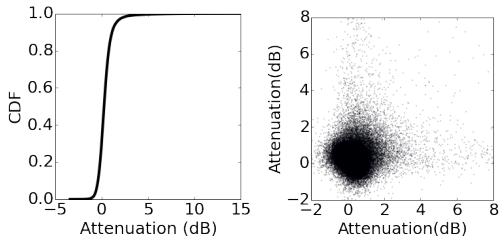
Over-engineering across data centers. We also study whether over-engineering varies by data center. Figure 3b plots power levels of the ten largest data centers in our dataset. Each color represents a data center. We see that RxPower distributions and thus over-engineering levels are similar for every data center.

Optical power variation over time. Over the course of our study, we found power levels of a link vary little over time. Figure 4a illustrates RxPower over time for two sample transceivers. As shown, for each transceiver, RxPower remains mostly stationary over time. Figure 4b plots the CDF of maximum minus minimum RxPower value for all transceivers. The figure shows 78% of the links have a variation below 0.2 dB, and over 99% have a variation below 0.8 dB. Thus, links with high RxPower have consistently high RxPower, instead of power levels dropping intermittently (which would cause high BER during those times if we were to reduce their over-engineering).

2.3 Understanding Low Optical Power

In Figure 2, unlike TxPower, RxPower has a long tail, suggesting that a small fraction of links experience high attenuation and thus low RxPower. Figure 5a shows this effect directly by plotting the CDF of attenuation for all multi-mode links; single-mode results are similar. We compute attenuation as TxPower at the sender minus RxPower at the receiver. The figure shows most links have attenuation close to zero—the median is 0.26dB—but 0.44% have attenuation higher than 5dB.

Initially, we thought high attenuation would correspond to links that are long or have many optical connectors. To confirm this supposition, we studied *attenuation symmetry* between the two directions of a bi-directional optical link. If link length or connector count are to blame, because these factors are identical in both directions, high attenuation should be symmetric. Figure 5b shows a scatter plot where the two axes represent attenuation levels in different directions. We see that attenuation is low and symmetric



(a) *CDF of Attenuation* (b) *Bi-directional Attenuation*

Figure 5: *Attenuation is nearly zero and symmetric for most links. Interestingly, high attenuation is asymmetric.*

for most links.[3] But it is asymmetric when it is high. When one side has high attenuation, the other side does not.

This behavior suggests that high attenuation does not stem from link length or connector count but due to poor or dirty connectors or fiber damage—connectors and fibers are different in the two directions. To confirm this hypothesis, we analyzed hundreds of repair records for poor optical links. We found two main root causes for low RxPower: (i) dirty fiber connectors (which must be cleaned to fix the problem); (ii) damage to the fiber's cladding, the outside layer of the fiber that protects the actual core. The finding that high attenuation rarely stems from long links is consistent with our later experiments showing current transceivers have low BER even on links that are longer than their specification.

Dependence on link location. We also find that low RxPower links are scattered across the data center uniformly and randomly, and they are not correlated with a specific switch brand or topology tier. To confirm, we compute the percentage of switches having links with RxPower at the bottom 0.01% of Figure 2. We then uniformly and randomly select 0.01% of links with any RxPowers and again compute the percentage of switches to which they belong. If the two numbers match, it suggests links with RxPower at the tail are scattered uniformly and randomly across switches. We repeat this analysis for 100 values between 0.01%-tile and 1%-tile tail and observe the same result. Figure 6 shows that the numbers based on this independence assumption closely match the data. If some switches were more likely to have links with low power levels, the "Low RxPower data" curve would be consistently lower than the "Uniform Random" curve. This observation, together with our observation on the root causes of high attenuation, suggests that dirty connectors and damaged fiber can show up anywhere in the data center.

3 Reducing Over-engineering and Cost

Our measurements above reveal that the optical layer of DCNs today is heavily over-engineered. This over-

[3] Attenuation levels below 0 are due to (unbiased) calibration error in TxPower and RxPower sensors. The reported power is within $\pm$ 2 dB of the actual value.

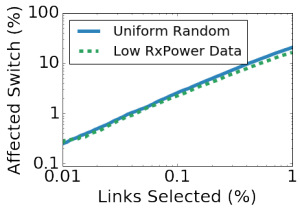


Figure 6: *Links with low RxPower are uniformly and randomly scattered across switches.*

engineering does not come for free; DCNs are expensive to build, with transceivers accounting for 48-72% of the network cost, depending on link speed and link length distribution (Table 1). We shared our measurement results with experts in transceiver manufacturing companies and learned of many possibilities to lower the price of transceivers (e.g., changing transceiver hardware, relaxing test requirements, reducing labor and packaging cost).

However, we also learned that immediate realization is difficult because of two inter-twined challenges. First, transceiver manufacturers cannot reliably estimate cost savings without carefully crafting and optimizing the entire manufacturing chain, and they are reluctant to do so without standardization or firm commitments from DCN operators. Second, since any method of reducing over-engineering can cause some gray links (with BER higher than 10^{-12}), DCN operators are reluctant to commit unless the cost savings are known in advance to be high, and there is a way to protect sensitive applications against gray links.

In this paper, we solve both challenges. First, we demonstrate immediate cost savings are possible by allowing commodity transceivers to be "stretched," that is, using them for longer than their currently-specified distances. Second, we devise a system for routing and forwarding packets where some links may be gray. Leveraging application and path diversity in DCNs enables applications to use paths according to their loss-tolerance (§4).

Below, we explain how stretching reduces cost and show how to engineer a stretched network.

Cost reduction through stretching. Figure 7 shows the price and specified reach of different standard transceiver technologies available in the market for 10, 40, and 100Gbps. The price of 10 and 40Gbps transceivers represents the average across three volume retailers [14, 13, 7] and the price of 100Gbps transceivers is from one retailer [10]. As shown, the standard includes discrete reaches, with the price of transceivers increasing as reach is increased. Longer reach requires more expensive components (e.g., narrow spectrum laser, cooling module) and manufacturing processes.

Figure 8 illustrates how stretching transceivers reduces cost. The solid line is price for standard reach, and the dashed line shows the stretched reach. When the reach is stretched from R_1 to R_1', we don't have to pay more to use

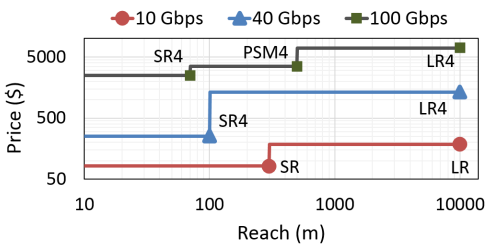


Figure 7: *Price and reach of standard transceivers for 10Gbps, 40Gbps and 100Gbps technologies. Both X and Y axes are in log scale. The label beside each data point is the name of the technology. PSM4 [16] is not yet an IEEE standard but is an agreement between 12 companies.*

an R_2-rated transceiver for distances between R_1 and R'_1 and, hence, we can save cost. The exact savings depends on how much transceivers can be stretched and the relative pricing of different options. We quantify the savings for available standard technologies in §7.

We don't expect transceivers' lifetimes to be shortened by stretching because lasers are designed for a long life-time, and TxPower does not decrease over time (96.7% of transceivers' TxPower does not change more than 0.2 dB over our 10-month measurement period).

Determining how much to stretch. To determine how much transceivers in a DCN can be safely stretched, we use a metric called *network reliability bound* (NRB). NRB is a lower bound on the expected fraction of paths that the DCN administrator desires to be good (i.e., those where all links have BER below 10^{-12}). We compute the maximum stretch for a transceiver based on NRB and *i*) the maximum number of links in a DCN path, and *ii*) the BER distribution expected for different stretch levels. We show in §7 how to compute this distribution using expected attenuation distribution.

This computation is best illustrated in an example. Suppose our goal is for NRB to be 95%, which can be translated to each path being good with a probability of at least 95%. In a 3-stage Clos network, where the maximum number of hops is four, this goal can be met if each link is good with probability of $\sqrt[4]{95\%} = 99.7\%$ (assuming that links being good or bad is independent of location, as we showed in §2.3). From the BER distribution for different stretch levels, we can now determine the stretch at which 99.7% of the links will be good. We use this value as the maximum stretch for a transceiver. In reality, the network will have more than 99.7% good links because many links where "stretched" transceivers will be used are shorter than the maximum stretch.

Need for software layer protection. NRB enforces a lower bound on the fraction of good paths, allowing a small fraction of gray paths. To preserve application performance, a naive solution is to simply turn gray links off and thus remove all gray paths. But even if

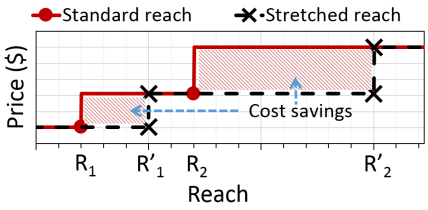


Figure 8: *Illustration of how stretching transceivers reduces cost. When R1-rated transceiver is stretched from R_1 to R'_1, we do not have to use (the more expensive) R_2-rated transceiver for link lengths between R_1 and R'_1.*

the fraction of gray links is small, turning them off can result in 20 to 50% capacity loss for certain ToR-pairs (§7). Instead, we propose that a software system be used to effectively utilize such gray links, while protecting loss-sensitive applications. The design of RAIL, described next, accomplishes this goal.

4 Overview of RAIL

RAIL is a system for routing and forwarding in a DCN, where *i*) links have a range of link packet error rate (LPER); and *ii*) applications have different requirements for path packet error rate (PPER). In such a network, RAIL ensures an application is routed only through *paths* with its desired reliability level (or better). Thus, sensitive applications (i.e., RDMA-based ones) are routed only along the most reliable paths while tolerant ones (i.e., UDP-based ones) can be routed through any path. RAIL is a general solution that is agnostic to why gray links occur; they could be cause by any method of reducing over-engineering, including stretched transceivers, cheaper hardware or cheaper fiber.

We want our solution to be: *i*) readily deployable, i.e., requiring minimal changes to current infrastructure; and *ii*) practical, i.e., having low overhead and complexity. To appreciate these constraints, we consider two extreme design possibilities with respect to how much hosts need to know about the network. The first is source routing, in which hosts are responsible for selecting paths (i.e., sequence of links to the destination) through the network that meet application needs. The challenge with this approach is that hosts will need an up-to-date view of network topology and LPER. With hundreds of thousands of links and frequent link failures [34], maintaining such a view at every host is highly complex.

In the second design, hosts are not told anything about the network (as is the case today). To deliver reliable communication, they use network coding transparently; i.e., application traffic to a given destination is coded such that it can withstand some fraction of loss. The problem with this approach is that it sets up an unwanted trade-off between coding overhead and the loss rate experienced by traffic. Hosts do not know in advance which path a given flow might take (due to ECMP routing). If they encode every flow to a level that guarantees the most sensitive

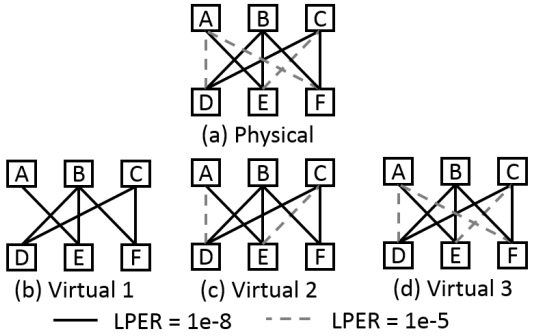


Figure 9: *Example of RAIL's virtual topologies. Solid lines are links having* 10^{-8} *LPER. Dashed gray lines are links having* 10^{-5} *LPER. RAIL maintains three virtual topologies (b, c, d) depending on Path Packet Error Rate (PPER).*

applications do not suffer (i.e., encoding based on the least reliable path), the coding overhead will be onerous for most paths. If they encode based on the average path loss rate, sensitive applications that traverse worse paths will suffer. It is possible to adapt by learning path loss rate based on what is actually experienced by a flow, but this is complicated by the fact that most flows are short and do not generate enough packets for robust learning.

To be deployable and practical, RAIL explores a design point in the middle. On the same physical topology, it builds *k virtual topologies*. Each topology falls in a different *class* representing a bound on the worst Path Packet Error Rate (PPER) in the topology. PPER represents the packet corruption probability for a complete ToR-to-ToR path and is derived from each hop's LPER. The first-class topology is made exclusively of the most reliable paths in the physical topology, the second-class topology permits all paths with PPER less than a bound, and the *k*-th class topology may include all paths in the physical topology. Figure 9 illustrates this concept with *k*=3, that is, three virtual topologies. Within a virtual topology, routing and forwarding will follow the same protocol as DC operators prefer today for their physical topology (e.g., ECMP over equal hop paths). Figure 9a shows the physical topology with two types of links: some with LPER 10^{-8} and three gray links with LPER 10^{-5}. RAIL creates three virtual topologies shown in Figure 9b, c, and d. Virtual topology (b) guarantees PPER of 2×10^{-8}. Virtual topology (c) guarantees PPER of 1×10^{-5} since its worst paths are A-D and C-E. Virtual topology (d) guarantees PPER of 2×10^{-5} since its worst path is D-A-F. Thresholds of these virtual topologies are configurable.

Virtual topologies are exposed to end hosts as different (virtual) interfaces. Applications (or the hypervisor on their behalf) bind to the interface that reflects their reliability criteria. Thus, the most-sensitive applications (e.g., RDMA-based ones) may bind to the interface for the first-class topology, TCP flows may bind to the second-class topology, and bulk transfers may bind to the *k*-class topology. Beyond making this choice, hosts are not aware of RAIL.

Applications that bind to less reliable topologies may be already robust to small amounts of loss they may experience, or they can be made that way by using a version of TCP that is robust to corruption-based losses [22, 44]. If this is not the case, however, RAIL includes a module that uses coding to transparently enhance traffic reliability.[4] Designing this module is an easier task than the coding-based option outlined above; flows going over less reliable topologies are likely longer, giving our module a chance to learn the path being used and select an appropriate coding level.

In the following section, we describe RAIL's design in more detail. As will become clear, it requires no changes to existing switches and only small changes to the host software, and then only if the error correction module is used.

5 RAIL in Detail

This section describes virtual topologies, routing and forwarding, and error correction mechanisms in RAIL. It also provides guidelines on configuring RAIL.

5.1 Virtual Topology Construction

Virtual topologies in RAIL share the same physical fabric but offer different guarantees for the maximum packet corruption rate along any of their paths. Our task is to build K virtual topologies, where the k-th topology offers the worst-case PPER of L_k. K and $\{L_k\}$ are selected by DCN operators based on their applications (see §5.4).

These topologies are built by the RAIL controller, which maintains an up-to-date view of each link's LPER by polling the switches. Since link qualities are relatively static (§2), maintaining such a view is straightforward. In addition to providing a worst-case PPER guarantee, we want each virtual topology to include as many links as possible to maximize its capacity. Because optimally finding such a topology is computationally expensive, we resort to a fast, greedy algorithm. Speed is important because, while link qualities are relatively static, links do fail frequently [34], and we need to recompute virtual topologies when links fail or recover.

Our algorithm has K rounds, one per topology. In each round, it starts with a set of candidate links for the topology and iteratively removes low reliability links until the required guarantee can be met. We start with the K-th class (least reliable) topology. For it, all links (that are currently alive) are initial candidates. We find the ToR-to-ToR path with the worst PPER, and if that PPER is higher than the required bound, we remove the link (on the path) with the worst LPER. Such link-removal iterations are repeated until the worst-case path meets the required bound. We then begin the next round, for the topology that

[4] Retransmitting lost packets is another (more efficient in terms of byte overhead) way to improve traffic reliability. We explore coding because it offers a faster, proactive way to recover lost data.

is one class lower, starting with links not removed in the previous round.

The speed of the above algorithm depends on how quickly we can check whether a topology meets a PPER requirement. Tracking every path's PER takes at least $O(n^3)$ in a n-ToR Clos topology (as there are $O(n^2)$ ToR pairs with $O(n)$ paths between each pair). Such running time is computationally infeasible on realistic network controllers. Our algorithm speeds up this running time to $O(n \log n)$ and is linear in the number of links in the topology. The intuition is that because of a DCN's highly structured topology and its simple routing scheme, we can track the worst path without tracking the PPER on every path.

Our algorithm goes through each switch exactly once and computes three values on each switch, from the bottom stage all the way to the top stage. The three values computed on switch s are (1) up_s: PPER for the worst monotonic upward path from any ToR to s, (2) $down_s$: PPER for the worst monotonic downward path from s to any ToR, and (3) top_s: PPER for the worst up-down path for which s is the highest stage switch.

Definition 1. *Children of switch s are the set of switches on a lower stage than s, with direct links to s.*

$up_s \leftarrow 0$
$down_s \leftarrow 0$
$top_s \leftarrow 0$
for $c \in Children(s)$ **do**
  $up_s \leftarrow max(up_s, 1-(1-up_c)(1-LPER_{c\to s})$
  $down_s \leftarrow max(down_s, 1-(1-LPER_{s\to c})(1-down_c))$
end
for $c,d \in Children(s),\ c \neq d$ **do**
  $tops_s \leftarrow max(top_s, 1-$
    $(1-up_c)(1-LPER_{c\to s})(1-LPER_{s\to d})(1-down_d))$
end

Algorithm 1: Update up_s, $down_s$, top_s on switch s.

After those three numbers are calculated, our algorithm outputs the worst path by picking the worst top_s among all switches in the network. The worst PPER is simply $max_{s \in all\ switches}(top_s)$. The correctness proof and running-time analysis appear in Appendix B.

5.2 Routing and Forwarding

To simplify routing and forwarding, we use a non-overlapping IP address space within each virtual topology. We configure switches such that, when routing or forwarding for a topology, they ignore links that are not part of that topology. The exact mechanism will depend on the routing paradigm used by the data center. If the data center uses a distributed protocol such as BGP to compute paths [41], we configure BGP to not announce prefixes for a virtual topology over links that are not part of it. No RAIL-specific changes are made to switch software, and they will forward packets as they do today (e.g., ECMP).

If the data center uses an SDN controller to centrally compute forwarding paths, we can either instantiate one controller per virtual topology or use one network-wide controller programmed to not use certain links for given prefixes.

Our approach may create uneven load on links because different links are part of different topologies. Load is uneven even without our modifications, however, and traffic engineering is required to balance it [59, 19, 52]. We leave the task of extending traffic engineering to account for virtual topologies for future work.

5.3 Error Correction

When the application or transport protocol is not robust to small amounts of PPER (corruption-based loss), RAIL's error correction module can be used to guarantee high performance in exchange for slight bandwidth overhead. This module is completely transparent to applications.

As argued earlier, given the diversity of PPERs across paths, it is important that error correction be based on the PPER of each path, rather than being guided by the worst-case or average PPER in the virtual topology. Recall that because of ECMP-hashing, hosts are not aware of the path taken by a flow.

RAIL's error correction module learns the PPER in two steps. First, as soon as a new flow starts, the source host sends a `traceroute` probe with a header (5-tuple) identical to that of the flow. This probe reveals the path taken by the flow. We use special DSCP bits in the IP header of the probe packet to indicate to the destination host module that it should not deliver the packet to the application. Second, the error correction module queries RAIL's controller for PPER of the path.

Ideally, the error correction module should be able to use bit-level forward error correction (FEC) for each packet. However, this approach does not work in practice because today's switches drop packets when CRC checksum fails. As we are seeking an immediately deployable solution, we do not consider this option. The main benefit is that bit-level FEC has low coding overheads. As we show in §7, the bandwidth overhead of our error correct module is already low enough.

RAIL sends "parity" packets after every n data packets, where n is based on the path packet loss rate (see below). We use XOR encoding because it is lightweight and known to be effective [53]. That is, after every n data packets, the sender sends a packet whose content is the XOR of the previous n packets. In this coding scheme, as long as n out of the $n+1$ packets are successfully delivered, the receiver can recover the original n packets. Losing two or more packets within a group of $n+1$ packets results in data loss.

If PPER is p, the probability of having two or more losses among $n+1$ packets is $1-(1-p)^{n+1}-(n+1)p(1-p)^n$. We pick n such that this probability is lower than the desired post-recovery loss probability t (experienced by applications). Any path with $p < t$ does not use any error correction. To show an

example of computing n, we first quantify t for a particular transport. Suppose TCP's performance degrades when the loss rate is above 0.1%; therefore, we can pick t=0.1%. For a path loss rate of $p = 0.3\%$, we would choose n to be 14 so that the post-recovery loss rate is again $0.092\% < 0.1\%$. The bandwidth overhead in this case is 7.1%.

For a given virtual topology, we include all paths meeting the loss criteria; thus, most paths are as reliable as the IEEE standard requires. Even if coding overhead is high for a particular flow, the average overhead will be small. We show later (§7) that the number of good paths for which error correction is unnecessary dominates the total number of paths.

5.4 Configuring RAIL

While it is up to individual operators to configure the number and loss rate guarantee of virtual topologies in RAIL, based on their applications, we offer a simple recommendation. It is based on the double observation that many applications use TCP, and the performance of some TCP variants degrade noticeably only when loss rate exceeds 0.1% [48, 26]. We see this behavior in our experiments, and it is consistent with what others have reported. That is why some operators completely switch off links with error rates higher than 0.1% [58].

We recommend that data centers be configured with three virtual topologies. The first-class topology should provide paths with the same reliability as today, equivalent to where each link has BER less than 10^{-12}. This can be used to carry the most sensitive applications, such as RDMA. The second-class topology should provide paths with PPER below 0.1%, and it should carry applications like short TCP flows. Finally, the third-class topology should provide paths with PPER below 10%, and unless the application uses loss-tolerant transport, it should be error corrected. When the application is a long TCP flow, it should be error corrected to a reliability of t=0.1%. RAIL always put RDMA traffic on first-class topology and never uses error-correction code with RDMA. A corollary of our recommendation is that any link with PPER above 10% will be turned off entirely. Such links are rare (§7).

6 Implementation

Our implementation of the RAIL controller and the error correction module includes the following features. The controller learns link PERs from CRC error counters. It does not distinguish between optical- versus electrical-related corruption and provides protection from both. It constructs three virtual topologies with worst-case PPER of 0.0001%, 0.1% and 10%. The controller computes routing tables globally based on the virtual topologies and pushes rules to switches accordingly.

Our implementation of error correction sits below the kernel TCP/IP stack so that it is oblivious to the transport protocol. We implement it as a driver for tun/tap device in the Linux kernel. (On Windows, a WinSock kernel device driver may be used.) The driver keeps a buffer of size n so that it can decode the coded packet if needed and deliver packets to higher layers in order. It keeps a fine-grained timer such that if a missing packet is not recovered within a short time window, the next packet is delivered to the transport protocol (e.g., TCP). This delivery may trigger a recovery at the transport layer. Such transport-layer retransmissions are new packets for our error module.

To identify packets for coding and decoding, we insert a 4-byte header after the IP header containing a sequence number. Once the encoding rate is negotiated, coded packets and data packets have separate sequence numbers. The parity packet has the last sequence number in each group of n. As the error correction module knows the exact path to the destination, it performs cross flow error correction among all the flows with the same path.

7 Evaluation

Our evaluations are divided into three categories. First, we use a testbed and an optical simulator to quantify the stretch of two widely used short reach transceivers and compute the resulting BER and potential cost savings (§7.1). Next, we evaluate the impact of stretching these transceivers on the overall network path quality (§7.2). Finally, we show RAIL preserves applications' performance in a network with gray links (§7.3). Our results demonstrate that RAIL has minimal impact on overall network quality and application performance, while reducing total network cost by up to 10% for 10Gbps networks and up to 44% for 40Gbps networks.

7.1 Stretching Existing Technologies

We experiment with eight IEEE-standard short-reach 10Gbps and 40Gbps transceivers (two brands for each speed and two units of each brand). Some manufacturers provide non-standard technologies with reach values that are not supported by IEEE. These transceivers can likely be stretched as well if they follow similar specification practices, but evaluating individual non-standard transceivers is beyond the scope of this paper. Our goal rather is to demonstrate that commodity transceivers are over-engineered and can be stretched beyond their specification. We also experimented with a standard 100Gbps transceiver. Those results are preliminary and appear in Appendix C.

Experimental methodology. Our stretch experiments are based on a testbed and an optical simulator. Our testbed has one 10G [2] and one 40/100G switch [3], four 10G-SR transceivers [8, 4], four 40G-SR4 transceivers [9, 4] and a set of OM3 fibers [5] of lengths between 10m to 1000m. We focus on SR (short reach) technologies as they are viable candidates for stretching; the reach of LR (long reach) technologies is longer than typical maximum DC link lengths.

To emulate long fibers of different lengths, as shown in Figure 10, we concatenate multiple short cables with

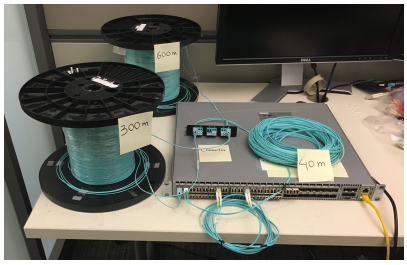
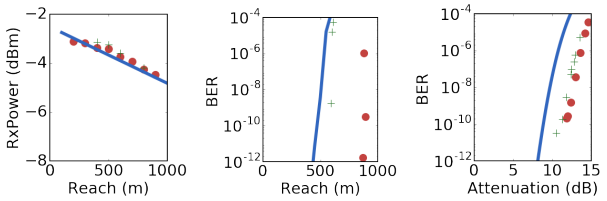


Figure 10: *Our 10G-SR testbed. We stitch short fibers together with fiber connectors to emulate long fibers. We concatenate 600m, 300m, 40m fibers together to emulate a 940m fiber. Two 300m 10G-SR Finisar transceivers [8] are attached to the fiber on both ends. Switch ports are on and we can send traffic through with BER of around* 10^{-8}.



(a) *RxPower vs. reach* (b) *BER vs. reach* (c) *Attenuation vs. BER*

Figure 11: *Validation for 40G-SR4 on OM3 fiber. Blue line corresponds to simulations. Circles and crosses are testbed results for transceivers from different manufacturers. (a) RxPower as a function of fiber length (no added attenuation). (b) BER as a function of fiber length (no added attenuation). (c) BER as a function of added attenuation when fiber is at transceiver's specified reach.*

fiber connectors. To emulate additional attenuation in real environments, due to more/dirty connectors or damaged fiber, we insert a variable attenuator which adds additional attenuation from the distribution shown in Figure 5a.

Since our transceivers do not directly report BER, to infer their BER, we measure packet corruption rate (LPER) with full-sized, line rate traffic for five minutes. We then calculate BER from LPER using a simplified model:[5] $LPER = 1-(1-BER)^{PACKET\ SIZE}$.

Testbed experiments are useful to provide coarse data on how link BER changes with different link lengths and attenuation levels. To explore the parameter space in fine granularity and to eliminate hardware quality differences between manufacturers, we use VPI [15], a standard optical simulator for data transmission system. (We cannot use a close-form formula to compute BER based on fiber length because of complex dispersion effects.) Our simulations model laser characteristics and a laser driver in the sender, modal and chromatic dispersion in the fiber, loss on the connector, and receiver sensitivity and dispersion equalization chips in the receiver. We configure these parameters based on the transceivers' specification sheet.[6]

For both 10Gbps and 40Gbps, we validate our simulator along three dimensions: RxPower, BER, and attenuation. Figure 11 shows our validation results for 40G-SR4; the results are similar for 10G-SR4. Figure 11a shows RxPower as fiber length increases. The scatter dots are testbed results, and the solid line is the result of our simulator. The figure shows that the simulator is able to closely match the RxPower of both transceiver brands for all fiber lengths. Figure 11b shows BER as fiber length increases. Since BER depends on a transceiver's sensitivity to modal dispersion, the two transceivers do not necessarily have to match. Hence, we configure our simulator to be the most sensitive one of the two brands and use the more conservative results in our evaluations for the rest of this paper. Similarly, Figure 11c shows that our simulations capture BER vs. additional attenuation conservatively. In this experiment, we use a 100m fiber, the design limit of our 40G-SR4 transceivers, and introduce additional attenuation using a variable attenuator.

BER distribution versus link length. We can now derive the distribution of BER that a link will observe as a function of its length. This distribution is a function of both link length and additional attenuation caused by dirty connectors or damaged fiber in real deployments. To simulate how stretched links impact BER in real world, we add the attenuation distribution seen in our measurement data in §2. This method is conservative because it assumes that all the attenuation measured in the wild is caused by factors other than link length. In practice, link length contributes as well, and our simulator already includes link length.

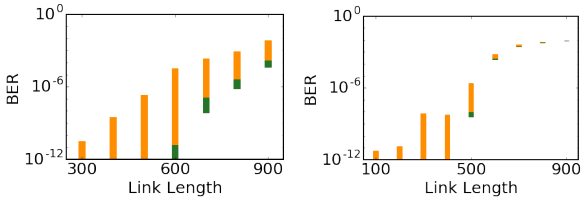
Figure 12 shows the BER distribution that will occur for various link lengths. The BER is represented as a bar for each link length. The top of the bar represents when extra attenuation is 99.9%-tile value (from the attenuation data) and the middle, which separates the two colors, when it is 99%-tile. As the figure shows, when we use 10G-SR, which is rated for 300m, on 500m links, at least 99% of these links will have BER less than 10^{-12}. At the same performance level, we can stretch 40G-SR4, which is rated for 100m, to 400m.

Cost savings. The level of stretch that a network should use depends on the trade-off between cost savings and performance, which we quantify using the NRB metric defined in §3. More stretch means more cost savings, but it also means that a larger fraction of paths is gray.

We illustrate this trade-off using a standard 3-stage Clos network. A DCN's total cost includes the equipment costs (i.e., transceivers, fibers, switches) and switch power consumption. Switches are $90/port [50], multi-mode

[5]This model may overestimate BER, and this would underestimate potential for stretch because of two factors. First, some packet corruption events may be due to non-optical issues. Second, Ethernet uses line code (e.g., 64b/66b for 10Gbps network), and any bit flip in the code causes an extra 2 bits to be corrupted in the future, possibly in subsequent packets.

[6]Our simulation files are available online [17]; other researchers can use these to simulate optical links in DCNs.



(a) *10G-SR* (b) *40G-SR4*

Figure 12: *BER distribution for different link lengths. The top of the orange bar is 99.9 percentile BER, and the bottom of the green bar is 0 percentile BER. The two colors meet at 99% BER. The green portion is not visible when BER is below* 10^{-12}.

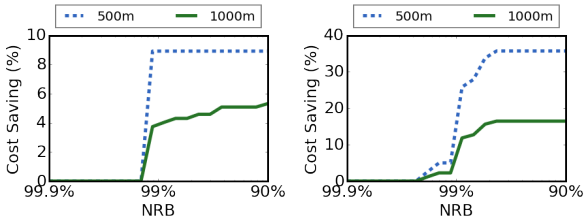
Device	Unit Cost	Count	Total Cost
300m 10G-SR Transceiver	$82.3	19660	$1.6M
10km 10G-LR Transceiver	$187.2	13108	$2.5M
Multi-mode Fiber (10Gbps)	$0.44/m	1475 km	$0.6M
Single-mode Fiber (10Gbps)	$0.21/m	2621 km	$0.6M
Switch (10Gbps)	$90/port	32768 ports	$2.9M
Power(150W/Switch, 3 years)	$0.07/KWh	5 GWh	$0.4M

Table 1: *Total DCN cost breakdown for 512 ToR, 512 aggregation and 256 core switches Clos network with 10Gbps technology. Link length is drawn from uniform distribution between 0-500m. With 10Gbps technology, transceivers account for 48% of the total DCN cost.*

and single-mode fiber cost $0.44 and $0.21 per meter respectively [6]. Each switch consumes around 150 Watts (this includes energy consumed by transceivers). With 32-port switches, we can build a full bisection 3-stage fat tree network with 512 ToR, 512 aggregation, and 256 core switches. There are 8192 ToR-aggregation links and 8192 aggregation-core links. The link length distribution depends on the physical layout of the DCN. For links under 300m, 10G-SR is used with multi-mode fiber. For links above 300m, 10G-LR is used with single-mode fiber. Table 1 provides a breakdown of the cost of these DCNs. As the table shows, transceivers represent 48% of the total cost of the network. If the same network were using a 300m 10GBase-SR transceiver stretched to 500m (for example), all the 300–500m links could use this transceiver instead of the more expensive 10GBase-LR one.

Figure 13 shows the cost reduction versus NRB for 0-500m uniform and 0-1000m uniform link length distributions. For these plots, we compute the stretch level from NRB as outlined in §3 and then the cost savings from the stretch level. Our measurement data (§2) show NRB in current networks is 99.9%. We see that cost savings are significant when NRB drops to 99%, and the incremental gain is small beyond 95%. Thus, in later evaluations, we only consider two degrees of stretch, low stretch for NRB=99% and high stretch for NRB=95%, which correspond to stretch levels in Table 2.

Next, we study how link length distribution affects the amount of cost savings. Figure 14a shows the resulting cost



(a) *10Gbps* (b) *40Gbps*

Figure 13: *Total DCN cost reduction on a 3-stage fat tree network (10Gbps, 40Gbps) with 512 ToRs assuming 0-500m and 0-1000m uniform link length distribution. The cost reduction depends on the amount of stretch.*

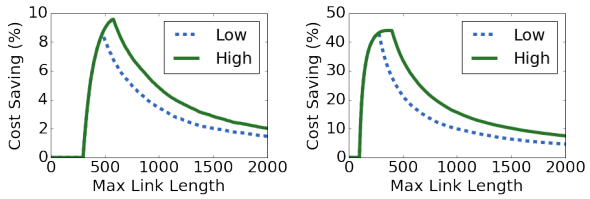
Technology	No Stretch	Low	High
10G-SR (OM3)	300m	480m	580m
40G-SR4 (OM3)	100m	280m	400m

Table 2: *Optical technologies' maximum reach under different degrees of stretch.*

savings for a 10Gbps network. When no link is longer than 300m, stretch does not result in any savings because we never use stretched technologies. The cost saving peaks at 480m for low stretch and 580m for high stretch, the lengths at which most fractions of links can use shorter reach technologies beyond their design reach. When link length distribution concentrates on long links, the amount of cost savings decreases because most of the links need to use long reach technologies anyway, and cost on long reach technologies dominates the total cost of the network.[7] Overall, stretching short reach technology reduces the total cost DCN-wide up to 10% (1 million dollars) depending on the link length distribution. The savings are lower for low stretch, because fewer 10GBase-LR transceivers can be changed to 10GBase-SR transceivers.

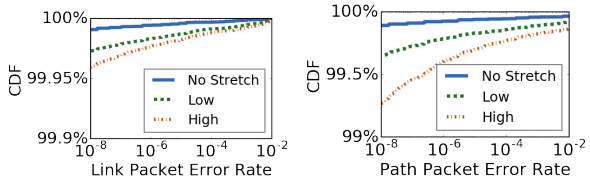
Figure 14b shows the cost results for 40Gbps networks. Except for transceiver costs, which are listed in Figure 7, we assume the cost and energy consumption of 40Gbps components is 4× higher than their 10Gbps counterparts—40Gbps components often bundle four 10Gbps components. While we assume multi-mode fiber is $1.32 per meter, we keep the cost of single mode fiber the same because 40Gbps single-mode transceivers use wavelength division multiplexing. Transceivers account for 72% of the total DCN cost for 0-500m uniform link length distribution. The overall trend of cost savings is similar to that in the 10Gbps network. The total cost savings on 40Gbps networks can be up to 44% (21 million dollars) depending on the link length distribution. The cost reductions are higher for these networks because *i*) there is a higher cost difference between short and long reach technologies,

[7] If the network uses non-standard, intermediate reach (e.g., 500m) transceivers, we could have stretched them to cover longer distances (e.g., 500m to 1km) as well.



(a) *10Gbps* (b) *40Gbps*

Figure 14: *Total DCN cost reduction on a 3-stage fat tree network (10Gbps, 40Gbps) with 512 ToRs assuming uniform link length distribution. Maximum cost savings for 10/40Gbps is achieved when max link length is 580/400m.*



(a) *LPER* (b) *PPER*

Figure 15: *(a) CDF of LPER. (b) CDF of PPER when oversubscription is 4.*

ii) a larger fraction of the links can make use of shorter reach technology, and *iii*) transceivers account for a larger fraction of total DCN cost based on our cost assumptions.

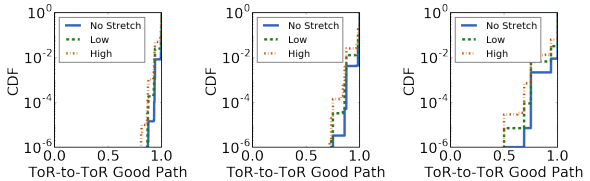
7.2 Characterizing a Stretched Network

NRB ensures by design that the network has a certain minimum fraction of good paths. We now provide a more detailed characterization of a stretched network in terms of the distribution of its link and path qualities. These distributions depend on the exact link length distribution in the network. So that our results can be reproduced, we use 0-500m uniform link length distribution on a 512-ToR Clos network (as in §7.1). Results with link lengths drawn from our cloud provider's network were similar. We study 40Gbps below; 10Gbps networks behave similarly.

Link qualities. Figure 15a shows the CDF of LPERs with and without stretch. We see that even with high stretch, only 0.05% of the links have LPER worse than 10^{-8} (which corresponds to 10^{-12} BER). Only 0.01% of the links have LPER higher than 10%, our guideline for switching off links. As reference, we note that at any given time, roughly 0.08% of the links are down for other reasons in our data centers.

Path qualities. To study path characteristics in a stretched network, we simulate three different oversubscription levels. When the oversubscription level is 1 (4, 16), we have 512 (256, 128) aggregation switches and 256 (128, 64) core switches. Each switch has 32 ports.

We assume the links in the topology have LPERs as per the distribution above. We showed earlier that low RxPower (and thus high LPER) levels are not correlated with switches and appear independent across links. For



(a) *Oversub = 1* (b) *Oversub = 4* (c) *Oversub = 16*

Figure 16: *CDFs of the fraction of good paths for different ToR-ToR pairs across 50 different topologies. The Y axis is log scale to capture the tail behavior.*

each oversubscription level, we generate multiple topologies with different randomized mappings from the LPER distribution and present results aggregated across them.

Figure 15b shows the CDF of PPER for a 512-ToR topology with an oversubscription of four. The results are similar for other oversubscription levels; 99% of paths in the topology have PPER below 10^{-8}.

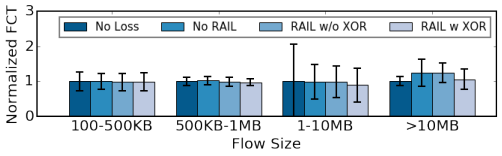
Worst-case experience of loss-sensitive applications. An overall high fraction of good paths is not sufficient to ensure good performance for loss-sensitive applications. For every pair of ToRs that exchange traffic for such applications, there must be enough good paths. Figure 16 shows the CDF of the ratio of good paths to total paths across ToR pairs. A good path means PPER less than 4.8×10^{-8}, equivalent to 10^{-12} BER on each link of a path in a 3-stage fat tree.

We see that when the oversubscription is equal to one, the tail 0.01% of the ToR-to-ToR pairs still has 83% of the good paths remaining. This fraction decreases as oversubscription increases because ToR switches now have fewer uplinks to aggregation switches. If one such uplink has high LPER, it impacts a higher fraction of paths from this ToR to other ToRs. However, even when the oversubscription is 16, the tail 0.01% of the ToR-to-ToR paths still has 70% of good paths left. On the flip side, these data also demonstrate that simply turning gray links off can halve the capacity between some ToR pairs.

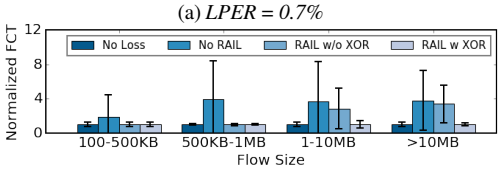
7.3 Application Performance with RAIL

We study the effectiveness of RAIL in preserving application performance using experiments on a small but realistic testbed. Our testbed emulates a 3-stage fat tree network with four ToRs, four aggregation switches, and two core switches. The ToR and aggregation switches are spread across two pods. Each port in the topology is a 10Gbps SPF+ port, and each link has two Finisar FTLX8571D3BCL [8] multi-mode transceivers connected via OM3 fiber [5]. The switches implement ECMP routing. One host is attached to each ToR switch, runs Ubuntu 14.04 with Linux kernel version 3.19, and uses TCP CUBIC [36].

We emulate a gray, high-BER link using an optical attenuator and change the location of the attenuator in different experiments. We use the virtual topology configuration



(a) *LPER = 0.7%*



(b) *LPER = 5%*

Figure 17: *Normalized flow completion time. Error bars are standard deviations of the normalized flow completion time. Flows shorter than 1MB are high priority flows. High priority flows are not affected by packet corruption with RAIL. Low priority flows are protected by XOR coding.*

guideline above, but because it is difficult to finely control the BER using an attenuator, we obtain configurations in which the first- and second-class topologies are identical and contain highly-reliable paths. The third-class topology contains the high-BER link(s). We use iperf to send TCP flows between arbitrary end hosts, with a flow size and inter-arrival time distribution from prior work [20]. TCP flows smaller than 1MB bind to the second- class topology; longer flows bind to the third-class topology.[8]

Figure 17 shows flow completion times (FCT) binned by flow size and normalized to the case of a completely lossless network ("No Loss"). When the network is lossy, without RAIL ("No RAIL"), we see that FCTs are higher for all flow sizes, especially when the loss rate is high. RAIL without error coding ("RAIL w/o XOR") is able to protect only high-priority (short) flows because it routes them over the reliable, second-class topology. With error coding, RAIL ("RAIL w XOR") is able to protect low-priority flows as well. The performance experienced by all flows matches that of the lossless network.

Error correction overhead. Finally, we study the bandwidth overhead of RAIL's error correction. Recall that we set the target post-recovery loss rate to 0.1%. Thus, error correction only kicks in for paths with PER above 0.1%. For paths with PER between 0.1% and 3%, we use standard XOR code with n computed based on error rate. For PPER from 3–10%, we simply replicate every packet three times.

The average bandwidth overhead of our coding scheme is below 0.1% for the 512-ToR topology with 0-500m and 0-1000m uniform link length distribution. This low overhead is the reason we use a simple coding method in RAIL instead of more efficient methods based on retransmissions or bit-level FEC (forward error correction).

[8]This mapping between flow size and topology is only for our experiments. In reality, we expect applications to bind to the desired virtual interface. RAIL does not try to guess flow sizes.

8 Related Work

Our work draws on several themes of previous work.

Measuring optical links. Many researchers have studied optical WAN for properties such as dispersion [30, 57, 25, 39, 56], temperature variations [38, 42], and packet loss and inter-arrival times [31, 43]. Ghobadi et al. study optical signal quality in WAN and, like us, find an overprovisioned optical layer [33]. In contrast, however, our focus is on DCNs, where the environment and technology are different. We believe we are the first to study this optical layer.

Reducing cost of optics in DCNs. We are inspired by other efforts in the industry to lower the cost of optics in DCNs. Facebook [27] is pushing for a new standard for low cost 100Gbps transceivers. Their initial observation is similar to ours: the DCN is a milder operating environment than traditional telecom networks. Corning [28] also observed, using stochastic attenuation models, that DCN link qualities can be disparate and it is possible to extend the reach on some fraction of links. We complement these efforts with a detailed characterization of optical links in operational DCNs, proposing a way to reduce cost without hardware changes and developing a system to preserve application performance if some links turn gray due to reduced over-engineering.

Virtual topologies. The concept of virtual topologies over the same physical infrastructure has been leveraged in other contexts, such as simplifying the specification of network policies in software-defined networking (SDN) [40, 51, 46] or "slicing" the network to isolate users [23, 54, 47]. We use this concept to build topologies with different reliability guarantees.

Reliable systems atop unreliable components. There is a long-standing tradition of building reliable systems using (cheaper) unreliable components and masking unreliability from applications using intelligent software techniques. Classic examples include building reliable storage systems using disks that are individually unreliable [49, 32]. Our work follows this tradition, though the set of techniques it uses are specific to its domain.

9 Conclusions

Our measurements show that optical links in data center networks are heavily over-engineered. This over-engineering is not only expensive but also unnecessary because of application and path diversity in DCNs. Many applications can tolerate small amounts of loss, and loss-sensitive applications can be supported as long as some (not all) paths between ToR pairs are reliable. We find that reducing optical over-engineering simply by using transceivers beyond their specified length can reduce network cost by up to 10% for 10Gbps networks and 44% for 40Gbps networks. Moreover, when coupled with the traffic routing and error correction mechanisms of RAIL, there is negligible loss in application performance.

Acknowledgments

We thank Keren Bergman, David Bragg and Jamie Gaudette for sharing insights on optics. We thank Hui Ma and Shikhar Suri for deployment of our monitoring system. We thank our shepherd George Porter and the anonymous reviewers for their feedback. This work was partially supported by the NSF (CNS-1318396 and CNS-1518702).

References

[1] AOI 100G MPO MMF 850nm 70m Transceiver. http://www.high-tech.co.jp/common/sys/product/product00445_01.pdf.

[2] Arista 7050S. https://www.arista.com/assets/data/pdf/Datasheets/7050S_Datasheet.pdf.

[3] Arista 7060CX. https://www.arista.com/assets/data/pdf/Datasheets/7060X_7260X_DS.pdf.

[4] Arista Optics Modules and Cables. https://www.arista.com/assets/data/pdf/Datasheets/arista_transceiver_datasheet.pdf.

[5] Corning OM3 Fiber. https://www.corning.com/media/worldwide/coc/documents/Fiber/PI1468_07-14_English.pdf.

[6] Duplex Zipcord Fiber Optic Cable. http://www.fs.com/c/duplex-zipcord-fiber-optic-cable-1249.

[7] Finisar. http://www.mouser.com/finisar-corporation/.

[8] FINISAR FTLX8571D3BCL. https://www.finisar.com/optical-transceivers/ftlx8571d3bcl.

[9] FS.COM QSFP 40G Transceiver. http://www.fs.com/products/36309.html.

[10] HPC Optics. https://www.hpcoptics.com.

[11] IEEE 10 Gb/s Ethernet task force. http://grouper.ieee.org/groups/802/3/ae/public/blue_book.pdf.

[12] Optical monitor MIB. http://www.oidview.com/mibs/9/CISCO-OPTICAL-MONITOR-MIB.html.

[13] PCWholeSale. http://www.pc-wholesale.com/.

[14] ROBOfiber. http://www.robofiber.com/.

[15] VPI. http://www.vpiphotonics.com/index.php.

[16] 100G PSM4 Specification. http://www.psm4.org/100G-PSM4-Specification-2.0.pdf, 2014.

[17] RAIL VPI simulation files. https://github.com/railnsdi2017/rail_VPI, 2016.

[18] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In *SIGCOMM*, 2008.

[19] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese. CONGA: Distributed Congestion-aware Load Balancing for Datacenters. In *SIGCOMM*, 2014.

[20] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pFabric: Minimal Near-optimal Datacenter Transport. In *SIGCOMM*, 2013.

[21] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. FAWN: A Fast Array of Wimpy Nodes. In *SOSP*, 2009.

[22] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *SIGCOMM*, 1996.

[23] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. In *SIGCOMM*, 2006.

[24] E. Brewer, L. Ying, L. Greenfield, R. Cypher, and T. Ts'o. Disks for Data Centers. Technical report, Google, 2016.

[25] H. Bulow, W. Baumert, H. Schmuck, F. Mohr, T. Schulz, F. Kuppers, and W. Weiershausen. Measurement of the maximum speed of PMD fluctuation in installed field fiber. In *OFC*, 1999.

[26] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. BBR: Congestion-Based Congestion Control. *ACM Queue*, 2016.

[27] A. Chakravarty, K. Schmidtke, S. Giridharan, J. Huang, and V. Zeng. 100G CWDM4 SMF Optical Interconnects for Facebook Data Centers. *Conference on Lasers and Electro-Optics*, 2016.

[28] X. Chen, J. Abbott, D. Powers, D. Coleman, and M.-J. Li. Statistical Treatment of IEEE spreadsheet model for VCSEL-multimode fiber transmissions. *OECC/PS*, 2016.

[29] N. El-Sayed, I. A. Stefanovici, G. Amvrosiadis, A. A. Hwang, and B. Schroeder. Temperature Management in Data Centers: Why Some (Might) Like It Hot. *SIGMETRICS Perform. Eval. Rev.*, 2012.

[30] R. J. Feuerstein. Field Measurements of Deployed Fiber. In *OFC*, 2005.

[31] D. A. Freedman, T. Marian, J. H. Lee, K. Birman, H. Weatherspoon, and C. Xu. Exact Temporal Characterization of 10 Gbps Optical Wide-area Network. In *IMC*, 2010.

[32] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. In *SOSP*, 2003.

[33] M. Ghobadi, J. Gaudette, R. Mahajan, A. Phanishayee, D. Kilper, and B. Klinkers. Signal characteristics in Microsoft optical backbone. In *OFC*, 2016.

[34] P. Gill, N. Jain, and N. Nagappan. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *SIGCOMM*, 2011.

[35] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM*, 2009.

[36] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.*, 2008.

[37] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In *NSDI*, 2010.

[38] H. Ji, J. H. Lee, and Y. C. Chung. System Outage Probability Due to Dispersion Variation Caused by Seasonal and Regional Temperature Variations. In *OFC*, 2005.

[39] M. Karlsson, J. Brentel, and P. Andrekson. Long-term measurement of PMD and polarization drift in installed fibers. *Journal of Lightwave Technology*, 2000.

[40] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, N. Gude, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang. Network Virtualization in Multi-tenant Datacenters. In *NSDI*, 2014.

[41] P. Lahiri, G. Chen, P. Lapukhov, E. Nkposong, D. Maltz, R. Toomey, and L. Yuan. Routing Design for Large Scale Data Centers: BGP is a better IGP! https://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf.

[42] J. C. Li, K. Hinton, P. M. Farrell, and S. D. Dods. Optical impairment outage computation. *Opt. Express*, 2008.

[43] T. Marian, D. Freedman, K. Birman, and H. Weatherspoon. Empirical characterization of uncongested optical lambda networks and 10GbE commodity endpoints. In *DSN*, 2010.

[44] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *MobiCom*, 2001.

[45] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 2008.

[46] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker. Composing Software-defined Networks. In *NSDI*, 2013.

[47] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary. NetLord: A Scalable Multi-tenant Network Architecture for Virtualized Datacenters. In *SIGCOMM*, 2011.

[48] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. In *SIGCOMM*, 1998.

[49] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD*, 1988.

[50] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica. A Cost Comparison of Datacenter Network Architectures. In *CoNEXT*, 2010.

[51] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *SIGCOMM*, 2013.

[52] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *SIGCOMM*, 2011.

[53] L. Rizzo. Effective Erasure Codes for Reliable Computer Communication Protocols. *SIGCOMM Comput. Commun. Rev.*, 1997.

[54] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the Production Network Be the Testbed? In *OSDI*, 2010.

[55] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *SIGCOMM*, 2015.

[56] S. Woodward, L. Nelson, M. Feuer, X. Zhou, P. Magill, S. Foo, D. Hanson, H. Sun, M. Moyer, and M. O'Sullivan. Characterization of Real-Time PMD and Chromatic Dispersion Monitoring in a High-PMD 46-Gb/s Transmission System. *Photonics Technology Letters, IEEE*, 2008.

[57] S. Woodward, L. Nelson, C. Schneider, L. Knox, M. O'Sullivan, C. Laperle, M. Moyer, and S. Foo. Long-Term Observation of PMD and SOP on Installed Fiber Routes. *Photonics Technology Letters, IEEE*, 2014.

[58] X. Wu, D. Turner, G. Chen, D. Maltz, X. Yang, L. Yuan, and M. Zhang. NetPilot: Automating Datacenter Network Failure Mitigation. In *SIGCOMM*, 2012.

[59] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat. WCMP: Weighted Cost Multipathing for Improved Fairness in Data Centers. In *EuroSys*, 2014.

[60] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion Control for Large-Scale RDMA Deployments. In *SIGCOMM*, 2015.

Appendix

A Expected Fraction of Good Paths

Definition 2. *A path is good when all unidirectional links on the path are good.*

Lemma 1. *For a path with h hops, when each hop has the probability of at least l to be good and independent, the path is good with the probability of at least l^h.*

Proof. Every hop's quality is independent. □

Theorem 1. *When every path is good with the probability of at least P_{each}, the expected fraction of good paths is at least P_{each}.*

Proof. Let P_i be the probability of a path i to be good. Let's define an indication variable I_p such that

$$I_p = \begin{cases} 0 \text{ , if path p is not good,} \\ 1 \text{ , if path p is good} \end{cases}$$

If the network has n paths, the expected fraction of paths is

$$\text{Expected fraction of good paths} = E\left(\frac{\sum_{p \in \text{all paths}} I_p}{n}\right)$$

Using the linearity of expected value, we get

$$\text{Expected fraction of good paths} = \frac{\sum_{p \in \text{all paths}} E(I_p)}{n}$$

Because $E(I_p) \geq P_{each}$,

$$\text{Expected fraction of good paths} \geq \frac{nP_{each}}{n} = P_{each}$$

□

Theorem 2. *If each link has the probability of at least l to be good and independent, and the longest path in the network has h hops, then the expected fraction of good paths is at least l^h.*

Proof. Use Lemma 1 to get a lower bound of the probability of any path to be good. Then use Theorem 1. □

B Find the Worst Path in Clos Topology

We seek an efficient solution to the following problem: Assuming ECMP routing, given a subset of links on a Clos network and every link's unidirectional packet error rate (LPER), find the worst end-to-end path with the highest path packet error rate (PPER).

This problem can be solved efficiently because of the data center's unique topology and its simple ECMP routing scheme. We describe our algorithms assuming the Clos topology has an oversubscription ratio of 1 across all stages. All our results hold when oversubscription is introduced.

Definition 3. *A Clos network is a multi-stage network. A switch at (i)th stage can only connect to switches at neighbor stages (i.e., $(i \pm 1)$th stage). Every stage has n switches, except for the top stage which has $\frac{n}{2}$ switches. All switches in the network have $2k$ ports. The network has $\log_k n$ stages. Every switch except on the top stage has k ports facing the upper stage and k ports facing the lower stage. Switches on the top stage have all $2k$ ports facing the lower stage.*

Today, data center operators chose ECMP up-down routing as the basic routing algorithm. In this method, packets first travel to an upper stage switch and then down to the destination top-of-rack (ToR) switch. ECMP requires packets to take one of the shortest paths.

Definition 4. *A path from ToRs (stage = 1) switches a to b ($a \neq b$) is called an up-down path when there is a switch c on the path such that the stage number of each switch monotonically increases from a to c and monotonically decreases from c to b.*

Clos topology has a unique property, in that every up-down path can actually be chosen to forward traffic. All up-down paths between the same ToR-pair have exactly the same number of hops. When we want to find the worst path, we only have to consider all the up-down paths.

The worst path is defined as the following.

Definition 5. *A worst path is the up-down path with highest PPER. PPER of path p is* $1-\prod_{l\in p}(1-LPER_l)$.

It is computationally infeasible to track PPER for every path in the network. Tracking PPER takes at least $O(n^3)$ running time because a Clos network has $O(n^3)$ paths (i.e., $O(n)$ paths for every ToR pair and there are $O(n^2)$ ToR pairs). As an alternative, the worst path computation must quickly neglect paths with no hope of becoming the worst path in the run-time.

Our algorithm 1 identifies the worst path with a running time of $O(n\log n)$ and is provably optimal in asymptotic running-time. In this section, we prove the path produced by the algorithm is indeed the worst path; we then prove no faster algorithm exists.

Lemma 2. *The output of Algorithm 1 is an up-down path.*

Proof. We only need to show $\forall s$,top_s is a valid path. From Algorithm 1, top_s is nothing but a monotonically up-going path to s and a monotonically downward path from s.

Thus, top_s is an up-down path if the source and the destination ToR of top_s are different. In Algorithm 1, we see when we calculate top_s, we enforce the direct children of s to be different. In Clos network, this means the source and the destination ToR are in different branches of the Clos (subtree of fat tree) containing non-overlapping sets of ToRs. Thus, top_s is an up-down path. □

Theorem 3. *The output of our algorithm is the worst path.*

Proof. Proof by contradiction. Our algorithm outputs path p. The worst path is p' such that $PPER_{p'} > PPER_p$.

Without loss of generality, p' is an up-down path from a' to b'. Because p' is also an up-down path, by definition, there must be a c' on p' such that c' is the highest stage switch on p', a' to c' is a monotonic upward path, and c' to b' is a monotonic downward path.

When our algorithm goes through switch c', there are two situations. (1) $p' = top_{s'}$ (2) $p' \neq top_{s'}$. Let's talk about both situations.

If $p' = top_{s'}$, because $PPER_p = max_{s\in all\ switches}(top_s)$, $PPER_p \geq top'_s = p'$. This contradicts the assumption that $PPER_{p'} > PPER_p$.

If $p' \neq top_{s'}$ and p' is valid, p' must includes two children of s'. Then, it must be the case that $p' < top_{s'}$ because, otherwise, p' will be chosen when computing $top_{s'}$. Because $PPER_p = max_{s\in all\ switches}(top_s)$, then $PPER_p \geq top_{s'} > p'$. This contradicts the assumption that $PPER_{p'} > PPER_p$.

Overall, $PPER_{p'} \leq PPER_p$. Thus, the output of our algorithm is the worst path.

□

Theorem 4. *The running time of our algorithm is* $\Theta(n\log n)$.

Proof. Our algorithm passes through each switch once, and the update algorithm is $\Theta(k^2)$. Because k is constant, the update part is in the order of number of switches, $\Theta(n\log_k n)$. The comparison part compares top_s for all switches which takes another $\Theta(n\log_k n)$. Thus, our algorithm finishes in $\Theta(nlog_k n)$. Because k is constant, our algorithm finishes in $\Theta(n\log n)$. □

Theorem 5. *Any algorithm that can compute the worst path has a running time of at least* $\Theta(n\log n)$.

Proof. Proof by contradiction. Assume an algorithm exists that computes the worst path with running-time less than $\Theta(n\log n)$.

We construct an adversarial oracle that always returns LPER = 0.5 for every link queried by the algorithm. After the algorithm finishes, the algorithm returns a worst path p. Because the number of links is in the order of $\Theta(nlog_k n)$, the algorithm does not have enough time to read LPER on every link. There must be a fraction of links that are not read by the algorithm. Let l be one of the unidirectional links whose LPER is not read by the algorithm.

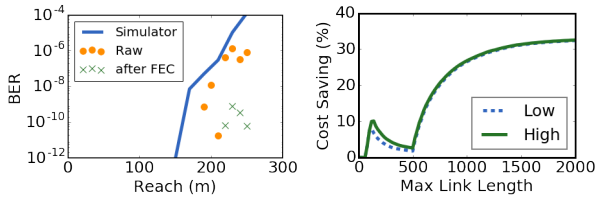
There are two situations here. (1) l is on p; (2) l is not on p. We discuss both situations.

If l is on p, the oracle sets $LPER_l \leftarrow 0$ and LPER of all remaining links (those not read by the algorithm) at 0.5. Thus, l cannot be the worst path because it has lower PPER than other paths. This contradicts the assumption that p is the worst path.

If l is not on p, the oracle sets $LPER_l \leftarrow 1$ and LPER of all remaining links (those not read by the algorithm) at 0.5. Thus, l must be on the worst path of any path going through l has higher PPER (PPER = 1) than any path that does not go through l. This contradicts the assumption that p is the worst path.

Overall, the algorithm cannot output the correct worst path without taking at least $\Theta(n\log n)$.

□



(a) *Reach Test* (b) *Cost Saving*

Figure 18: *18a shows the reach test for 100G-SR4. Blue line is simulated transceiver from VPI. Yellow dots are raw BER and green dots are corrected BER. 18b shows cost saving for uniform random link length distribution for different max length.*

C Over-engineering in 100Gbps

We repeat our stretch experiments using 100G-SR4 transceivers [1] (standard reach 70m) and observe similar levels of over-engineering. However, an important distinction between 100Gbps and 10/40Gbps technologies is the presence of a Forward Error Correction (FEC) module in 100Gbps switches. This means the pre-FEC BER requirement is reduced from 10^{-12} to 5×10^{-5} and the FEC module boosts BER back to the standard on every hop. Figure 18a shows pre- and post- FEC BER as we stretch the fiber length without additional attenuation. Interestingly, even pre-FEC BER (Raw) is lower than 10^{-12} at 180m; this is 2.5 times higher than the standard reach, once again confirming the degree of over-engineering. We simulate this transceiver in VPI and conservatively bound the stretch to 110m and 130m to achieve *network reliability bounds* of 99% and 95% respectively where FEC is off.

From simulation in VPI, we show PSM4 technology can be stretched from 500m to 2km. Figure 18b shows the cost saving for uniform random link length distribution. Before 70m, there is no cost saving because all links are covered by 100G-SR4. The cost saving peaks at 150m, the stretched reach, because the largest fraction of the link can use 100G-SR4. Cost saving decreases at this point, because 500m PSM4 technology's cost is cheap. Cost saving increases after 500m because PSM4 technologies can replace higher cost 100G-LR4 technologies between 500-2km. Overall, we find the savings in 100G can be up to 30%.

When FEC is turned on, the link quality distribution among all the links is narrower. The amount of over-engineering remains the same, which means stretching transceivers can still reduce the cost of DCN. However, switching gray links off is likely enough to protect the network. We leave this problem for future investigation.