Physical Deployability Matters

Jeffrey C. Mogul
Google

John Wilkes
Google

Abstract

While many network research papers address issues of deployability, with a few exceptions, this has been limited to protocol compatibility or switch-resource constraints, such as flow table sizes. We argue that good network designs must also consider the costs and complexities of deploying the design within the constraints of the physical environment in a datacenter: *physical* deployability. Traditional metrics of network "goodness" do not account for these costs and constraints, which might explain why some otherwise attractive designs have not been deployed in real-world datacenters.

CCS Concepts

• **Networks** → **Physical topologies**.

Keywords

Network physical deployability

ACM Reference Format:
Jeffrey C. Mogul and John Wilkes. 2023. Physical Deployability Matters. In *The 22nd ACM Workshop on Hot Topics in Networks (HotNets '23), November 28–29, 2023, Cambridge, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3626111.3628190

1 Introduction

Networking researchers have long talked about "deployability," either asserting that their novel designs are easy to deploy into existing networks, or arguing whether deployability should or should not be a criterion for reviewing papers, or declaring that a clean-slate design is preferable to trying to accommodate existing practice.

What has not received as much attention from the research community is the *physical* deployability of a datacenter network design: is a design feasible to deploy within the constraints of the physical environment in a datacenter, at scale and at reasonable cost? We argue that this is a valid and necessary criterion for judging network designs, that it can be made at least somewhat rigorous, and that there are both solved and challenging unsolved problems for the networking research community to understand.

We focus on datacenter networks, although physical-deployability issues certainly affect wireless networks [35] and WANs.

We explore whether deployability considerations might explain the non-deployment of otherwise attractive designs, such as expander graphs, in real-world datacenters. We also discuss technologies and techniques that would simplify physical deployment, and we speculate about the possibilities for devising well-specified objectives and metrics that could guide future research.

Designers of datacenter networks care about time-to-deploy, repair and maintenance processes, evolvability, and other non-functional factors. Physical deployability significantly affects all of these; network designs that fail to explicitly address physical deployability can be hard to assess.

What's the best way to measure and improve physical deployability? We don't claim to have a full solution to either (likely there isn't a single "solution"). Instead, we challenge researchers to broaden their scope of "deployability" and how to address it.

Context: Google's networks are larger than those that academic researchers can directly study, and than most enterprise networks. Therefore, our focus on large-scale deployment challenges, and on automated processes, might seem irrelevant to some readers. However, many research papers describing novel network designs do focus on scalability; we argue that a design that scales only on paper, and not in physical reality, is not truly scalable. We hope to expand, not reduce, the range of worth research topics.

We believe that many of these issues will inevitably afflict smaller networks, if they haven't already.

2 What is "physical deployability," and why does it matter?

Traditional "deployability" discussions have focused on protocol-level concerns: can this packet format traverse the entire network path successfully [40, 51]? do these routing protocols have compatible metrics, and will their implementations avoid pathologies like loops and black holes? can secure routing or naming mechanisms be deployed incrementally with any benefits [31]? will deploying this novel congestion-control algorithm cause harm to other flows [52]? will a new mechanism or a larger scale fit into existing hardware tables [54]? Concerns about deployability of new mechanisms inside switches have motivated OpenFlow [33][1] and P4 [8], both of which have led to many publications.

[1]The paper states "Today [2008], there is almost no practical way to experiment with new network protocols (e.g., new routing protocols, or alternatives to IP) in sufficiently realistic settings (e.g., at scale carrying real traffic) to gain the confidence needed for their widespread deployment."

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

HotNets '23, November 28–29, 2023, Cambridge, MA, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0415-4/23/11.
https://doi.org/10.1145/3626111.3628190

In contrast, with "physical deployability" we focus on *interactions between network equipment (switches, cables, connectors, etc.) and the physical world*: where do things fit? how do we get things in the right places at the right times? can humans manipulate these parts without undue toil, without harm to themselves or to the equipment, and without errors? what if we want robots to do the work instead?

For example, a perhaps apocryphal story depicted in the movie "Hidden Figures" relates that in 1961, NASA's IBM 7090 was too big to fit through the doorway [25]. Even today, that constraint limits how many network-equipment racks can be conjoined and pre-cabled off-site.

In large-scale datacenters, physical processes (even if carried out by humans) are managed by complex automation systems, which plan the placement and connectivity of computing equipment, including networking equipment; order the correct materials (e.g., cables pre-built to proper lengths); instruct the humans or robots where and when to place and connect equipment; and validate that everything is in its proper place. These automated systems embed a variety of assumptions about equipment, cables, and their interfaces with the physical world, and so are not infinitely capable.

Physical deployability also depends on whether existing automation can handle a novel or modified design, or whether it can easily be reconfigured without extensive software engineering. Notably, physical-automation software can be difficult to test, because testing with real components is expensive, while simulations of physical reality are often too imprecise to detect minor mistakes (such as an attempt to run a network cable through a space that is just a little too small to accommodate the safe bending radius of the cable).

This is not a "rigorous definition of deployability," as one reviewer requested. We're not sure how to construct that, or if it's even possible, but we think that could itself be worthy of further work.

Internally, we use metrics such as "time to deploy" (hours of effort), cost to deploy, and "first-pass yield" (i.e., what fraction of deployed switches or links actually work without further repair). But even time-to-deploy is hard to define, depending on how much parallelism is possible and how one decides when a task is finished. And researchers without hyperscale networks to play with will need proxy metrics.

2.1 Post-deployment activities

Closely tied to physical deployments are three other processes: repairs, expansion, and decommissioning ("decom"). Large datacenters must support in-service repairs – you cannot shut off the entire network if a switch or cable fails – and the feasibility and risk of in-service repairs depends on the network design. For example, if one port on a multi-port line-card fails, the whole card needs to be replaced, requiring all of the other ports on the card to be drained. Does the network design tolerate this kind of correlated downtime? And just as deployment is automated, so is the repair process: automated tools must reliably determine which parts need replacing, dispatch a human operator to the right location with correct instructions, trigger drains only during the smallest possible window, and take remedial actions after a part is replaced (e.g., updating MAC addresses; re-installing firmware).

Many datacenter owners initially install a moderate amount of server and switch capacity, and then incrementally deploy more via "expansion" processes. Expanding a live network has both costs and risks, which have been addressed by prior work. For example, Zhao *et al.* optimized the expansion process for a Clos network [56]. Jellyfish [47] and Xpander [50] set expandability as a fundamental architectural goal. The FatClique paper [55] offered physical-deployment metrics, but only in the context of expansions.

It's often necessary to partially decommission network equipment in a datacenter. (Decommissioning an entire datacenter is conceptually much easier, but rarely necessary or desirable.) We decommission switches and other equipment when they reach the end of their support lifetime (e.g., the vendor no longer provides security updates) or because they are functionally obsolete or fully depreciated, and we want to replace them. Physically removing switches or, especially, cables from a running network is risky – one might accidentally remove the wrong thing, or damage something nearby. Therefore, when we must add cables, to support higher bandwidths or port counts, we seldom remove old ones. Instead, we provision enough space in cable trays for several generations.

It is surprisingly hard to automate a decom procedure, because it can be hard to know for sure what cannot be removed. (E.g., we can only remove a cable bundle once none of the affected ports are still in service, and none are planned to be in service soon.)

2.2 Supply chain

Physical deployability also intersects with supply-chain challenges. For example, if the network design (including switch stack and SDN controllers) support fungible hardware (the ability to replace one part with another, without other consequences), then a supply-chain problem at one vendor can be resolved by buying compatible parts from another. (AWS has stated that fungibility is a fundamental principle in their network designs [10], and Meta uses the same switch chassis in all layers of their datacenter design, with a second vendor to complement their own switch design [4].) A desire for fungibility might mean not taking advantage, in a network design, of special features only available from one vendor. (Many prefer Ethernet over InifiniBand because the latter lacks multiple vendors with interoperable products [43]).

Datacenter operators also must protect the physical integrity of network equipment against malicious actors. Because network switches and SDN controller machines are physical items that travel along a supply chain, they are inherently vulnerable to security threats during the journey. Supply-chain attacks that modify hardware have been alleged [11]; attacks that remotely modify flash have been

substantiated [17, 18, 30, 42], and these could just as easily happen along the supply chain. Post-deployment physical attacks can also be carried out by malicious insiders (including adding "sniffers" on unencrypted links, or adding rogue NICs or linecards). To protect against physical attacks requires support for tamper-resistance and continuous auditing of hardware and firmware.

2.3 Automated physical deployment

Why do we care so much about automated physical deployment? Even when human operators do all of the physical work, we can save a lot of time by automating all of the preparatory work, and automating the planning of operator actions so that they don't have to waste time (e.g., repeatedly walking from one place to another, or getting in each other's way). When physical deployment is slow, it strands much-more-expensive machine capacity (a machine without a network connection is "stranded" capital). An extra 5 minutes per thing adds up quickly when you have to install 10k things (that would be about 1 week of added time).

Slow deployment also makes network capacity planning harder, because demand forecasts become inaccurate over relatively short timescales. If we install too little capacity, machines are stranded; if we install too much, it wastes money. Fungibility also helps here, by avoiding deployment delays when a part needs to be substituted.

3 What makes it hard?

Most network research, and much of the initial design phases for real networks, takes place at a level of abstraction that conceals a large set of constraints imposed by physical deployment. Abstraction is unquestionably necessary, but the hidden constraints mean that designs that look appealing on paper can turn out to be infeasible. This can force complex tradeoffs between preserving abstract-network properties (e.g., support for a range of traffic matrices or reliability in the face of switch and link failures) and respecting those constraints. Before we discuss how to cope with that paradox, in §5, we first look at some of the constraints.

Broadly, physical constraints fall into a few categories:

- Physical feasibility: will things actually fit into the available space and power? Will a design work with easily-obtainable parts?
- Human factors: can deployment and repairs be done with reasonable levels of human effort? without too many errors? and without exposing humans to harm?
- Operational challenges: will we be able to keep the network running reliably?
- Support for evolution: will we be able to change the network, possibly in radical ways, without having to rip it out entirely? or even while it is in active service?
- Financial: can we meet the other constraints only by spending a lot of money?

Since this paper itself is subject to page-count constraints, here we discuss just some of the specific constraints.

3.1 Physical constraints

Especially as link rates increase, constraints on the cabling between network ports become more challenging. The physical extent of large-scale datacenters exacerbates this: longer cables bring more problems. One can get higher bandwidths across longer distances with optical connections, but optical transceivers are power-hungry and quite expensive (and perhaps less reliable than electrical connections).

Therefore, large-scale high-speed networks often mix copper cables (cheaper, but infeasible except at short distances) and optical fiber cables (expensive, but can span hundreds of meters or more). This leads to complex optimization problem, since some network topologies gain shorter cable runs (on average) at the cost of more switch hops (and thus somewhat more latency). Mudigonda *et al.* [37] presented algorithms for this optimization problem. Agarwal *et al.* [1] extended that to consider how cables run through trays between racks, but left for future work the problem of accounting for the plenum capacities of the racks.

Higher-speed copper cables also tend to be thicker, to preserve signal integrity. Amazon reported [10] that the 2.5m cables they used within switch racks went from a 6.7mm outside diameter (OD) for 100Gbps to an 11mm OD for 400Gbps. This makes them much stiffer, and their cross-sectional area increases by 2.7X. Such cables are much harder (or impossible?) to fit into a rack full of switches (they report using 256 cables in a rack).[2] Therefore, they switched to active electrical cables, decreasing the cable thickness while still being cheaper and more reliable than optical intra-rack cabling.

Viable cable lengths can also be reduced by the insertion losses from patch panels and optical circuit switches (e.g., 0.5dB to 1.0dB in Telescent's switches [49]). This conflicts with some of the benefits of inserting patch panels or OCSs to mitigate other deployment issues (see §4.1).

Meta has also described how they use multiple approaches to cabling in response to various physical constraints [14].

Some papers have proposed using free-space optics [23] or 60GHz wireless links [57] within datacenters. While these avoid the physical challenges of cables, these too suffer from real-world issues. Free-space optics require unobstructed paths between racks, which is hard to guarantee; at higher speeds, they also might expose human eyes to damage. 60GHz wireless links probably cannot be packed tightly enough to entirely replace large bundles of fibers.

Cable installation can be tedious, especially on the datacenter floor, which is not the best work environment. Singh *el al.* report savings of almost 40% (capex + opex) and weeks of delay by using regular, pre-constructed bundles of cables [44].

For the same reason, intra-rack cables are often preinstalled before a rack full of switches is delivered. In some cases, it can be helpful to pre-cable a conjoined *pair* of racks (representing an atomic unit of network capacity). However, this can conflict with floor-space constraints limiting a row to

[2] A thicket of cables can also impair airflow in a rack of switches.

an odd number of racks – thus an installation with an even number of racks leaves valuable space unoccupied. (Also, double-wide racks don't always fit through doors.)

Since many things with localized extent can go wrong in datacenters (clumsy humans, cable cuts, small fires, etc.) we avoid physical single points of failure (SPOFs) in longer inter-rack cable runs. Maintaining full physical-path diversity adds complexity to fiber routing.

3.2 Human constraints

Things are always changing in a large datacenter; a colleague observed that these are "active construction sites." This level of activity poses both a risk to equipment (cables can be inadvertently or accidentally disturbed) and to humans.

Most networking research does not account for human interactions with in-deployment or in-service network equipment. Real designs must consider safety (e.g., equipment weight, hot surfaces, electrical shock, eyesight risk from high-powered optics, and hearing risk from noisy fans), and how many people at a time can work on one rack.

Human workers rely on a mixture of training and automation (to tell them what to do next, and to verify that it was done correctly). A network design that is hard to explain in training materials, or to the software engineers developing automation, is likely to suffer from more outages or longer deployment times. Not enough attention has been paid to good user interfaces, especially those that synchronize steps between human tasks and automated ones. (For example, an SDN control plane should avoid routing traffic through racks where a human is actively making changes – but should resume using those paths as soon as they are stable.)

3.3 Operational constraints

Physical components of networks fail relatively often at scale, especially when one considers not just switch failures, but also NIC failures, cabling disruptions, and power-distribution failures [16, 19, 45].[3] Thus, network operators must plan for frequent post-deployment physical operations, and this has implications for network design.

While many network data-plane and control-plane designs are good at rapidly detecting and routing around failures, and getting better at root-causing failures to specific components [28, 41] mitigation techniques generally cannot tolerate large numbers of concurrent failures. Therefore, network availability depends on mean time to repair (MTTR), an inherently physical problem.

One tradeoff to make, during network design, is the size of a physical unit of repair (i.e., what part of the network needs to be "drained" of traffic during the repair). For example, while using higher switch radixes supports lower hop-count designs, that also means that one switch repair takes more ports out of service, even if only one port has failed.

Repairs usually rely on spare parts, and supply-chain issues motivate parts fungibility: the ability to replace one part with a slightly different one (e.g., from a different vendor). Fungibility implies a need to design a network without depending on the best available parts, but rather the second-best. This could, for example, reduce the allowable length for a cable.

Low MTTR depends on good automation, to guide repair operations and validate the results before returning links and switches to service. A network design that abstracts too many physical details conceals physical-world failure domains (e.g., shared power feeds). Exposing the abstract-to-physical mapping via well-designed representations (see §5.3) can help automated systems manage repairs appropriately (and also manage supply-chain fungibility).

3.4 Evolution constraints

Some datacenter networks, especially at small to medium scales, are fully designed before installation. However, there are significant advantages to being able to evolve a design in-place. For example, one can incorporate newer technologies as they become available, for higher bandwidth or lower cost (and for lower power consumption). Or, one can adapt the topology to shifting traffic demands, such as those induced by large-scale machine learning.

In-place evolution leads to heterogeneity, often in ways that are hard to predict during the initial design. For example, a network might end up incorporating switches with multiple radixes, or different line rates. Ideally, then, a network design should support heterogeneity; Curtis *et al.* described how to do this for Clos networks [12]. Poutievski *et al.* describe how routing traffic via an intermediate "transit block" can avoid having to add low-speed ports on high-speed switches [39]. Singla *et al.* incorporate heterogeneous link speeds into their upper-bounds analysis [46].

Datacenter networks can also outlive the useful service lifetime of components like switches, especially if a vendor ceases to support a switch. Replacing a component with a much newer one, while often necessary or advantageous, can create a cascade of other issues. For example, when replacing an SDN controller machine, the newer machine might not support the older NIC, requiring a port-speed upgrade at a connected switch, and possibly cabling changes as well.

(We distinguish *evolution* from *expansion*, the one aspect of deployability that has been widely discussed [47, 50, 55, 56].)

Evolutionary pain can come from fundamental challenges (e.g., scaling link speeds beyond the ability of copper cables to carry signals between racks) or contingent ones – "this is how we've always done things" or "our automation software made too many assumptions." While the latter might be viewed as self-inflicted, they are also inevitable; a network design aimed at a long service lifetime probably needs to account for them.

3.5 Financial constraints

Although large scale datacenter networks are much less expensive than the machines they connect [24], they are not

[3] We note that published data on failure rates in datacenter networks, especially including non-switch components, appears to be scarce.

cheap, and that leads to pressure to reduce their costs, including how expensive they are to deploy.

One result is the desire to deploy the network incrementally, to avoid paying depreciation on unused capital equipment, to defer decisions about how much capacity is needed, and to allow that capacity demand to be fulfilled by faster, cheaper technology as it becomes available. In turn, that requires that the network be able to handle multiple design variations simultaneously – and the use of heterogeneous port counts and line rates complicates the network design problem [46].

A network that can be upgraded, added to, and (partially) decommissioned while it is in use, needs software tooling to support near-arbitrary deployment sequences, and a great deal of attention to safety: the connectivity that networks provide is somewhat antithetical to the traditional approaches that use isolation to reduce the blast radius of an error or fault.

Over time, incrementally-deployed networks can accumulate elements from multiple technology generations, and it is rarely cost-effective at scale to replace or upgrade computers just because the network designers want that. That increases the overall system's complexity, and can act as a drag on engineering development velocity, as well as how fast and cheaply the network can evolve.

4 Case studies

We include three case studies to illustrate some of the issues raised in this paper.

4.1 Indirection helps

Some prior work has argued for "flat" [26, 29] datacenter topologies, where ToRs are directly connected to each other, rather than through a hierarchy (as in a Clos or leaf-spine network). This reduces path lengths, leading to lower intrinsic latency, and perhaps less congestion. However, Marty *et al.* found that directly connecting ToRs, as is the case with a pure flattened butterfly topology, was operationally challenging, because racks are often added or removed, which impacted network operations [32]. The performance benefits of flat networks might not outweigh their operational costs.

Indirection also helps at the upper levels of Clos networks. Zhao *et al.* described how using a layer of patch panels between the aggregation blocks and spine blocks in a large Clos made it a lot easier to expand the network incrementally, because the topology can be expanded or modified "without walking around the data center floor or requiring the addition or removal of existing fiber" [56].

Poutievski *et al.* showed that replacing these patch panels with a relatively slow optical circuit switch (OCS) not only further eases expansions, but also supports frequent changes to the capacity between aggregation blocks, to respond to changing and uneven inter-block traffic demands. (In real networks, inter-rack and inter-block demands are often persistently and highly non-uniform; networks need the flexibility to cope with time-varying non-uniformity.)

4.2 Why aren't expanders in wide use?

Papers describing expander-graph datacenter networks, such as Jellyfish [47], Slim Fly [7], and Xpander [50], have shown that these networks outperform Clos and leaf-spine networks in theoretical and simulation analysis. However, we have not found any descriptions of such networks being deployed in commercial practice. Why not?

We suspect (although this would be hard to prove!) that physical-deployability concerns limit the practical attractiveness of expander graphs. Jellyfish seems to deter the pre-placement of intra-datacenter fiber between potential switch locations, which is a significant contributor to deployment efficiency. It can be challenging to compute the lengths and bundling for pre-deployed fiber, and Jellyfish's use of regular random graphs makes that "highly non-trivial" [50].

The Xpander and FatClique![55] papers indeed discuss physical deployment issues, especially cabling complexity. Both assert that unlike Jellyfish, they allow effective cable bundling. However, while frequent and rapid incremental addition of machine racks is a financial necessity (§3.5), Xpander requires as many as $d/2$ links to be rewired each time a d-port ToR is added (or removed). (It is also unclear whether Xpander supports mixing ToRs of several radixes, as might be necessary in a long-lived network.)

Non-physical reasons can discourage use of expander-graph networks. Eight years passed between the Jellyfish paper and when Harsh *et al.* described "the first implementation of a routing scheme on standard hardware for expanders or flat networks in general" [26]. They also demonstrated that there are "flat topologies (other than expanders) that outperform leaf-spine networks" at moderate scales.

Singla *et al.* proposed changing the connectivity of VL2 [20] so that instead of connecting ToRs only to aggregation switches, they distribute ToR connections among both aggregation and core switches, and find that this significantly increases the number of ToRs supported by a given network size [46]. However, they do not consider the physical-deployability constraints that would make this difficult; e.g., the increased length of some ToR uplink cables might force the use of more-expensive optics, and might complicate the use of pre-assembled fiber bundles. (Google reported similar issues with the Aquila design [15].)

A "fear factor" may also deter enterprise deployments of novel network designs; the lack of such deployments does not inherently prove that expander graphs are hard to deploy. In fact, given the attractive theoretical properties of expander and flat graphs, we expect further research will find a way to preserve these properties (in part) while resolving the physical-deployment challenges. Google's "evolved" Jupiter design [39] replaces spine blocks with a reconfigurable expander graph between aggregation blocks, while keeping a traditional Clos hierarchy at lower layers.

4.3 Design changes to live networks

Google's original Jupiter [44] used a set of aggregation blocks fully connected to a set of spine blocks (each block composed of packet switches), connected by an OCS layer (see §4.1).

More recently, we developed a modified Jupiter design, in which aggregation blocks are directly connected via the OCS layer; this avoids the considerable cost of the spine blocks, although it does require a more complex control plane [39].

To convert the existing Jupiters from fat-trees to the direct-connect design, technicians must change how fibers connect to OCS units (i.e., disconnect the fibers to spine blocks and add connections to aggregation blocks). To do this in a live network, we temporarily drain traffic from each OCS rack, then technicians perform the complex task of moving a lot of fibers without breaking or mis-connecting any of them, and then we un-drain the rack. This process takes multiple hours of human labor per rack, across many racks.

We draw two lessons from this experience. First, the useful lifetime of a network can exceed the lifetime of its original design; indirection made it much easier to "redesign" a live network. Second, an SDN control plane can do more than update flow tables; it can also coordinate between demand forecasts, availability requirements, manual operations segmented into low-impact chunks, the necessary drains/undrains, and automated testing for wiring errors.

5 What could make things easier?

Between existing experience and some speculation about the future, we see several broad categories of innovations, and areas for further research, that would make physical deployment easier in datacenter networks:

- Technical innovations: new or improved technologies, both hardware and software.
- Process innovations: better ways to organize deployment processes.
- "Digital twins": per Wikipedia, "a digital representation of an intended or actual real-world physical product, system, or process (a physical twin) that serves as the effectively indistinguishable digital counterpart of it for practical purposes, such as simulation, integration, testing, monitoring, and maintenance" [53]. The term was introduced by Grieves [22].

5.1 Technical innovations

Several existing or potential technical innovations potentially can make deployment easier and less error-prone.

For example, while a lot research has covered designs and uses for high-speed optical circuit switches (e.g., [6, 34]), these have not become commercially available. On the other hand, various companies sell relatively slow OCSes [9, 49], and (as mentioned in §4.1), Poutievski *et al.* showed how these can be used to enable both topology engineering and faster incremental deployment [39].

Zhao *et al.* described algorithms to compute a minimal-effort plan to update patch panels in a Clos [56], an NP-complete problem.

Some vendors offer "active" or "intelligent" patch panels [3, 27], which monitor the status of patch-panel connections and can assist with remote or automated diagnosis of faults. However, these are more expensive than passive patch panels, and possibly vulnerable to software bugs.

5.2 Process innovations

Our experience has been that deploying new designs into a datacenter network is much easier if they are substantially similar to prior designs, and when we can easily tell how they do vary. This is especially true for supporting new designs in deployment-automation software. But defining "substantially similar" has proved challenging, because (1) seemingly minor changes can create major deployment headaches, (2) designs can vary in a large number of dimensions, and (3) the range of potential designs is large.

We initially hoped to be able to define a multi-dimensional "capability envelope," representing the variability that our automation software could handle without changes, but that effort stalled due to the number of dimensions, and the many "dimensions" that could not be represented using simple metrics such as length, weight, or line speed.

Instead, we have some initial experience that by moving knowledge about a design out of automation code, and into a declarative data representation, we can at least detect out-of-envelope designs because we cannot represent them without schema changes. Our work on using declarative, structural models of networks [36] to support change management [2] has sometimes helped us spot hard-to-support network designs because we had no existing way to model them. We made these discoveries much earlier than if we had had to study our (imperative) software to deduce where the problems would lie. However, this migration of knowledge from code to data is slow and sometimes difficult to get right.

5.3 Digital twins

The later we detect a problem with either a new product being introduced to our networks, or with a new installation of, or change to, an existing product, the harder it is to clean up the mess. When we discover problems late, often we have made changes to the physical world that are slow and expensive to unwind, especially without causing more problems (see the discussion of "decom" in §2.1).

The digital-twin concept has seen widespread use in many industries [21], but relatively little use in networking, even though (as this paper describes) there are plenty of opportunities to apply it – especially for simulating physical-world operations before we commit to a new design or to a planned deployment. Our goal, towards which we have made just a little progress, is to be able to rapidly test whether an abstract design violates physical-world constraints. In our experience, the costs to remediate mistakes increase dramatically if we only discover them late in these processes.

Digital-twin approaches are not easy to introduce, since they depend on accurate and detailed "multi-physics" modeling spanning network design, physical characteristics of

network (and other) equipment, power, cooling, and the spaces where equipment is deployed. Ideally, these would be automatically derived from high-level (abstract) specifications whenever possible. To be complete, we would also need (possibly simplified) models of human (or robot) capabilities.

One barrier to creation of a comprehensive digital twin is that much of the necessary data exists in a variety of *ad hoc*, poorly-documented, and ambiguous formats – spreadsheets, CAD drawings, proprietary design systems, etc. Based on our experience with network modeling, it is possible to migrate from legacy representations to well-curated ones, but it's also hard to create future-proof taxonomies.

Another barrier is that existing data is often incomplete or wrong. We would greatly benefit from new methods to validate such data, perhaps by inferring design rules that were never formally stated (analogous to prior work on bug-finding [13]). But that cannot fix mundane errors such as recording the wrong position for a rack (which means that another rack might not fit where it is intended); that will require better techniques for measuring the physical world.

Modeling multiple aspects in isolated representations is not sufficient. Grieves highlights the value of a "Unified Repository [...] that will link the two products together." [22] In our experience, without rigorously standardized definitions and identification systems, such linkage falls back onto heuristics, which often cause errors and are hard to maintain. So, while pieces of the representational universe exist (e.g., MALT for network structure [36], Brick for the structure of buildings [5]), today these remain isolated islands.

We would also need multi-physics simulation engines – not to simulate how packets or routing updates propagate through a network, but rather to simulate how network equipment fits into 3D space (including details such as how cables are routed between racks). Many of these simulators already exist for various industries; integrating these with models of network structures (e.g., to know the connections between switches, or to know which switches are vulnerable to common-mode failures) would be a crucial step.

As noted in §2.1, we must also support decom. Our experience is that a multi-dimensional "twin" is especially useful here, to ensure that physical removal of components from a live network does not inadvertently cause an outage. Testing a decom process on a real deployment is especially challenging, because of this risk. Testing on a twin, while it cannot provide perfect coverage, would be much safer and cheaper.

We do formal and informal postmortems of deployment mistakes and delays. *Almost all of these could have been averted if we could do multi-layer digital-twin dry runs.*

5.4 Defining metrics

The interactions between network designs and physical deployment issues are complex. Designers, and especially researchers without access to large datacenters, would benefit from well-defined metrics and objectives for evaluating proposed designs. Also, given that many network-design decisions are complex enough to require ILP or similar solvers, expressing physical-deployability goals via metrics enables these solver-based approaches.

Another value of deployability metrics is that they might reduce fears about adopting novel designs. If one can predict that a new design will not create major deployability problems, network operators might be more open to it.

In §5.2 we described "capability envelopes" as a possible objective; we still believe that these could be developed, at least for some dimensions.

Zhang *et al.* defined several metrics for "lifecycle management complexity," including the number of re-wiring steps for an expansion, and the number of re-wired links per patch panel (see also [56]). To those, one might add locality metrics (e.g., number of patch panels touched).

One might also define "diversity-support" metrics; e.g., the number of different link speeds or switch radixes that can be included in one network without severe problems.

We also need to represent the tradeoff between day-1 costs and longer-term costs, since a hard-to-evolve design might be sufficiently cheaper up-front to merit its use.

Unfortunately, because we cannot always anticipate how future evolution in network designs will affect deployability, it seems impossible to define a closed set of deployability metrics; there will always be room for new work in this area.

6 Other prior work

Several previous publications (but relatively few in the past decade) have at least touched on the interactions between physical deployment issues and network design.

Facebook's Robotron paper [48] focuses on the deployment of network-device configuration, and does not discuss physical deployment or repairs. However, the paper does discuss the "FBNet" representation, which describes (*inter alia*) the relationship between physical components of a router or switch, and abstract entities such as BGP sessions.

Andreyev *et al.* discuss how Meta reduces space and power requirements while scaling up their datacenter network [4].

Popa *et al.* compared various datacenter topologies based on various costs, including an estimated "labor cost" for running cables [38], based on "discussions with operators that suggested the dominant expense in cabling is due to the human cost of manually wiring equipment." However, they apparently did not consider the use of pre-built cable bundles, which subsequently in Singh *et al.* was reported to significantly decrease cost and installation time [44].

Disclosure of ethical concerns

This paper raises no ethical concerns, describes no research on human subjects, or even robot subjects, and does not involve user data or PII.

Acknowledgments

We would like to thank Brighten Godfrey, Andrew Ferguson, Ankit Singla, Chi-yao Hong, and the reviewers for helpful suggestions.

References

[1] Rachit Agarwal, Jayaram Mudigonda, Praveen Yalagandula, and Jeffrey C. Mogul. 2015. *An Algorithmic Approach to Datacenter Cabling*. Technical Report HPL-2015-8. HP Laboratories. https://www.hpl.hp.com/techreports/2015/HPL-2015-8.pdf.

[2] Mohammad Al-Fares, Virginia Beauregard, Kevin Grant, Angus Griffith, Jahangir Hasan, Chen Huang, Quan Leng, Jiayao Li, Alexander Lin, Zhuotao Liu, Ahmed Mansy, Bill Martinusen, Nikil Mehta, Jeffrey C. Mogul, Andrew Narver, Anshul Nigham, Melanie Obenberger, Sean Smith, Kurt Steinkraus, Sheng Sun, Edward Thiele, and Amin Vahdat. 2023. Change Management in Physical Network Lifecycle Automation. In *Proc. USENIX Annual Technical Conference (USENIX ATC 23)*. Boston, MA, 635–653. https://www.usenix.org/conference/atc23/presentation/al-fares

[3] Alder Networks. [n.d.]. Active Patch Panels. http://www.aldernetworks.co.uk/activepatchpanels.html.

[4] Alexey Andreyev, Xu Wang, and Alex Eckert. 2019. Reinventing Facebook's data center network. https://engineering.fb.com/2019/03/14/data-center-engineering/f16-minipack/.

[5] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, Mario Berges, David Culler, Rajesh Gupta, Mikkel Baun Kjærgaard, Mani Srivastava, and Kamin Whitehouse. 2016. Brick: Towards a Unified Metadata Schema For Buildings. In *Proc. ACM Intl. Conf. on Systems for Energy-Efficient Built Environments*. 41–50.

[6] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. In *Proc. SIGCOMM*. 782–797.

[7] Maciej Besta and Torsten Hoefler. 2014. Slim Fly: A Cost Effective Low-Diameter Network Topology. In *Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis*. 348–359.

[8] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95.

[9] CALIENT Technologies. 2022. CALIENT'S OPTICAL CIRCUIT SWITCH: Data Sheet. https://www.calient.net/wp-content/uploads/2022/06/Datasheet_Calients-Optical-Circuit-Switches.pdf.

[10] Stephen Callaghan and JR Rivers. 2022. AWS re:Invent 2022 - Dive deep on AWS networking infrastructure (NET402). (video) https://www.youtube.com/watch?v=HJNR_dX8g8c.

[11] Thomas Claburn. 2021. Supermicro spy chips, the sequel: It really, really happened, and with bad BIOS and more, insists Bloomberg. https://www.theregister.com/2021/02/12/supermicro_bloomberg_spying/. *The Register* (2021).

[12] Andrew R. Curtis, S. Keshav, and Alejandro Lopez-Ortiz. 2010. LEGUP: Using Heterogeneity to Reduce the Cost of Data Center Network Upgrades. In *Proc. CoNEXT*.

[13] Dawson Engler, David Yu Chen, Seth Hallem, Andy Chou, and Benjamin Chelf. 2001. Bugs as Deviant Behavior: A General Approach to Inferring Errors in Systems Code. In *Proc. SOSP*. 57–72.

[14] João Ferreira, Naader Hasani, Sreedhevi Sankar, Jimmy Williams, and Nina Schiff. 2018. Fabric Aggregator: A flexible solution to our traffic demand. https://engineering.fb.com/2018/03/20/data-center-engineering/fabric-aggregator-a-flexible-solution-to-our-traffic-demand/.

[15] Dan Gibson, Hema Hariharan, Eric Lance, Moray Mclaren, Behnam Montazeri, Arjun Singh, Stephen Wang, Hassan Wassel, Zhehua Wu, Sunghwan Yoo, Raghuraman Balasubramanian, Prashant Chandra, Michael Cutforth, Peter James Cuy, David Decotigny, Rakesh Gautam, Alex Iriza, Milo M. K. Martin, Rick Roy, Zuowei Shen, Ming Tan, Ye Tang, Monica C Wong-Chan, Joe Zbiciak, and Amin Vahdat. 2022. Aquila: A unified, low-latency fabric for datacenter networks. In *Proc. NSDI*.

[16] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. 2011. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *Proc. SIGCOMM*. 350–361.

[17] Dan Goodin. 2020. Custom-made UEFI bootkit found lurking in the wild. https://arstechnica.com/information-technology/2020/10/custom-made-uefi-bootkit-found-lurking-in-the-wild/.

[18] Dan Goodin. 2022. Discovery of new UEFI rootkit exposes an ugly truth: The attacks are invisible to us. https://arstechnica.com/information-technology/2022/07/researchers-unpack-unkillable-uefi-rootkit-that-survives-os-reinstalls/.

[19] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or Die: High-Availability Design Principles Drawn from Google's Network Infrastructure. In *Proc. SIGCOMM*.

[20] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. In *Proc. SIGCOMM*. 51–62.

[21] Samuel Greengard. 2019. Digital Twins Grow Up. https://cacm.acm.org/news/238642-digital-twins-grow-up/fulltext.

[22] Michael Grieves. 2014. Digital twin: manufacturing excellence through virtual factory replication. available at https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf. *White paper* 1, 2014 (2014), 1–7.

[23] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-Space Optics. In *Proc. SIGCOMM*. 319–330.

[24] James Hamilton. 2010. Overall Data Center Costs. https://perspectives.mvdirona.com/2010/09/overall-data-center-costs/.

[25] Robin Harris. 2017. Hidden Figures and the IBM 7090 computer. https://www.zdnet.com/article/hidden-figures-and-the-ibm-7090-computer/.

[26] Vipul Harsh, Sangeetha Abdu Jyothi, and P. Brighten Godfrey. 2020. Spineless Data Centers. In *Proc. HotNets*. 67–73.

[27] Rakesh K. 2011. What is an Intelligent Patch Panel & What are its advantages? https://excitingip.com/2632/what-is-an-intelligent-patch-panel-what-are-its-advantages/.

[28] Srikanth Kandula, Dina Katabi, and Jean-Philippe Vasseur. 2005. Shrink: A Tool for Failure Diagnosis in IP Networks. In *Proc. SIGCOMM Workshop on Mining Network Data*. 173–178.

[29] John Kim, William J. Dally, and Dennis Abts. 2007. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. In *Proc. ISCA*. 126–137.

[30] Mark Lechtik, Vasily Berdnikov, Denis Legezo, and Ilya Borisov. 2022. MoonBounce: the dark side of UEFI firmware. https://securelist.com/moonbounce-the-dark-side-of-uefi-firmware/105468/.

[31] Robert Lychev, Sharon Goldberg, and Michael Schapira. 2013. BGP Security in Partial Deployment: Is the Juice Worth the Squeeze?. In *Proc. SIGCOMM*. 171–182.

[32] Michael R. Marty, Philip Wells, Hong Liu, and Dennis Abts. 2023. Retrospective: Energy Proportional Datacenter Networking. In *50 Years of the International Symposia on Computer architecture (selected papers)*.

[33] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (March 2008), 69–74.

[34] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. 2017. RotorNet: A Scalable, Low-Complexity, Optical Datacenter Network. In *Proc. SIGCOMM*. 267–280.

[35] Karen Miranda, Antonella Molinaro, and Tahiry Razafindralambo. 2016. A survey on rapidly deployable solutions for post-disaster networks. *IEEE Communications Magazine* 54, 4 (2016), 117–123.

[36] Jeffrey C Mogul, Drago Goricanec, Martin Pool, Anees Shaikh, Douglas Turk, Bikash Koley, and Xiaoxue Zhao. 2020. Experiences with Modeling Network Topologies at Multiple Levels of Abstraction. In *Proc. NSDI*.

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

[38] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. 2010. A Cost Comparison of Datacenter Network Architectures. In *Proc. CoNEXT*.

[39] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Connor, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yamsura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. 2022. Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking. In *Proc. SIGCOMM*.

[40] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proc. NSDI*. San Jose, CA, 399–412.

[41] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C. Snoeren. 2017. Passive Realtime Datacenter Fault Detection and Localization. In *Proc. NSDI*. 595–612.

[42] Bruce Schneier. 2023. BlackLotus Malware Hijacks Windows Secure Boot Process. https://www.schneier.com/blog/archives/2023/03/blacklotus-malware-hijacks-windows-secure-boot-process.html.

[43] Scott Schweitzer. 2015. 5 Reasons Infiniband Will Lose Relevance After 100G. https://technologyevangelist.co/2015/11/23/5-reasons-infiniband-will-lose-relevance-at-100g/.

[44] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proc. SIGCOMM*.

[45] Rachee Singh, Muqeet Mukhtar, Ashay Krishna, Aniruddha Parkhi, Jitendra Padhye, and David Maltz. 2021. Surviving Switch Failures in Cloud Datacenters. *SIGCOMM Comput. Commun. Rev.* (May 2021), 2–9.

[46] Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. 2014. High Throughput Data Center Topology Design. In *Proc. NSDI*. 29–41.

[47] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *Proc. NSDI*.

[48] Yu-Wei Eric Sung, Xiaozheng Tie, Starsky HY Wong, and Hongyi Zeng. 2016. Robotron: Top-down Network Management at Facebook Scale. In *Proc. SIGCOMM*.

[49] Telescent Inc. [n.d.]. G4 NTM. https://www.telescent.com/products.

[50] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. 2016. Xpander: Towards Optimal-Performance Datacenters. In *Proc. CoNEXT*. 205–219.

[51] Weitao Wang, Masoud Moshref, Yuliang Li, Gautam Kumar, T. S. Eugene Ng, Neal Cardwell, and Nandita Dukkipati. 2023. Poseidon: Efficient, Robust, and Practical Datacenter CC via Deployable INT. In *Proc. NSDI*. USENIX Association, Boston, MA, 255–274.

[52] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. In *Proc. HotNets*. 17–24.

[53] Wikipedia contributors. 2023. Digital twin — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Digital_twin&oldid=1157427837. [Online; accessed 14-June-2023].

[54] Minlan Yu, Jennifer Rexford, Michael J. Freedman, and Jia Wang. 2010. Scalable Flow-Based Networking with DIFANE. In *Proc. SIGCOMM*. 351–362.

[55] Mingyang Zhang, Radhika Niranjan Mysore, Sucha Supittayapornpong, and Ramesh Govindan. 2019. Understanding Lifecycle Management Complexity of Datacenter Topologies. In *Proc. NSDI*. Boston, MA, 235–254.

[56] Shizhen Zhao, Rui Wang, Junlan Zhou, Joon Ong, Jeffrey C Mogul, and Amin Vahdat. 2019. Minimal Rewiring: Efficient Live Expansion for Clos Data Center Networks. In *Proc. NSDI*.

[57] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y. Zhao, and Haitao Zheng. 2012. Mirror Mirror on the Ceiling: Flexible Wireless Links for Data Centers. In *Proc. SIGCOMM*. 443–454.