Precise Data Center Traffic Engineering with Constrained Hardware Resources

Shawn Shuoshuo Chen, *Carnegie Mellon University;* Keqiang He, *Airbnb;*
Rui Wang, *Google;* Srinivasan Seshan and Peter Steenkiste, *Carnegie Mellon University*

https://www.usenix.org/conference/nsdi24/presentation/chen-shawn

This paper is included in the
Proceedings of the 21st USENIX Symposium on
Networked Systems Design and Implementation.

April 16–18, 2024 • Santa Clara, CA, USA

978-1-939133-39-7

Open access to the Proceedings of the
21st USENIX Symposium on Networked
Systems Design and Implementation
is sponsored by

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Precise Data Center Traffic Engineering with Constrained Hardware Resources

Shawn Shuoshuo Chen ♠ Keqiang He ♦ Rui Wang ♣ Srinivasan Seshan ♠ Peter Steenkiste ♠

♠ *Carnegie Mellon University* ♦ *Airbnb* ♣ *Google*

Abstract

Data center traffic engineering (TE) routes flows over a set of available paths following custom weight distributions to achieve optimal load balancing or flow throughput. However, as a result of hardware constraints, it is challenging, and often impossible for larger data center networks, to precisely implement the TE weight distributions on the data plane switches. The resulting precision loss in the TE implementation causes load imbalances that can result in congestion and traffic loss.

Instead of treating all flows equally, we adapt the hardware resource allocation to a flow's traffic volume and its contribution to the overall precision loss. We intelligently prune select ports in weight distributions and merge identical distributions to free up hardware resources. Evaluation using realistic traffic loads shows that our techniques approximate ideal TE solutions under various scenarios within 7% error, compared to a 67% error for today's state-of-the-art approach. In addition, our design avoids traffic loss triggered by switch rule overflow. Finally, the execution time is 10× faster than the current approach.

1 Introduction

Data center networks (DCNs) have resorted to using links with abundant bandwidth, topologies with rich connectivity, and operation at enormous scale to meet growing application needs. However, these additional resources come at a significant cost and DCN operation must ensure efficient utilization of the network infrastructure. Traffic engineering (TE) plays a critical role in addressing this efficiency need by routing traffic over carefully chosen paths. Existing TE designs consist of two parts: (1) calculating the optimal plan for mapping of flows to links, which is called the *TE solution*; (2) implementing the TE solution on data plane switches, which we call *TE implementation*. Past TE research [3–5, 9, 22, 31, 38] has mostly focused on the TE solution, but the TE implementation impacts operational efficiency greatly and, despite this, has received much less attention.

Equal-Cost Multi-Path (ECMP) [30] is the most widely adopted method to implement TE solutions because of its universal hardware support. Weighted-Cost Multi-Path (WCMP) [73] is an extension of ECMP that supports a weighted split of traffic among paths by using forwarding table entry replication. This enables finer-grained traffic control but can consume much more space in the switch hardware table. As a result, two problems occur. First, some groups (the data structure that implements weight distributions in the switch) used by the TE solution cannot be installed in the switch after the switch group table becomes full, leading to traffic loss. Second, if the weights are adjusted to use less space in the group table, the resulting TE implementation may have poor TE performance. We refer to the difference between link utilization of the TE solution and of the TE implementation as the *precision loss*.

Recently, spine-free DCN designs [7, 45, 46] propose to replace the traditional Clos topology with clusters connected as a full mesh. Spine-free design reduces the total cost of ownership and accelerates DCN evolution, but we found that they also increase usage on the group table, exacerbating WCMP's performance issues.

We seek to answer this question: *how to implement TE solutions with high precision given constrained hardware space?* Adjusting weight distributions of groups proves effective in lowering table space requirement. However, when we compared the groups retrieved from production switches with the TE solution, we found that the precision loss is often high. We analyzed the differences, which led to several insights on how to reduce the group table usage with lower precision loss: (1) Groups that serve significant traffic contribute disproportionally to the precision loss. They should be given more group table space. (2) Some groups have very skewed weight distributions and consume many table entries. Normal weight adjustment that preserves all paths struggles to avoid high precision loss. However, certain paths can be pruned from the distribution to significantly lower the table space requirement, often with little impact on precision loss. (3) After adjusting weights, some groups can become identical. Deduplicating such groups frees up group table space without incurring precision loss.

We develop three heuristics based on the insights: (1) traffic-aware resource allocation, (2) prune paths, and (3) deduplica-

♦ This work was done when Keqiang He was at Google.

tion. We then implement two algorithms that map TE solutions to switch hardware, applying the above heuristics. They both aim to minimize precision loss while adjusting group weights to meet the hardware resource constraint. Heuristic 2 is invoked conditionally when the resource constraint is otherwise impossible to satisfy. The two algorithms make different tradeoffs: one has very low precision loss, while the other runs faster but with higher precision loss in some cases.

Since evaluation in production DCNs could impact user traffic, we built a data center TE evaluation framework named FabricEval that allows us to run controlled experiments using production-like DCN topologies and traffic loads. Our implementations of the algorithms are compared to the state-of-the-art approach [73] under both common and extreme DCN configurations. Results show that our design implements TE solutions with no more than 7% precision loss, while using the state-of-the-art approach can have up to 67% precision loss. Furthermore, our design achieves a 10× execution speed improvement over the existing approach and successfully avoids traffic loss triggered by exhausted hardware resources.

In summary, this paper makes the following contributions:

- We quantify the TE precision loss problem for a variety of DCN topologies, scale, and traffic patterns. Our study identifies five root causes of precision loss (§3).
- We present two algorithms with different speed and optimality tradeoffs that can generate high precision TE implementations, at the same time complying with hardware resource constraints (§4).
- We develop a TE evaluation framework that mirrors Google's production DCNs and TE system and use it to evaluate our design (§5).
- We release the source code of our algorithm implementation and the evaluation framework FabricEval [1].

2 Background

We provide an overview of how data center traffic engineering systems are designed, as well as the hardware resource constraint these systems face. We then describe how the emerging spine-free DCN architecture differs from the traditional Clos-based design from a traffic engineering perspective.

2.1 Traffic engineering in data center networks

End hosts in DCNs are densely connected by many paths traversing different switches. The TE system acts as a centralized controller that manages all links/switches and routes traffic end-to-end via available paths. It looks for an optimal plan to fulfill a list of goals, including serving all traffic `demands` (bytes to send between each src-dst pair), maximizing flow throughput, balancing link loads etc. The TE problem is typically formulated as a multi-commodity flow (MCF) optimization problem. Though it is theoretically possible to obtain a TE solution by solving an MCF with comprehensive constraints, e.g., switch group table size limits, the scale of today's DCNs makes this approach prohibitively

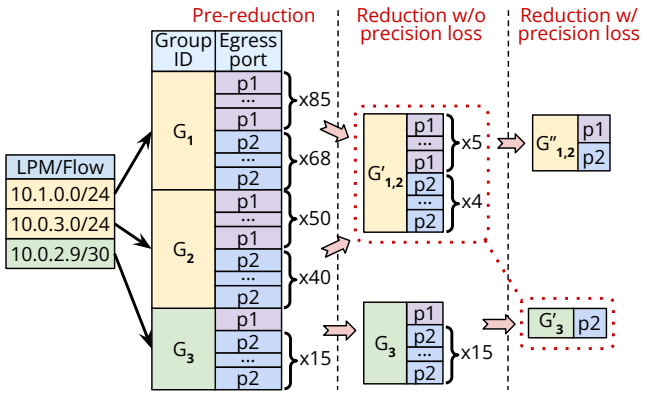


Figure 1: Hardware representation of flows/groups, and a demonstration of group reduction.

expensive. Common TE systems break the task into two steps: first generate a *TE solution* to the MCF problem without hardware constraints, and then map the TE solution to a data plane *TE implementation*. Some TE systems generate solutions hierarchically [18] for better scalability, but this does not impact the TE implementation step—the focus of this paper.

The TE solution specifies for each demand which paths to use and how much traffic to place on each path. When paths diverge at a switch, the demand is split by a ratio (weight distribution) according to the routing decision. The TE system updates each switch along the paths to reflect the routing decision. To map the TE solution to a TE implementation, the end-to-end TE solution first needs to be broken down into switch-local routing decisions. Next, switch-local decisions are translated to switch rules, namely longest prefix match (LPM) entries and groups, and installed on switches using switch control plane APIs (e.g., OpenFlow/SDN [44]).

An LPM/flow entry points to a group that is used for packet forwarding (see the pre-reduction phase in Figure 1). A group consists of a set of ports, each with an integer weight reflecting the rounded fractional traffic volumes assigned to each path. Weight distributions are achieved via port/entry replication since all entries in a group are selected by uniform hashing with equal probability at runtime. When ECMP is used, each port in a group uses exactly one entry in the group table, which is very efficient. With WCMP, the entries a port uses is proportional to its weight in the group. It is easy to see that the number of entries needed depends strongly on the weight ratios, e.g., G_1=$\{p1{:}85; p2{:}68\}$ will require a lot of entries.

The group table space is a scarce resource. Table 1 lists the group table capacity of a few types of switches found in production DCNs (also called fabric). Given the large number of flows, and the use of WCMP in today's DCNs, it is hard to avoid exhausting the group table. A group reduction algorithm is needed to reduce group sizes—the sum of entries used by each group—so the TE solution fits in the group tables. The group reduction algorithm should generate a new weight distribution that closely approximates the original one. For example in Figure 1, groups G_1-G_3 can be reduced to $G'_{1,2}$ and G'_3 with

Table 1: Switch hardware profile of an example deployment.

Switch generation	Port speed	Group table size
Gen. 1	40 Gbps	4096 entries
Gen. 2	100 Gbps	16384 entries
Gen. 3	200 Gbps	32768 entries

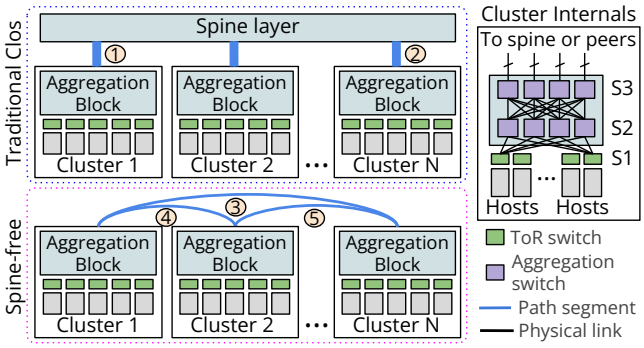


Figure 2: Traditional Clos and spine-free DCN topology. A path segment is an abstraction of parallel physical links.

minimal precision loss, while lowering table usage from 259 to 10 entries (more details given in §3 and §4). The state-of-the-art group reduction algorithm is called *WCMP TableFitting* [73].

2.2 Spine-free data center networks

Until recently, DCNs have used a traditional Clos topology that connects clusters to a spine layer, as illustrated in Figure 2 (top left). Inside each cluster, a non-blocking tree topology connects a large number of hosts. In this paper, we consider a 3-stage cluster layout, as illustrated in Figure 2 (right). Stage 1 (S1) switches are also called ToR switches. Stage 2 (S2) and stage 3 (S3) switches are aggregation switches. The S2 and S3 switches in each cluster also form an aggregation block.

Clos DCNs can use all links leaving a cluster to reach another cluster (assuming no failure) because they belong to the same shortest path. For example, for cluster 1 to reach cluster N in the traditional Clos DCN in Figure 2, there is only one shortest path: ①→② through the spine. The spine is often the link speed bottleneck of all paths because of infrequent upgrades. TE in Clos DCNs is thus straightforward, with all links running at the same speed after speed auto-negotiation, flows are spread across links in the shortest path in the form of either ECMP or very simple weight distributions (e.g., due to asymmetric link striping or some out-of-service links). In a cluster with egress demands, each switch has one flow entry matching the aggregated IP prefix of each destination cluster. Every flow references an individual group that contains ports used for reaching the next-stage switches, e.g., S1→S2 and so on[1]. Groups on S3 switches contain ports to the spine.

The spine layer is expensive and requires frequent upgrades to keep up with the growing traffic demand. To overcome this cost disadvantage, DCNs are moving to a spine-free topology enabled by optical circuit switching [64] that directly connects the clusters using a full mesh, as illustrated in Figure 2 (bottom left). Note that the intra-cluster topology and number of links coming out of a cluster remain the same.

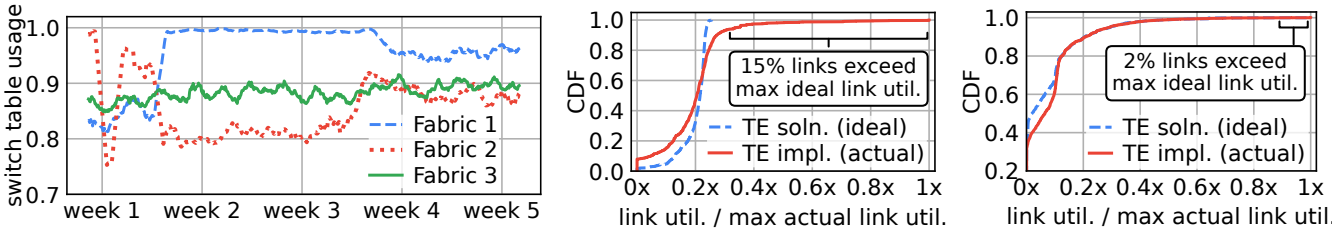
TE in spine-free DCNs is different and more challenging to implement than in Clos DCNs in two ways. First, link speed differences hidden by the slow spine layer are now exposed. This translates to more skewed weight distributions in groups, which consume more hardware resources. Second, spine-free DCNs have far fewer direct-connect links in the shortest path between clusters. In order to support high traffic volumes, non-shortest-path forwarding must be adopted to utilize all links. In the spine-free DCN of Figure 2, cluster 1 can reach cluster N via the direct path ③, as well as multiple indirect paths such as ④→⑤ through cluster 2. However, graph theory shows that longer paths lead to higher bandwidth overhead and lower flow throughput [61]. Therefore, the length of non-shortest paths is limited to two hops, i.e., a flow can transit through at most one intermediate cluster to reach the destination. We term these two types of paths *direct* and *indirect* paths. In practice, TE avoids using all indirect paths simultaneously since this leads to a Valiant Load Balancing (VLB) scheme [72] with higher link loads, although VLB does reduce the group table footprint. The benefits of combining VLB with WCMP are left for future work.

For both direct and indirect paths, switches of the source cluster need to install flows and groups the same way as in Clos DCNs. We call these `src`-typed flows/groups. In addition, the intermediate cluster in an indirect path uses separate `transit`-typed flows/groups to forward traffic from source clusters to the final destination. Indirect-path traffic only traverses the aggregation block. `transit` flows are installed on all S2/S3 (but S1) switches to match on the source and destination prefixes. Corresponding `transit` groups reflect indirect-path traffic in the direction of S3→S2→S3.

3 TE precision loss challenges

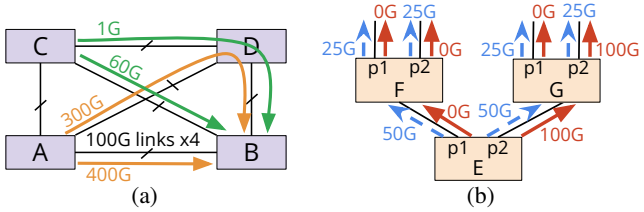
To illustrate the pressure on group tables, Figure 3a shows the group table utilization on Google's production spine-free fabrics. We see that group table usage can be as high as 90-100%. However, high table usage does not directly represent the negative impact on traffic. The key network-level metrics are the traffic load and loss on each link. We observed that high table usage strongly correlates with high link utilization. For example, Figure 3b shows the actual link utilization in a spine-free fabric in which the TE implementation produces a max table utilization of 90%. We see that the TE implementation results in a much more imbalanced link utilization distribution compared to the (ideal) TE solution. Moreover, 15% of the links exceed the max link utilization. The worst few links see an actual utilization 5 times higher than that of the TE solution, resulting in congestion loss. Figure 3c shows that a Clos fabric of similar scale also experiences

[1] A small, constant number of static flows/groups are installed on each switch to handle intra-cluster and ingress (destination) traffic. They are not an important factor in the discussion of this paper, hence ignored.



(a) (b) (c)

Figure 3: (a) 30-day time series of maximum group table usage in 3 arbitrary production fabrics. Usage is normalized to the table size of each switch. (b) Normalized link utilization CDF plotted from the TE solution and TE implementation in one of the spine-free fabrics. (c) Normalized link utilization CDF in a traditional Clos fabric of scale similar to the fabric in (b).



(a) (b)

Figure 4: (a) Abstract view of a small spine-free network. (b) Cascading precision loss in a 2-stage aggregation block. Desired traffic is in dashed blue, actual traffic is in solid red.

precision loss due to group table constraints, although the problem is less severe. We focus our discussion on a spine-free TE setting, though our findings also apply to Clos DCNs.

We compared a large number of TE solutions and their result and identified five key challenges that exacerbate precision loss: (1) Large network scale leads to more groups sharing hardware resources; (2) TE solutions are generated without considering hardware constraints and often contain skewed weights that are difficult to reduce; (3) Heterogeneous DCN hardware subjects parts of the network to higher precision loss; (4) Diverse paths in spine-free TE result in larger group sizes; and (5) The cascading impact of errors in one stage to the next in multi-stage DCNs increases precision loss multiplicatively. We now explain these challenges in detail.

① **Scalability challenge.** The total number of flows/groups to install on a switch increases with the number of clusters because each cluster has demands for virtually all clusters. As the number of groups in a table increases, each group has to use fewer entries, resulting in higher precision loss overall.

Considering the Clos and spine-free DCNs in Figure 2, hosts in each cluster are usually assigned continuous IP addresses that aggregate into a single prefix. The number of distinct prefixes equals the number of clusters N. In Clos DCNs, each switch needs to support N-1 `src` flows/groups, one for each destination cluster. In spine-free DCNs, each switch not only has N-1 `src` groups, but also one `transit` group for each indirect path traversing it. In the worst case, a cluster can be used by all (N-2) source clusters to reach (N-1) destination clusters indirectly. This sums up to (N-1)(N-2) `transit` groups on each aggregation switch.

② **Skewed weights challenge.** Since the TE system is not aware of any hardware resource constraint, the TE solution it generates may include highly skewed traffic distributions. Consider the example in Figure 4a, demand C→B has a total volume of 61G. The TE solution assigns 1G to path C→D→B and 60G to path C→B—a rather skewed split because path C→A→B has no slack capacity left and at least two paths must be used for diversity. Aggregation blocks A, B, C, and D each has four physical S3 switches. The four switches in each block are connected pair-wise (via four links) to the four switches in each of the other blocks. As a result, the `src` group for demand C→B on the S3 switches in C has a weight distribution of 0.25 : 15, which is rounded to 1 : 15.

Demand C→B experiences precision loss for two reasons. First, rounding weights to integer as required by the hardware triggers precision loss. We could also use 1 : 60, but this increases the group size. Second, if we cannot afford 16 entries, we can reduce the larger weight while retaining all member ports in the group, which is a common group reduction behavior. However, this further increases precision loss and leads to oversub(scription) on link C→D. In the worst case, 1 : 15 is reduced to 1 : 1 (ECMP), resulting in an oversub of $\frac{\text{actual volume}}{\text{desired volume}} = \frac{1/2 \cdot (0.25+15)}{0.25} = 30.5$ times more traffic on link C→D. Alternatively, precision loss can be lowered if certain ports are removed from the group. We discuss this option as a heuristic in §4.

③ **Heterogeneity challenge.** Cloud data centers are expanded incrementally to deal with growing traffic demand. Newly installed clusters are always equipped with the latest generation switch hardware, which means different generations of hardware co-exist in the DCN. Switches differ in link speeds and group table sizes, as demonstrated in Table 1. This heterogeneity aggravates precision loss in two ways: (1) The smaller table size of older switches forces groups to be reduced more heavily; (2) Mixed link speeds make group sizes larger on newer generation switches.

Gen. 3 switches have bandwidth 5× higher than Gen. 1 switches, but group table size is 8× larger. This means newer switches can support more groups with less group reduction if flow/group counts scale with bandwidth; however, older switches may will struggle to store the necessary state.

In spine-free DCNs, a switch can see multiple speeds across its ports since they may peer with switches of different

generations. The TE system seeks to balance link loads by placing more traffic on links with higher speed. A switch with mixed link speeds (e.g. Gen 3 switches with connections to slower hardware) increases the skew of group weights. This is defined as the max-to-min weight ratio of a group. The larger the ratio, the more skewed a group is. Our experiments (§5.4) show that in a baseline spine-free topology, the average skew ratio on Gen. 1 switches, which don't suffer from this issue, is 484 and the max ratio is 1176. In contrast, for Gen. 3 switches, the average ratio is 1187 and the max ratio is 2676.

④ **Path diversity challenge.** Group pruning can result in a significant reduction of path diversity which may result in high link utilization when there are traffic shifts or link failures. [52] proposes a technique that accounts for potential demand shifts by imposing a minimum level of path diversity for each flow. We incorporate this technique as a path diversity constraint into our MCF formulation[2] that generates TE solutions. With this constraint, TE solutions must use at least a certain percentage of the available paths to serve each demand. This percentage is referred to as the *spread*. While this constraint makes the TE solution more resilient, it degrades the solution optimality.

We compared the TE solutions generated by the MCF without and with a path diversity constraint for the same traffic load. Findings suggest that the standard formulation tends to use as few paths as possible. Many groups in the standard solution contain only one member port because the demand is small enough to fit onto one path. These groups use only one entry in the group table. Solution to the modified formulation uses several paths as the diversity constraint mandates. With a *spread*=50%, flows use more paths, which increases the pressure on the group table. As mentioned in the Heterogeneity challenge, larger groups will be reduced more heavily to fit into the same table space, hence causing high precision loss. We present a quantitative study in §5.3.

⑤ **Cascading precision loss challenge.** The multi-stage DCN topology has a cascading effect that amplifies precision loss multiplicatively as traffic flows from upstream switches to downstream switches.

To understand the cascading effect, consider the example in Figure 4b. Switches E, F and G form a 2-stage topology inside an aggregation block. A demand of 100G needs to be sent out from switch F and G. The TE solution (illustrated in dashed blue) divides the demand uniformly on upstream (S2) switch E, i.e., it sends 50G to both F and G. Next, downstream (S3) switches F and G each uniformly split the traffic between their two ports. However, to conserve group table entries, the TE implementation places all 100G on link E→G (shown in solid red) while switch F receives zero traffic. The oversub of link E→G is 2, while the undersub of link E→F is 0. This oversub-/undersub is inherited by the downstream links on switches F and G. The two ports on switch F are undersubscribed with zero traffic, regardless of the groups installed on F. On the other hand, even if switch G implements a perfect ECMP group, port G-p1 and G-p2 will each carry 50G, which yields an oversub of 2. This is the baseline precision loss inherited from switch E.

This example only shows the cascading precision loss inside one aggregation block. For indirect paths, the aggregation block of intermediate clusters inherits the baseline precision loss from the upstream aggregation block in source clusters.

[2] §A.1 details the formulation. Equation 2f is the path diversity constraint.

4 Design

We now present two algorithms that lower the TE precision loss compared to current group reduction solutions. Our algorithms, called *Direct Mixed-Integer Reduction (DMIR)* and *Iterative Greedy Reduction (IGR)*, aim to convert a set of original input groups to weight-reduced output groups so that the sum of the group sizes does not exceed hardware limit. DMIR builds on top of mixed-integer programming (MIP) to approximate groups to the TE solution. Specifically, it uses an MIP solver (e.g., Gurobi [24]) to directly find the optimal weight assignment for each group. Since MIP-based solutions have exponential time complexity, we also develop IGR which greedily searches for the smallest-sized groups that best approximate the TE solution in polynomial time. IGR is similar to WCMP TableFitting but uses additional heuristics.

Both algorithms achieve group reduction by decreasing group weights relatively. They also use three heuristics to deal with challenging scenarios (§3) in the reduction process. We first describe the heuristics in §4.1 and then introduce the algorithms in §4.2 and §4.3.

4.1 Shared heuristics

The heuristics described next are used in both algorithms. The first two, Group Sharing and Group Pruning, directly reduce the number of entries used in the group table. The third heuristic, Table Carving, ensures that high-volume flows have sufficient group entries to accurately distribute traffic.

Group Sharing. Group table usage depends on both the number of groups and the number of entries per group. As the network size increases, the number of groups to install on a switch increases, requiring more hardware resources (see §3 Scalability challenge). This heuristic reduces the number of groups by eliminating duplicates. Two groups are identical if they use the same ports and weights for each port. While `src` and `transit` flows/groups must be kept isolated by a design requirement, flows of the same type can share the same group entries to distribute their traffic. This optimization does not incur any penalty, since it does not change the traffic distribution. For example, G_1 and G_2 in Figure 1 can be reduced to the same $G'_{1,2}$, which is shared by the two flows.

We find that while `src` groups generally differ, `transit` groups often become identical after group reduction. For instance, the TE solution in Figure 4a leverages both direct and indirect paths to fulfill demands A→B and C→B. Two `transit` groups are installed on switches of aggregation block D, one for path C→D→B and one for path A→D→B. Since

D→B is a common segment, the same ports are used in both `transit` groups. The two groups only differ in pre-reduction port weights. Considering any S3 switch in D, it only has one port connected to B, as explained in the Skewed Weights challenge. The `transit` group for C→D→B places 0.25G on this port while the `transit` group for A→D→B puts 75G on it. Both groups can be reduced to single-port ECMP groups with weight 1, so they become identical. Generally speaking, `transit` groups for indirect paths with a common segment are identical post-reduction.

In production DCNs, groups are reduced batch by batch as the TE system generates new groups in response to the ever-changing network conditions. Since Group Sharing itself does not incur TE precision loss, it should always be invoked to conclude a round of reduction. The saved table space can subsequently benefit the next batch.

Group Pruning. Sometimes not all groups in the TE solution fit in the table, even if they are reduced to ECMP. In this case, some groups are not installed, and their corresponding traffic is dropped by the switch. Barring this heuristic, group reduction algorithms do not proactively reduce groups beyond ECMP because they aim to preserve the path diversity. But as a last resort to avoid dropping traffic, our algorithms invoke this Group Pruning heuristic that prunes ports from groups.

The key question is which port to prune. There exist multiple victim selection policies, e.g., prune the first port or a random port. The strategies are easy to implement, but they are not necessarily good choices. For example, a port carrying 90% of the group traffic might be pruned, impacting precision loss significantly. Our pruning policy considers both group size reduction (which impacts the path diversity of the group) and the increase in precision loss. Similar to WCMP TableFitting, our goal is to limit the max port oversub in a group. For IGR, we prune the port that results in the least increase in max oversub across all member ports, i.e., the port with the smallest weight (traffic volume). For example, p1 in G_3 of Figure 1 can be pruned—the size of pruned G'_3 becomes one, while the remaining p2 is merely oversubscribed by 1.016×. As discussed later in §4.2, the best port to prune in DMIR is not always the one with the smallest weight, but rather the port that yields the optimal objective.

We expect Group Pruning to be especially useful in addressing the Skewed Weights and Heterogeneity challenges. The reason is that both challenges lead to small weights on some ports, which can be pruned with minimal impact on precision loss. Our evaluation in §5.8 shows that Group Pruning has very little impact on path diversity.

Table Carving. This heuristic addresses the Path Diversity and Cascading Precision Loss challenges. Groups are forced to use more egress ports to satisfy the path diversity constraint. This means groups used by flows with a low traffic volume may have a lot of non-zero weights and large sizes, even though their contribution to the overall precision loss is more limited than high-volume groups. We develop Table Carving to ensure high-volume groups are prioritized over others and can receive sufficient resources during group reduction.

Algorithm 1 *DMIR*($\{G_i\}, T$), $1 \le i \le n$. $\{G_i\}$ is a set of `src` or `transit` groups. T is the available table space.

1: *// Step 1: Table Carving.*
2: **for** $i = 1$ **to** n **by** 1 **do**
3: $\quad T_i = \text{MAX}(len(G_i), \left\lfloor \frac{\text{SUM}(G_i)}{\sum_i \text{SUM}(G_i)} \cdot (T - \sum_i len(G_i)) \right\rfloor)$
4: *// Step 2: single-group optimization.*
5: $\{G'_i\} = \{G_i\}$
6: **for** $i = 1$ **to** n **by** 1 **do**
7: $\quad G'_i = \text{SINGLEGROUPMIP}(G_i, T_i)$ // Equation 1.
8: *// Step 3: reclaim and redistribution.*
9: **for** $G'_i = G'_1, \ldots, G'_n \in \text{SORT}(\{G'_i\})$ **do**
10: $\quad unused = \text{RECLAIMUNUSED}(\{G'_i\})$
11: $\quad$ **if** $unused > 0$ **then**
12: $\quad\quad G'_i = \text{SINGLEGROUPMIP}(G_i, T_i + unused)$
13: *// Step 4: Group Sharing.*
14: **return** DEDUP($\{G'_i\}$)

Table Carving allocates table entries exclusively to each group. A minimum number of entries are allocated per-group to avoid that low-volume groups end up with too few entries and little path diversity. Remaining entries are allocated to groups in proportion to their traffic volume. Note, a group will receive an allocation at least equal to the number of entries required by its ECMP form. Some low-volume groups are likely to be reduced to ECMP (or even get pruned) if they are only allocated ECMP-sized entries. Nevertheless, the minimum allocation is a tradeoff between protecting high-volume groups and avoiding starving low-volume groups.

While $G'_{1,2}$ in Figure 1 can be pruned to $G''_{1,2}$ to further reduce table usage without hurting path diversity, reducing G_3 to G'_3 is preferred because it carries less traffic. In addition to the Path Diversity challenge, this heuristic also helps address Cascading Precision Loss. Protecting the high-volume groups in the upstream switches through Table Carving effectively limits the baseline precision loss on the downstream switches.

4.2 Direct mixed-integer reduction

DMIR is a parallel group reduction algorithm. For a given set of input groups, DMIR performs a reduction on each group individually while ensuring the overall resource constraint is not violated. Algorithm 1 presents the structure of DMIR. Reduction happens in four steps: (1) The Table Carving heuristic allocates table space T to each group (line 2-3). (2) DMIR instantiates a number of single-group MIP optimization problems (described below) and solves them in parallel to get final reduced groups (line 5-7). (3) Table entries unused by reduced groups are reclaimed (line 10) and redistributed to other groups, potentially allowing precision improvements over that of the original allocation (line 12). (4) Final groups are deduplicated using the Group Sharing heuristic (line 14).

Algorithm 2 *IGR*($\{G_i\}, T$), $1 \le i \le n$.

```
1:  // Step 1: Table Carving, see Algorithm 1
2:  // Step 2: single-group reduction.
3:  θ = 1.00 // oversub limit.
4:  {G'_i} = SORT({G_i}) // sort by descending group size.
5:  while SUM({G'_i}) > T or SUM(G'_i) > T_i, ∃i do
6:      for i = 1 to n by 1 do
7:          G'_i = REDUCESINGLEGROUP(SORT(G_i), T_i, θ)
8:          if SUM({G'_i}) ≤ T and SUM(G'_i) ≤ T_i, ∀i then
9:              return DEDUP({G'_i})
10:     θ += 0.05
11:     PRUNEGROUPIFSTUCK(G_i_not_fit)
12: // Step 3: Group Sharing.
13: return DEDUP({G'_i})
14: function REDUCESINGLEGROUP(G = {w_i}, T_i, θ)
15:     G'* = G' = {w'_i} = (1...1) // G'*: optimal G'.
16:     oversub_min = ∞
17:     for w'_1 = 1 to wmax = ⌈ w_1·T_i / SUM(G) ⌉ by ⌈ wmax / ITER ⌉ do
18:         for i = 2 to p by 1 do
19:             w'_i = ⌈ (w'_1 / w_1) · w_i ⌉
20:         oversub = MAXPORTOVERSUB(G, G')
21:         if oversub < oversub_min then
22:             UPDATE(oversub_min = oversub, G'* = G')
23:         if SUM(G') > T_i or oversub_min ≤ θ then
24:             break
25:     return G'*
```

The objective of single-group MIP optimization is to minimize the difference between an original group G and the reduced group G' such that G' uses no more than its allocated resources. Equation 1 is the single-group MIP formulation. w_i and w'_i are port weights of groups G and G'. p is the number of ports on the switch. T_G is the allocated table entries for group G'. The formulation treats G and G' as two p-dimensional vectors such that $G = (w_1, w_2, \cdots, w_p)$ and $G' = (w'_1, w'_2, \cdots, w'_p)$. This brings G and G' into the same vector space $\mathbb{R}^p$ for convenient comparison. Note that this formulation also organically integrates Group Pruning with the common path-diversity-preserving reduction behavior by allowing w'_i to be 0.

$$\begin{aligned} \textit{minimize} \quad & \sum_{i=1}^{p} \left| \frac{w_i}{\sum_{i=1}^{p} w_i} - \frac{w'_i}{\sum_{i=1}^{p} w'_i} \right| \\ s.t. \quad & \sum_{i=1}^{p} w'_i \le T_G \\ & w'_i \in \mathbb{Z}^+, \forall i \in \{1,\ldots,p\} \end{aligned} \tag{1}$$

The difference between G and G' can be measured by various metrics, such as cosine similarity, L1-norm, Kullback-Leibler divergence etc. We choose L1-norm $||G - G'||_1$ since finding the best G' is a combinatorial optimization problem and L1-norm's linearity makes it computationally tractable. This metric is different from that used by WCMP TableFitting and IGR: restricting the max port oversub of a group. While restricting max port oversub is an intuitive objective—as long as none of the ports exceeds the oversub limit, the overall precision loss is bounded—subjecting all ports to the same oversub limit does not always produce the optimal group reduction outcome (as seen in the Skewed Weights challenge).

An alternative to single-group MIP is directly solving all groups in a monolithic formulation[3]. This is nevertheless infeasible because with so many decision variables and constraints, it takes the solver more than days to find a solution.

In the third step, DMIR collects the unused entries from each group after reduction. This is in order to make further improvements on some groups by rerunning reduction with extra hardware resources. Some groups have unused entries because these entries do not significantly improve the solution quality, so we reclaim them for other groups as follows. After the initial reduction, we sort groups by their objective metric (line 9). The group with the largest difference from its original group is ranked on top. All the unused entries are allocated to the top group, and group reduction is rerun with the larger space limit. It is possible that the top group only consumes a few or none of the unused entries. If so, unused entries are reclaimed again and redistributed to the next group in line. This process repeats until either all groups are reduced again or all previously unused entries have been used.

4.3 Iterative greedy reduction

IGR has two objectives: a primary one to keep reduced group sizes under resource limit, and a secondary one to restrict the max port oversub in each group to an upper bound. Algorithm 2 describes the three-step structure of IGR. Step 1 is the same Table Carving heuristic of Algorithm 1. In step 2, IGR iteratively reduces all groups (line 5-11) until both conditions are met: (1) sum of all group sizes meets the table limit T, and (2) each group size meets its allocated space limit T_i. Each group is reduced individually by the REDUCESINGLEGROUP function. IGR is greedy because reduction could terminate in the middle of an iteration once the two conditions are met (line 8-9), leaving some groups reduced more than the rest. Some groups may fail to meet the upper bound θ on port oversub, especially when θ is very tight in the initial iterations. IGR relaxes θ in each iteration (line 10) by a constant step size. A more relaxed θ enables REDUCESINGLEGROUP to reduce the group size more. If the reduced group size stops decreasing after θ has been relaxed for *thresh*=3 consecutive times, we consider the reduction "stuck". *thresh* is a tunable parameter that allows us to balance between aggressive reduction and more port oversub tolerance. When the reduction is stuck, the Group Pruning heuristic is invoked (line 11).

REDUCESINGLEGROUP starts the search for a final group from the original group's ECMP form (line 15). It locks the relative weight ratio between ports in the group under reduction G'. In each iteration, ports in G' are sorted by their

[3] See §A.2 for a comprehensive monolithic formulation.

weights in descending order (line 7). Once the assigned weight of the first port w'_1 is determined, all the remaining port weights are determined based on the relative weight ratio (line 18-19). The number of iterations performed on each group is limited to a small constant $ITER$. REDUCESINGLEGROUP conducts the search in increments of $\lceil \frac{wmax}{ITER} \rceil$ instead of the smallest increment of one (line 17), to ensure the search process finishes within $ITER$ iterations. This effectively mitigates the long convergence time (see §5.6) that iterative-reduction-style algorithms suffer from—there are many groups (Scalability challenge) and many ports per group (Path Diversity challenge) to iterate through. REDUCESINGLEGROUP makes the best effort to meet the oversub limit and returns the smallest group it has found when it hits the group size limit (line 21- 24).

5 Evaluation

In §5.1, we describe FabricEval, a data center TE evaluation framework that we created. The DMIR and IGR algorithms are compared to WCMP TableFitting under a wide range of network conditions in §5.3-§5.5. We also present group reduction speed as another metric in §5.6. §5.7 investigates the application layer impact. §5.8 inspects the side effect of Group Pruning. §5.9 discusses the contribution of each heuristic.

5.1 FabricEval evaluation framework

FabricEval is a network-level data center TE evaluation framework. It models the entire data center TE pipeline end-to-end: from taking snapshots on network states and traffic demands to generating TE solutions, group reduction and finally implementing the solution on data plane. FabricEval employs a modular design that allows us to plug in different TE solving algorithms, group reduction algorithms, and switch hardware models with different configurations. This enables us to compute both the ideal link utilization expected by the TE solution and the actual link utilization from different group reduction algorithms, on all links. It also tracks table usage and traffic loss due to missing forwarding entries from switches.

In FabricEval, topology inputs are represented as Protobuf [49]. Traffic demands are represented as plain matrices. Inputs are passed to a TE solver module, which by default runs the same TE algorithm used in our production fabrics. The generated TE solution is organized by cluster and forwarded to a group reduction module of each cluster. Finally, reduced groups are installed on the corresponding switches. FabricEval is implemented in 15,820 lines of Python code. Since FabricEval is not a packet-level simulator, we use *ns-3* [51] to study application layer metrics.

Open source releases. The source code of FabricEval, DMIR, IGR and our implementation of WCMP TableFitting is released online [1].

5.2 Experimental setup

All configurations used in the evaluation are developed from a *baseline* production-like configuration that includes (1) a 65-cluster heterogeneous spine-free fabric, (2) a path diversity spread of 50%, and (3) a traffic matrix (TM) representing the total egress/ingress demands between ToR pairs. Due to security and privacy reasons, we cannot publish most results from production fabrics and traffic traces. However, we carefully tuned FabricEval to match production results. Using FabricEval also avoids the difficulty of running controlled experiments on live fabrics. All experiments in this paper are conducted on a Dell PowerEdge R720 server, with two 20-core Intel Xeon E5-2680 CPUs, 128 GB of memory and 1.2 TB of disk.

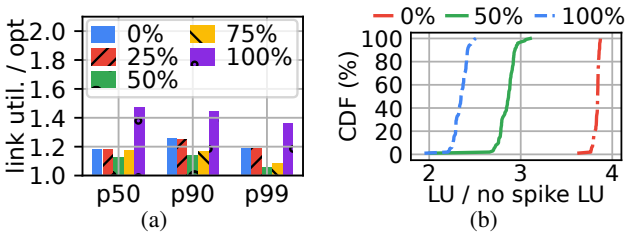


Figure 5: (a) 50% spread achieves best link utilization using production traffic trace and routing settings. (b) larger spreads are more robust against traffic spikes.

Network topology. The *baseline* fabric features 2.6K switches and 83.2K links. Each cluster in the fabric has 32 S1 switches. Each S1 has 32 ports—8 facing the S2 switches and 24 facing the host machines. The aggregation block in each cluster has 4 S2 and 4 S3 switches interconnected as a nonblocking FatTree; each has 128 ports—64 facing up and 64 facing down. The 64 up-facing ports on S3 are evenly spread to 64 peer S3 switches. The S2 switches connect to each S1 with 2 links.

Traffic matrix. The *baseline* TM captures properties found in production traffic traces. Clusters have unequal egress and ingress demands—storage clusters, for example, typically have a small ingress but a large egress because applications often read more data than they write. The demands follow an exponential distribution. The traffic volume between a cluster pair is characterized by the Gravity model [52, 55, 63, 70]. A cluster's total egress/ingress volume is distributed proportionally among peer clusters, weighed by each peer's total volume. Incremental data center expansion could leave a few clusters deployed but having zero demand. They can still be used by other clusters for transit.

5.3 Spread and path diversity

The Path Diversity challenge in §3 focuses on balancing resilience to traffic changes and optimal efficiency. This tradeoff depends on the spread. To determine an appropriate spread for our evaluation, we run production trace using realistic routing settings (i.e., TM fed into TE comprises peak demands of the past 24 hours and TE is recomputed every hour) on a large spine-free production fabric[4]. Figure 5a shows the link utilization for this realistic scenario, normalized relative to the optimal TE solution computed against the instantaneous

[4] §D.3 includes link utilization data for more production fabrics.

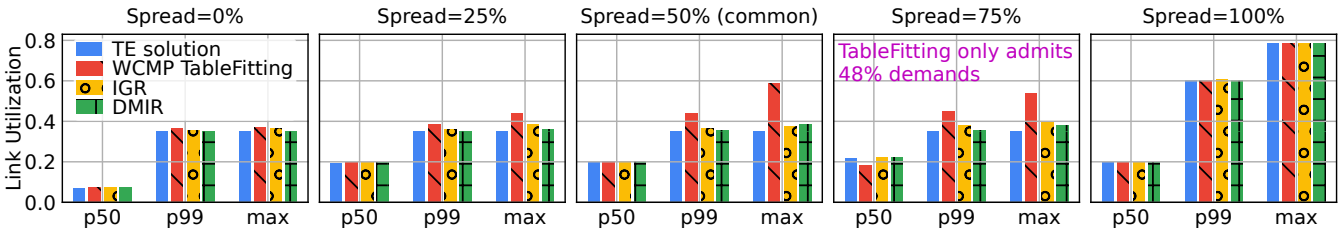


Figure 6: Median and tail link utilization with different path diversity.

Table 2: Summary of various fabrics used in FabricEval.

name	topology	# Gen1	# Gen2	# Gen3	link failure
Fabric 1	spine-free	22	22	21	none
Fabric 2	spine-free	11	11	11	none
Fabric 3	spine-free	65	0	0	none
Fabric 4	spine-free	22	22	21	1%
Fabric 5	Clos	21	21	22	1%
Fabric 6	random graph	22	22	21	none

demand (i.e., assuming traffic measurement and TE solution programming have zero delay). We find that 50% spread achieves link utilization closest to OPT (1.14× OPT) because higher spreads rely on costly indirect paths and lower spreads are fragile to TM changes. To understand how spread values affect TE solution's *robustness*, we further study more extreme cases, such as when 25% of the original demands spike by 4×. We define robustness as the link utilization post-spike divided by the pre-spike link utilization for the same spread. Figure 5b shows this ratio for each link in the same production fabric for 0% (no) spread, 50% and 100% (full) spread. As expected, 100% spread is most robust since spikes are spread throughout the network. 50% spread is significantly more robust than 0%. Overall, 50% spread proves to be a reasonable tradeoff between TE optimality and robustness.

Figure 6 compares the link utilization of IGR and DMIR with that of the TE solution (assuming no table size constraint) and WCMP TableFitting (TF) for five spread values. The results are collected on the *baseline* fabric and TM. In each graph, the y-axis shows the link utilization of the 50th (p50) and 99th (p99) percentile, and the maximum (max) link in the network. We see that with 0% spread, all algorithms come close to the link utilization of the TE solution, which is not surprising since most flows have a group size of one. As the spread increases, the link utilization difference between the TE solution and the group reduction algorithms increases. With a 50% spread, WCMP TF's link utilization is 25.4% higher at p99 and 67.1% higher at max. DMIR sees link utilization 1.6% higher at p99 and 9.1% higher at max. IGR increases over the TE solution by 4% and 7.4%, respectively. The result at 75% spread largely resembles that at 50% spread, except for a seemingly counter-intuitive link utilization improvement from WCMP TF [5].

[5] WCMP TF's result at 75% spread does not reflect a true improvement. The reason is that the groups generated by WCMP TF do not fit in the switch table. The overflow groups cause the associated traffic being dropped. This benefits the groups that do fit.

As the spread further increases, link utilization continues to deteriorate due to increased pressure on the group tables. At 100% spread, all algorithms achieve a link utilization equal to the TE solution. Since 100% spread forces demands to spread across all paths, groups become identical and after deduplication, they easily fit in the table with minimum precision loss.

5.4 Heterogeneous fabrics

In order to understand how different fabric configurations affect TE precision, we construct a list of fabrics that differ in scale, hardware and/or topology, as described by Table 2. Fabric 1 is the *baseline* fabric. Fabric 6 is a Jellyfish-like fabric with random graph topology [61] using the same hardware as fabric 1.

Figure 7 shows how results differ across these fabrics. Compared to fabric 1, all three algorithms in fabric 2 and 3 operate close to the TE solution in terms of both median and tail utilization. This confirms our analysis in §3 that Scalability and Heterogeneity both have an impact on precision loss. The tail link utilization in fabric 3 is noticeably higher than other fabrics of same scale because the same TM is applied to a fabric with lower capacity. Note that DMIR's max link utilization in fabric 3 is 10% higher than the TE solution, while WCMP TF and IGR are 12.4% and 3% higher, respectively. This is caused by a 120-second timeout on DMIR. Due to the nature of MIP problems, we use a timeout to ensure DMIR terminates within a duration comparable to the other two algorithms, which can lead to a suboptimal solution. Without this timeout, DMIR can still find an optimal solution within 10 minutes in all the experiments we run.

Looking at fabric 1 and 4, we find that the impact of failure is mostly on tail link utilization. The max link utilization under failure has increased by up to 63% over that without failure. Among the three algorithms, DMIR and IGR are more affected than WCMP TF. Compared to fabric 1, precision loss in fabric 5 is moderate—WCMP TF is 9% higher than the TE solution, yet IGR and DMIR are still effective in bringing this down to 1.3%. This aligns with our production observation that precision loss in Clos fabrics is not as grave as spine-free fabrics. Precision loss in fabric 6 is high. This is due to the asymmetry and imbalance introduced from random link assignment. WCMP TF has a max link utilization 55.1% higher than the TE solution, while IGR and DMIR lower this

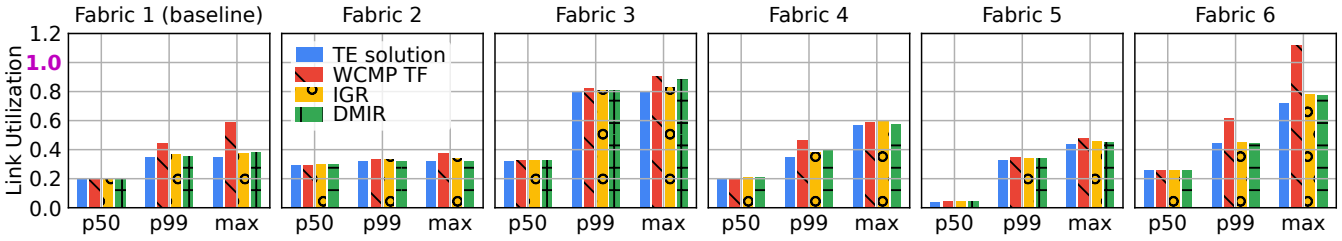


Figure 7: Link utilization distribution in various network fabrics.

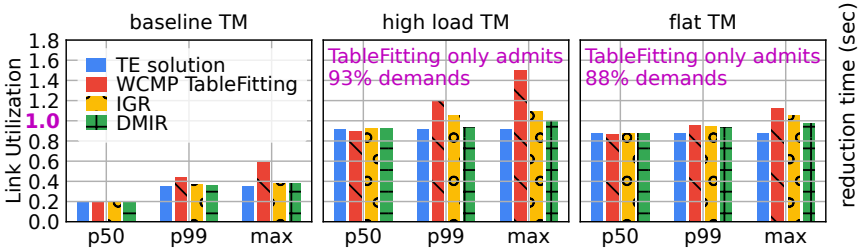


Figure 8: Link utilization distribution with different traffic patterns.

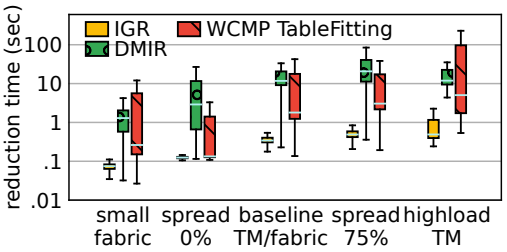


Figure 9: Comparison of group reduction speed.

gap to 8.4% and 7.5%. Generally speaking, IGR and DMIR are more effective when the topology is irregular.

5.5 Traffic pattern

The *baseline* TM has an average cluster ingress/egress volume that is about 40% of the cluster's bisection bandwidth. 15% of the latest generation clusters are configured to be in expansion mode with empty demands. For comparison, we created two more TMs: a high load TM and a flat TM. The high load TM is generated in the same way as the *baseline* TM, except that its average cluster volume is raised to 70% of the bisection bandwidth and all clusters have non-empty demands. The flat TM does not follow the Gravity model. Instead, each cluster has an equal ingress and egress demand of 7.68 Tbps (240 Gbps per ToR) that is uniformly spread across all peer clusters. There are no empty clusters in the flat TM. We run all three TMs on the *baseline* fabric with 50% spread.

With the high load and flat TMs, WCMP TF results in traffic loss due to group table overflow, as shown in Figure 8. As much as 7% and 12% of the total demands are dropped, respectively. In the high load TM scenario, even with a fraction of groups rejected, WCMP TF increases the actual link utilization over the TE solution by up to 62.1%. On the other hand, DMIR increases actual link utilization by only 9.4% over the TE solution at max, and IGR increases by 19%. This result is consistent with the increase found in the *baseline* TM scenario. The flat TM results are slightly better: WCMP TF has a max link utilization 28.4% higher than the TE solution, while it is 11.1% for DMIR and 20.2% for IGR.

All three algorithms will result in congestion loss due to oversub on some links. For example, in the high load TM scenario, WCMP TF's actual max link utilization is 1.49, almost 50% over the physical link capacity, while that of DMIR and IGR are 1.007, and 1.096. This is however expected given that the max link utilization of the TE solution is over 0.9. Any error in group reduction could lead to traffic exceeding the link capacity. While the average cluster volume is configured to 70% of the bisection bandwidth, link utilization of the TE solution ends up much higher. We attribute this to the blocking property of the spine-free topology.

5.6 Execution speed

The execution speed of the IGR and DMIR algorithms is important since they are used online. Each switch runs group reduction twice, first on the `transit` groups and then on the `src` groups. We measure the total time to complete both reductions on each switch. A number of representative configurations are selected to cover the common and corner cases. They include: (1) fabric 2 (small-scale) in §5.4, (2) 0% spread in §5.3, (3) the *baseline* fabric/TM in §5.5, (4) 75% spread in §5.3, and (5) the high load TM in §5.5.

Figure 9 plots the distribution of per-switch group reduction time as standard boxplots with max/min, first/third quartile and median. As can be seen, the reduction time on each switch spans a wide range: there is about two orders of magnitude difference. The reduction time differs significantly across switches as well as between different group types. On switches with smaller tables it take longer to fit the groups since there is a tighter table size constraint. The `transit` groups are easier to reduce than `src` groups because they are mainly ECMP-like groups. If a cluster has a zero demand, there is no `src` group to be installed on the switches in this cluster. The execution time is then solely up to `transit` reduction.

DMIR and WCMP TF have overlapping ranges. In some scenarios, such as 0% and 75% spread, DMIR is slower than WCMP TF, while in other cases, they are on par. IGR is the fastest among all three. It generally completes group reduction in under a second. Overall, IGR is 17-42× faster than DMIR and 10× faster than WCMP TF by comparing the median reduction time.

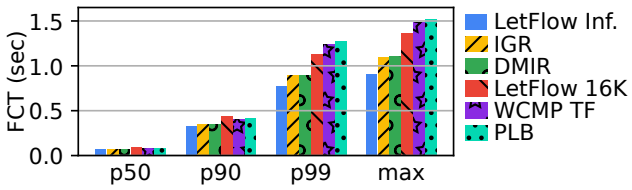


Figure 10: *ns-3* flow completion time.

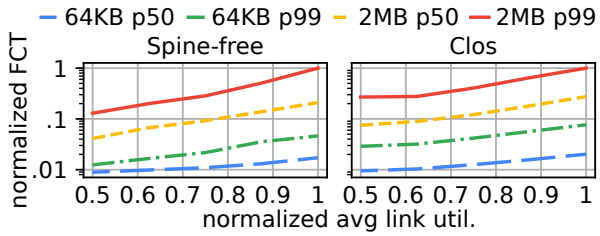


Figure 11: Production flow completion time vs. link load.

5.7 Impact on flow completion time

Thus far, we have shown our approach is effective at meeting TE's link utilization objectives. We now explore the impact our approach has on flow completion times. We extend our FCT comparison to include flowlet-based and host-based schemes. The traffic trace is constructed by sampling the flow sizes and start times from published production measurement studies [8, 35, 41, 56, 69] while ensuring that the total flow size of each cluster adds up to its demand in the TM.

In contrast to ECMP/WCMP, TE approaches such as [5, 37, 38, 65] balance load at the flowlet level [34]. While this finer grain load balancing could improve FCT, most large-scale DCN operators adopt some form of ECMP/WCMP [2, 22, 47, 58, 60] instead of a flowlet-based approach. We attribute this to the universal vendor support for ECMP/WCMP and the simplicity of deployment, e.g., no parameter tuning is required. We focus on LetFlow [65], a recent flowlet proposal that provides significant FCT performance gains.

We implemented IGR, DMIR, WCMP TF and LetFlow in *ns-3*, and use the *baseline* fabric/TM. As per [65], each switch uses a flowlet table that maps a 5-tuple hash to an egress port id. ECMP groups are still required to track ports that can reach the destination for each flow aggregate. When the inactive interval in a flow exceeds the pre-configured timeout, the flowlet entry expires and a new egress is randomly selected from the corresponding group. The best timeout value is picked by sweeping through a range of timeout values. We assume the flowlet table is subject to hardware resource constraints similar to WCMP. A 32K-entry memory is split in half—16K for groups and 16K for flowlets[6].

Figure 10 shows that IGR and DMIR reduce the p99 FCT of WCMP TF by 28.1% and 27.8%, respectively. LetFlow with an unlimited table size outperforms all other candidates, shortening the p99 FCT of WCMP TF by 37.8%. With a 16K table size constraint, LetFlow underperforms IGR and DMIR but remains better than WCMP TF by 8.7% in p99 FCT. To put this improvement into perspective, we run flows of two sizes in two production fabrics, then we steadily increase average link utilization of the fabric, the p50/p99 FCTs at every link load are collected. Figure 11 illustrates how FCT normalized to each flow's min RTT correlates with link utilization (more data in §D.4). On the more loaded side of the spectrum, a 10% reduction in link utilization can translate to a 20-40% reduction in p99 FCT.

Another class of designs that has reported significant FCT gains is host-based TE (e.g., PLB [53]). [53] shows that PLB can fully correct load imbalance caused by ECMP between two links with 1:2 capacity difference. However, with capacity difference of 1:100, PLB can only correct it to 1:3.5, and p99 FCT is 2.3× higher than that of a load balanced scenario. Figure 10 indicates that PLB alone achieves FCT on par with WCMP TF and we hold the same view as [53] that in-network WCMP and host-based PLB are complementary in performance benefits.

5.8 Group Pruning's impact on path diversity

A potential side effect of Group Pruning is that heavily pruned groups might lose a lot of paths. This reduced path diversity could impact the network's resilience to link failures. To understand the impact of pruning, we record the paths/ports used by all groups in the *baseline* fabric/TM experiment. Figure 12 shows that IGR prunes less than 0.01% of the total groups, while DMIR prunes 0.07% of them. DMIR's pruning decision is more aggressive than IGR: 48 groups (out of 1350624) are pruned down to a single port. In contrast, groups pruned by IGR only lose 1-3 ports. More results can be found in §D.5.

Figure 12 also shows the fraction of traffic impacted by Group Pruning. Impacted means that the traffic is steered away from pruned paths, but is not dropped. Despite DMIR's more aggressive pruning, the total traffic impacted (area under curve) is lower than IGR. We attribute this to DMIR's tradeoff between path diversity and lower traffic impact.

Overall, the small number of aggressively pruned groups and tiny fraction of traffic impacted suggests that the negative impact of Group Pruning is minor.

5.9 Ablation study

Now, we evaluate how much each of the heuristics in §4 contributes to the overall precision improvements. We use the *baseline* fabric/TM and 50% spread. First, the contribution of Group Sharing is evaluated by disabling this heuristic in both DMIR and IGR. Figure 13 illustrates the significant impact of Group Sharing on IGR: 5× decrease in max link utilization. DMIR is improved by 1.7×. Additionally, Group Sharing reduces the O(N^2) `transit` group entries (e.g., 162K) to no more than 64 (see §D.2 for details). Next, the contribution of Table Carving is demonstrated in Figure 14. Instead of standard Table Carving, the algorithms are modified to carve the table in constant sizes: each group receives an equal share of the table space. IGR and DMIR with standard Table Carving perform

[6] Cisco ACI fabric with Nexus 9300 switches supports flowlets under the feature named Dynamic Load Balancing [13]. A total of 4096×8-way ECMP entries (see Table 7 in [14]) are available on chip.

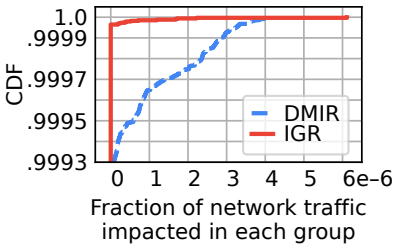


Figure 12: Impact of reduced path diversity on network traffic.

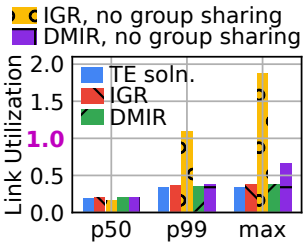


Figure 13: Group Sharing's contribution to TE precision.

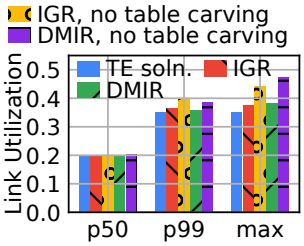


Figure 14: Table Carving's contribution to TE precision.

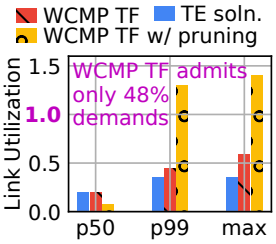


Figure 15: Impact of Group Pruning.

1.2× and 1.24× better than the constant version. Finally, we look at the contribution of Group Pruning in Figure 15. 75% spread is used to reflect how Group Pruning avoids traffic loss. The pruning logic is hard to completely disable in both algorithms, so instead, we add Group Pruning to WCMP TF. With Group Pruning, WCMP TF can avoid group installation failures and the consequent traffic loss entirely, at the cost of a 2.4× increase in link utilization. However, this increase is specific to certain scenarios and can be offset by other improvements.

5.10 Recommendation

While both our algorithms perform well, DMIR generally has lower precision loss than IGR. Though its advantage is usually small, some extreme scenarios can benefit from DMIR. For example, in the high load TM scenario (§5.5), DMIR is the only option that avoids link utilization above 1.0, except for a slight violation on the worst link. In contrast, IGR is always faster than DMIR, so it responds better to traffic load changes. Because of that, we recommend generally using IGR, except for difficult scenarios where low precision loss matters.

6 Related work

Data center TE. Many data center network designs [2–4, 22] leverage flow-level ECMP [30] to achieve failure resilience and efficient bandwidth utilization. However, ECMP is known to perform poorly with asymmetry and heterogeneity. WCMP [73] improves over ECMP. Niagara [36] distributes traffic using flow rules instead of group rules. DASH [32] splits traffic via comparison-based hash space partition. It avoids WCMP-style entry replication but requires arithmetic operation and P4 [11] support. Our work assumes fixed-function switches but also applies to programmable switches.

Flare [34], CONGA [5], HULA [38], Clove [37], LetFlow [65], and Contra [31] leverage the elasticity of flowlets to balance traffic load across paths. In networks with different path latencies [12, 17, 42, 45, 50, 66, 67], one has to carefully select the flowlet detection timeout to avoid reordering. Flowlets are also susceptible to traffic characteristics, e.g., [43, 62] find that RDMA traffic [6, 19, 23, 27, 39, 47, 74] exhibits weak flowlet pattern. As described in LetFlow [65], flowlet switching requires switch support [13], which is not as widely available as ECMP. There are also works like DRILL [20] and RPS [16] that operate at packet-level.

Host-based TE approaches like Presto [26], Hermes [68], MPTCP [54], FlowBender [33], RePaC [71] and PLB [53] assume either no in-network support or merely ECMP. PLB [53] shows that our work complements host-based TE.

Spine-free data center. Dragonfly [40], Dragonfly+ [59], Slim Fly [10], Slingshot [15], Aquila [21] have studied direct-connect topology networks. However, they primarily focus on HPC networks. Meanwhile, most deployed DCNs [2, 22] employ a Clos topology with a spine. Harsh et al. [25] discusses ECMP-based TE in spine-free data centers. Sirius [7] and RotorNet [46] are spine-free but spread traffic using extended VLB [72]. Jupiter [52] explores advanced TE in spine-free DCNs. We believe our work is among the first to improve TE precision loss in spine-free DCNs.

Wide area network TE. Wide area network TE faces similar switch hardware constraints. SWAN [28] allocates traffic demands to a set of tunnels, which are limited by the number of supported switch rules. B4 TE [29] generates groups that consume 14× the switch table space. It has to decouple traffic splitting rules across two stages in its Clos fabric to reduce per-switch table consumption. Meta's Express Backbone [48] and Edge Fabric [57] run centralized TE on BGP and ECMP. We found no available data on their TE precision loss.

7 Conclusion

Precision loss is an inherent problem when implementing traffic engineering with limited switch hardware resources and the shift towards spine-free topologies exacerbates the problem. We introduce two group reduction algorithms that offer different tradeoffs in terms of precision and execution time. Both algorithms use heuristics to address challenges, such as the hardware-agnostic nature of TE algorithms, switch heterogeneity, and path diversity. Our evaluation shows they achieve significant improvements in various network scenarios compared to the current solution.

Acknowledgments. We thank Adithya Abraham Philip, Zico Kolter, Akshitha Sriraman, Miguel Ferreira, Wei Bai, Brian Chang, Weiyang Wang, Min Yee Teh, Nandita Dukkipati and the anonymous NSDI reviewers for their feedback. We also thank engineers from the Google NetInfra team for their support. This research is sponsored by the U.S. Army Contracting Command under award number W911QX20D0008.

References

[1] FabricEval TE Evaluation Framework, 2024. https://github.com/shuoshuc/FabricEval.

[2] Anubhavnidhi Abhashkumar, Kausik Subramanian, Alexey Andreyev, Hyojeong Kim, Nanda Kishore Salem, Jingyi Yang, Petr Lapukhov, Aditya Akella, and Hongyi Zeng. Running bgp in data centers at scale. In *NSDI*, pages 65–81, 2021.

[3] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review*, 38(4):63–74, 2008.

[4] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. *Proceedings of NSDI 2010: 7th USENIX Symposium on Networked Systems Design and Implementation*, pages 281–295, 2010.

[5] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al. Conga: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 503–514, 2014.

[6] Wei Bai, Shanim Sainul Abdeen, Ankit Agrawal, Krishan Kumar Attre, Paramvir Bahl, Ameya Bhagat, Gowri Bhaskara, Tanya Brokhman, Lei Cao, Ahmad Cheema, Rebecca Chow, Jeff Cohen, Mahmoud Elhaddad, Vivek Ette, Igal Figlin, Daniel Firestone, Mathew George, Ilya German, Lakhmeet Ghai, Eric Green, Albert Greenberg, Manish Gupta, Randy Haagens, Matthew Hendel, Ridwan Howlader, Neetha John, Julia Johnstone, Tom Jolly, Greg Kramer, David Kruse, Ankit Kumar, Erica Lan, Ivan Lee, Avi Levy, Marina Lipshteyn, Xin Liu, Chen Liu, Guohan Lu, Yuemin Lu, Xiakun Lu, Vadim Makhervaks, Ulad Malashanka, David A. Maltz, Ilias Marinos, Rohan Mehta, Sharda Murthi, Anup Namdhari, Aaron Ogus, Jitendra Padhye, Madhav Pandya, Douglas Phillips, Adrian Power, Suraj Puri, Shachar Raindel, Jordan Rhee, Anthony Russo, Maneesh Sah, Ali Sheriff, Chris Sparacino, Ashutosh Srivastava, Weixiang Sun, Nick Swanson, Fuhou Tian, Lukasz Tomczyk, Vamsi Vadlamuri, Alec Wolman, Ying Xie, Joyce Yom, Lihua Yuan, Yanzhao Zhang, and Brian Zill. Empowering azure storage with RDMA. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 49–67, Boston, MA, April 2023. USENIX Association.

[7] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. Sirius: A flat datacenter network with nanosecond optical switching. *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 782–797, 2020.

[8] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, page 267–280, New York, NY, USA, 2010. Association for Computing Machinery.

[9] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Microte: Fine grained traffic engineering for data centers. In *Proceedings of the seventh conference on emerging networking experiments and technologies*, pages 1–12, 2011.

[10] Maciej Besta and Torsten Hoefler. Slim fly: A cost effective low-diameter network topology. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 348–359, 2014.

[11] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, jul 2014.

[12] Shawn Shuoshuo Chen, Weiyang Wang, Christopher Canel, Srinivasan Seshan, Alex C. Snoeren, and Peter Steenkiste. Time-division tcp for reconfigurable data center networks. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 19–35, New York, NY, USA, 2022. Association for Computing Machinery.

[13] Cisco. Cisco application centric infrastructure fundamentals, 2015. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/aci-fundamentals/b_ACI-Fundamentals/m_fundamentals.html#concept_F280C079790A451ABA76BC5C6427D746.

[14] Cisco. Cisco nexus 9000 series nx-os verified scalability guide, release 9.3(12), 2023. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/93x/scalability/guide-9312/cisco-nexus-9000-series-nx-os-verified-scalability-guide-9312.html.

[15] Daniele De Sensi, Salvatore Di Girolamo, Kim H. McMahon, Duncan Roweth, and Torsten Hoefler. An in-depth analysis of the slingshot interconnect. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

[16] Advait Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. On the impact of packet spraying in data center networks. In *2013 Proceedings IEEE INFOCOM*, pages 2130–2138, 2013.

[17] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, page 339–350, New York, NY, USA, 2010. Association for Computing Machinery.

[18] Andrew D. Ferguson, Steve Gribble, Chi-Yao Hong, Charles Killian, Waqar Mohsin, Henrik Muehe, Joon Ong, Leon Poutievski, Arjun Singh, Lorenzo Vicisano, Richard Alimi, Shawn Shuoshuo Chen, Mike Conley, Subhasree Mandal, Karthik Nagaraj, Kondapa Naidu Bollineni, Amr Sabaa, Shidong Zhang, Min Zhu, and Amin Vahdat. Orion: Google's software-defined networking control plane. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, April 2021.

[19] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, Fei Feng, Yan Zhuang, Fan Liu, Pan Liu, Xingkui Liu, Zhongjie Wu, Junping Wu, Zheng Cao, Chen Tian, Jinbo Wu, Jiaji Zhu, Haiyong Wang, Dennis Cai, and Jiesheng Wu. When cloud storage meets RDMA. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 519–533. USENIX Association, April 2021.

[20] Soudeh Ghorbani, Zibin Yang, P. Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. Drill: Micro load balancing for low-latency data center networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 225–238, New York, NY, USA, 2017. Association for Computing Machinery.

[21] Dan Gibson, Hema Hariharan, Eric Lance, Moray McLaren, Behnam Montazeri, Arjun Singh, Stephen Wang, Hassan M G Wassel, Zhehua Wu, Sunghwan Yoo, Raghuraman Balasubramanian, Prashant Chandra, Michael Cutforth, Peter Cuy, David Decotigny, Rakesh Gautam, Alex Iriza, Milo M K Martin, Rick Roy, Zuowei Shen, Ming Tan, Ye Tang, Monica Wong-Chan, Joe Zbiciak, and Amin Vahdat. Aquila: A unified, low-latency fabric for datacenter networks. pages 1249–1266. USENIX Association, 4 2022.

[22] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 51–62, 2009.

[23] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. Rdma over commodity ethernet at scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 202–215, New York, NY, USA, 2016. Association for Computing Machinery.

[24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[25] Vipul Harsh, Sangeetha Abdu Jyothi, and P. Brighten Godfrey. Spineless data centers. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, HotNets '20, page 67–73, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Keqiang He, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella. Presto: Edge-based load balancing for fast datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 465–478, New York, NY, USA, 2015. Association for Computing Machinery.

[27] Zhiqiang He, Dongyang Wang, Binzhang Fu, Kun Tan, Bei Hua, Zhi-Li Zhang, and Kai Zheng. Masq: Rdma for virtual private cloud. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[28] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. *SIGCOMM Comput. Commun. Rev.*, 43(4):15–26, aug 2013.

[29] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. B4 and

after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 74–87, New York, NY, USA, 2018. Association for Computing Machinery.

[30] C. Hopps. Rfc2992: Analysis of an equal-cost multi-path algorithm, 2000.

[31] Kuo-Feng Hsu, Ryan Beckett, Ang Chen, Jennifer Rexford, and David Walker. Contra: A programmable system for performance-aware routing. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 701–721, Santa Clara, CA, February 2020. USENIX Association.

[32] Kuo-Feng Hsu, Praveen Tammana, Ryan Beckett, Ang Chen, Jennifer Rexford, and David Walker. Adaptive weighted traffic splitting in programmable data planes. In *Proceedings of the Symposium on SDN Research*, pages 103–109, 2020.

[33] Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, page 149–160, New York, NY, USA, 2014. Association for Computing Machinery.

[34] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. Dynamic load balancing without packet reordering. *SIGCOMM Comput. Commun. Rev.*, 37(2):51–62, mar 2007.

[35] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The nature of data center traffic: Measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, page 202–208, New York, NY, USA, 2009. Association for Computing Machinery.

[36] Nanxi Kang, Monia Ghobadi, John Reumann, Alexander Shraer, and Jennifer Rexford. Efficient traffic splitting on commodity switches. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pages 1–13, 2015.

[37] Naga Katta, Aditi Ghag, Mukesh Hira, Isaac Keslassy, Aran Bergman, Changhoon Kim, and Jennifer Rexford. Clove: Congestion-aware load balancing at the virtual edge. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, page 323–335, New York, NY, USA, 2017. Association for Computing Machinery.

[38] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*, pages 1–12, 2016.

[39] Daehyeok Kim, Tianlong Yu, Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Shachar Raindel, Chuanxiong Guo, Vyas Sekar, and Srinivasan Seshan. FreeFlow: Software-based virtual RDMA networking for containerized clouds. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 113–126, Boston, MA, February 2019. USENIX Association.

[40] John Kim, William J. Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. *Proceedings - International Symposium on Computer Architecture*, pages 77–88, 2008.

[41] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. Hpcc: High precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 44–58, New York, NY, USA, 2019. Association for Computing Machinery.

[42] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. Scheduling techniques for hybrid circuit/packet networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, New York, NY, USA, 2015. Association for Computing Machinery.

[43] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. Multi-Path transport for RDMA in datacenters. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 357–371, Renton, WA, April 2018. USENIX Association.

[44] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, mar 2008.

[45] William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C. Snoeren, and George Porter. Expanding across time to deliver bandwidth efficiency and low latency. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 1–18, Santa Clara, CA, February 2020. USENIX Association.

[46] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 267–280, New York, NY, USA, 2017. Association for Computing Machinery.

[47] Rui Miao, Lingjun Zhu, Shu Ma, Kun Qian, Shujun Zhuang, Bo Li, Shuguang Cheng, Jiaqi Gao, Yan Zhuang, Pengcheng Zhang, Rong Liu, Chao Shi, Binzhang Fu, Jiaji Zhu, Jiesheng Wu, Dennis Cai, and Hongqiang Harry Liu. From luna to solar: The evolutions of the compute-to-storage networks in alibaba cloud. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 753–766, New York, NY, USA, 2022. Association for Computing Machinery.

[48] Henry Kwok Mikel Jimenez Fernandez. Building express backbone: Facebook's new long-haul network, 2017.

[49] Jeffrey C. Mogul, Drago Goricanec, Martin Pool, Anees Shaikh, Douglas Turk, Bikash Koley, and Xiaoxue Zhao. Experiences with modeling network topologies at multiple levels of abstraction. In *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, NSDI'20, page 403–418, USA, 2020. USENIX Association.

[50] Matthew K. Mukerjee, Christopher Canel, Weiyang Wang, Daehyeok Kim, Srinivasan Seshan, and Alex C. Snoeren. Adapting TCP for reconfigurable datacenter networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 651–666, Santa Clara, CA, February 2020. USENIX Association.

[51] NS-3 Consortium. ns-3 network simulator, 2023. https://www.nsnam.org/.

[52] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. Jupiter evolving: Transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 66–85, New York, NY, USA, 2022. Association for Computing Machinery.

[53] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani. Plb: Congestion signals are simple and effective for network load balancing. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 207–218, New York, NY, USA, 2022. Association for Computing Machinery.

[54] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving datacenter performance and robustness with multipath tcp. *SIGCOMM Comput. Commun. Rev.*, 41(4):266–277, aug 2011.

[55] Matthew Roughan. Simplifying the synthesis of internet traffic matrices. *SIGCOMM Comput. Commun. Rev.*, 35:93–96, 10 2005.

[56] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. Inside the social network's (datacenter) network. *SIGCOMM Comput. Commun. Rev.*, 45(4):123–137, aug 2015.

[57] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 418–431, New York, NY, USA, 2017. Association for Computing Machinery.

[58] Hua Shao, Xiaoliang Wang, Yuanwei Lu, Yanbo Yu, Shengli Zheng, and Youjian Zhao. Accessing cloud with disaggregated Software-Defined router. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 1–14. USENIX Association, April 2021.

[59] Alexander Shpiner, Zachy Haramaty, Saar Eliad, Vladimir Zdornov, Barak Gafni, and Eitan Zahavi. Dragonfly+: Low cost topology for scaling datacenters. In *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, pages 1–8, 2017.

[60] 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. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 183–197, New York, NY, USA, 2015. Association for Computing Machinery.

[61] Ankit Singla, Chi Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. *Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation*, pages 225–238, 2012.

[62] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. Network load balancing with in-network reordering support for rdma. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 816–831, New York, NY, USA, 2023. Association for Computing Machinery.

[63] Paul Tune and Matthew Roughan. Spatiotemporal traffic matrix synthesis. *SIGCOMM Comput. Commun. Rev.*, 45:579–592, 8 2015.

[64] Ryohei Urata, Hong Liu, Kevin Yasumura, Erji Mao, Jill Berger, Xiang Zhou, Cedric Lam, Roy Bannon, Darren Hutchinson, Daniel Nelson, Leon Poutievski, Arjun Singh, Joon Ong, and Amin Vahdat. Mission apollo: Landing optical circuit switching at datacenter scale, 2022.

[65] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. Let it flow: Resilient asymmetric load balancing with flowlet switching. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 407–420, Boston, MA, March 2017. USENIX Association.

[66] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. C-through: Part-time optics in data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, page 327–338, New York, NY, USA, 2010. Association for Computing Machinery.

[67] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 739–767, Boston, MA, April 2023. USENIX Association.

[68] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. Resilient datacenter load balancing in the wild. In *Proceedings of the 2017 SIGCOMM, Los Angeles, CA, USA*, August 2017.

[69] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. High-resolution measurement of data center microbursts. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, page 78–85, New York, NY, USA, 2017. Association for Computing Machinery.

[70] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. *SIGMETRICS Perform. Eval. Rev.*, 31:206–217, 6 2003.

[71] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. Hashing linearity enables relative path control in data centers. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 855–862. USENIX Association, July 2021.

[72] Rui Zhang-Shen and Nick McKeown. Designing a predictable internet backbone with valiant load-balancing. In *Proceedings of the 13th International Conference on Quality of Service*, IWQoS'05, page 178–192, Berlin, Heidelberg, 2005. Springer-Verlag.

[73] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. Wcmp: Weighted cost multipathing for improved fairness in data centers. *Proceedings of the Ninth European Conference on Computer Systems*, 2014.

[74] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 523–536, New York, NY, USA, 2015. Association for Computing Machinery.

Appendix

A Formulations

A.1 Data center TE formulation

The TE system solves a multi-commodity flow (MCF) problem on a network $G=(V,E)$ with a set of vertices/switches V and edges/links E. Assume that there are k commodities between sources and destinations, denoted as $(s_i,t_i,d_i), 0 \le i \le k$. Each commodity has demand d_i between source s_i and destination t_i. The objective of the TE system is to balance load (i.e., link utilization) across all links. This objective can be linearized as minimizing the maximum link utilization. Below is the path-based MCF formulation.

$$
\begin{aligned}
\textit{minimize} \quad & u_{max} && \text{(2a)}\\
s.t. \quad & u(x,y) \le u_{max}, \forall (x,y) \in E && \text{(2b)}\\
& u(x,y) = \frac{\sum_{i=1}^{k} \sum_{p \in \mathbb{P}_i^{(x,y)}} f_{ip}}{c(x,y)}, \ \forall (x,y) \in E && \text{(2c)}\\
& \sum_{i=1}^{k} \sum_{p \in \mathbb{P}_i^{(x,y)}} f_{ip} \le c(x,y), \ \forall (x,y) \in E && \text{(2d)}\\
& \sum_{p \in \mathbb{P}_i} f_{ip} = d_i, \ \forall i \in \{1,...,k\} && \text{(2e)}\\
& f_{ip} \le d_i \cdot \frac{c_p}{S \cdot \sum_{p \in \mathbb{P}_i} c_p}, \ \forall p \in \mathbb{P}_i && \text{(2f)}\\
& f_{ip} >= 0, \ \forall p,i && \text{(2g)}
\end{aligned}
$$

u_{max} is the maximum link utilization of the network, $u(x,y)$ is the link utilization of link $(x,y) \in E$. $c(x,y)$ represents the capacity of link (x,y). f_{ip} is a portion of commodity i (i.e., flow) assigned on path p, where p includes multiple links between the source and destination. $\mathbb{P}_i$ is the set of all paths between s_i and t_i of commodity i. $\mathbb{P}_i^{(x,y)}$ is the subset of $\mathbb{P}_i$ where all paths contain link (x,y). c_p represents the capacity of path p, which is the bottleneck capacity across all links in this path.

Equation 2d is the (optional) link capacity constraint. If a commodity has a high demand such that no valid assignment exists to meet this constraint, the problem is considered infeasible. However, if we are willing to overload a link, this constraint can be removed. Equation 2e is the flow conservation constraint. It indicates that all demands must be fully served, the network cannot hold or drop any demand. Equation 2f is the path diversity constraint mentioned in §3. It enforces that no more than a certain fraction of demand d_i is assigned to flow f_{ip}. The fraction is proportional to the total capacity of the paths used to serve d_i, which is controlled by a spread parameter $S \in (0,1]$. When S is close to 0, f_{ip} becomes essentially unbounded. When S approaches 1, flows for commodity i are forced to use all available paths. Note that S cannot be set to 0, but a small enough value is sufficient to completely remove this constraint, as f_{ip} is upper bounded by Equation 2e instead.

A.2 Multi-group monolithic reduction

To extend the discussion in §4.2, we now consider another group reduction formulation. Unlike Equation 1, this formulation attempts to combine all the single-group reduction problems into one monolithic formulation that can be directly solved by calling the MIP solver. Since this monolithic formulation needs a single optimization objective, we construct a weighted sum of L1-norm based on each group's individual L1-norm, where the weights are the total traffic volume carried by each group. The intuition is that groups with more traffic contribute more to the overall precision loss metric. Therefore, the MIP solver should spend more effort in optimizing those groups. Equation 3 shows the comprehensive formulation.

$$
\begin{aligned}
\textit{minimize} \quad & \sum_{i=1}^{n} \Big(\sum_{j=1}^{p} w_{ij} \Big) \cdot \sum_{j=1}^{p} \left| \frac{w_{ij}}{\sum_{j=1}^{p} w_{ij}} - \frac{w'_{ij}}{\sum_{j=1}^{p} w'_{ij}} \right| \\
s.t. \quad & \sum_{i=1}^{n} \sum_{j=1}^{p} w'_{ij} \le T \\
& w'_{ij} \in \mathbb{Z}^{+}, \forall i \in \{1,...,n\}, j \in \{1,...,p\}
\end{aligned}
\qquad \text{(3)}
$$

$G_i = (w_{i1}, w_{i2}, ..., w_{ip})$ is the original group i, w_{ij} is the original weight on port j of group i. Similarly, w'_{ij} is the reduced weight on port j of group i. T is the total available table space on the target switch. This formulation only enforces the sum of group sizes across all reduced groups to meet the table space limit, there is no per-group space limit and the Table Carving heuristic is not invoked.

Multi-group monolithic formulation scales poorly as mentioned in §4.2. For a large-scale production DCN, we fail to find a solution to the monolithic formulation after running the latest version of Gurobi solver for days. On the other hand, with a one-hour timeout set on Gurobi, the resulting solution is considerably worse compared to that obtained from single-group MIP.

B Example snippet of `TEIntent`

The TE system in large-scale DCNs usually divides the TE solution into batches, where each batch contains instructions of traffic split for all demands in a cluster. In FabricEval, such a batch is referred to as a `TEIntent`. The instruction of traffic split for one demand in a cluster is referred to as a `PrefixIntent`. A `TEIntent` can contain multiple `PrefixIntents`. Each `PrefixIntent` needs to specify its type (`src` or `transit`). This information is required to later translate the `PrefixIntent` to groups. The `PrefixIntent` also specifies the name of the destination aggregation block it wants to reach. Here the name of the destination cluster is not used instead because the `PrefixIntent` indicates the ports to use in the aggregation block. It should be consistent to use names of aggregation blocks for source, transit, and destination. Finally, the `PrefixIntent` lists all the northbound ports on S3 used to forward the demand, and the amount of traffic to be placed on each port in Mbps. The Protobuf format of `TEIntent` and `PrefixIntent` is as follows.

```
message PrefixIntent {
  enum PrefixType {
    UNKNOWN = 0;
    SRC = 1;
    TRANSIT = 2;
  }

  message NexthopEntry {
    // Name of next-hop port.
    string nexthop_port = 1;
    // Traffic volume in Mbps.
    double weight = 2;
  }

  string dst_prefix = 1;
  uint32 mask = 2;
  string dst_name = 3;
  PrefixType type = 4;
  repeated NexthopEntry nexthop_entries =5;
}

message TEIntent {
  // Name of cluster.
  string target_cluster = 1;
  // A list of PrefixIntents.
  repeated PrefixIntent prefix_intents = 2;
}
```

We also capture an actual `TEIntent` forwarded to one of the clusters in an experiment in §5.4. Due to the large Protobuf size, only a small snippet is demonstrated below.

```
te_intents {
  target_cluster: "toy3-c1"
  prefix_intents {
    dst_name: "toy3-c2-ab1"
    type: SRC
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i1-p1"
      weight: 25.5
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i2-p1"
      weight: 25.5
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i3-p1"
      weight: 25.5
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i4-p1"
      weight: 25.5
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i1-p3"
      weight: 0.765625
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i2-p3"
      weight: 0.765625
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i3-p3"
      weight: 0.765625
    }
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i4-p3"
      weight: 0.765625
    }
  }
  prefix_intents {
    dst_name: "toy3-c3-ab1"
    type: SRC
    ...
  }
  prefix_intents {
    dst_name: "toy3-c2-ab1"
    type: TRANSIT
    nexthop_entries {
      nexthop_port: "toy3-c1-ab1-s3i1-p9"
      weight: 3095.288
    }
    ...
  }
}
```

The network entities follow a naming scheme of [network name]-[cluster name]-[aggregation block name]-[switch stage][switch name]-[port name]. Each entity is indexed in the topology, hence its name is always the entity type + index. For example, in next-hop port "toy3-c1-ab1-s3i1-p1", "toy3" is the network name, "c1" identifies cluster 1, "ab1" is the only aggregation block in cluster 1, "s3i1" refers to the first switch in stage 3, and "p1" is the first port on this switch.

C Time complexity of IGR

Step 1 of IGR (Table Carving, line 2-3 of Algorithm 1) computes $len(G_i)$, $\text{SUM}(G_i)$ and $\sum_i \text{SUM}(G_i)$. This requires iterating over all p port weights in all n groups, so it takes O(pn). Function REDUCESINGLEGROUP runs at most $ITER$ iterations in the outer loop. The inner loop iterates over each port. So REDUCESINGLEGROUP takes O(p·$ITER$). Each iteration of oversub relaxation (line 5-11) takes O(pn·$ITER$). In the worst case, oversub relaxation stops when all groups are pruned until one port is left (zero port left means traffic loss). Here we make a reasonable assumption that a set of singleton groups must fit. This means the while loop (line 5) runs at most O(pn) times, hence making step 2 O($p^2n^2 \cdot ITER$). Step 3 Group Sharing requires scanning all groups and their ports

Table 3: Group table space usage on sampled S1/S2/S3 switches by different group types.

stage	group type	original usage	post-reduction
S1	`src`	8 entries	8 entries
S2	`src`	1746624 entries	64 entries
S3	`src`	2241913 entries	18285 entries
S1	`transit`	0 entry	0 entry
S2	`transit`	162112 entries	64 entries
S3	`transit`	792738 entries	60 entries

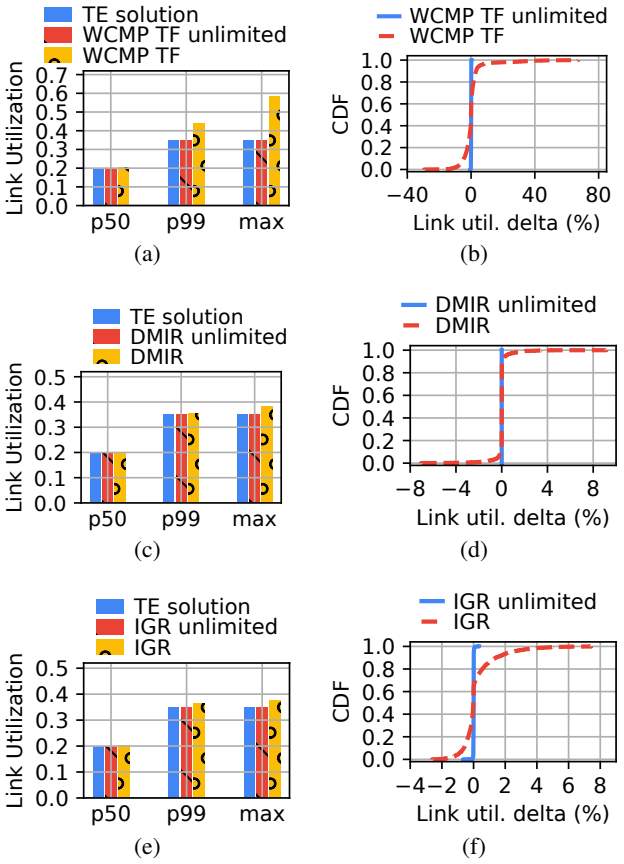


Figure 16: (a, c, e) Median and tail link utilization if running original group reduction algorithm vs. running it with infinite group table size, so that there is only precision loss caused by rounding weights. (b, d, f) Per-link utilization delta over the TE solution.

to mark duplicates, so it is O(pn). Overall, the time complexity of IGR is $O(pn+p^2n^2 \cdot ITER+pn)$, which is polynomial.

D Extra evaluation

D.1 Quantization error

Rounding fractional weights in groups to integers introduces a minor yet acceptable quantization error. We choose to express weights in Mbps so that the quantization error of rounding is at most $\pm$1Mbps. Weights in Kbps or bps can further reduce quantization error, but it comes at a cost where the Protobuf is bloated with larger integers. Figure 16 shows that the quantization error of Mbps granularity is sufficiently small.

We repeat the *baseline* fabric/TM experiment using three group reduction algorithms, WCMP TableFitting, DMIR and IGR, with unlimited group table space on each switch. In other words, the original fractional weights in each group only need to be rounded to integers, but will not be reduced in size—as there is infinite group table space to store any arbitrary number of entries. The precision loss over the TE solution from each group reduction algorithm with unlimited table space reflects the pure impact of rounding.

For instance, WCMP TableFitting is significantly improved with unlimited table space. The TE implementation is almost identical to the TE solution. Per-link utilization delta over the TE solution is also reduced to close to zero, which means there is very minimum oversub and the implementation overall is accurate. Similarly, for DMIR and IGR, the implementation is also improved, although the improvement is not as large as WCMP TableFitting. The TE precision loss (i.e., quantization error) achieved by all three group reduction algorithms is generally under 0.01%.

D.2 Table usage

Table 3 lists the group table usage statistics from sampled S1/S2/S3 switches in the *baseline* fabric from §5.4. The original usage column indicates the number of entries required if the table space is unlimited (pre-group-reduction). The post-reduction column indicates the actual number of entries consumed found on the switch (after running group reduction). We can see that table usage on S1 switches is low and group reduction is irrelevant here. This is because the groups are always ECMP on S1 switches and a shared group is used for serving all demands. As mentioned in §4, the Group Sharing heuristic significantly reduces the number of entries required by `transit` groups to no more than 64, on both S2 and S3 switches. The `src` groups on S3 switches see a table usage reduction from 2241913 to 18285—a 122× reduction!

D.3 Spread in TE solution

Figure 17 demonstrates the link utilization relative to optimal link utilization under shifting traffic demands for 7 different production spine-free fabrics. As can be seen, 50% spread generally yields the best link utilization comparing to other spreads. In some fabrics, e.g., Fabric A and Fabric D, 25% or 75% spread can achieve a link utilization similar to 50% spread due to the traffic demands on these fabrics are either more stable or more uncertain.

Figure 18 plots the CDF of per-link utilization delta under different spreads. Some links are severely under-utilized and some are severely over-utilized by up to 65%. However, the relative percentage change in link utilization does not always translate into high absolute link utilization because the traffic volume carried on such links can be low. We are more interested in the absolute link utilization than per-link delta.

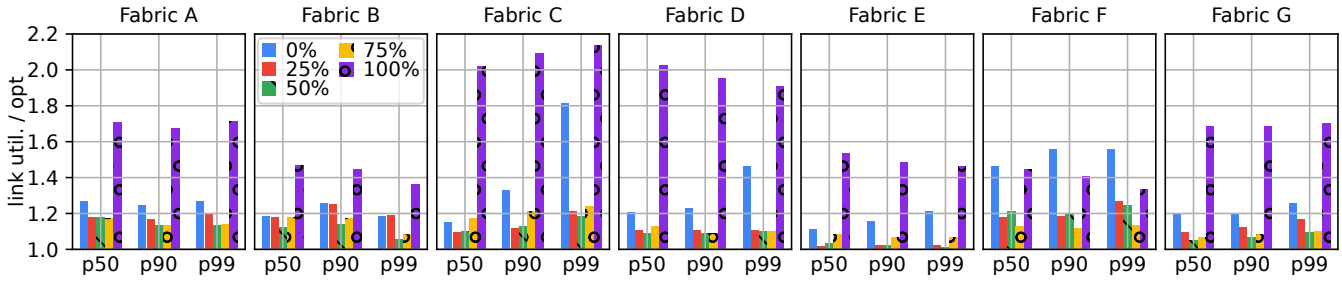


Figure 17: Production link util. of various TE spreads for 7 different fabrics.

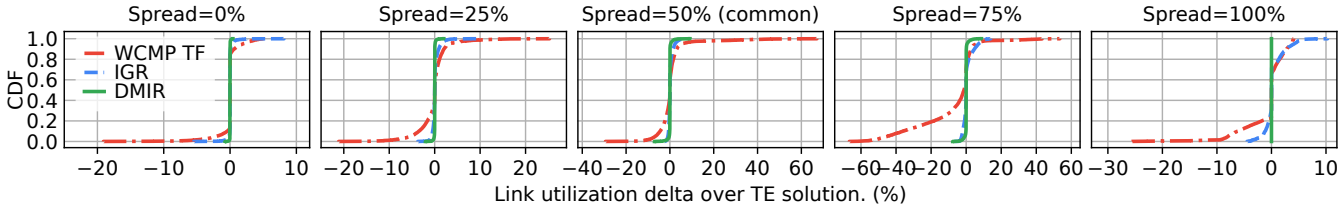


Figure 18: CDF of each link's delta over TE solution

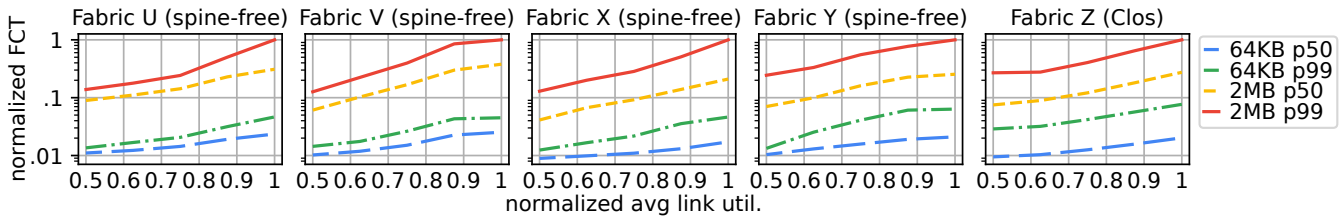


Figure 19: Production flow completion time vs link util. for 5 fabrics.

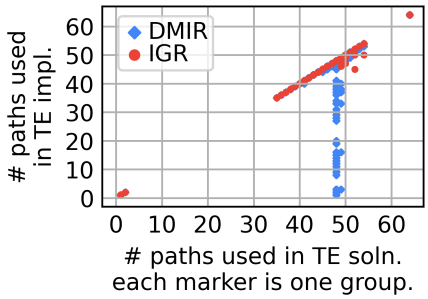


Figure 20: Original path diversity (x-axis) vs. reduced path diversity (y-axis) for each group of 64 ports.

D.4 Flow completion time

Figure 19 extends Figure 11 in §5.7 and shows a total of 5 production fabrics. The first 4 fabrics are spine-free fabrics of different sizes and the last one is a large traditional Clos fabric. As we can see, these fabrics share the same general trend—tail flow completion time of large flows could increase drastically as the average link load increases. In terms of network topology, spine-free fabrics see no difference than the Clos fabric, which confirms that the tail FCT increase is a universal trend.

D.5 Reduced path diversity

Group Pruning is a mechanism to further reduce group sizes while sacrificing path diversity. With the pruning policy introduced in §4.1, we try to minimize the impact on path diversity by pruning the small weights in the groups. Figure 12 and Figure 20 confirm that the chosen pruning policy indeed has very minor impact on path diversity. We obtain all the groups in the *baseline* fabric and plot a scatter plot of their used paths in the TE solution vs. TE implementation. As Figure 20 shows, groups (markers) on the $y = x$ line are not pruned and see no reduction in path diversity. Those not on the line have reduced path diversity. The closer a group is to the x-axis, the more reduction in path diversity it has. DMIR is more aggressive in pruning, some groups could have only a few paths left. However, such groups only constitute less than 0.003% of the total groups (48 out of 1350624) and their traffic impact is insignificant as discussed in §5.8.

One might argue that even 48 groups (of one single port left) would be a blast radius too large. If link failure does happen, traffic on these groups will be dropped. However, these 48 groups are distributed across different switches, they do not share the same port/link. The probability for all of them to fail simultaneously is extremely low. On the other hand, we have implemented a protective lower bound in the Group Pruning

heuristic. If users prefer not to have single-port groups, they can set this lower bound to a number greater than 1 (e.g., 2). In which case, Group Pruning will stop when there are only two ports left in the group. The larger this lower bound, the less likely the group is going to drop its traffic (unless all ports in the group fail). But this also comes with a disadvantage: groups may no longer fit in the table if the lower bound is set too large.