On low-latency-capable topologies, and their impact on the design of intra-domain routing

Nikola Gvozdiev, Stefano Vissicchio, Brad Karp, Mark Handley
University College London (UCL)

ABSTRACT

An ISP's customers increasingly demand delivery of their traffic without congestion and with low latency. The ISP's topology, routing, and traffic engineering, often over multiple paths, together determine congestion and latency within its backbone. We first consider how to measure a topology's capacity to route traffic without congestion and with low latency. We introduce *low-latency path diversity* (LLPD), a metric that captures a topology's flexibility to accommodate traffic on alternative low-latency paths. We explore to what extent 116 real backbone topologies can, regardless of routing system, keep latency low when demand exceeds the shortest path's capacity. We find, perhaps surprisingly, that topologies with good LLPD are precisely those where routing schemes struggle to achieve low latency without congestion. We examine why these schemes perform poorly, and offer an existence proof that a practical routing scheme can achieve a topology's potential for congestion-free, low-delay routing. Finally we examine implications for the design of backbone topologies amenable to achieving high capacity and low delay.

CCS CONCEPTS

• **Networks** → **Routers**; **Network resources allocation**; **Traffic engineering algorithms**; **Wide area networks**; **Physical topologies**; **Network dynamics**; **Public Internet**;

ACM Reference Format:
Nikola Gvozdiev, Stefano Vissicchio, Brad Karp, Mark Handley. 2018. On low-latency-capable topologies, and their impact on the design of intra-domain routing. In *SIGCOMM 2018, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3230543.3230575

SIGCOMM 2018, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230575

1 INTRODUCTION

In recent years, low-latency communication has taken on a new importance. Latency matters, not just for obviously latency-sensitive applications such as telephony or games [15], but also for applications such as web browsing. Much effort has been put into reducing buffer bloat [35], improving TCP loss recovery [10], and deploying congestion control that tries not to build queues [6]. Content providers often reduce latency by moving content closer to users. ISPs can also reduce latency by choosing low-propagation delay paths, though as queueing inflates latency, traffic must be placed on low-delay paths so as to avoid causing congestion.

An ISP has two design choices available that determine the congestion and delay experienced by traffic within its backbone: the topology itself and the placement of traffic on that topology, as determined by a combination of routing and traffic engineering.[1] There has been little systematic study of the interaction between a topology's design and the behavior of routing schemes when run on it. A topology's designer must, even if only implicitly, take into account how the routing system will behave on that topology. Similarly, a new routing system's designer would have in mind (again, perhaps implicitly) topologies on which routing should perform well. Each of these approaches starts by fixing a "legacy" design (either the routing or the topology) and attempts to tailor the other to it. If either legacy design isn't a good fit with placing traffic to avoid congestion and achieve low latency, the ability of the ensemble to meet those aims will suffer.

In this paper we break with this approach and develop a routing-agnostic, first-principles understanding of the sorts of ISP topologies that fundamentally have the potential to deliver time-varying traffic demands with low latency and without congestion: namely, those with *diverse low-latency paths.* We quantify the extent to which 116 real ISP backbone topologies from the Internet Topology Zoo [29] exhibit this potential. From there, we explore in detail how well today's widely known routing systems manage to exploit these same ISP topologies' inherent potential for congestion-free, low-latency traffic placement. We find, somewhat surprisingly, that on topologies with diverse low-latency paths—precisely those with the greatest potential of this sort—status-quo schemes from shortest-path routing to B4 [25] and MinMax traffic engineering (e.g., TeXCP [27]) arrive at traffic placements that suffer congestion or high latency stretch. We reveal why

[1] In the interest of brevity, we will often refer to this combination as routing.

these routing designs encounter these poor outcomes on these promising topologies.

Further, a routing scheme that aims for congestion-free, low-latency traffic placements must not congest links when traffic demands vary over time. A simple way to guard against overloading links when demands increase is to enforce *headroom*: to reserve some minimum fraction of each link's capacity to accommodate foreseen but rare demand increases. Putting capacity aside to soak up demand spikes, however, can be seen as *changing the topology*; a capacity-aware routing scheme may move some traffic to longer paths when the capacity of a short-delay path is "reduced." We explore this interplay, and show that one may view the design space of congestion-free, low-delay routing schemes as falling along a continuum. At one extreme is a notional scheme that employs no headroom on any links—and thus achieves the lowest delay a given topology can offer, at the expense of risking congestion when demand increases. At the other are MinMax schemes, which by definition leave as much headroom on links as possible. These are resilient to congestion caused by demand changes, at the expense of using longer paths. In an ISP setting, where demand isn't known perfectly in advance, it is an open question where on this continuum a practical routing scheme should lie. Is there a sweet spot with enough headroom to cope with demand variability, yet not so much that paths are needlessly circuitous, incurring high latency?

We sketch an approach to routing on path-diverse topologies that achieves their potential more fully, while coping with demand variability. We do not claim that this approach is ready for deployment, or has every engineering detail worked out. Rather, our contributions lie in:

- revealing the nuanced interaction between a topology's path diversity and routing schemes that aim to deliver low latency without congestion;
- revealing exactly why existing routing schemes cannot unlock the low-latency potential of path-diverse topologies;
- identifying the central role of headroom in effecting a necessary trade-off between avoiding congestion and reducing path latency when traffic demands vary; and
- characterizing a routing approach that avoids the pathologies to which existing approaches fall prey on path-diverse topologies, that parsimoniously yet safely applies headroom to cope with demand variability, and that is computationally tractable at ISP scale.

This approach to routing can perform well on all topologies, but it performs especially well where the topology offers a good diversity of low-delay paths. We speculate that such topologies may be rare today because they have been hard to use effectively with existing routing schemes. The adoption of techniques similar to those presented may eliminate this obstacle to building more "mesh-like" network topologies well suited to low-latency, congestion-free traffic delivery.

2 ASSESSING TOPOLOGIES' POTENTIAL FOR LOW LATENCY

If an operator wishes to build a network well-suited to providing robust low-delay communication, how would they measure the extent to which they had succeeded? One could say a topology offers low latency if the shortest paths between points of presence (PoP) lie close to the corresponding great circle routes, but this falls short as a metric for two reasons:

- Geographic, geopolitical, and economic constraints limit where links can reasonably be provisioned.
- Shortest paths may end up congested if demand diverges from that envisaged during provisioning, leading to queuing delays and loss. Avoiding congestion without massive over-provisioning requires using alternate, longer paths.

We don't claim any deep insight into geopolitical or economic constraints that limit link deployment. For now, let us consider only network links that exist in real ISPs.

What we would really like is a metric, agnostic to both routing and traffic, that characterizes how well suited a topology is to providing *robust* low-latency communication. Although the shortest paths in a network may not be ideal, they are the best paths we are sure are viable to provision. How well suited is a network topology to providing low-latency service under traffic loads that are not trivial to route?

To derive such a metric we start from a network map that includes all PoPs and link latencies. For each PoP pair, we compute the lowest latency path. Then for each link on the path, we consider the latency cost to route around that link if it were congested. If the map contains link capacities, we must also take these into account. For example, it is unreasonable to consider a 1 Gb/s link as providing a viable alternate to a congested 100 Gb/s path. We consider an alternate path as a *viable alternate* if its bottleneck has at least the capacity of the bottleneck on the shortest path. If there are multiple alternate paths, we progressively add the n lowest latency alternate paths until their min-cut is sufficient to form a viable alternate. When this is necessary, we consider the propagation delay of the alternate to be that of the n^{th} lowest latency alternate.

We define *path stretch* to be the fraction d_a/d_s, where the viable alternate path's propagation delay is d_a and the delay of the shortest path between the two PoPs is d_s. We set a threshold for path stretch—for example, we may consider a path stretch of 1.4 to be acceptable—and measure alternate path availability (APA), defined as the fraction of links on the shortest path that can be routed around without exceeding this stretch limit. Each PoP pair gives an APA data point in the range from zero (no links can be routed around without excessive delay) to one (all links can be routed around). A CDF of those data points characterizes the network. The resulting curve gives insight into the availability of low-latency alternate paths, and is scale-invariant, so can be used to compare networks of different size and geographic scale. This curve

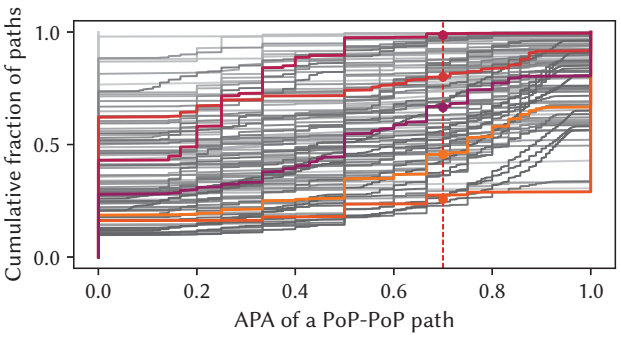


Figure 1: CDF curves of APA for all networks, given path stretch limit of 40%. Five random curves are highlighted.

captures the feasibility of routing around hotspots caused by congestion without dramatically inflating delay.

Figure 1 shows a CDF for each of the 116 networks with diameter greater than 10 ms in the Topology Zoo (augmented with computed link latencies [16]).[2] Networks vary considerably in how well they provide low-latency alternate paths. Consider the x-axis value of 0.7; this indicates paths where 70% of links can be routed around without excessive delay. A corresponding y-axis value of 0.25 indicates that 75% of paths have low-latency alternates that route around at least 70% of the hops. Thus topologies whose curves are to the lower right on this graph provide usable path diversity.

A few curves are horizontal lines; these are clique topologies. We understand these to be overlay networks; for example, one is an older network provisioned using ATM virtual circuits. Overlays are not really interesting from our point of view: the ISP likely uses the overlay technology to provision on demand, rather than rely on intra-domain routing.

To reduce each curve to a single metric for each network, we compute *low latency path diversity* (LLPD) as follows.

$$LLPD = \frac{\text{number of PoP pairs with APA} \geq 0.7}{\text{total number of PoP pairs}}$$

The choice of 0.7 here is not crucial; as Figure 1 shows, the rank ordering does not change greatly if we choose a different threshold in the upper half of the distribution.

An LLPD of close to one indicates that for most PoP pairs, we can route around most of the links on their shortest path without incurring excessive delay. Conversely, we observe an LLPD of close to zero usually indicates a more tree-like network. Networks with mid-range LLPD often consist of wide rings: while they have path diversity, the latency cost of going the "wrong way" around the ring can be high.

Networks with high LLPD typically fall in two categories. Some are well interconnected, resembling a two-dimensional grid. An example of this class is GTS's network in central Europe, shown in Figure 2. Others, such as Cogent, span more than one continent, with good path diversity between continents. The long latency baseline between continents makes it easier to score well on latency stretch, but high LLPD also requires good connectivity within continents.

[2]The Topology Zoo is not without limitations; some topologies are rather old, and PoP locations are often unverified. Nevertheless, it gives a useful view of diverse WAN topologies over time; even older topologies elucidate then-current backbones' delay characteristics.

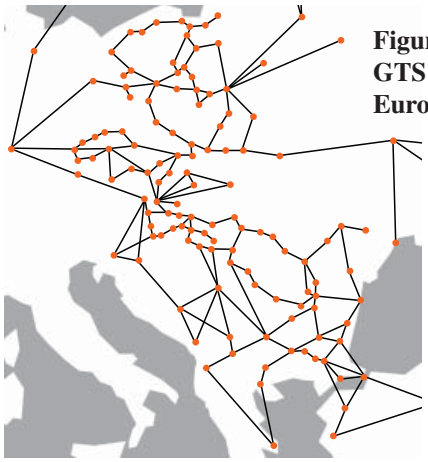


Figure 2: GTS's Central Europe topology.

3 PATH DIVERSITY IS HARD TO USE

In earlier work [21] we showed using small synthetic examples that two-dimensional grid networks can be hard to route, as they inadvertently concentrate traffic. We use LLPD to understand to what extent this is a problem in real networks.

For each topology from the Topology Zoo we synthesize 100 traffic matrices, each representing a moderate load for that network's available capacity. To do so, we use a variant of the gravity model [39]. This model generates traffic aggregates between PoP pairs according to a Zipf distribution, as real-world traffic has been characterized. The original model, however, ignores geographic proximity between endpoints. By contrast, many content providers today place content geographically near to users, yielding greater traffic *locality*. To see how traffic locality affects routing, we extend traffic matrix generation with a locality parameter. The original gravity model dictates the ingress and egress traffic volumes at each PoP; our extension moves load among aggregates that span different distances according to the locality parameter. For values greater than zero we redistribute some traffic from longer-distance flows to shorter-distance ones. Specifically, a locality parameter of ℓ allows short-distance flows to increase by ℓ times their original demand.[3] We find that a locality of one suffices to add significant locality, while larger values tend to under-load long-distance links too much to justify their presence in the topology. Unless stated otherwise, we use a locality value of one in our analyses.

[3]We express these constraints in a simple linear program whose solution yields per-aggregate traffic volumes; we refer the interested reader to [20] for full details.

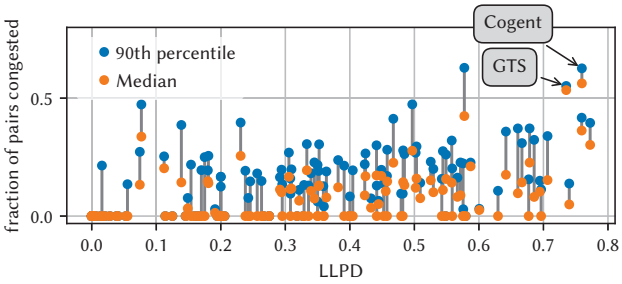


Figure 3: Networks with high LLPD tend to concentrate traffic when using SP routing.

We scale each traffic matrix so that the network is moderately loaded, but not close to being overloaded. The goal is that with optimal routing it is still (just) possible to route the network without congestion if all traffic increases by 30%. This gives a network where, if we minimize maximum link utilization, the min-cut has 23% headroom[4]. In most topologies this corresponds to a median link utilization of 20-30%.

Shortest path routing. We first look at how shortest-path routing [33, 36] performs when link costs are proportional to delay. Figure 3 shows the median and 90^{th} percentile of congested source-destination pairs across all topologies and traffic matrices, when networks are sorted by their LLPD value (on the x-axis). The figure shows that under moderate load shortest-path routing tends to concentrate traffic in networks with multiple low-latency paths (high LLPD).

One conclusion is that networks with good LLPD are not designed to be used with shortest-path routing. Such networks have many low-latency alternative paths, and it seems likely that they have evolved to be run with a traffic engineering scheme able to use these alternative paths. To understand the interplay between topology and routing, we need to examine them using active load-dependent routing systems.

Latency optimality. In Figure 4 we show the performance of active routing schemes. The top of each graph is the same as in Figure 3; it shows the fraction of paths that are congested. The bottom half of each graph is inverted and shows latency stretch, calculated as $\sum_f d_f / \sum_f d_{f,sp}$, where d_f is the delay seen by flow f when routed by the scheme, and $d_{f,sp}$ is the shortest path latency between that source and destination.

Figure 4(a) refers to an optimal routing scheme, where optimality is expressed as minimizing the sum of the propagation delays seen by all flows. Specifically, this scheme minimizes

$$\sum_a n_a \sum_{p \in P_a} x_{ap} d_p \quad (1)$$

subject to the constraints that no link is overloaded and all flows are routed. Here, n_a is the number of flows in aggregate a, P_a are the paths a may take, d_p is the propagation delay of path p, and x_{ap} is the fraction of a's traffic routed on path p.

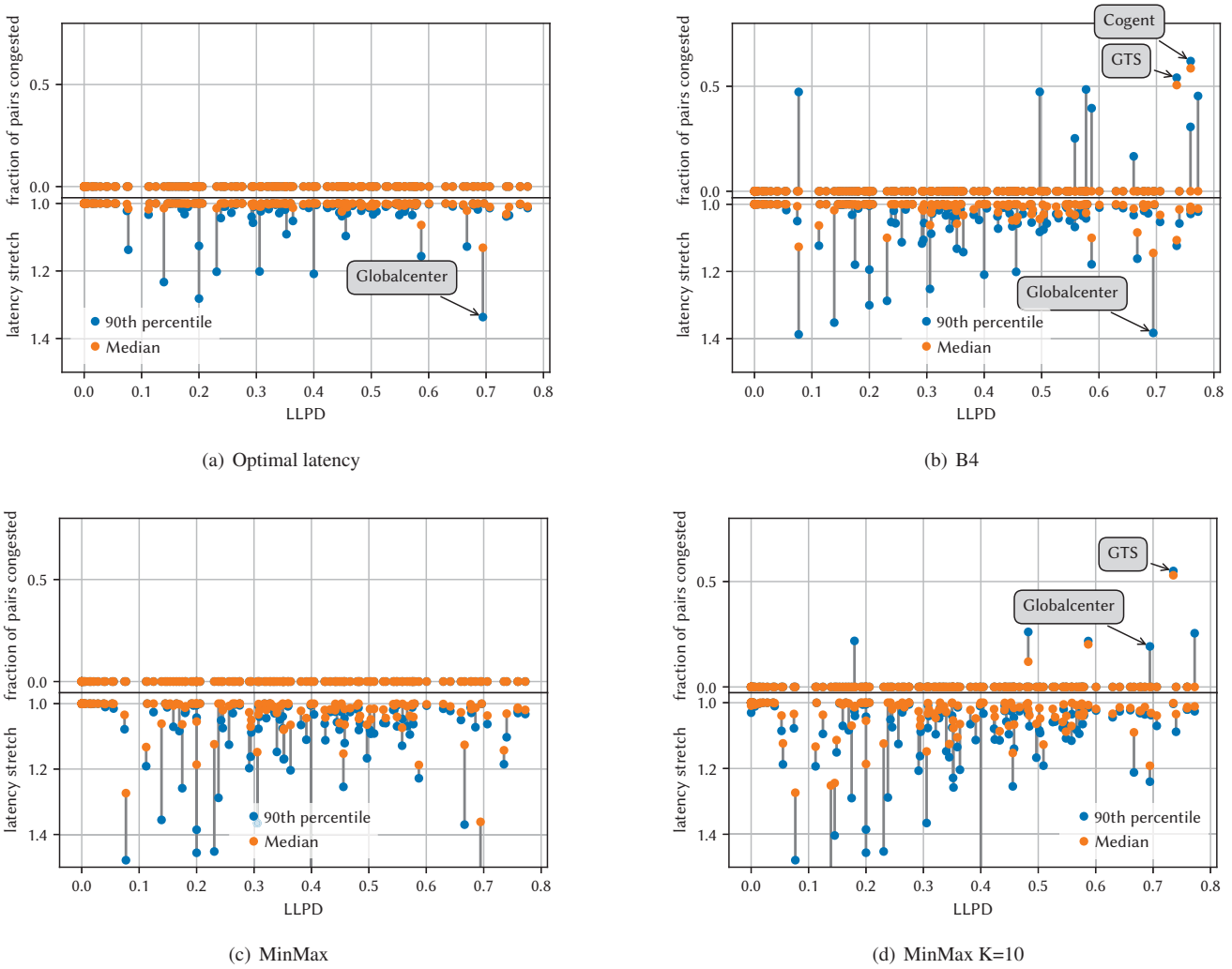
[4]Min cut load is 77%, so the traffic can increase by a factor of $\frac{1}{0.77} = 1.3$.

Figure 4(a) shows that it is possible to route all traffic without causing excessive delay stretch. An exception, Globalcenter, is a full-mesh topology, so likely is an overlay network where it makes little sense performing dynamic routing at the IP level. Grid-like networks such as GTS and diverse intercontinental networks like Cogent that were prominent in Figure 3 give low delay with this sort of optimal routing, which can make very effective use of their low-delay path diversity.

Greedy low latency routing. How do deployed traffic engineering schemes perform? Automatic bandwidth allocation for MPLS-TE [42, 43] considers one aggregate at a time, and places each aggregate on its shortest non-congested path. B4 [25] uses a central controller to assign traffic from aggregates with a slightly improved algorithm. It starts by incrementally placing traffic from each aggregate onto its shortest path. This is done in parallel for all aggregates. When an aggregate's shortest path fills up, B4 starts allocating that aggregate onto the next shortest path, and so forth. Hence, while it considers low-latency paths first, B4 still uses a greedy algorithm. B4 includes prioritization for subsets of traffic. We give all traffic equal priority, as it is generally unclear how an ISP should prioritize demands. In the following, we focus on B4 but the same observations also hold for MPLS-TE.

Figure 4(b) shows the performance of B4 on the topologies from the Topology Zoo, with the same parameters as in Figure 4(a). B4 matches the optimal performance on many of the simpler networks. However, for most of the networks with mid-range LLPD, B4 gives slightly sub-optimal latency. Even more interestingly, it induces congestion on some of the networks with greatest path diversity: for GTS and Cogent, in particular, more than half of B4's paths cross a saturated link in the median case. Clearly, B4's greedy strategy frequently becomes locked into local minima in these topologies.

In [21] we reported a similar effect, and showed a synthetic topology susceptible to Braess's paradox [3]. We initially suspected that this was what was happening here too, but in fact there are other more likely local minima that can trap B4. Consider the part of GTS's network shown in Figure 5. This is a central part of this network, and a large number of aggregates flow through this region. Consider the aggregate from Veszprem (V) to Gyor (G). As B4 allocates traffic, link 1 fills up in the eastbound direction, occupied by the green and many blue aggregates. B4 would normally then start to allocate capacity on the second-best paths. For the blue aggregate of traffic flows, this is possible. However, if there are more red aggregates than blue ones, B4's algorithm will have already filled link 2 in the westbound direction with red traffic. There is no spare capacity for the green traffic as both link 1 eastbound and link 2 westbound are full, and these are the only links out of V. Of course, this example is a simplification of the real traffic allocation. In reality, the red and blue aggregates are hundreds of different aggregates, and



(a) Optimal latency

(b) B4

(c) MinMax

(d) MinMax K=10

Figure 4: Effects of active routing on congestion and delay.

other flows, not shown, are also present. However, the figure captures the basic cause and effect of B4's greedy choices.

The example shows that B4 cannot avoid congestion in this well-connected part of the network. In contrast, an optimal placement would move red traffic aggregates onto the fractionally longer path through G, allowing room for the green traffic on link 2, and so avoiding congestion.

Even when B4 can fit the traffic, latency can be excessive. In Figure 6, two aggregates share a bottleneck link on their shortest paths. B4 will allocate this link equally between the two aggregates until it fills, then start filling the next-shortest path for each aggregate. However, the next-shortest paths have different latency costs, with the blue aggregate needing to take a long detour. It would have been better to allow the blue aggregate to remain on its shortest path, and move more of the red aggregate to its second-best path, as there is minimal latency cost to the red aggregate from doing so.

MinMax based routing. Other traffic engineering schemes such as TeXCP [27] and MATE [11] take the MinMax approach. A pure MinMax approach optimizes traffic placement so as to minimize the maximum link utilization. This is insufficient, as it does not generate unique solutions—many possible placements may have the same maximum link utilization, including ones with very suboptimal high-latency paths. One way to obtain a practical routing system is to minimize the sum of path latencies as a tie-break between traffic placements with equal maximum link utilization.

By definition, MinMax will fit the traffic if it is possible to do so, and Figure 4(c) shows no congestion. However, by focusing on utilization first and only using latency as a tie-break, many aggregates suffer significantly higher latency than they would with optimal routing (see Figure 4(a)). The reason is not complicated: to reduce maximum link utilization, some aggregates are forced over circuitous paths.

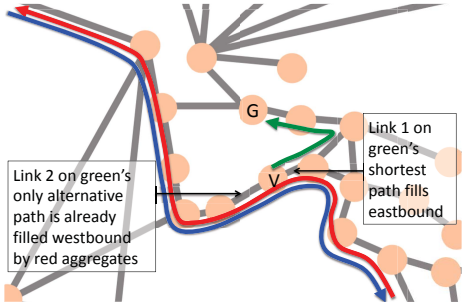


Figure 5: Inadvertent congestion on GTS using B4.

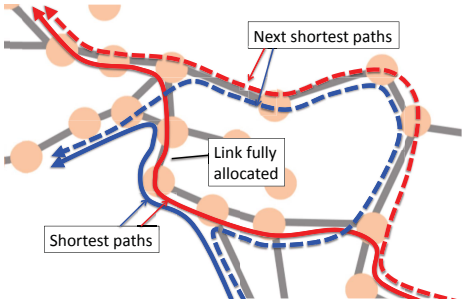


Figure 6: Excessive latency in the GTS topology using B4.

To prevent long paths from being selected unnecessarily, routing schemes such as TeXCP limit path choice to the k shortest paths. The intuition is that if long paths are never given to the MinMax algorithm, a good balance will be struck between reducing latency and minimizing peak utilization.

Figure 4(d) shows the results of running the MinMax algorithm using latency to tie-break, but supplying only the ten shortest paths, as suggested by TeXCP. For most networks with lower LLPD, there is little difference between full MinMax and MinMax with $k = 10$. These networks have little low-latency path diversity, hence some of the ten shortest paths are long. For networks with high LLPD, things are more interesting. Limiting path choice clearly improves latency, though it is still worse than under B4. However, now MinMax can no longer always avoid congestion; networks with high LLPD have a very large number of possible, often non-disjoint, paths, so simply limiting choice to the k best for a constant k is insufficient to avoid congestion.

4 THE HEADROOM DIAL

So far we have considered traffic as a fixed quantity that can be packed into a network. Real network traffic is neither constant in rate nor entirely predictable. A plausible option for a practical routing system is to reserve some minimum fraction of each link's capacity to accommodate foreseen but rare demand increases. We refer to this fraction as headroom.

Living on the edge. Let us first examine how minimizing delay uses links' capacity. We consider again the GTS network, which has high LLPD. Figure 7 shows CDFs of link utilization using latency-optimal placement and our MinMax formulation, the two extreme approaches in terms of link

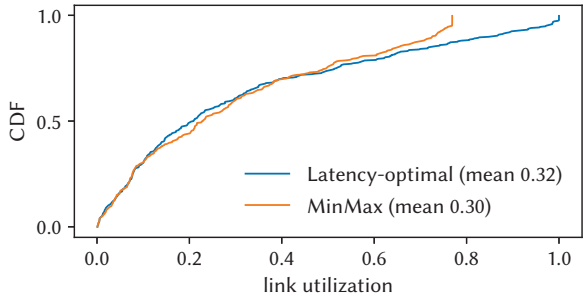


Figure 7: Link utilization in GTS's median case in Fig 4.

utilization, for the median of GTS's traffic matrices from Figure 4. The median latency stretch for this topology is 15% for MinMax and 4% for latency-optimal routing. Despite this, Figure 7 shows that most links are lightly loaded and exhibit similar utilization under both schemes. Clearly, what matters is how loaded the most desirable links are.

Figure 7 also highlights that the few busiest links are loaded very close to 100% in the optimal routing scheme. No real network would be deliberately operated with such extreme link utilizations, since traffic variability would cause (short-term) queuing which in turn would add delay. In practice some degree of headroom must be left on links.

We can regard headroom as a dial that can be controlled by the routing system. We can calculate latency-optimal paths for a given value of headroom by simple scaling down link capacities by the chosen headroom and running the optimal routing scheme on the modified topology. With headroom set to zero, we get the latency-optimal curve, but short-term queuing will adversely affect traffic. If we set headroom to the value MinMax calculates as the maximal free capacity on the busiest links (about 23% in Figure 7), then the latency-optimal algorithm converges with MinMax, giving identical traffic placements. In between the two lies the viable range of traffic placements that all fit the traffic, but which trade off latency against headroom to accommodate traffic variability.[5]

Two key questions emerge from this view of headroom:

- How much headroom can be left before it starts to greatly impact latency?
- How much headroom is needed to statistically multiplex busy links without causing excessive short-term queuing?

Headroom vs. latency. To see the effect of increased headroom on latency, consider Figure 8. This plot shows the median latency stretch as headroom is increased, when performing latency-optimal routing. To see the trend more clearly, we start with a slightly less loaded network - one in which the traffic matrix could be scaled by a factor of 1.65 before it is no longer possible to fit the traffic (i.e., the min-cut of the network is loaded at 60%). We then progressively increase the reserved headroom in steps from 0% reserved headroom

[5]Figure 4 shows that B4 and MinMaxK10 sometimes lie outside this range.

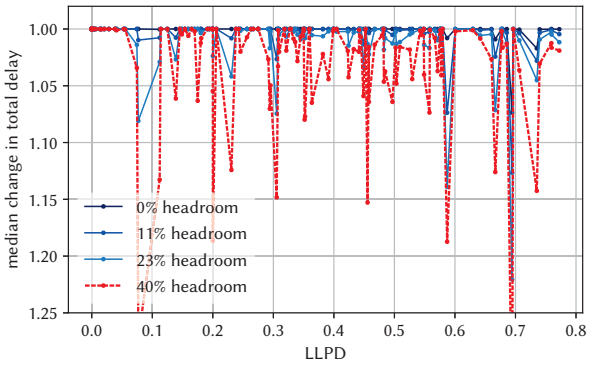


Figure 8: Latency stretch as headroom is increased.

to 40%. For this traffic level, with 40% headroom the latency-optimal placement converges with the MinMax placement.

The most prominent spikes with high LLPD are again from the clique networks; as noted before, these are less interesting because they are overlay networks, so have alternative ways to mitigate congestion. With the exception of these cliques, the other networks show relatively little delay stretch as headroom increases. This is the case even for networks with high LLPD. Only as headroom finally reaches the extreme of MinMax does delay stretch really increase greatly.

The implication is that it is probably unnecessary to live right on the ragged edge of triggering congestion to get paths with reasonably low latency. At the same time, minimizing headroom will normally decrease latency, so it is likely to be worthwhile actively estimating how much headroom is really needed to avoid significant transient queues building.

How much headroom is needed? Any load-dependent routing system must use estimates of traffic volumes to make its routing decisions. These estimates are inevitably imperfect. Suppose, for example, that the routing system recalculates routes every minute. Two factors need to be considered. First, how predictable is the mean traffic rate from minute to minute? Second, how well does short-term variability of aggregates sharing each link statistically multiplex? If we can answer these questions, we can decide how much headroom needs to be allocated when calculating paths, so as to minimize propagation latency while avoiding queuing latency.

Prior studies indicate that mean traffic demands are predictable over minute-long timescales on a WAN, and are more predictable than demands on a LAN [37]. Furthermore, a more recent study of Google's WAN [22] measures a typical backbone link's utilization, which varies less than 10% from minute to minute. To see whether the same conclusions might hold for ISP traffic, we analyzed the best publicly available traffic data. These CAIDA packet traces date from 2013 to 2016, and contain all packet headers from four 10 Gbps links within a U.S. Tier-1 ISP's backbone [5]. For each link we have 40 one-hour traces, typically ranging from 1 to 3 Gbps.

Algorithm 1: Predicting next minute's mean level.

```
prev_value                          // Value measured last minute
prev_prediction                     // Value predicted last minute
decay_multiplier ← 0.98             // 2% decay when level drops
fixed_hedge ← 1.1                   // 10% hedge against growth
scaled_est ← prev_value * fixed_hedge;
if scaled_est > prev_prediction then
    next_prediction ← scaled_est;
else
    decay_prediction ← prev_prediction * decay_multiplier;
    next_prediction ← max(decay_prediction, scaled_est);
```

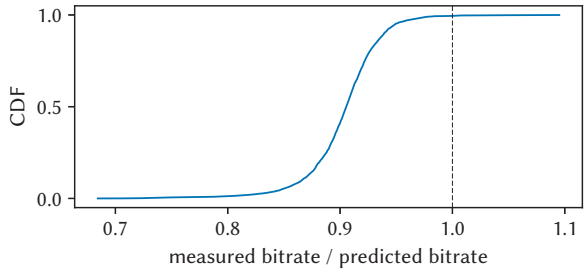


Figure 9: Predictions of mean traffic level (Tier-1 ISP).

We compute the mean traffic level for each minute and apply Algorithm 1 to predict the mean rate in the next minute. This implements a simple conservative strategy: the estimate increases in line with values measured during the last minute, and decays slowly when the measured rate decreases. The aim is aggregates can grow by 10% before exceeding our target.

Figure 9 shows a CDF across all the CAIDA traces of measured mean bitrate in the next minute divided by predicted bitrate. If the traffic were constant, all values would be $1/1.1 = 0.91$. The traffic is very predictable on minute-to-minute timescales: only 0.5% of the time does the actual traffic exceed the target, and then never by more than 10%.

From these traces, we tentatively conclude that 10% link headroom may be sufficient to allow changes in mean traffic rate from minute to minute. When several such aggregates are placed on the same 10Gbps or 100Gbps core link, it is very unlikely they will all exceed their predicted values simultaneously, so in many cases less headroom may be needed. There is a limit to what we can conclude from such traces though: although they do measure Tier-1 backbone traffic, we simply do not know if they are typical of other ISPs.

We also see significant variability on sub-second timescales. We measure the bit-rate from the CAIDA traces each millisecond, and calculate the standard deviation of these values for each minute. Figure 10 is a scatter plot of the standard deviation in minute t plotted against the standard deviation in minute $t + 1$. Different colors map to different traces, though some colors are reused. The absolute value of standard deviation spans a large range, but the points are tightly clustered

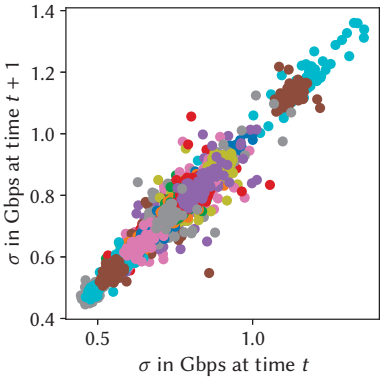


Figure 10: Minute to minute change of the standard deviation of traffic rate.

around the $x = y$ line, indicating that traffic variability does not greatly change from one minute to the next. In these traces, the variability of each aggregate can therefore be characterized so that a routing system can predict how aggregates statistically multiplex. In turn, such a prediction can be used to adjust the headroom needed on a link-by-link basis.

5 DEALING WITH SCALABILITY AND TRAFFIC VARIABILITY CHALLENGES

Given the inability of existing routing schemes to leverage topologies' potential for low latency traffic delivery, the obvious question is whether it is possible to design a practical routing system that both computes low-delay paths and automatically fine-tunes the headroom dial.

Since we aim for close-to-optimal paths, a centralized design seems a promising starting point. Any centralized load-dependent routing system must progress through three stages: measure network demand; calculate paths; install paths in routers. Others have deployed working solutions of this form, especially for modern private WANs [23, 25, 28].

We are primarily interested in the feasibility of the low-delay path calculation stage. To calculate paths, the centralized controller needs traffic measurements. We assume that ingress routers can measure traffic, and can report traffic volumes and approximate numbers of flows to the controller. For the former information, they periodically send the controller batches of traffic counter values. Those values can be collected by reading hardware counters multiple times each second, as demonstrated in [8, 32]. For the latter, ingress routers can use known techniques to track the number of flows per aggregate – e.g., through specific hardware support [12] or estimations from sampled mirrored packet streams [9].

Given this data, the controller has to calculate the necessary headroom, reserve this capacity, and then optimize traffic placement so as to minimize latency subject to the constraint of avoiding congestion. This is challenging for two reasons. First, the optimization itself may be expensive on large networks that provide good LLPD, since by definition they have many (low-latency) paths. Second, computing necessary headroom involves understanding how aggregates

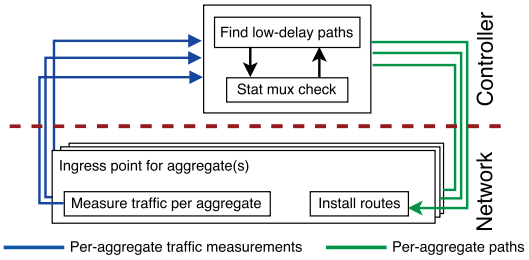


Figure 11: High-level overview.

statistically multiplex, and this in turn depends on which aggregates might share a link. How should an intra-domain routing system design address these two challenges?

Design overview. For the path computation to scale on large, dense networks and to account for aggregates' multiplexing, a centralized controller can iterate through three phases, as shown in Figure 11:

(1) find the best low latency solution, using current estimates of needed headroom.
(2) appraise how well the proposed solution statistically multiplexes on busy links.
(3) tweak headroom when multiplexing is unsatisfactory, and repeat from (1).

This approach requires that finding low latency solutions and checking statistical multiplexing are both fast, as online route computation needs to run in seconds.

Path optimization. Given tentative values for headroom, formalizing the optimization problem is relatively simple. One way to do it is to cast path selection as a multi-commodity flow problem, with one commodity per aggregate, in the spirit of Bertsekas et al. [2]. However, the size of this optimization model scales with the product of number of aggregates and number of links, hence this approach may quickly become impractical for large networks with high LLPD.

An alternative path-based formulation is shown in Figure 12. This is not only more efficient, but also better suited to iterating for adjusting headroom. Despite being similar to a standard path-based formulation of the multi-commodity flow problem, a few differences are worth noting.

If aggregates' demands globally exceed the capacity of possible paths, congestion cannot be avoided. In this case the formulation spreads traffic as equally as possible across all links, as expressed by the last term of the objective function.

Otherwise, the formulation selects the lowest-delay paths subject to the constraint that they avoid congestion; M_2 is large to ensure this term dominates. Once this term is satisfied, the delay minimization term determines the choice of paths. This term has two factors: the first factor minimizes the sum of path latencies, while the second factor guarantees uniqueness of the solutions, hence predictability of routing. If there are two aggregates competing for a link, and moving either of the

secondary goal: minimize latency

tie-breaker: minimize delay stretch

primary goal: avoid congestion

if congestion is unavoidable: minimize total overload

$$\min \sum_a n_a \sum_{p \in P_a} x_{ap}(d_p + \frac{d_p M_1}{S_a}) + M_2 O_{max} + \sum_l O_l$$

$$\sum_a \sum_{p \in P_a} x_{ap} B_a < C_l O_l \quad \forall l \in L \qquad \text{per-link overload definition}$$

$$1 \leq O_l < O_{max} \quad \forall l \in L \qquad \text{maximum overload definition}$$

$$\sum_{p \in P_a} x_{ap} = 1 \quad \forall a \in A \qquad \text{all the aggregates must be routed}$$

$a \rightarrow$ aggregate $B_a \rightarrow$ bandwidth of aggregate $n_a \rightarrow$ flows in aggregate

$p \rightarrow$ path $d_p \rightarrow$ path delay $S_a \rightarrow$ shortest path delay $l \rightarrow$ link $C_l \rightarrow$ link capacity

$P_a \rightarrow$ possible paths for aggregate $x_{ap} \rightarrow$ fraction of a on path $O_l \rightarrow$ overload of link

Figure 12: Linear Program for latency optimization.

two to a longer path yields the same total delay, this second factor ties-breaks by moving the aggregate whose RTT is already larger – e.g., giving more predictable latency based on geography, and fitting better with how CDNs work. To ensure this is only a tie-break, M_1 is a very small constant.

To directly optimize this model using a linear program (LP) solver would require all possible paths for all aggregates be considered. Such a direct approach is inefficient for large networks with high LLPD, which are precisely the ones which we expect to benefit the most from this optimization approach.

Fortunately, this is not necessary, given the path-based objective function above. Consider an aggregate that does not fit on its shortest path. An optimal solution places as much traffic as possible on the shortest path, then allocates the rest to the next-best path. If the next-best path is not congested, adding further paths for this aggregate serves no purpose, as they will never be used. Essentially, there is a "delay threshold" for each aggregate, beyond which paths of longer delay will never be used. We don't know this delay threshold a-priori, but we can learn it from successive runs of the LP solver.

The above approach is represented in Figure 13. We associate each aggregate with the list of its k shortest paths, where initially $k = 1$. We formulate an LP where possible paths for each aggregate are those from its list. We then solve the LP and look for links that are maximally overloaded – i.e., such that $O_l = O_{max} > 1$. For all aggregates that cross those links we extend the list of paths by generating shortest paths for an increasing k, run the LP again and repeat. We iterate until we find paths with no overloaded links.

Even though this approach involves multiple runs of the LP optimization, it actually runs very quickly because the number of variables (paths) in each run is small. The bottleneck is not the linear optimizer, but the k shortest paths algorithm [49], the results of which can be readily cached. This yields sub-second runtimes even with tens of thousands of aggregates.

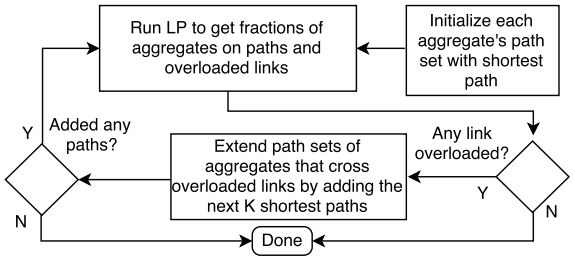


Figure 13: Obtaining paths and per-path aggregate fractions, assuming each aggregate's demand is known.

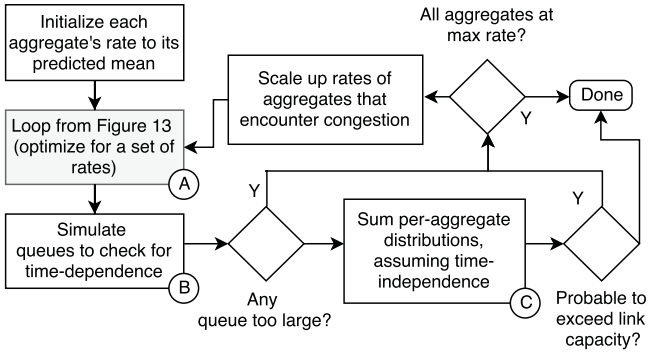


Figure 14: Iteration to assess statistical multiplexing.

Link multiplexing check. Given a representative demand B_a for each aggregate, the iterative LP approach will compute a latency-optimal solution. Unfortunately aggregates cannot be characterized by their mean bitrate; doing so will cause excessive queuing if no headroom is allocated. Since different aggregates have different variability, the headroom needed to allow for statistical multiplexing depends on which aggregates are placed onto each link. However, to determine which aggregates should share a link, the LP optimizer has to know how much headroom to leave. How can we break this cycle?

Our strategy is shown in Figure 14. First we compute a prediction of the mean rate for each aggregate, based on its measured behavior from the last minute. Using mean values as an initial estimate of B_a, we perform a preliminary optimization of aggregate placement onto paths. We then use the short timescale traffic measurements from the ingress routers to assess whether these aggregates will statistically multiplex well enough on each link. If this test passes for all links, then the traffic placement is good. If it fails for any link, we scale up B_a for those aggregates traversing that link, and re-optimize (Ⓐ in Figure 14). Scaling up aggregates serves to add headroom, but only for those aggregates that don't multiplex well. The alternative—scaling down the link speed to add headroom—is less effective, as it prevents other less variable aggregates being chosen to use the link instead.

Aggregates may fail to statistically multiplex, either because traffic bursts are temporally correlated [13], or because

these aggregates are very variable and statistically likely to exceed capacity of the links they share. Testing for temporal correlation (Ⓑ in Figure 14) is simple. For each aggregate, the controller has measurements of data transmitted in each 100 ms period. It simply sums the values from each aggregate for each corresponding 100 ms period to test if the link's bandwidth would be exceeded. If it is, the excess traffic would be queued, so is carried over to the next 100 ms period. The controller rejects any solution yielding transient queuing delays that exceed a maximum allowed value (say, 10 ms).

To evaluate uncorrelated multiplexing (Ⓒ in Figure 14), we can treat aggregates as random processes, each with a different discrete distribution given by its 100 ms bandwidth measurements. When these aggregates multiplex on a link, we care about the resulting multiplexed bandwidth distribution. We treat each aggregate's measurements as a probability mass function (PMF). For each link, we take the convolution of the PMFs of aggregates that cross that link, and examine the convolved PMF. If the probability that this PMF exceeds the link's capacity is below a threshold, we can conclude that the aggregates will multiplex well enough to fit. The threshold comes again from the maximum queue we wish to allow—if we allow 10 ms queues and the measurements span an interval of 60 seconds, the threshold would be $\frac{10}{60000} = 0.00016$.

Despite having tens of thousands of aggregates, each with a different PMF, two optimizations allow all the needed convolutions to be performed in milliseconds. First, we don't test multiplexing on a link if the sum of the 100ms peak traffic levels of all aggregates placed on that link does not exceed the link capacity: those links are indeed guaranteed to pass both tests. Second, since convolution in the time domain is equivalent to multiplication in the frequency domain, we can transfer data to the frequency domain using a Fast Fourier Transform (FFT), multiply the frequencies, and invert the FFT to get the convolved distribution. This algorithm runs in $O(N \log N)$ time, where N depends on the quantization applied to the discrete time-domain data. We experimentally found that 1024 levels per distribution yields good performance.

Feasibility. For conciseness, we refer to the iterative latency-optimal routing algorithm described above as Low Delay Routing (LDR). To be practical, LDR must be able to calculate paths quickly on large, dense networks. Figure 15 shows CDFs of the runtime of the LDR algorithm on the Topology Zoo networks whose LLPD exceeds 0.5; these are the hardest to route. Several of these topologies are of significant scale: the greatest number of nodes is 197 (90th percentile: 74 nodes) and the greatest number of links is 243 (90th percentile: 96 nodes). The "LDR" curve includes caching of the k shortest paths, whereas the "cold cache" curve shows the first run, before the cache is populated. For comparison, the "link-based" curve shows that a multi-commodity flow formulation of the same optimization is about two orders of magnitude slower. We conclude that the LDR approach is fast enough to use in online centralized routing systems.

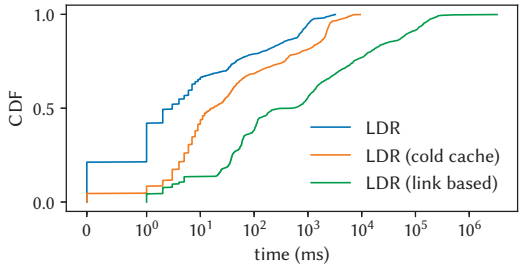


Figure 15: Run time of optimization algorithms.

6 LATENCY, LOCALITY, LOAD & LLPD

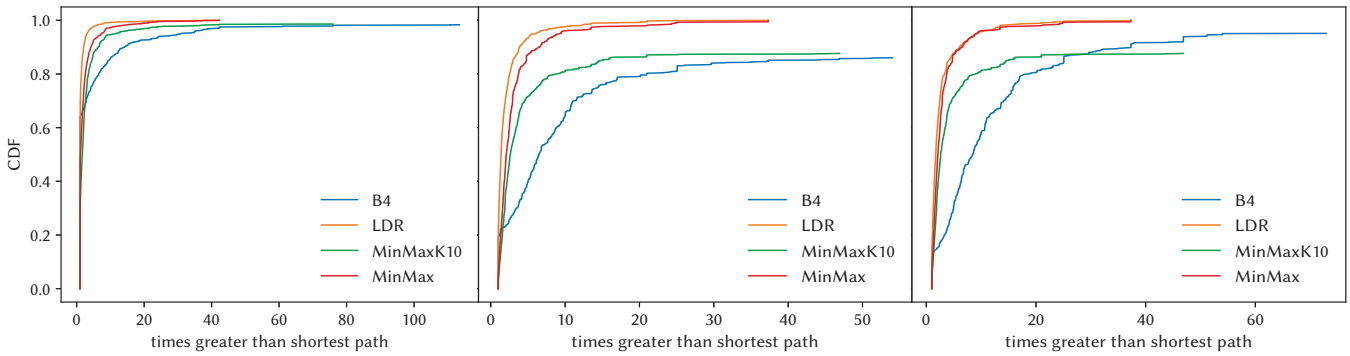
With a practical latency-optimal routing system to compare against, let us revisit the landscape to better understand the interactions between topology, traffic locality, load and latency.

In general, the higher the LLPD, the harder it is to route a network using shortest path routing, but the more options an active load-dependent routing system has to move load around. With more options, heuristic algorithms such as B4 and MinMaxK10 can get stuck in a local minimum. Under low load this is not usually a problem, but as load increases and the routing system moves more traffic onto longer paths, getting stuck in a local minimum becomes more likely.

Under very high load we see that unrestricted MinMax becomes close to optimal, as options for re-routing become limited. However, under low loads MinMax chooses circuitous routes as it tries to minimize peak link utilization.

Traffic locality also plays a role here, though different routing schemes are affected differently. If we set the locality parameter to zero, increasing the mean distance that traffic travels, we observe that B4 becomes significantly less optimal as it fills the best long distance paths first, forcing some sub-optimal routings thereafter. On networks with low LLPD, when locality is low, both MinMax algorithms may increase latency significantly. This is common on topologies with large rings, where MinMax often routes traffic the long way round the ring in pursuit of reduced utilization on the shorter path. Conversely, increasing locality beyond our default value of one has little effect on any routing scheme.

Finally, we must discuss headroom in the context of B4. B4 was designed for use on Google's network with controlled traffic sources. If we wish to use it for ISP networks, we will also need to add headroom. Our formulation for assessing traffic predictability and statistical multiplexing is quite general, so we can also apply it to B4 to calculate desired headroom. When we do so, we find that headroom interacts with B4 in an interesting manner. Consider again the GTS topology in Figure 5, where B4 became congested. If we allow, say, 10% headroom, B4 will stop short of saturating all the links on the first pass, and move on to placing some of the long



(a) LLPD < 0.5, no headroom (b) LLPD > 0.5, no headroom (c) LLPD > 0.5, 10% headroom

Figure 16: Maximum path stretch

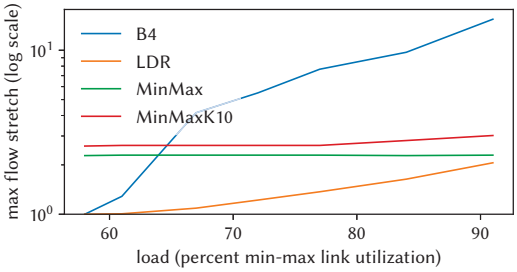


Figure 17: Effect of load on median latency stretch

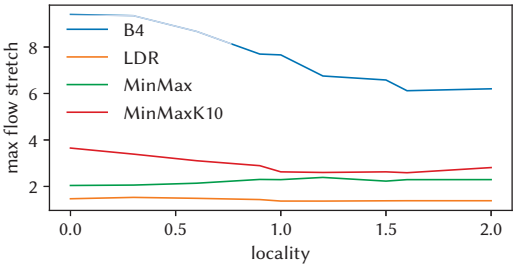


Figure 18: Effect of locality on median latency stretch

distance traffic on the slightly longer path via G. If the traffic from G→V that B4 failed to place can fit within the reserved headroom, then it can still be routed after all other traffic has been placed. Of course, the headroom was there to mitigate transient congestion, and now less headroom is available, so queuing is more likely. Still, we observe that congestion is less likely with B4 when explicitly allocating headroom.

Let us examine some measurements that illustrate the observations above. First, how does LLPD and headroom affect latency stretch? Figure 16 shows CDFs of the *maximum path stretch* for each traffic matrix under the same conditions as Figure 4 (min-cut 0.7 load and locality 1). The different curves illustrate the effects of LLPD and headroom. In Figure 16(a) we see the networks with low LLPD. These networks have few low-latency alternate paths; for some topologies and traffic matrices the maximum stretch is very high - over 100x in the limit with B4. There is not much to choose between the four algorithms here, as the topologies don't provide many routing options. No headroom is reserved in Figure 16(a), but adding 10% headroom makes little difference to the CDF.

Figures 16(b) and 16(c) show the networks with high LLPD. Where the CDF fails to reach 1.0, this indicates that in the remaining scenarios the routing system could not find a placement that would fit all the traffic. This happens with both B4 and MinMaxK10. When headroom is added, B4 can fit traffic in a wider range of scenarios, though these graphs don't capture the degree to which B4 eats into the supposedly reserved headroom to do so. B4 also pays a latency price in such cases. LDR with headroom and MinMax give very similar maximum stretch; this is mostly because our MinMax formulation optimizes for latency once its utilization goal has been satisfied. As we saw in Figure 8, LDR with headroom and MinMax exhibit very different *median* latency stretch.

To understand how robust the results are as load changes, we examine the median latency stretch across all traffic matrices in the topologies with LLPD greater than 0.5. In Figure 17 we observe how this changes as we increase overall load. B4 is quite sensitive to high load levels in these networks, but the other schemes are not. Note that at low load, when everything fits on the shortest path, B4 is optimal, whereas at high load, the MinMax and optimal curves converge.

Figure 18 shows the same metric as we adjust the locality of the traffic matrix. A locality of zero tends to load long distance links more, whereas localities above one tend to load local links more. B4 is especially sensitive to congesting the wide-area links, so a traffic matrix with low locality tends to hurt latency. The MinMax algorithms don't congest the wide-area links, but they do load-balance more traffic off them than is strictly necessary, so they also exhibit increased latency with low locality. All the schemes give better results when

the traffic matrix exhibits higher locality, though the MinMax curves are rather level with locality greater than 1.5.

7 RELATED WORK

The research community has previously devoted attention to topology, routing, and demand in ISPs' backbones, but prior work has typically discussed each of these aspects in isolation. For example, studies of WAN topology design and optimization [4, 18, 38] ignore the network's routing system. We have shown that it is critical to consider the interplay between topology and routing—e.g., the routing system may impose constraints on how the topology can be evolved (§8).

Measurement studies [34, 41] consider the impact of factors including topology and (BGP) routing policies on inter-domain path latency. In this work we focus on intra-domain routing, and show that it significantly impacts path latency in WANs. The flattening of the AS-level hierarchy [17] and the rapid increase in popularity of Content Delivery Networks are also likely to magnify the impact of intra-domain routing.

Most prior intra-domain routing systems [11, 27, 30, 46] focus on maximizing spare link capacity, a problem profoundly different from the delay minimization one we study. These two problems' different natures also motivate oblivious and semi-oblivious approaches [7, 31]—while maximizing spare capacity is inherently more robust to demand fluctuations, delay minimization requires judicious application of headroom to keep paths as low-latency as practicable while maintaining slack capacity for demand spikes.

We are far from the first to look at low-delay routing. Gallager [14] provides a general LP-based definition of path optimization problems whose objective functions can encompass delay. Follow-on contributions [26, 45, 47] consider the problem of minimizing per-packet latency, assuming that delay is a function of the load on traversed links—which is the case if those links are bottlenecks. In the WAN scenario this is often not the case, as most flows that traverse the network are already bottlenecked at a slower access link.

Alternative mechanisms, e.g., based on specific queuing schemes [19] or new transport layer mechanisms [48, 50], can also provide low latency, especially to delay-sensitive, short-lived flows. However, those mechanisms require precise information about latency-sensitive and latency-insensitive flows that ISPs typically do not have (but see §8).

We are also not the first to advocate for the importance of delivering Internet traffic with low delay. Recent contributions span specific intra-datacenter solutions [1] to more general techniques for use in networked systems [44], and calls to arms for the research community [40]. We contribute a systematic study of the interplay between routing and topology, and leverage that understanding to describe a new approach to routing that can best exploit low-latency path diversity.

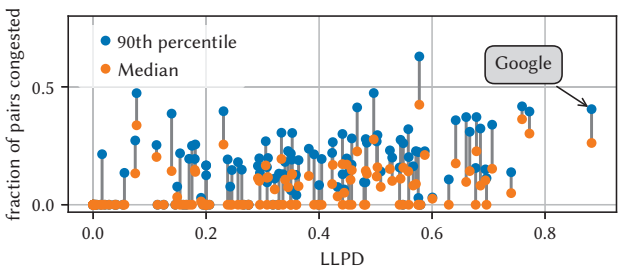


Figure 19: Same shortest-path routing data as in Figure 3, but with Google's topology (LLPD = 0.875) added.

8 DISCUSSION

We have explored the interplay between the diversity of low-latency paths in a topology and the ability of a routing scheme to exploit that diversity to achieve congestion-free, low-delay traffic delivery. While we have taken an initial step toward routing that better harnesses this diversity in today's ISP topologies, important questions in this area remain.

Modern enterprise networks. The Topology Zoo consists largely of transit networks from recent decades, most of which were not designed with dynamic, latency-minimizing routing in mind. How do state-of-the-art enterprise networks compare? We examined a wide-area global enterprise network owned by Google [24]. In Figure 19 we revisit the behavior of delay-proportional shortest-path routing by augmenting Figure 3 with results for Google's network. The new datapoint clearly exhibits the greatest LLPD among all topologies and, unsurprisingly, cannot be routed using shortest paths alone. Google's own B4 in fact performs nearly optimally on this network without exhibiting the pathologies in Section 3. We conjecture that this topology was explicitly designed for dynamic latency-minimizing routing. We believe it to be an important existence proof that it is possible and economically viable to build a high-LLPD network that spans the globe. We note though that an enterprise network can control traffic at endpoints, so demand may be more predictable than at an ISP.

Does routing influence topology? The ISP topologies we have studied were designed to be used with pre-existing routing schemes. Have routing systems' limitations constrained how networks themselves have grown? Apart from the example of B4 and SNet, we cannot definitively answer this question without deploying an optimal routing system and waiting a decade or so to see how ISPs upgrade their networks. Nor can we accurately determine which topology upgrades might be likely; we have no model for the economic and geopolitical constraints that gate new link deployment. We can, however, examine the extent to which topology upgrades enable better service from today's routing systems. When adding links to a topology in principle ought to improve service but in practice does not, an ISP wouldn't likely choose to

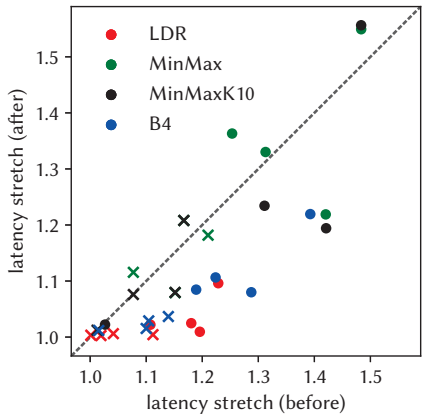


Figure 20: Latency benefits of network growth

grow the network in that way. If, however, the ISP had a routing system that could harness those added links to improve service, the ISP would see benefit in adding them.

We examined four networks that are difficult to route with low latency, even with optimal traffic placement. The networks chosen are those from Figure 4(a) with high latency, but we exclude those with clique topologies, to which we cannot add links. We use the LLPD metric to determine which additional links might confer the greatest benefit. Of all the links to be possibly added, we add the one that gives the greatest increase in LLPD. We then repeat this process until the number of links has increased by 5%. As each link added improves LLPD, the resulting network will, in principle, be more amenable to low-latency routing.

Figure 20 shows how much the different routing schemes profit. As before, load is 0.7 and locality 1. The x-axis and y-axis respectively show the latency stretch on the original and enhanced networks. We plot points for each routing scheme: crosses show median stretch, and dots the 90th percentile. Ideally, all points would be close to the x-axis. At the very least, we would hope for the points to be below the $x = y$ line, indicating that adding links reduces latency.

Only LDR fully takes advantage of the new links, giving median latency stretch very close to unity. For three of the networks, LDR's 90th percentile is less than all other routing systems' median latency. B4 can also take advantage of new links, though it is far from perfect. Both MinMax algorithms fare much worse. In some cases, adding new links that improve LLPD actually *increases* latency, as both algorithms use the links to load balance more widely.

We conjecture on the basis of these preliminary experiments that the routing scheme does determine which links are best for an ISP to add. Although we cannot be sure that limitations in today's routing systems prevent ISPs from deploying lower-latency topologies, it seems that may be the case.

Limits to LLPD's applicability. We formulated LLPD as a simple metric to *retrospectively* assess how well an existing topology supports the delivery of traffic with low delay. What about *prospectively,* to determine which links to add or increase in capacity when evolving a topology? We don't believe LLPD is always the best instrument for predicting which evolved versions of a topology offer the lowest latency.

Consider an Asia-centered network already with high LLPD that stretches to Europe in the West and the US in the East. Adding a single non-redundant transatlantic link would reduce latency for some Europe↔US traffic, but may actually reduce LLPD, as there is no low-latency alternate path available.

Even if adding a link increases LLPD, without a routing scheme such as LDR that can effectively use path diversity, latency may not decrease, as shown in Figure 20. Where such a routing scheme is used, if forecast traffic matrices are also available, then the optimized value of LDR's objective in Figure 12 provides a better metric to evaluate the impact of the adding of new links on latency. Combining this metric with other constraints to reflect economic and other costs of link deployment may be a fruitful direction for future work on optimizing the evolution of a topology.

Extension to differentiated traffic classes. Not all flows are equal; some applications may be more latency sensitive than others, though it is not always easy for an ISP to know which are which. If an ISP does know which flows should be prioritized, it is straightforward to extend our optimization framework to split aggregates according to priority, and to modify the LP constraints and weights so as to prioritize giving low latency paths to flows that will benefit most.

Generality of building blocks. We believe that LDR's iterative growth of the set of paths used to route an aggregate and its convolution technique for determining headroom should both be of use in other low-delay routing systems. For example, B4 assumes no variation in traffic demands, as it is designed for an enterprise setting in which the routing controller has global knowledge of all sources' exact rates. The convolution approach to headroom could be useful in adapting B4 to the ISP setting. And while MinMax K10's fixed choice of the ten lowest-delay paths is bound to be too great or too small for some aggregates, iteratively growing the path set for MinMax per aggregate, subject to a bound on delay stretch, should help MinMax avoid needless detours.

We hope our work can be a first step toward enabling the deployment of ISP topologies that are better than today's for the provision of low-latency service, but remain unbuilt because today's routing systems cannot fully harness their path diversity.

ACKNOWLEDGMENTS

We thank Michael Walfish, our shepherd Michael Schapira, and the anonymous reviewers for their helpful comments. Nikola Gvozdiev was supported by a Google European Doctoral Fellowship.

REFERENCES

[1] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. 2010. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Proc. USENIX NSDI*.

[2] Dimitri P. Bertsekas, Robert G. Gallager, and Pierre Humblet. 1987. *Data networks*. Vol. 2. Prentice-hall Englewood Cliffs, NJ.

[3] Dietrich Braess. 1968. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung* 12, 1 (01 Dec 1968), 258–268.

[4] Robert S. Cahn. 1998. *Wide Area Network Design: Concepts and Tools for Optimization*. Morgan Kaufmann Publishers Inc.

[5] CAIDA. 2018. Internet Data – Passive Data Sources. (2018). https://www.caida.org/data/passive/

[6] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Yeganeh, and Van Jacobson. 2017. BBR: Congestion-Based Congestion Control. *Commun. ACM* 60, 2 (2017), 58–66.

[7] Marco Chiesa, Gábor Rétvári, and Michael Schapira. 2016. Lying Your Way to Better Traffic Engineering. In *Proc. ACM CoNEXT*.

[8] Andrew R Curtis, Jeffrey C Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. 2011. DevoFlow: Scaling flow management for high-performance networks. *ACM SIGCOMM Computer Communication Review* 41, 4 (2011), 254–265.

[9] Nick Duffield, Carsten Lund, and Mikkel Thorup. 2003. Estimating Flow Distributions from Sampled Flow Statistics. In *Proc. ACM SIGCOMM*.

[10] Nandita Dukkipati. 2013. Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses. Internet Draft. (2013).

[11] Anwar Elwalid, Cheng Jin, Steven Low, and Indra Widjaja. 2001. MATE: MPLS adaptive traffic engineering. In *Proc. IEEE INFOCOM*.

[12] Cristian Estan and George Varghese. 2002. New Directions in Traffic Measurement and Accounting. In *Proc. ACM SIGCOMM*.

[13] Sally Floyd and Van Jacobson. 1993. The Synchronization of Periodic Routing Messages. In *Proc. ACM SIGCOMM*.

[14] Robert G. Gallager. 1977. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE Transactions on Communications* 25, 1 (Jan. 1977), 73–85.

[15] Riot games. 2016. Fixing the Internet for real time applications: part II. https://engineering.riotgames.com/news/fixing-internet-real-time-applications-part-ii. (2016).

[16] Steven Gay, Pierre Schaus, and Stefano Vissicchio. 2017. REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms. *CoRR* abs/1710.08665 (2017). arXiv:1710.08665 http://arxiv.org/abs/1710.08665

[17] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. 2008. The Flattening Internet Topology: Natural Evolution, Unsightly Barnacles or Contrived Collapse?. In *Proc. ACM PAM*.

[18] Wayne D. Grover. 2004. *Mesh-Based Survivable Networks*. Prentice Hall PTR.

[19] Liang Guo and Ibrahim Matta. 2001. The War between Mice and Elephants. In *Proc. IEEE ICNP*.

[20] Nikola Gvozdiev. 2018. Traffic Matrix Generator. https://github.com/ngvozdiev/tm-gen. (2018).

[21] Nikola Gvozdiev, Stefano Vissicchio, Brad Karp, and Mark Handley. 2017. Low-Latency Routing on Mesh-Like Backbones. In *Proc. ACM HotNets-XVI*.

[22] Avinatan Hassidim, Danny Raz, Michal Segalov, and Ariel Shaqed. 2013. Network utilization: The flow view. In *Proc. IEEE INFOCOM*.

[23] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving High Utilization with Software-Driven WAN. In *Proc. ACM SIGCOMM*.

[24] 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. 2018. Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in a Software-Defined WAN. In *Proc. ACM SIGCOMM*.

[25] Sushant Jain, Alok Kumar, Subhasree M, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim W, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. B4: Experience with a Globally-Deployed Software Defined WAN. In *Proc. ACM SIGCOMM*.

[26] Umar Javed, Martin Suchara, Jiayue He, and Jennifer Rexford. 2009. Multipath protocol for delay-sensitive traffic. In *Proc. IEEE COMSNETS*.

[27] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. 2005. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *Proc. ACM SIGCOMM*.

[28] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. 2014. Calendaring for Wide Area Networks. In *Proc. ACM SIGCOMM*.

[29] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. 2011. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (october 2011), 1765 –1775.

[30] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proc. ACM SIGCOMM*.

[31] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. 2018. Semi-Oblivious Traffic Engineering: The Road Not Taken. In *Proc. NSDI)*.

[32] Jeffrey C Mogul and Paul Congdon. 2012. Hey, you darned counters!: get off my ASIC!. In *Proc. ACM HotSDN*.

[33] J. Moy. 1998. OSPF Version 2. RFC 2328. (April 1998). http://www.ietf.org/rfc/rfc2328.txt

[34] Wolfgang Mühlbauer, Steve Uhlig, Anja Feldmann, Olaf Maennel, Bruno Quoitin, and Bingjie Fu. 2010. Impact of routing parameters on route diversity and path inflation. *Computer Networks* 54, 14 (2010), 2506–2518.

[35] Kathleen Nichols and Van Jacobson. 2012. Controlling Queue Delay. *Commun. ACM* 55, 7 (2012), 42–50.

[36] Dave Oran. 1990. OSI IS-IS Intra-domain Routing Protocol. RFC 1142. (1990).

[37] Yi Qiao, Jason Skicewicz, and Peter Dinda. 2004. An empirical study of the multiscale predictability of network traffic. In *Proc. IEEE HPDC*.

[38] Bruno Quoitin, Virginie Van den Schrieck, Pierre François, and Olivier Bonaventure. 2009. IGen: Generation of Router-level Internet Topologies through Network Design Heuristics. In *Proc. IEEE ITC*.

[39] Matthew Roughan. 2005. Simplifying the synthesis of internet traffic matrices. *ACM SIGCOMM Computer Communication Review* 35, 5 (2005), 93–96.

[40] Ankit Singla, Balakrishnan Chandrasekaran, P. Brighten Godfrey, and Bruce Maggs. 2014. The Internet at the Speed of Light. In *Proc. ACM HotNets-XIII*.

[41] Neil Spring, Ratul Mahajan, and Thomas Anderson. 2003. The Causes of Path Inflation. In *Proc. ACM SIGCOMM*.

[42] Richard Steenbergen. 2013. MPLS RSVP-TE Auto-Bandwidth: Practical Lessons Learned. https://www.nanog.org/sites/default/files/tues.general.steenbergen.autobandwidth.30.pdf. (2013). Accessed: 2017-10-31.

[43] Pete Templin. 2006. MPLS Traffic Engineering. https://www.nanog.org/meetings/nanog37/presentations/pete-templin.pdf. (2006). Accessed: 2017-10-31.

[44] Ashish Vulimiri, Philip Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. 2013. Low Latency via Redundancy. In *Proc. ACM CoNEXT*.

[45] Srinivas Vutukury and JJ Garcia-Luna-Aceves. 1999. A simple approximation to minimum-delay routing. In *ACM SIGCOMM Computer Communication Review*, Vol. 29. ACM, 227–238.

[46] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. 2006. COPE: Traffic Engineering in Dynamic Networks. In *Proc. ACM SIGCOMM*.

[47] Zheng Wang and Jon Crowcroft. 1990. Shortest path first with emergency exits. In *ACM SIGCOMM Computer Communication Review*, Vol. 20. ACM, 166–176.

[48] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2010. Design, implementation and evaluation of congestion control for multipath TCP. In *Proc. USENIX NSDI*.

[49] Jin Y Yen. 1970. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quart. Appl. Math.* 27, 4 (1970), 526–530.

[50] Wenxuan Zhou, Qingxi Li, Matthew Caesar, and P. Brighten Godfrey. 2011. ASAP: A Low-latency Transport Layer. In *Proc. ACM CoNEXT*.