TANGO: Secure Collaborative Route Control across the Public Internet

Henry Birge-Lee, Sophia Yoo, Benjamin Herber, Jennifer Rexford, and Maria Apostolaki, *Princeton University*

https://www.usenix.org/conference/nsdi24/presentation/birge-lee

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

TANGO: Secure Collaborative Route Control across the Public Internet

Henry Birge-Lee
Princeton University
birgelee@princeton.edu

Sophia Yoo
Princeton University
sophiayoo@princeton.edu

Benjamin Herber
Princeton University
bherber@alumni.princeton.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

Maria Apostolaki
Princeton University
apostolaki@princeton.edu

Abstract

As the demands of modern latency-critical applications grow, major service providers are seeking to meet those demands by expanding their infrastructure to the edge and offering global connectivity through private WANs or Network-as-a-Service solutions. Unfortunately, these approaches are costly for smaller edge networks and lead to Internet consolidation. Worse, since the public Internet suffers from limited visibility and control over interdomain routing, smaller edges today are left with poor alternatives outside of joining the hypergiants. As a new alternative, we introduce TANGO, which enables smaller edges to expose paths and exert route control over the public Internet *without relying on third parties or cooperation from the Internet core*, to dynamically meet the performance needs of their customers. We show that, using *collaboration*, TANGO edges can jointly *(i)* expose more BGP-compliant wide-area paths via coordinated BGP advertisements; *(ii)* collect fine-grained, trustworthy telemetry using cryptographically-protected custom headers; and *(iii)* dynamically reroute traffic in the data plane. TANGO innovates in both the control and data planes, and runs on a programmable switch or in eBPF. Our Internet-scale experiments uncover rich path diversity, exposing paths that outperform the default BGP path 75-100% of the time for 20 edge pairs across multiple continents, while reducing latency by up to 39% compared to the default.

1 Introduction

Modern networked applications, from self-driving cars to online gaming and video conferencing, have strict requirements of high reliability and low latency [11, 34, 41, 54]. To satisfy these needs, hypergiants continually expand their private network infrastructure closer to the edge, effectively optimizing client experience. For instance, Google not only operates multiple Points of Presence (PoPs) globally, connecting data centers to the rest of the Internet via peering, but also partners with ISPs to deploy Google-supplied servers inside the ISP networks [37]. Similarly, Azure proposed deploying physical infrastructure inside enterprise premises to optimize ingress traffic [43]. Edge networks such as enterprises and small clouds, however, are unable to continuously expand their infrastructure and are forced to resort to alternatives such as Network-as-a-Service (NaaS) [7] from global cloud providers, outsourcing their traffic management while inheriting third-party practices and security/privacy policies. This private-WAN trend leads to increased industry consolidation, benefiting larger companies and well-served regions while leaving smaller edge networks with limited negotiation power, reduced growth opportunities, and increased vulnerability to outages [40, 75]. The natural question arises: *Is moving to private infrastructure the only way to meet growing application requirements, or can the public Internet rise to the challenge?*

To answer this, we must first understand the obstacles preventing an edge network from extracting more performance and route control in today's public Internet. Edge networks have very limited available path diversity. BGP (*i.e.,* the default Internet routing protocol) selects a single path per destination prefix based on crude (often performance-unaware) criteria [82]. While multi-homed ASes can optimize the first hop of their path [9, 10, 33], they are unable to tap into the Internet's full path diversity without collaboration from the Internet core. SD-WAN solutions [1, 2, 7] that combine multi-homing, overlay routing, and multicast techniques are still limited to BGP-default paths. Even assuming an edge network could forward traffic via adequately distinct paths, identifying performance opportunities requires accurate and trustworthy monitoring, which is impossible in practice for a single edge. Indeed, measurements at the hosts [53, 55] or even at the border of a stub network [15, 35] are affected by the performance of both the forward and reverse path and are inflated by the load on the receiver's access network, their hardware, and even the application itself, hiding performance differences between paths. Active probing might avoid some of this noise, but probes can be preferentially treated, hence unreliable [18]. Worse yet, some ASes might attempt to fool any measurement infrastructure to attract more traffic or hide their outages.

To overcome these obstacles, we present TANGO, an edge-to-edge route-control scheme that relies on *cooperation* between pairs of edge networks (*e.g.,* enterprises and data centers). We ob-

serve that collaboration between edge networks occurs naturally in today's Internet, either among sites of the same organization or between pairs of enterprises, creating many real-world opportunities for TANGO. TANGO exploits this collaboration to expose multiple routes per destination that are already installed in core routing tables (but not used by BGP) by advertising multiple prefixes for the same destination. While TANGO edges cannot explicitly change how on-path routers forward traffic, they can remotely guide the propagation of BGP advertisements for each prefix via surgical use of BGP communities and path poisoning, in an *automated manner and with no prior topology knowledge*.

TANGO also performs accurate and trustworthy wide-area monitoring between two edges, another building block towards reliable and real-time route control. Edges can operate TANGO nodes at their border gateways to piggyback telemetry information (metadata) on every packet at the sending edge [42]. This metadata is then stripped away at the receiving edge, before the packet is forwarded to its destination. In this manner, TANGO edges obtain accurate *wide-area*, *one-way* measurements unpolluted by reverse path metrics, noisy access networks, or application glitches. Further, relying on metadata makes TANGO *protocol-agnostic* (does not rely on TCP semantics) and scalable (does not keep per-flow or packet state [47, 79]). Most importantly, since TANGO operates over the untrusted public Internet, it provides *trustworthy* telemetry using shared book ciphers and secure OTP-protected route updates directly in the data plane.

TANGO has potential for many modern use cases. For example, a cloud provider without its own private WAN can run TANGO across the Internet between its data centers for dynamic route control. Emerging online gaming services can establish agreements with remote edge networks to optimize latency and jitter for their customers over the Internet, without investing in on-premise infrastructure or pairing with a cloud provider [80]. Edge networks leveraging federated learning to train models without sharing data with each other or with a cloud provider [19] can use TANGO to optimize communication between each participating network and the parameter server, which is often a bottleneck [46] or even security hazard [63]. Finally, datacenters running miners, validators, or decentralized exchanges can use TANGO to improve their pair latency and path diversity (hence their performance and security [13, 14, 29, 48, 60, 71]), without sacrificing their decentralized nature by moving to a single cloud.

As an end-to-end system, TANGO allows edges to optimize interdomain traffic according to their desired objectives, providing them with the means to *(i)* forward their traffic through paths they did not know existed; *(ii)* accurately measure relative loss and delay even in the presence of adversaries; and *(iii)* securely reroute traffic in real-time. This work does not seek to innovate on the *path-selection* algorithm, nor does it make claims on its stability and optimality. In fact, we find that TANGO can yield non-trivial benefits for TANGO edges in the wild even with a simple control loop, *e.g.,* selecting the path that maintains significantly lower latency for over 100ms. While, in theory, greedily optimizing routes based on local preferences might impact path conditions, we believe that in practice, independent edge pairs are less likely to affect other traffic by congesting links, especially compared to alternative large cloud systems with heavier traffic loads [43, 61, 76].

To reap all these benefits, two cooperative edges only need to *(i)* deploy lightweight TANGO logic at their border gateway, which controls routing between the cooperative edges and *(ii)* have access to a BGP speaker which can advertise a set of prefixes. While TANGO is highly deployable, since it can run on either a programmable switch or with eBPF, its modular design further eases adoption. In fact, each or a subset of the components can be independently deployed and directly benefit adopters. For example, TANGO's trustworthy telemetry scheme can be independently used to reliably measure loss and delay, verify service level agreements (SLAs), or detect violations of network neutrality. Similarly, clouds and distributed enterprises can use our TANGO's path-finding algorithm alone to expose paths they did not know existed.

In our ethically-conducted Internet-scale experiment between 23 pairs of TANGO nodes in globally-distributed Vultr data centers [5], TANGO's automated path discovery tool exposed 3-12 distinct paths beyond the BGP default. Interestingly, for 20 pairs, one or more TANGO-uncovered paths outperformed the default for 75-100% of the time, with some improving one-way latency by up to 39% (§6.1). Meanwhile for 6 pairs, an alternative path improved latency by at least 20% or more for over 10 hours on average. We also estimated with large-scale simulations across 999,000 randomly chosen pairs in the Internet topology that TANGO can expose at least two new paths for 98.6% of tested pairs, without collaboration from the Internet core (§6.2). Finally, we ran TANGO end-to-end between a switch and eBPF deployment on two continents, showing the practicality and performance of our real-time routing control in the wild (§7).

Contributions Beyond a Preliminary Workshop Paper: This work builds on our previous position paper that outlined the pressing need for edge networks to control their Internet routing and presented the high-level idea of cooperative routing [21]. This paper extends our earlier work by *(i)* developing an *automated* mechanism for identifying and exposing paths with no topology information; *(ii)* designing and implementing a data-plane mechanism for reliable monitoring that malicious on-path attackers cannot deceive; *(iii)* conducting an Internet-scale measurement study showcasing the benefits of real-time route control, in addition to a large-scale simulation showing the route diversity TANGO can expose.

2 TANGO Problem Setting

In this section, we explore important challenges of optimizing routing in today's Internet and highlight key requirements for a secure and practical interdomain route-control system.

2.1 Challenges of Today's Internet

Lack of Route Control: Despite the rich path diversity of the Internet (§6.2), the default interdomain routing choices of an edge network are limited to its direct neighbors. With standard BGP, each AS only exposes one path to each neighbor independently of performance, and so a single-homed network has no choice beyond the *single* BGP route its direct provider offers for each destination IP prefix. Meanwhile, a multi-homed network might only select among very few providers, which can have common bottlenecks, making such solutions limited [10, 15]. Any source routing protocol or multipath extension to BGP requires the participation of multiple ASes, making deployment difficult. Similarly, MPTCP only operates once paths are exposed, and it is protocol-specific.

Inaccurate Measurements: Collecting accurate performance measurements that are suitable for comparing wide-area paths is challenging. First, end-to-end measurements are often dominated by issues in the edge network (*e.g.,* wireless interference or local congestion) or on the end hosts themselves (*e.g.,* an overloaded machine) which, in essence, add noise to the wide-area path performance. Second, bidirectional metrics such as round-trip time (RTT)—whether collected by end-hosts or by network devices—are hard to decompose into separate metrics for the two one-way paths. Instead, separate measurements are required for path selection which is naturally one-way. Finally, measurement strategies often rely on protocol semantics (*e.g.,* TCP sequence and acknowledgment numbers), which do not generalize to all traffic, *e.g.,* QUIC [44], thus reducing the chances of reliable passive measurements or even ignoring the performance of certain protocols.

Untrusted Network: Route control over the public Internet (*i.e.,* an untrusted network) requires consideration of on-path adversaries. On-path adversaries may try to fool the monitoring infrastructure (or any data-driven system [49]) for monetary gains (*e.g.,* to attract more traffic to their path to generate higher revenue, perform traffic analysis, or hide poor performance to avoid SLA violations) [18]. Secure monitoring using cryptography at scale is very challenging in today's networking hardware.

Poor Deployability: The many proposals for optimizing Internet routing are notoriously hard to deploy in practice, creating a pressing need for low-cost and readily deployable solutions. Existing approaches often require core networks to run a new variant of BGP [26], deploy additional overlays [8], or run an entirely new routing protocol [56]. For instance, public overlay networks (*e.g.,* RON [12]) and future Internet architectures (*e.g.,* SCION [56]) require worldwide deployment, extra infrastructure (with associated costs), end-to-end coordination, and overheads for software processing on end-hosts.

2.2 TANGO Design Requirements

To overcome these challenges in the wide-area setting, we propose TANGO, a platform allowing edge networks (*e.g.,* small data centers and enterprises) to optimize interdomain routes. The following constraints drive TANGO's design.

Incentive Compatibility (§4, §5.1): TANGO should only expect cooperation from edge networks that actively benefit from routing optimizations (*i.e.,* source and destination networks of exchanged traffic). Thus, it should be transparent and should not rely on support from ISPs or intermediate ASes in the Internet core. In addition, the entry-level investment for individual networks to use TANGO should be minimal.

Plug-and-Play Control (§4): TANGO should enable edge networks to leverage Internet path diversity without requiring them to have multiple providers/peers, knowledge of the wide-area topology or expertise in advanced routing techniques. Observe that private-WAN approaches typically rely on highly connected PoPs and knowledge of the intermediate topology to control routing over available paths [43]. This is impractical for smaller edge networks connected over the Internet, where only incomplete topology approximations are available [6, 38]. Moreover, TANGO must *automatically* discover and expose paths *without* requiring manual input from the operator.

Accuracy & Timeliness (§5.1, §5.3): TANGO should allow participating networks to accurately measure paths and react in a timely manner to changes, dynamically choosing different routes based on collected performance measurements.

Trustworthiness (§5.2): TANGO should be robust against adversaries attempting to influence routing decisions by making a path appear more performant than it is. Concretely, we assume adversaries can intercept (and thus modify) packets on at most $n-1$ of n paths, and they can observe (eavesdrop) on any path. For more details on manipulation and replay attacks possible under our threat model, we refer readers to §A.

3 TANGO Overview

Using an intuitive example, we describe key insights and innovations TANGO employs to satisfy the above requirements.

Example: Consider an enterprise network ASX in Fig. 1 that wants to *temporarily* offload real-time computing of user information to a small cloud in ASY. This cloud is particularly reliable and meets ASX's computing needs, but does not operate an edge close to ASX. Despite the existence of alternative low-latency paths from ASX to ASY, the BGP default path via AS1-AS2-AS3 incurs prohibitively high tail latency for ASX's real-time needs. ASX could benefit from using the cloud in ASY, if only it could forward its traffic via one of the alternative paths (*e.g.,* via AS5). Unfortunately, under BGP, ASX cannot use an alternative path, and relying on an SD-WAN raises concerns with privacy and cost.

TANGO edges discover new paths and tunnel traffic over them: TANGO exposes path diversity by treating IP prefixes as *routes* (as opposed to distinct destinations), and using unique prefixes to reach the same destination via distinct paths. To

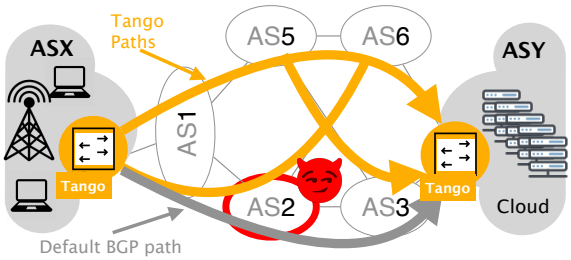


Figure 1: *ASX (enterprise) and ASY (cloud) communicate over the BGP default (gray) path via AS2-AS3.* TANGO *exploits collaboration between ASX and ASY to expose additional paths (orange).*

expose distinct paths, TANGO edges collaboratively use advanced BGP routing techniques available in today's networks (e.g., path poisoning and BGP communities). Note that TANGO *automatically* constructs BGP advertisements to discover paths making TANGO *plug-and-play* (§4). It also does not require the cooperation of any networks on the path other than the collaborative edges, and is thus *incentive compatible*.

As illustrated in Fig. 1, TANGO exposes three distinct paths from ASX to ASY by advertising three distinct prefixes from ASY, which are already installed in core BGP routing tables, but unused by BGP. The TANGO receiver (ASY) controls propagation of these advertisements through the Internet with BGP communities (§4). As a result, the TANGO sender (ASX), can control which path it uses to send traffic to ASY by tunneling application traffic over the preferred path (§5.1). ASY decapsulates tunneled packets and forwards them toward their final destination. Even though the TANGO sender can select a different path for each packet, the BGP announcements made by the TANGO receiver are **stable** and each statically represent a distinct path. Thus, there are no BGP updates when the TANGO sender chooses to change paths (preventing BGP route flapping). In practice, TANGO is symmetric, and both edges can optimize bidirectional traffic.

TANGO edges collaboratively measure delay and loss: Exposing path diversity is only the first step toward intelligent route control. ASX would need to monitor the performance of the four exposed paths to decide how to route traffic. TANGO provides highly accurate monitoring, as it operates at the edge of each network, avoiding access-network noise, *e.g.,* from wireless links. This gives TANGO an advantage over traditional end-host measurements that are notoriously inaccurate due to variable loss and delay within each network or probing techniques that can be deceived by ASes preferentially treating probes [18]. To passively measure delay and loss, TANGO adds the timestamp of when a packet left the sending edge network to every packet, along with a unique sequence number. Upon receipt, the receiving edge determines *(i) relative latency* between paths by calculating the difference between the time of packet receipt and the timestamp carried by the packet and comparing this with measurements from other paths; and *(ii)* loss by checking for missed segments (out-of-order TANGO sequence numbers).

TANGO offers trustworthy loss and delay measurements: A rational (or malicious) AS, say AS2 in Fig. 1, might try to fool TANGO into routing traffic through her infrastructure, not by improving the performance of her network but by compromising the monitoring or rerouting infrastructure. Since she cannot preferentially treat monitoring packets, as all packets are used for monitoring, she will try to fake lower delay by modifying the timestamps carried in the packets. Although adopting typical security primitives (*e.g.,* signatures, encryption) is challenging due to memory and computation constraints of modern high-speed hardware (*e.g.,* programmable switches), TANGO protects both timestamps and sequence numbers of each packet from tampering. To do so, TANGO leverage multiple insights. First, observe that timestamps and sequence numbers progress predictably. This allows TANGO to precompute and prepopulate signatures with more flexible, memory-rich software. Additionally, observe that adversaries want to make their path look superior and thus have nothing to gain from replaying old signatures (*e.g.,* from old timestamps). This enables TANGO to be resilient against replay attacks.

TANGO supports fast and secure route updates: While one-way measurements are collected at the TANGO receiver (ASY in Fig 1), the TANGO sender (ASX) decides which path packets will take *e.g.,* based on per-class performance objectives. Instead of sending raw or summarized measurements back to the TANGO sender (ASX), the receiver node ASY computes the best path for each traffic class according to ASX objectives and freshly collected measurements. If the newly computed best path is different from the current one, ASY issues a separate route update to the sender in the data plane, broadcasting the update packet over all paths to the sender, for increased update reliability. To prevent an on-path adversary from tampering with the reroute updates, we use one-time-pads directly in the dataplane.

We stress that the need for secure data-plane measurements is not particular to TANGO. In fact, many data-driven systems have been shown to be vulnerable to on-path adversaries [49]. Still, existing solutions focus solely on making monitoring scalable and accurate rather than secure [15, 35, 47, 79], effectively overlooking trustworthiness.

4 Unveiling Path Diversity with PATHFINDER

TANGO surpasses the limited path diversity offered with BGP by employing a novel recursive algorithm, PATHFINDER, which exposes BGP-compliant paths via *static* BGP announcements for different IP prefixes. Adhering to TANGO's design requirements (§2.2), PATHFINDER does not assume collaboration of the Internet core or knowledge of the wide-area topology. PATHFINDER runs on two TANGO edges with the minimum capability of announcing a single BGP prefix from one edge, and observing the AS path(s) for that prefix from the other edge. Increasing the test prefixes reduces the time it takes for PATHFINDER to expose all paths, but a typical duration using a single test prefix is $\approx$ 30 min. PATHFINDER will run at bootstrap of TANGO to expose paths and rerun on rare occasions *e.g.,* if the TANGO

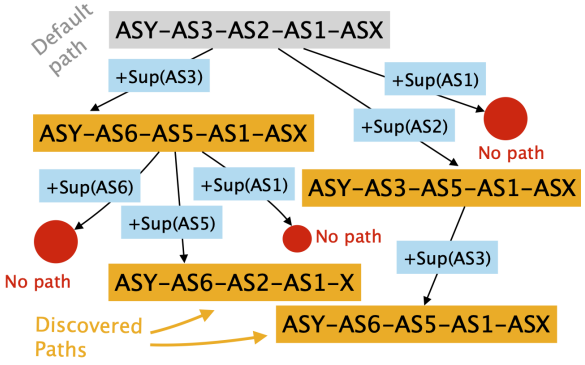


Figure 2: *Finding paths via iterative advertisement suppression.*

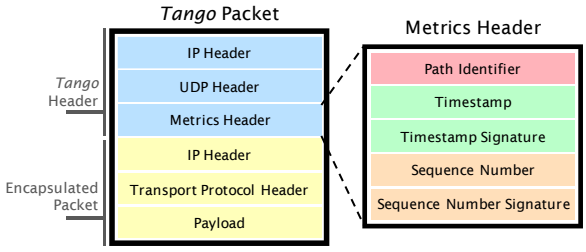


Figure 3: TANGO *tunnels application packets in an IP header specifying an interdomain path, a static UDP header to ensure consistent ECMP behavior, & a metrics header with signatures for integrity protection.*

sender receives a BGP update regarding a route exposed by PATHFINDER. Rerunning TANGO does not require TANGO to go offline as long there is at least one unused BGP prefix.

PATHFINDER exposes previously unknown paths through the Internet by advertising prefixes to the TANGO sender edge while *strategically blocking (suppressing) export of the BGP best-path*. PATHFINDER leverages two commonly supported route suppression methods: *(i)* BGP communities and *(ii)* BGP path poisoning. Community-based filtering involves attaching BGP communities supported by major transit providers (*e.g.,* those discussed in [24]) to suppress route exports (via no-export communities) or to lower route preferences so certain ASes do not use previously preferred routes. To support community-based filtering, PATHFINDER needs to be keyed with specific values of action communities supported by its upstreams and major transit providers. This can be obtained from publicly available routing guides. BGP path poisoning exploits BGP loop detection to prevent select ASes on the original path from importing the BGP announcements [22, 57]. While these techniques have different topology-dependent trade-offs[1], they are largely interchangeable from PATHFINDER's perspective, both accomplishing the algorithmic objective of suppressing a given BGP route.

PATHFINDER recursively updates the BGP advertisements announced by one edge (the destination) based on real-time feedback from the other edge (sender). Specifically, PATHFINDER finds unidirectional paths between two nodes: a traffic source and destination. PATHFINDER starts by making a "default" BGP announcement from theTANGO destination. This announcement reaches the TANGO source, which records the AS-path associated with this announcement. Note that without PATHFINDER, this would be the default and only BGP path available to the sender. Next, PATHFINDER *suppresses* the propagation of this route (using communities or path poisoning) to every AS on the recorded AS-path, effectively forcing the advertisement to find a different path to the source. For each path it finds, it recursively applies this algorithm.

[1]BGP communities can suppress individual links but are not honored over provider-customer or peer-peer links while AS-path-poisoning only suppresses at the AS granularity but affects the entire path.

As an illustration, to discover the paths in Fig. 1, ASX and ASY would need to jointly construct a graph like that in Fig. 2. In this graph, nodes contain BGP paths between ASX and ASY. The root node represents the default path, while leaves capture paths discovered by PATHFINDER by supressing ASes in the edges of the path to the node. Red dots represent advertisement attempts that did not result in a path (*i.e.,* the advertisement did not reach the sender)[2]. Recall from Fig. 1 that the default path from ASX to ASY is via ASX-AS1-AS2-AS3-ASY (advertisements are propagated via ASY-AS3-AS2-AS1-ASX). To discover additional paths, ASY will propagate multiple routes starting from one that suppresses the propagation to AS3. By poisoning AS3, the advertisement will follow the path via ASY-AS6-AS5-AS1-ASX, and thus this will be the route that ASX hears. Next, ASY suppresses AS6 in addition to AS3 which results in no available path to ASX (ASX will hear no route), as all paths traverse either AS3 or AS6. Having fully investigated routes that suppress AS3, PATHFINDER backtracks to advertising a new route suppressing AS2 and then AS1.

The algorithm described above finds paths between senders and receivers that are distinct at an AS level. Thus, PATHFINDER does not leverage path diversity *within* each AS, or the existence of multiple peering locations for an AS pair. These paths could be found by combining PATHFINDER with Paris Traceroute [16]) and could be used by TANGO by applying the source and destination port combos found by Paris Traceroute in the outer UDP header of the TANGO packets (the outer headers on TANGO packets are ignored by the receiving switch). However, the use and exploration of additional intra-domain paths is out of scope.

5 Secure, Metrics-Informed Dynamic Routing

With newly-exposed path diversity, TANGO can dynamically route traffic along paths best suited for given performance objectives, while being informed by fine-grained metrics. There are several challenges to accurate and trustworthy monitoring and rerouting. TANGO overcomes them with custom monitoring

[2]The graph is for illustrative purpose only and is constructed based on the information that PATHFINDER extracts (*i.e.,* is not part of its input).

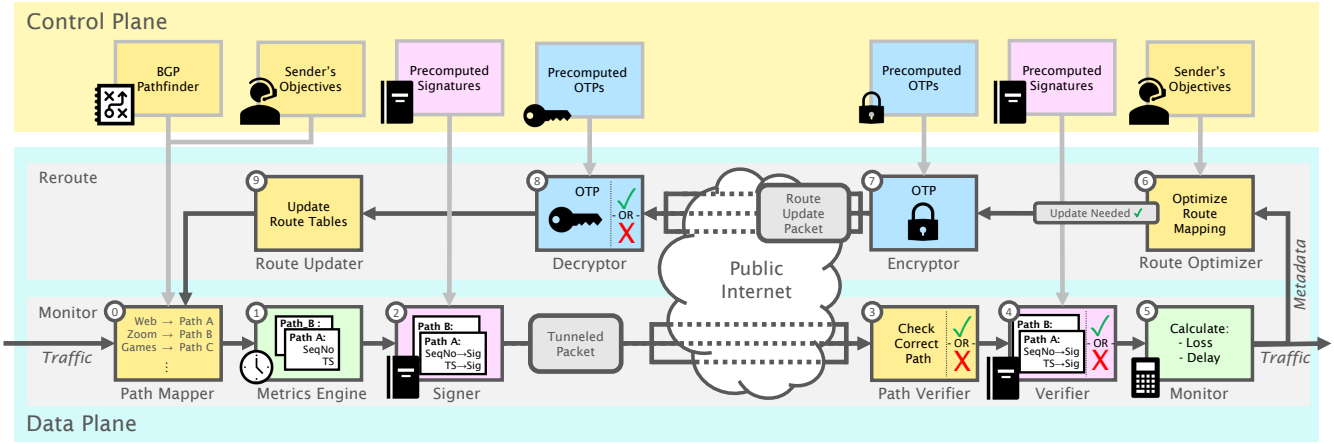


Figure 4: TANGO *is a hybrid control and data-plane approach to performant routing over the public Internet, informed by fine-grained, trustworthy telemetry.* TANGO *relies on cooperation: both edge networks share secret keys and per-class performance objectives.*

of one-way metrics (§5.1), trustworthy telemetry (§5.2), and real-time, tamper-proof route updates (§5.3). Fig. 4 illustrates the life cycle of a packet through each of these modules from a TANGO sender (left) to a TANGO receiver (right). We assume that TANGO routes traffic at the granularity of *traffic classes* to satisfy application-specific requirements.

5.1 Multi-Path Monitoring

Tunneling for Multi-Path Routing ⓪: TANGO tunnels packets through different physical paths by encapsulating them in a distinct per-path header (IPv4 or IPv6 headers) as shown in Fig. 3.[3] To route traffic according to its operator-designated traffic class objectives, TANGO maintains a mapping from class to path identifier (PID), which uniquely designates a path, specified by an IP header.

Custom Per-Path Monitoring ①⑤: In addition to the tunnel header, TANGO adds custom header fields to enable per-path performance monitoring [42]. Unlike prior works that rely on TCP semantics to measure performance [15, 28, 51, 62], TANGO adds a custom *Metrics Header* which contains a 3-bit PID, 12-bit timestamp, 32-bit PID and timestamp signature, 8-bit TANGO sequence number, and 1-bit sequence number signature. The *Metrics Header* is lightweight (7 bytes overall) and used to identify packet routes, calculate loss and delay, and facilitate trustworthy telemetry.[4] The timestamps and sequence numbers are defined as follows:

1. **Timestamps:** The TANGO sender tags a packet with its local time (t_1) in ms before tunneling it to its peer. Upon receipt, the peer node records its local time (t_2) and calculates the per-packet delay as $t_2 - t_1$. The timestamp carried by packets is at the ms granularity. Since Internet path latencies are on the order of tens of ms, more fine-grained measurements would not yield increased benefits. Also, since TANGO measures relative latency across paths, clock skew between nodes is irrelevant.
2. **Sequence Numbers:** The TANGO sender also tags packets with a monotonically increasing sequence number (*s_curr*) before tunneling. The receiver tracks the highest sequence number seen (*s_seen*), calculating loss as $s\_curr - s\_seen$. As TANGO sequences are not re-transmitted if dropped (unlike TCP sequence numbers), each dropped packet is only counted once. Out-of-order packets increase loss but should be rare among packets with a fixed header traveling edge-to-edge.

Aggregated Monitoring ⑤⑥: TANGO calculates per-path loss and delay metrics over an aggregation window of size *i*, by adding measurements to an aggregate sum. Upon the arrival of the *i*-th packet, the current sum will be persisted for later use in issuing reroutes, and the value will be reset. The operator may configure the window size, weighing tradeoffs in noise-resilience, network event response time, and security.

5.2 Secure Telemetry

Trustworthy Telemetry ②③④: To protect against tampering, the TANGO sender cryptographically signs all timestamps, PIDs, and sequence numbers, ensuring integrity and authenticity. Upon receipt, the TANGO receiver verifies each value to ensure they are untouched after transit over the public Internet. Signatures are path-specific, and hence secure against replay attacks: an on-path attacker cannot replace signatures with those they overhear from faster paths, as the signature is specific to the path (PID). Finally, any mapping (*i.e.,* pair of timestamps, sequence number, or PID to signature) that the adversary learns cannot be reused. The

[3] The TANGO header adds several bytes to each packet, which could potentially cause MTU issues. TANGO can resolve this in a similar way to other router-based encapsulation protocols by implementing Path MTU Discovery and responding with appropriate ICMP "fragmentation needed and DF set" packets, as specified in RFC 1191 [50].

[4] We developed these header sizes based on the parameters of the testbed we used. Even under more demanding production conditions, the anticipated header size will still be quite small compared to overall packet sizes.

sequence number changes per packet. The timestamp changes every ms and older timestamps are not useful as the attacker only tries to make her path appear faster.

Practical Challenges for Scalability: Cryptographically protecting metrics at packet line rate is nontrivial. Cryptographic primitives are resource-intensive to implement at scale. First, signing packets in the control plane requires laborious per-packet processing in software, while signing packets in the data plane would require multiple recirculations [27, 77, 78]—both too compute-intensive. Second, storing *precomputed* signatures directly on switch SRAM is too memory-intensive. Further, populating the data plane with precomputed signatures from the control plane creates read-write synchronization concerns between the data and control planes, as well as between TANGO sender and receiver.

The TANGO Signature Book ②④: To avoid online signature computation while still being memory-friendly, TANGO creates a *signature book* of precomputed signatures, and periodically populates smaller precomputed *signature blocks* from the signature book to the data plane. The signature book is used for two operations: signing (on the TANGO sender) and verifying (on the TANGO receiver). The following insights help us deal with the memory challenge. First, we observe that both sequence numbers and timestamps are predictable (monotonically increasing) over time, enabling efficient pre-computation and storage. Second, we observe that to reduce delay the adversary would only need to guess one timestamp per ms, while to hide a single packet drop she would need to correctly guess many consecutive signatures. Further, sequence numbers will eventually wrap around, thus making old mappings irrelevant to the adversary. Hence, TANGO uses 32-bit timestamp signatures and 1-bit sequence number signatures.

To deal with synchronization challenges, TANGO employs two strategies. First, it uses data-plane packets triggered by the control plane to quickly write signatures, instead of relying on control-plane write calls (*e.g.,* with gRPC). The control plane marshals precomputed signatures into large packets, which trigger block writes when processed in the data plane. Switch-specific constraints do not allow writing multiple indexes of the same register array, so signatures are sliced off packets in the data plane, which are recirculated until a null token is reached. This approach allows the control plane to update signatures faster than they are consumed by the data plane.

Second, TANGO splits its signature book into two blocks, where at any given time, one block is being written and one is being read, avoiding a race condition between reads and writes (see Fig. 5). This approach makes the sequence-to-signature mappings time-dependent and thus deterministic. Critically, however, sequence numbers are not reset for every new interval. While resetting would be more economical from the memory perspective, it would also allow the attacker to silently drop all packets at the end of each interval. Instead, TANGO stores the maximum signatures that can be consumed in each interval and ignores unused signatures. In other words, through every interval change, the sequence number continues to increase from its last value (Fig. 14 in §B).

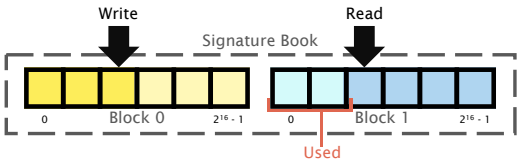


Figure 5: *To deal with synchronization issues between reads and writes,* TANGO *splits the signature book into two blocks, for reading and writing. The control plane writes fresh signatures to one block, while the data plane consumes the other for signing packets.*

5.3 On-Demand Reroutes in the Data Plane

Dynamic Route Updates ⑥⑨: The receiving TANGO node issues route updates to the sender based on aggregated path metrics and the sender's objectives for each traffic class. As reroutes should be infrequent, TANGO does not include these updates in the default TANGO header, rather opting for a custom packet containing the route update (*i.e.,* the new PID for the corresponding traffic class).

Trustworthy Route Updates ⑦⑧: Naturally, a route update could be the target of an on-path adversary that *(i)* tampers with route updates to direct traffic to desired paths; *(ii)* injects route updates to move traffic from a path; or even *(iii)* drops route updates to cause a denial-of-service to the reroute mechanism. Observe that the first two attacks are catastrophic for TANGO, since they have a direct effect on routing. Thus, to prevent tampering, the TANGO receiver (which issues the updates) encrypts route updates before transmission, while the original sender decrypts and verifies the update before applying. To also protect against dropping of route updates, TANGO broadcasts reroute packets over all available paths to ensure that at least one update will reach the sending node.[5]

OTP and Encoding Scheme: There are several challenges to encrypting updates: *(i)* limited compute and memory in the data-plane; and *(ii)* susceptibility of small ciphertexts to brute-force attacks. TANGO solves these problems by using precomputed OTPs with extension encoding scheme. This is a natural solution as route updates are relatively infrequent yet unpredictable. However, since the number of possible reroutes increases with both the number of paths and the number of traffic classes, storing route-update signatures could be memory inefficient. Most importantly, compromised security of route updates would be catastrophic even for a single packet, necessitating the perfect secrecy of OTP.

Concretely, a route update consists of an index that selects the OTP to XOR with the concatenated traffic class and PID (see Fig. 15 in §B). The control plane periodically updates OTPs, such that they are only used once. While OTP offers perfect secrecy, it is vulnerable to bit flips, meaning the attacker can change a few random bits in the traversed route update to change TANGO

[5] Threat model assumes at least one path with no on-path adversary (§2.2).

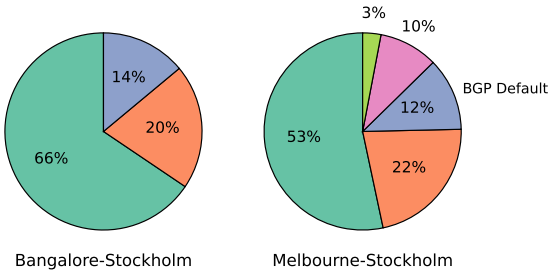


Figure 6: *Breakdown over 32 hours of how often a path had lowest relative latency. Here, the BGP default is beaten by alternative paths 100% of the time (Bangalore) and 88% of the time (Melbourne).*

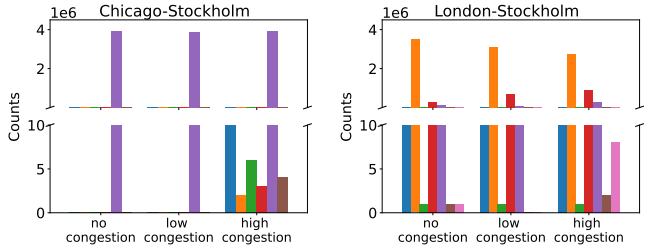


Figure 7: *Number of times different paths emerged as the best for a given congestion condition, per node pair. "no", "low", and "high" congestion conditions correspond to the lowest, middle, and upper tertiles, respectively, of one-way-delay experienced by the best path. While one path might dominantly be best under no/low congestion, during high congestion more paths emerged as the best path.*

behavior. To protect against this, TANGO independently encodes the traffic class and PID into 32-bit, sparse strings. The value to encode (*e.g.,* traffic class) selects a static, sparse-bit string and appends it to form a single, 32-bit string. The concatenation of the encoded bit strings are XOR'd with the OTP, and sent to the peer TANGO node alongside the update number. If an adversary tampers with the update number, traffic class bits, or the PID bits, the decrypted update or the encodings will be incorrect, and the update can be safely ignored. If the adversary were to now try and brute-force the update, they must now brute-force all encoding bits, which are tied to encoded values.

6 Internet-Scale Measurements

We showcase TANGO in the wild by deploying it with eBPF [17](§7) on 25 nodes in globally-distributed data centers of the cloud provider Vultr [5] (see Fig. 26 in §C). Through our Internet-scale measurement study (§6.1), we prove that TANGO uncovers *path diversity* even in a single-homed environment, where conventional BGP only offers a single path. We also show that TANGO uncovers *performance diversity* in these alternate paths, which often outperform the default BGP path by up to 100% of the time with up to 39% lower latency. To further validate and generalize these results, we also perform an Internet-scale simulation that confirms the rich path diversity available in the public Internet (§6.2). Together, these experiments showcase the benefit, practicality, and incremental deployability of TANGO.

Ethics Statement: We note that all testbed infrastructure in edge networks of our Internet-scale study are operated by the authors, and that traffic sent from TANGO nodes across the public Internet are transparent to intermediate ASes and can be processed as normal application traffic at reasonable sending rates (up to 1Mbps), raising no ethical issues.

6.1 Operational Deployment

Choice of Edge Network. We perform our measurement study with Vultr as a deployment convenience, however, many other types of edge networks can easily benefit from TANGO. Characteristics that would make an edge network particularly amenable to TANGO include having available prefix space (either IPv4 or IPv6) and either BGP communities *or* path poisoning capabilities. Note that the edge network does not require a provider that actively *engages* in BGP community announcements, as long as they are able to *transit* communities. This is often the default behavior in ISPs [68, 69].

Methodology: To measure available path diversity, we ran PATHFINDER on all bidirectional paths between 24 of the Vultr data centers (552 pairs in total)[6]. Vultr allows customers to make BGP announcements, supports several action communities (to control its own BGP behavior), and transits BGP communities (to potentially impact the behavior at remote ASes) [3, 4], but does not allow customers to do path-poisoning. Thus, we ran PATHFINDER using community-based suppression, inputting BGP suppression communities supported by Vultr as well as by several major transit providers (specifically ASes 3257, 6453, 4755, 3356, 1299, and 174).[7] While Vultr does export a full BGP route table to customers participating in its BGP services, it only provides a single next-hop at each datacenter: the Vultr upstream router. Thus, *by default, each source-destination pair would only have a single path, without* TANGO.

We also used TANGO to measure performance diversity between 23 Vultr pairs, with Vultr Stockholm fixed as the receiver. Specifically, we generated 1Mbps `iperf` UDP flows across 7 different TANGO-exposed interdomain paths for all 23 pairs, passively measuring latency and loss at 10 ms measurement intervals over a period of roughly 32 hours. We also repeated measurements with an additional 23 node pairs, using Vultr LA as the fixed receiver, and found similar results (§C). We present results for Stockholm measurements below.

6.1.1 Path Diversity over the Public Internet

How much path diversity can TANGO expose? Of the 552 node pairs explored, PATHFINDER exposed alternative paths for *503 pairs (91%), unveiling opportunities to benefit from multi-path routing*. The median number of paths available between two

[6] During PATHFINDER exploration, only 24 of 25 nodes were available.

[7] As there are no standard values for many BGP route suppression communities [24] these values had to be found by hand from routing guidelines.

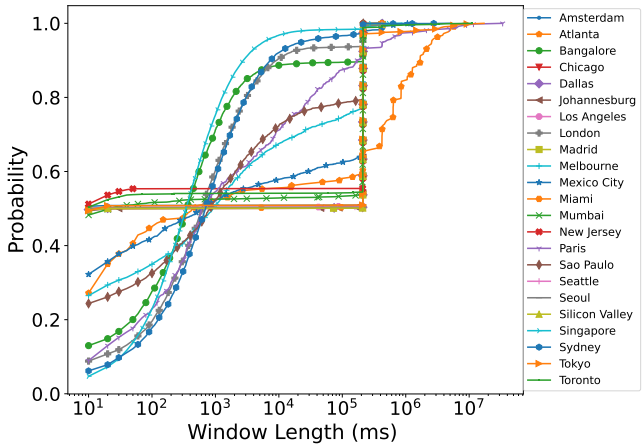


Figure 8: *On average across all 23 pairs, best paths lasted from 5-541s, with median durations of 10ms-105s.* TANGO *only chooses to perform dynamic rerouting for windows longer than 100ms.*

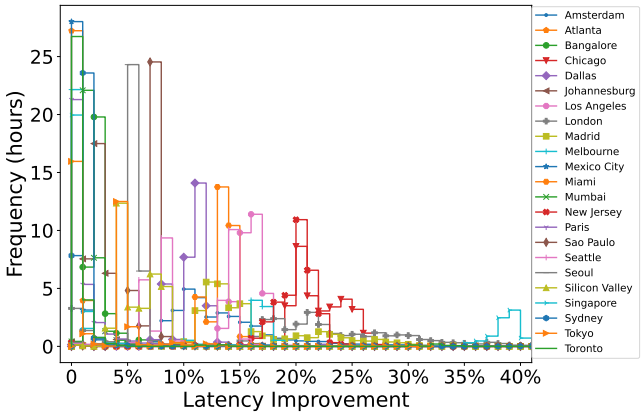


Figure 9: *Best paths that* TANGO *exposed outperform the BGP default by up to 22% on average, and up to 39% for some pairs.*

nodes was 3, and the average was 3.3. PATHFINDER also uncovered a sizeable long tail, as 84 different pairs had 6 or more paths and 3 pairs had 12 different paths between them (see Fig. 16 in §C). The average run time for PATHFINDER was quite low (<1h) as each pair only required 6.8 BGP announcements on average (with 5-min separations to account for propagation). While already encouraging, the number of paths can be significantly increased if Vultr were to allow BGP path-poisoning. Unlike community-based suppression which is only supported by some ASes and varies in implementation from AS to AS, AS-path poisoning is mandated by the BGP RFC [59], although some networks filter announcements with certain ASes poisoned [66].

6.1.2 Performance Diversity across Exposed Paths

How often are TANGO-discovered paths better than the default? We define the best path to be the one with lower relative one-way-delay compared to all other paths. Of the 23 pairs in our Vultr Stockholm measurements, *20 pairs had an alternative path that outperformed the default BGP path for a significant amount of time*: 100% of the time for 15 pairs, and 75-88% of the time for 5 pairs. This clearly shows the benefit of deploying TANGO. Of the remaining three pairs (Atlanta-Stockholm, Paris-Stockholm, Tokyo-Stockholm), the default path was dominantly best for only two pairs and was best only 55% of the time for the last pair. We also note that there were many pairs with *more than one* alternative path which outperformed the default, providing further path diversity and potential performance resilience: 7 pairs had two or more alternatives and 4 pairs had three or more alternatives. In Figs. 6 and 7, we visualize results for two node pairs, showing alternative paths outperforming the default. We include distribution charts for all 23 pairs in §C, Figs. 23 and 25.

How long is one path the best? We measured the window of time one path remained the best for each of the 23 Vultr pairs (Fig. 8). Path longevity varied across paths, likely due to performance variations from changing network conditions (*e.g.,* congestion). For 9 pairs, the median window duration was 350-925*ms* showing that dynamically updating the used path would be beneficial for forwarded traffic. For 11 pairs, the median duration was shorter at 10-30*ms*, most likely due to ECMP behavior between the nodes. The average duration for these 11 pairs ranged between 95-99*s* (from New Jersey, Mumbai), 121-221*s* (from Toronto, Tokyo), and 105-107*s* for the 7 other pairs. For the last 3 pairs, the median was much longer, lasting for 23-105*s*, while the average duration for each pair was 107s (from Johannesburg, Madrid, Chicago).

TANGO performs dynamic reroutes when a new best path emerges for longer than a given window threshold. The exact threshold can be configured per pair using such measurements. For instance, with this threshold, 10-30ms windows would not trigger a route update. Network operators can increase this threshold to provide more static long-term path optimizations, or further lower it for even faster dynamic updates.

By how much does the best path beat the BGP default? We measured the latency difference between the best and default paths, for each of the 23 Vultr pairs (Fig. 9). We found that for five pairs, the best path had at least 20% lower latency than the default for more than 6 hours within the 32-hour experiment. To better grasp the expected benefit of TANGO, consider that these nodes are in different sites of the same cloud. While as noted previously, there were two pairs for which the default path was best (Atlanta-Stockholm and Paris-Stockholm), for the remaining 21 pairs, *the non-default best path outperformed the default by an average of 1-22%* for durations of 0.9-24.54 *hours*. Further, the 75th percentile delay improvement was 24% for Chicago-Stockholm and 26% for London-Stockholm, lasting for 4.07*h* and 0.85*h*, respectively.

6.1.3 Event Analysis

While Fig. 9 demonstrates the benefit of running TANGO for the average-case, another major advantage of TANGO is to rapidly avoid problematic performance events by moving to other paths during disruptions. To quantify this benefit, we searched our measurements for significant performance degradation

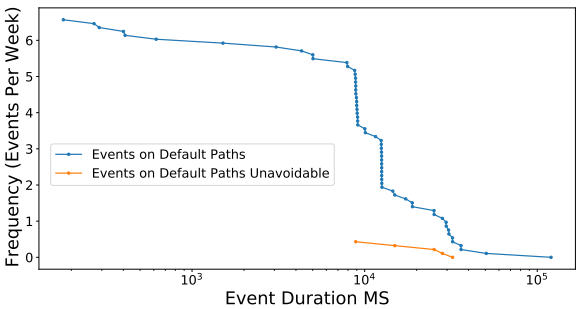


Figure 10: *Occurrence frequency vs. duration of high-loss performance degradation events on default BGP paths paths.* TANGO *could protect edges from four high-loss events on average per week.*

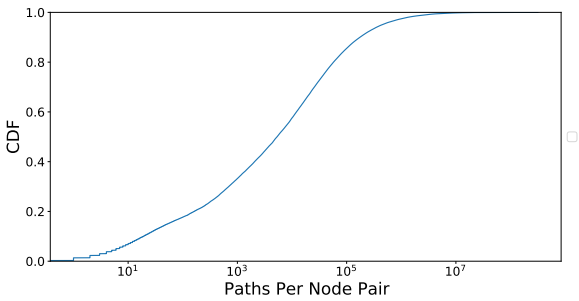


Figure 11: *The number of Gao-Rexford-compliant paths between 999k random topology node pairs.* PATHFINDER *exposed more than 2 unused BGP-compliant paths for 98.6% of pairs.*

events by filtering for 10 consecutive measurements (i.e., a 100ms window, with data points taken every 10ms apart) which were 20ms higher than the baseline one-way-delay on that path. We determined an event to be over once 10 consecutive measurements were below our event threshold. For each event, we computed the relative average one-way-delay and loss during the event. We also marked each event as avoidable if we saw there was another path between the same pair of nodes that was not experiencing a performance degradation event. We computed the frequency of events on paths by dividing the total number of events by the total duration of measurement collection.

From there, we further selected events with more significant performance degradation in loss or delay, potentially compromising real-time applications. We do expect that such events would cause customers to complain even if the average network performance is good. Concretely, we selected events with greater than 20% loss or 100% one-way-delay increase (latency events are discussed in §C.2). Fig. 10 presents a graph of event duration vs. frequency for loss events on default BGP paths (TANGO-discovered paths experienced similar events). In the absence of TANGO, events with 20% loss for longer than 8 seconds occurred over 5 times a week. Meanwhile, with TANGO-exposed paths and adaptive routing, such loss events can be reduced to *less than once a week on average*. It should also be noted that in our dataset, all unavoidable events occurred at a single sending node: Vultr Brazil (which may have seen a higher number of correlated events due to potentially less path diversity in developing regions like Brazil). *For all other nodes, every high-loss event encountered could have been avoided with* TANGO.

6.2 Internet-Wide Simulation

To understand potential path diversity across the Internet topology in an even more generalized setting, we counted Gao-Rexford-compliant [31] paths between randomly-chosen ASes in the March 2020 CAIDA Internet Topology Dataset [6]. To optimize the counting process, we separated a Gao-Rexford path into two sections: an initial part consisting of zero or more customer-provider links joined (optionally by a peer-peer link) to zero or more provider-customer links. With this in mind, for each BGP destination considered, we divided the topology into two sections: a "provider cone" that was reachable only traversing customer-provider links, and all other ASes outside the provider cone. We counted paths to all ASes in the provider cone via only customer-provider links by treating this as a Directed Acyclic Graph (DAG) based on the Gao-Rexford premise that there are no customer-provider loops [31]. We similarly counted paths from all other ASes to the source AS by treating the region outside the provider cone as a DAG but traversing provider-customer links (instead of customer-provider links). This gave us a path count to all ASes in the topology using either customer-provider or provider-customer links depending on whether the ASes were inside or outside the provider cone. Finally, we joined these two counts to form full paths by traversing each AS A in the provider cone and counting the potential paths that contained A in the last customer-provider link (the sum of the paths through ASes not in the provider cone of the peers and customers of A multiplied by the number of paths through the provider cone to reach A). This algorithm is scalabe and ensures counting distinctness, it is actually a lower bound on the number of paths in a given topology, as ASes in the provider cone can potentially be reached over peer-peer or provider-customer paths as well, which our counting methodology does not permit. Our counting is also limited by the accuracy of the CAIDA AS-Relationship dataset as is standard with many other Internet-scale simulation work [22–24, 70].

We ran our counting algorithm to count Gao-Rexford-compliant paths between 1000 randomly-chosen ASes producing 999,000 distinct source-destination pairs. We found today's Internet topology offers rich path diversity, as the median AS pair had 5,323 Gao-Rexford-compliant paths (Fig. 11). We largely attribute this richness to the high degree of certain vertices, including large IXPs, on the Internet graph; indeed, in a richly-connected topology paths grow exponentially. We, of course, do not advocate for TANGO to use more than a couple of them, yet this result shows high potential for multi-path routing across the wide area.

7 Internet-Scale Route Control

We showcase TANGO's real-time route control with an end-to-end experiment spanning two continents (North America and Europe), illustrating how TANGO leverages collected measurements to perform dynamic route control and avoid performance degradation

events (§7.1). We also present microbenchmarks that highlight the operational feasibility and efficiency of performing line-rate integrity protection with TANGO's switch prototype (§7.2).

Implementation: We implemented TANGO's data plane logic on modern switch hardware and with eBPF on standard Linux servers. Our switch prototype was implemented in 199 lines of Lucid code [67] and compiled to 1279 lines of P4 [25] targeted for an Intel Tofino programmable switch [36]. Our interchangeable eBPF version was written in 401 lines of code. Meanwhile, TANGO's control plane component and eBPF loading program was written in 524 lines. We have released TANGO's source code on GitHub[8].

Testbed: Our hardware testbed consisted of an Intel Tofino Wedge32X-BF programmable switch and several servers, each with a 20-core Intel Xeon Silver 4114 CPU and a Mellanox ConnectX-5 2×100Gbps NIC. Our sending edge network was based on Princeton University's campus in North America, with a server (running Ubuntu 20.04 and kernel version 5.4.0) sending main application traffic and generating background traffic flows, and the switch running TANGO data plane logic. Meanwhile, TANGO's receiving edge was deployed with eBPF on a standard Linux server at a Vultr data center in Stockholm, Sweden running Ubuntu 22.10 with kernel v5.19.0. The eBPF programs were built with Libbpf and Clang 15.0.6.

7.1 Dynamic Reroutes

To evaluate TANGO's end-to-end dynamic route control, we announced seven distinct IP prefixes from Vultr Stockholm via BGP, using community sets that we previously found were optimal for finding paths between Stockholm and other locations (§6.1). We did not have access to a BGP feed from our institution so we could not rerun our PATHFINDER algorithm specifically for this pair of nodes. We also noticed ECMP being used for outbound traffic from our institution. To benefit from this we explored the round-trip-time when sending traffic to different destination IPs within the same BGP prefix. We found two available ECMP paths for every announced BGP path. After enumerating these 14 paths, we observed some paths had seemingly-identical performance[9]. Ignoring redundant paths we had 12 distinct paths.

We generated keep-alive flows along all exposed paths while continually collecting per-path metrics. We also had one active flow, which we monitored for potential benefits from dynamic routing. We developed a simple control algorithm based on relative one-way-delay that would move the active flow to a different path if that path outperformed the current path for 10 consecutive measurements (i.e., 100ms for measurements taken every 10ms). We found this algorithm to be stable across the wide area, only rerouting during significant network degradation events. Once the control algorithm determined a route change was necessary, it produced a reroute data packet which was sent back to the TANGO switch node at our institution. This caused a register update on the switch, which rerouted the active flow to a different path.

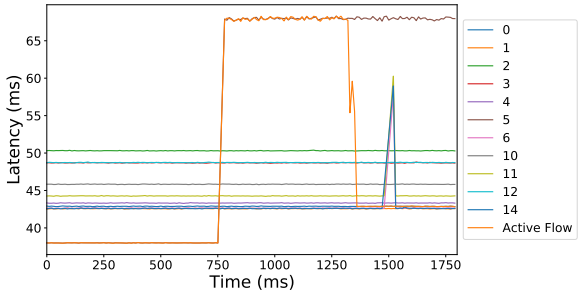


Figure 12: *One-way-delays of different paths and an active flow during our simulated performance degradation event*[10].

To observe TANGO's potential to dynamically avoid events like those discussed in § 6.1.3, we utilized another Intel Tofino switch running between our TANGO switch and the Internet to inject delay (with packet recirculations) on specific paths. To make our experiment more realistic, we replayed delay events that we had observed in the wild on paths that *(1)* had a similar one-way-delay as that of our institution to Stockholm and *(2)* had the largest number of disruptive events.

When we ran this experiment, our control algorithm started by moving the active flow from the BGP default path (path 0) to the optimally performing path (path 5). As the experiment continued, our delay mechanism began simulating a performance degradation event, causing path 5 to have its one-way-delay spike. *Within 650ms after the first sign of performance degradation, the observed active flow was moved back to path 0* (see Fig. 12). Even with a simple control loop, TANGO can reduce the duration of 90% of the high-loss events observed in §6.1.3, offering more than a 10-fold reduction in the median length (12s) of high-loss events. This response time could potentially be even faster if our prototype eBPF module generated reroute packets in the kernel as opposed to utilizing a separate user-space process (written in Python and taking approximately 400ms to initialize and execute), or if a more aggressive control algorithm was used.[11]

7.2 Data Plane Microbenchmarks

To demonstrate the operational efficiency of our integrity-protection scheme, we analyzed theoretical sizes for TANGO's signature book (§5.2), experimentally evaluating required control plane write speeds for book population. Our results prove that, for reasonable packet sizes, required block sizes refreshed every 5–15ms can comfortably be stored in the data plane and populated from the control plane (Fig. 13).

Book Block Hybrid Write Speed: We experimentally confirmed that the control plane can write an entire signature block across all

[8] https://github.com/PrincetonUniversity/tango-routing

[9] This is because we could not see the BGP path used by our institution so some BGP community combinations produced identical paths.

[10] It is likely a congestion-induced single-packet spike around 1500 ms.

[11] Our test flows ran UDP so reordering was not a prominent concern. If this rerouting algorithm is used on TCP traffic, flowlet boundaries can be taken into account to minimize overhead due to packet ordering changes.

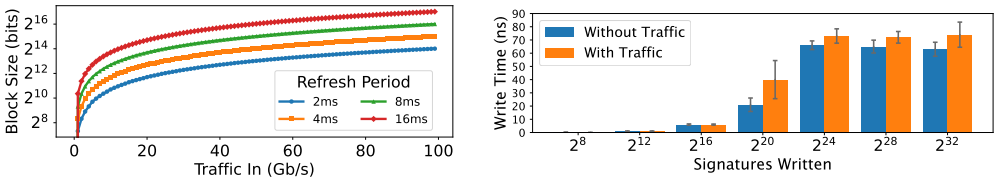


Figure 13: *Required signature block size* (left) *and write speed* (right) *reveal the practicality of a precomputed data plane signature scheme.*

books at speeds well below the refresh period, ensuring all signatures are refreshed before the data plane moves to the next block. Our first server issued control plane writes, the second generated background traffic at 100Gbps with DPDK [30], and the switch ran our slice-and-recirculate write behavior (§5.2) while tunneling out background traffic. We gathered write-time measurements by taking the time difference between the arrival of the first and last signature written to the block. As shown in Fig. 13, at a refresh period of 8ms at maximum port speed, the control plane can write 2^{16} signatures in less than 10ns over the data channel. Meanwhile, the required 2^{20} signatures for all blocks across all books can be written in approximately 20ns without background traffic and 40ns with it, while being well below the 8ms refresh limit.

8 Related Work

Performance-Driven Routing: Many prior works have explored performance routing; however, they suffer from poor deployability. Traffic engineering works such as TEXCP [39], EDGE FABRIC [61], and ESPRESSO [76] assume that the edge network is multi-homed. While EDGE FABRIC and ESPRESSO show the promise of SDN-based performance-driven routing, they can select between multiple already-existing BGP routes, thanks to global PoPs and peerings from Meta and Google. Such infrastructure is infeasible for smaller network operators and cloud providers. Other data-driven routing solutions such as BLINK [35] and SHORTCUT [64] allow fast failover in the data plane but do not improve performance. ROUTESCOUT [15] provides metric-driven dynamic routing in the data plane, but struggles to provide accurate metrics and also assumes multi-homed networks. AnyOpt [81] optimizes anycast catchment but does not advertise multiple prefixes for the same destination. PECAN [72] does advertise different routes to the same destination but only steers between them with DNS, preventing the fine-grained route control TANGO offers.

PAINTER [43] similarly uses multiple IP prefixes to advertise different routes for inbound traffic, but is designed for a cloud environment, which significantly changes the design space. First, as PAINTER runs in a highly-peered cloud, it can simply advertise distinct prefixes to different immediate neighbors, and does not attempt to find distinct paths after the initial hop (making it useless to single-homed networks). Second, PAINTER does not perform data-plane telemetry and leverages application-layer telemetry enabled by proxies running in edge networks. TANGO innovates on data-plane telemetry and does not require application proxies, while being protocol-agnostic and robust against malicious intermediate networks. Finally, TANGO is designed to be deployed in programmable data planes like P4 switches and smartNICs. PAINTER utilizes proxies running on traditional CPUs that do not have the scaling and cost benefits of data-plane hardware.

Secure Telemetry: Multiple solutions for real-time data-plane telemetry exist [45, 52, 58, 62], yet they are not designed with security in mind *i.e.,* their results could be compromised by an adversary. Moreover, they often require collaboration of all switches/routers in the path. For instance, INT (and later versions) [42, 65] collect fine-grained performance metrics at each hop, enabling informative network monitoring to operators, but require each switch to implement the protocol, which is not a reasonable assumption in the wide-area setting. A few secure telemetry solutions exist, such as Stealth Probing [18] and path-quality monitoring [32], but they do not provide the necessary fine-grained metrics for real-time, dynamic routing, and are not implementable in today's hardware.

Data-Plane Encyption: There are several general-purpose encryption schemes ported to the data plane; however, they are too resource-intensive. For example, the Advanced Encryption Standard *i.e.,* the de facto cipher for most Internet applications AES-TOFINO [27], utilizes the majority of Tofino memory resources and would need, optimally, 5 pipeline passes for each 16-byte block to encrypt. Even more lightweight cipher deployable on an ASIC such as SIMON AND SPECK [20], CHACHA [78], and HALFSIPHASH [77] are still too resource intensive. Beyond memory, they require several recirculations for every plaintext block to be encrypted. Other solutions, such as RAVEN [74] and PINOT [73], are application-specific and not easily extensible to other use-cases.

9 Conclusion

TANGO is the first route-control scheme to expose multiple wide-area paths without the cooperation of the Internet core, while offering accurate and trustworthy edge-to-edge measurements. Our Internet-wide experiments show there are significant benefits from optimizing routing on the public Internet, using TANGO-exposed paths not available with BGP. We show TANGO can run on a hardware switch or with eBPF, making it practical even for small networks.

Acknowledgments. We thank John Sonchack for his support with Lucid development. We are grateful to Ethan Katz-Bassett and Thomas Koch for their valuable feedback, and to our shepherd Italo Cunha and the anonymous reviewers for their insightful comments. This work was supported in part by Protocol Labs and by NSF GRFP Grant DGE-2039656.

References

[1] Managed sd-wan solutions for the cloud era. *https://www.aryaka.com/managed-wan-services/*.

[2] Versa networks. *https://versa-networks.com/sd-wan/cloud-wan/*.

[3] Announce your IP space with BGP and Vultr - Vultr.com. https://www.vultr.com/features/bgp/, 2022.

[4] AS20473 BGP customer guide. https://www.vultr.com/docs/as20473-bgp-customer-guide, 2022.

[5] SSD VPS servers, cloud servers and cloud hosting. https://www.vultr.com/, 2022.

[6] The CAIDA AS relationships dataset. https://www.caida.org/catalog/datasets/as-relationships/, 2023.

[7] HPE greenlake for aruba (NaaS). https://www.arubanetworks.com/solutions/naas/, 2023.

[8] Akamai. SureRoute. https://developer.akamai.com/article/sureroute, 2022.

[9] A. Akella, B. Maggs, S. Seshan, and A. Shaikh. On the performance benefits of multihoming route control. *IEEE/ACM Transactions on Networking (TON)*, 16(1):91–104, 2008.

[10] A. Akella, S. Seshan, and A. Shaikh. Multihoming Performance Benefits: An Experimental Evaluation of Practical Enterprise Strategies. In *USENIX Annual Technical Conference, General Track*, 2004.

[11] A. Alhilal, T. Braud, B. Han, and P. Hui. Nebula: Reliable Low-latency Video Transmission for Mobile Cloud Gaming. pages 3407–3417, 04 2022.

[12] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *ACM Symposium on Operating Systems Principles*, SOSP '01, page 131–145, 2001.

[13] M. Apostolaki, C. Maire, and L. Vanbever. Perimeter: A network-layer attack on the anonymity of cryptocurrencies. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I 25*, pages 147–166. Springer, 2021.

[14] M. Apostolaki, G. Marti, J. Müller, and L. Vanbever. SABRE: Protecting Bitcoin against Routing Attacks. In *Network and Distributed System Security Symposium (NDSS)*, 2019.

[15] M. Apostolaki, A. Singla, and L. Vanbever. Performance-driven internet path selection. In *ACM SIGCOMM Symposium on SDN Research (SOSR)*, page 41–53, 2021.

[16] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, page 153–158, New York, NY, USA, 2006. Association for Computing Machinery.

[17] S. authors. Suricata - eBPF and XDP. https://suricata.readthedocs.io/en/latest/capture-hardware/ebpf-xdp.html, 2018.

[18] I. Avramopoulos and J. Rexford. Stealth probing: Efficient Data-Plane security for IP routing. In *USENIX Annual Technical Conference*, Boston, MA, May 2006. USENIX Association.

[19] G. Bao and P. Guo. Federated learning in cloud-edge collaborative architecture: key technologies, applications and challenges. *Journal of Cloud Computing*, 11, 12 2022.

[20] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers. The simon and speck lightweight block ciphers. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2015.

[21] H. Birge-Lee, M. Apostolaki, and J. Rexford. It takes two to tango: cooperative edge-to-edge routing. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pages 174–180, 2022.

[22] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal. Bamboozling Certificate Authorities with BGP. In *USENIX Security Symposium*, 2018.

[23] H. Birge-Lee, L. Wang, D. McCarney, R. Shoemaker, J. Rexford, and P. Mittal. Experiences deploying Multi-Vantage-Point domain validation at let's encrypt. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4311–4327. USENIX Association, Aug. 2021.

[24] H. Birge-Lee, L. Wang, J. Rexford, and P. Mittal. SICO: Surgical Interception Attacks by Manipulating BGP Communities. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[25] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. In *ACM SIGCOMM Computer Communication Review*, 2014.

[26] J. M. Camacho, A. García-Martínez, M. Bagnulo, and F. Valera. BGP-XM: BGP Extended Multipath for Transit Autonomous Systems. *Computer Networks*, 57(4):954–975, 2013.

[27] X. Chen. Implementing AES encryption on programmable switches via scrambled lookup tables. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, SPIN '20, page 8–14, New York, NY, USA, 2020. Association for Computing Machinery.

[28] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford. Measuring tcp round-trip time in the data plane. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, SPIN '20, page 35–41, New York, NY, USA, 2020. Association for Computing Machinery.

[29] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[30] L. Foundation. Data plane development kit (DPDK), 2015.

[31] L. Gao and J. Rexford. Stable Internet Routing without Global Coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, 2001.

[32] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path-quality monitoring in the presence of adversaries. In *ACM SIGMETRICS*, page 193–204, New York, NY, USA, 2008. Association for Computing Machinery.

[33] D. K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *ACM SIGCOMM*, volume 34, pages 79–92. ACM, August/September 2004.

[34] F. Han, M. Wang, Y. Cui, Q. Li, R. Liang, Y. Liu, and Y. Jiang. Future Data Center Networking: From Low Latency to Deterministic Latency. *IEEE Network*, 36(1):52–58, 2022.

[35] T. Holterbach, E. C. Molero, M. Apostolaki, A. Dainotti, S. Vissicchio, and L. Vanbever. Blink: Fast connectivity recovery entirely in the data plane. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 161–176, Boston, MA, Feb. 2019. USENIX Association.

[36] Intel. Barefoot Tofino. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html.

[37] S. Jain. What's in a name? Understanding the Google Cloud network "edge". https://cloud.google.com/blog/products/networking/understanding-google-cloud-network-edge-points, 2021.

[38] J. Juen, A. Johnson, A. Das, N. Borisov, and M. Caesar. Defending Tor from Network Adversaries: A Case Study of Network Path Prediction. *Proceedings on Privacy Enhancing Technologies*, 2015(2):171–187, 2015.

[39] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *ACM SIGCOMM*, SIGCOMM '05, page 253–264, New York, NY, USA, 2005. Association for Computing Machinery.

[40] A. Kashaf, V. Sekar, and Y. Agarwal. Analyzing third party service dependencies in modern web services: Have we learned from the mirai-dyn incident? In *Proceedings of the ACM Internet Measurement Conference*, pages 634–647, 2020.

[41] M. A. Khan, E. Baccour, Z. Chkirbene, A. Erbad, R. Hamila, M. Hamdi, and M. Gabbouj. A Survey on Mobile Edge Computing for Video Streaming: Opportunities and Challenges. *IEEE Access*, 10:120514–120550, 2022.

[42] C. Kim, A. Sivaraman, N. P. Katta, A. Bas, A. Dixit, and L. J. Wobker. In-band network telemetry via programmable dataplanes. Industrial demo, ACM SIGCOMM '15, 2015.

[43] T. Koch, S. Yu, S. Agarwal, E. Katz-Bassett, and R. Beckett. Painter: Ingress traffic engineering and routing for enterprise cloud networks. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 360–377, New York, NY, USA, 2023. Association for Computing Machinery.

[44] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. B. Krasic, C. Shi, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. C. Dorfman, J. Roskind, J. Kulik, P. G. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, and W.-T. Chang. The quic transport protocol: Design and internet-scale deployment. 2017.

[45] M. Lee, S. Goldberg, R. R. Kompella, and G. Varghese. Finecomb: Measuring microscopic latency and loss in the presence of reordering. *IEEE/ACM Transactions on Networking*, 22(4):1136–1149, 2014.

[46] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[47] Y. Li, R. Miao, C. Kim, and M. Yu. FlowRadar: A better NetFlow for data centers. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 311–324, Santa Clara, CA, Mar. 2016. USENIX Association.

[48] A. Maria, Z. Aviv, and V. Laurent. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[49] R. Meier, T. Holterbach, S. Keck, M. Stähli, V. Lenders, A. Singla, and L. Vanbever. (self) driving under the influence: Intoxicating adversarial network inputs. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 34–42, 2019.

[50] J. Mogul and S. Deering. Path mtu discovery. RFC 1191, RFC Editor, November 1990. http://www.rfc-editor.org/rfc/rfc1191.txt.

[51] G. C. M. Moura, J. Heidemann, W. Hardaker, P. Charnsethikul, J. Bulten, J. M. Ceron, and C. Hesselman. Old but gold: Prospecting TCP to engineer and live monitor DNS anycast. In *Proceedings of the Passive and Active Measurement Workshop*, page to appear, virtual, Mar. 2022. Springer.

[52] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim. Language-directed hardware design for network performance monitoring. In *ACM SIGCOMM*, SIGCOMM '17, page 85–98, New York, NY, USA, 2017. Association for Computing Machinery.

[53] K. Nichols. pping (pollere passive ping), 2017.

[54] M. Osama, A. A. Ateya, S. Ahmed Elsaid, and A. Muthanna. Ultra-Reliable Low-Latency Communications: Unmanned Aerial Vehicles Assisted Systems. *Advances in Wireless Communications Systems, Information*, 13, 2022.

[55] S. Ostermann. tcptrace homepage. *http://www. tcptrace. org/*, 2007.

[56] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat. *SCION: A Secure Internet Architecture*. Springer Verlag, 2017.

[57] A. Pilosov and T. Kapela. Stealing the Internet: An Internet-scale Man in the Middle Attack. *NANOG 44*, 2008.

[58] T. Qiu, J. Ni, H. Wang, N. Hua, Y. R. Yang, and J. J. Xu. Packet doppler: Network monitoring using packet shift detection. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, New York, NY, USA, 2008. Association for Computing Machinery.

[59] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). RFC 4271, RFC Editor, January 2006. http://www.rfc-editor.org/rfc/rfc4271.txt.

[60] M. Saad, V. Cook, L. Nguyen, M. T. Thai, and A. Mohaisen. Partitioning attacks on bitcoin: Colliding space, time, and logic. In *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, pages 1175–1187. IEEE, 2019.

[61] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *ACM SIGCOMM*, 2017.

[62] S. Sengupta, H. Kim, and J. Rexford. Continuous in-network round-trip time monitoring. In *ACM SIGCOMM*, SIGCOMM '22, page 473–485, New York, NY, USA, 2022. Association for Computing Machinery.

[63] G. Severi, M. Jagielski, G. Yar, Y. Wang, A. Oprea, and C. Nita-Rotaru. Network-level adversaries in federated learning. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 19–27. IEEE, 2022.

[64] A. Shukla and K.-T. Foerster. Shortcutting fast failover routes in the data plane. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, ANCS '21, page 15–22, New York, NY, USA, 2022. Association for Computing Machinery.

[65] G. Simsek, D. Ergenç, and E. Onur. Efficient network monitoring via in-band telemetry. In *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 1–6, 2021.

[66] J. Snijders. Practical everyday BGP filtering with AS_PATH filters: Peer locking. *NANOG-67*, 2016.

[67] J. Sonchack, D. Loehr, J. Rexford, and D. Walker. Lucid: a Language for Control in the Data Plane. In *Proceedings of the 2021 ACM SIGCOMM Conference*. ACM, 2021.

[68] F. Streibelt, F. Lichtblau, R. Beverly, A. Feldmann, C. Pelsser, G. Smaragdakis, and R. Bush. Bgp communities: Even more worms in the routing can. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, page 279–292, New York, NY, USA, 2018. Association for Computing Machinery.

[69] F. Streibelt, F. Lichtblau, R. Beverly, A. Feldmann, C. Pelsser, G. Smaragdakis, and R. Bush. Bgp communities: Even more worms in the routing can. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, page 279–292, New York, NY, USA, 2018. Association for Computing Machinery.

[70] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: Routing Attacks on Privacy in Tor. In *USENIX Security Symposium*, 2015.

[71] W. Tang, L. Kiffer, G. Fanti, and A. Juels. Strategic latency reduction in blockchain peer-to-peer networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(2):1–33, 2023.

[72] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren. Quantifying the benefits of joint content and network routing. 41(1):243–254, jun 2013.

[73] L. Wang, H. Kim, P. Mittal, and J. Rexford. Programmable in-network obfuscation of traffic. *CoRR*, abs/2006.00097, 2020.

[74] L. Wang, H. Kim, P. Mittal, and J. Rexford. Raven: Stateless rapid ip address variation for enterprise networks. *Proceedings on Privacy Enhancing Technologies*, 2023, Jul 2023.

[75] Z. Yang, Z. Wu, M. Luo, W.-L. Chiang, R. Bhardwaj, W. Kwon, S. Zhuang, F. S. Luan, G. Mittal, S. Shenker, et al. {SkyPilot}: An intercloud broker for sky computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 437–455, 2023.

[76] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, V. Lin, C. Rice, B. Rogan, A. Singh, B. Tanaka, M. Verma, P. Sood, M. Tariq, M. Tierney, D. Trumic, V. Valancius, C. Ying, M. Kallahalla, B. Koley, and A. Vahdat. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. SIGCOMM '17, page 432–445, New York, NY, USA, 2017. Association for Computing Machinery.

[77] S. Yoo and X. Chen. Secure keyed hashing on programmable switches. In *Proceedings of the ACM SIGCOMM 2021 Workshop on Secure Programmable Network INfrastructure*, SPIN '21, page 16–22, New York, NY, USA, 2021. Association for Computing Machinery.

[78] Y. Yoshinaka, J. Takemasa, Y. Koizumi, and T. Hasegawa. On implementing chacha on a programmable switch. In *Proceedings of the 5th International Workshop on P4 in Europe*, EuroP4 '22, page 15–18, New York, NY, USA, 2022. Association for Computing Machinery.

[79] L. Yuliang, M. Rui, K. Changhoon, and Y. Minlan. LossRadar: Fast Detection of Lost Packets in Data Center Networks. In *CoNEXT*, New York, NY, USA, December 2016. ACM.

[80] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu. Improving Cloud Gaming Experience through Mobile Edge Computing. *IEEE Wireless Communications*, 26(4):178–183, 2019.

[81] X. Zhang, T. Sen, Z. Zhang, T. April, B. Chandrasekaran, D. Choffnes, B. M. Maggs, H. Shen, R. K. Sitaraman, and X. Yang. Anyopt: Predicting and optimizing ip anycast performance. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 447–462, New York, NY, USA, 2021. Association for Computing Machinery.

[82] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing cost and performance in online service provider networks. In *USENIX Networked Systems Design and Implementation*, 2010.

Appendix

A Adversarial Model

We describe in more detail a few concrete attacks that can be launched within the threat model outlined in §2.1.

1. **Manipulation Attacks**: An adversary on path P wants to make the one-way-delay of path P look smaller. The adversary launches the attack by modifying a timestamp t to timestamp $t+d$ which will make its one-way-delay appear to be reduced by d. Similarly, the adversary could modify the sequence numbers to hide loss and cover-up that some packets were dropped in her network.
2. **Replay Attacks**: An adversary is on paths P and Q. The latency from the sender to the adversary along P is shorter than the latency from the sender to the adversary along path Q. Latency from the adversary to the receiver is the same on path P and Q (P and Q may even share all hops after the adversary). At an instant t in time, the adversary sees packets with timestamp $t-p$ on path P and $t-q$ on path q. Since path P is faster, $t-p > t-q$. The adversary wants to improve the one-way-delay on path q and rewrites the timestamps for path Q to be $t-p$, reducing the delay on Q to the delay of P. Alternatively, the adversary could have a passive tap on path P.

B Additional TANGO Design Details

As shown in the pseudocode above (Alg. 1), TANGO senders perform the mapping from operator-provided objectives to routing traffic classes, add latency and loss metrics in custom headers, apply integrity-protecting signatures over the collected metrics, and tunnel the resulting packet over the public Internet to the destination edge.

Meanwhile, TANGO receivers decapsulate received metrics and verify signatures over the metrics, ensuring only tamper-free measurements are included in new optimization calculations (Alg. 2). After removing metadata, the receiver forwards the application traffic along to its final destination. It also issues route updates to the TANGO sender, if a better path has emerged and meets specified update thresholds (*e.g.,* the new path improves latency by at least 20% for more than 10 measurement cycles over 100 ms).

Algorithm 1 TANGO tunneling send behavior

```
1:  // Send an application packet to peer node
2:  function SEND(AppPacket)
3:      // Map packet to tunnel information
4:      TrafficClass ← GETCLASS(AppPacket)
5:      PathID ← GETPATH(TrafficClass)
6:
7:      // Collect metrics at time of processing packet
8:      Ts ← GETTIMENOWMS( )
9:      Seq ← GETANDINCREMENTSEQNUM( )
10:
11:     // Sign using books indexed by PathId and metrics
12:     TsSig ← TSBOOK[PathId][Ts]
13:     SeqSig ← SEQNUMBOOK[PathId][Seq]
14:
15:     // Form headers and send encapsulated packet
16:     { IpHdr, UdpHdr } ← GETTUNNELHEADERS(PathId)
17:     Metrics ← { PathId, Ts, TsSig, Seq, SeqSig }
18:     FORWARD(IpHdr, UdpHdr, Metrics, AppPacket)
19: end function
```

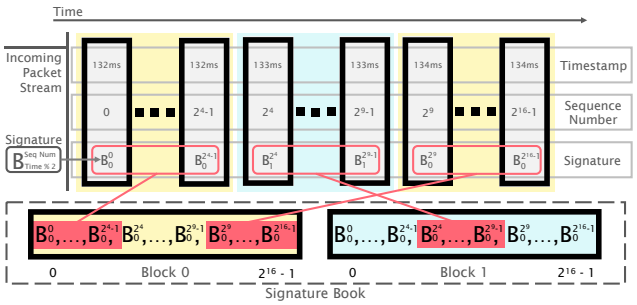


Figure 14: TANGO *ensures sequence numbers monotonically increase, even during block transitions, and only reset once all are expended. For example, assume block 0 is used on even timestamps and block 1 on odd. If, as the last sequence number of an even timestamp, the sequence number is* 2^4-1*, the next sequence number, now with an odd timestamp and therefore accessing block 1, is* 2^4*.*

C Supplementary Internet-Scale Results

C.1 Path Diversity: CDF

As described in §6.1.1, TANGO's PATHFINDER algorithm (detailed in §4) uncovered alternative paths between 503 globally-distributed Vultr pairs. The CDF of number of paths exposed per Vultr pair is included in Fig. 16. In some cases our BGP announcements appeared to be filtered as the source node used did not hear the BGP announcement made from the destination node. These

Algorithm 2 TANGO tunneling receive behavior

```
1:  // Verify tunneled packet and finish forwarding and reroute
2:  function RECEIVE(IpHdr, UdpHdr, Metrics, AppPacket)
3:      // Map packet to tunnel information
4:      TrafficClass ← GETCLASS(AppPacket)
5:      PathID ← Metrics.PathId
6:
7:      // Verify signatures
8:      ValidTsSig ← TSBOOK[PathId][Metrics.Ts]
9:      ValidSeqSig ← SEQNUMBOOK[PathId][Metrics.Seq]
10:     IsValidTs ← Metrics.TsSig == ValidTsSig
11:     IsValidSeq ← Metrics.SeqSig == ValidSeqSig
12:     if not IsValidTs or not IsValidSeq then
13:         return // Bail early if invalid
14:     end if
15:
16:     // Update metrics and finish forwarding packet
17:     { Delay, BestDelayId } ← UPDATEDELAY(Metrics.Ts)
18:     { Loss, BestLossId } ← UPDATELOSS(Metrics.SeqNum)
19:     FORWARD(AppPacket)
20:
21:     // Issue reroute request if path is not performant enough
22:     DoReroute ← CHECK(TrafficClass, Delay, Loss)
23:     if DoReroute then
24:         // Choose the best path for traffic class
25:         Id ← BEST(TrafficClass, BestDelayId, BestLossId)
26:         Update ← ENCRYPT(TrafficClass, BestPath)
27:
28:         // Request route update over all paths
29:         for { IpHdr, UdpHdr } in AllPaths do
30:             FORWARD(IpHdr, UdpHdr, Update)
31:         end for
32:     end if
33: end function
```

cases are recorded in the CDF as having a single available path because TANGO can still function by using the Vultr-provided IP address of instances to tunnel traffic. We also explored how many of these paths were found using only communities supported by our immediate transit provider Vultr and how many were found because of BGP community support at ASes further down the path (shown as separate traces in Fig. 16). If only Vultr-supported communities are used, the median pair of nodes only has two paths, but using transitive communities raises this by 50% to 3 paths. The maximum number of paths found between any two pairs of nodes increases from 6 to 12 showing the importance of community support at various nodes in the topology.

C.2 High-Latency Performance Degradation Events

In addition to looking for events with high loss we searched for events with high latency (i.e., a 100% latency increase over the baseline). We found these happened more frequently than high loss events. Fig. 17 shows these events. For high latency events, we found *all events were avoidable with* TANGO.

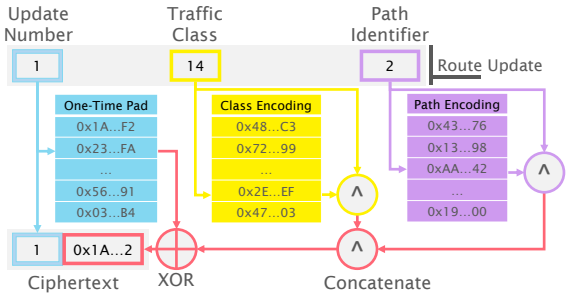


Figure 15: *Integrity-protected route updates. The encodings, if sufficiently sparse, prevent adversaries from brute-forcing the few bits encoding class/path, since overall encodings can easily be verified.*

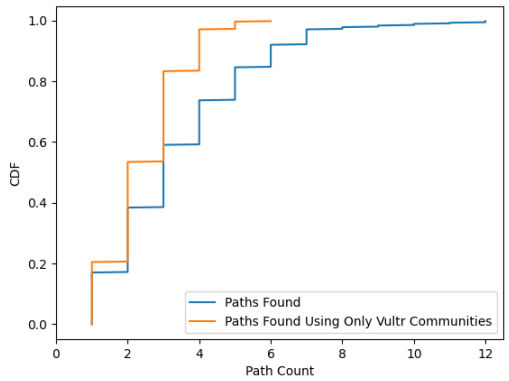


Figure 16: *Available paths identified with* PATHFINDER*, per Vultr pair using only communities supported by Vultr and communities supported by Vultr as well as ASes further down the path.*

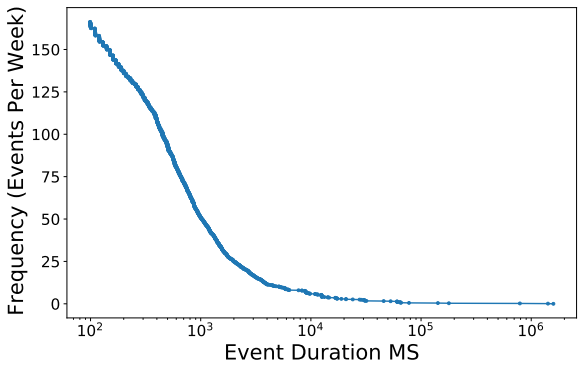


Figure 17: *Frequency vs duration of events with high latency.*

C.3 Performance Diversity: Vultr LA

We include results from our additional measurement study of 23 node pairs with Vultr LA as TANGO's fixed receiver.

How often are TANGO-discovered paths better than the de-

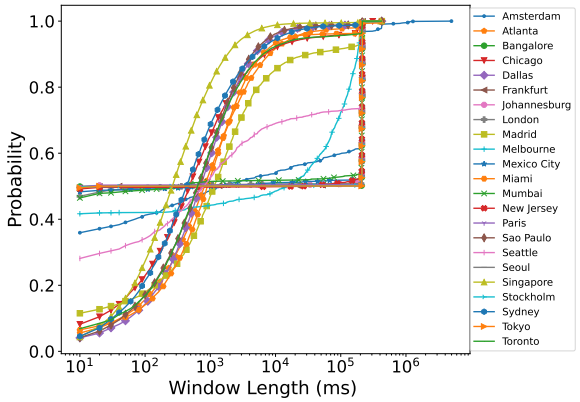


Figure 18: *Vultr LA Results: On average across all 23 pairs, best paths lasted from 1.8-107s, with median durations of 20ms-106s.* TANGO *only chooses to perform dynamic rerouting for best paths that last longer than 100ms.*

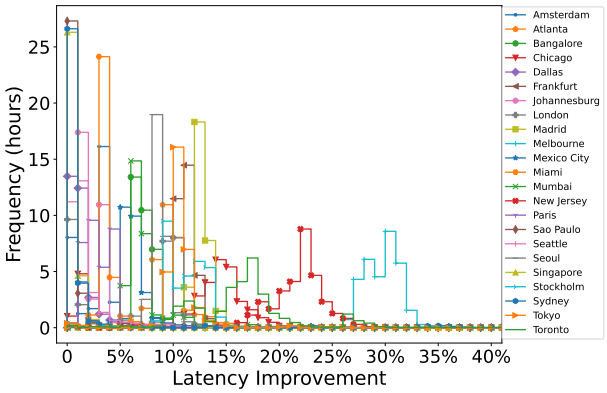


Figure 19: *Vultr LA Results:* TANGO*-exposed best paths outperform the BGP default by up to 29% on average, and up to 32% for some pairs.*

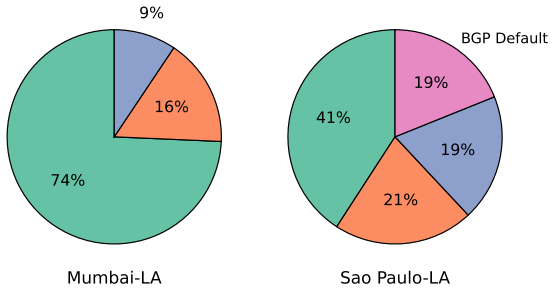


Figure 20: *Vultr LA Results: The BGP default path is often beaten by one or more alternative paths.*

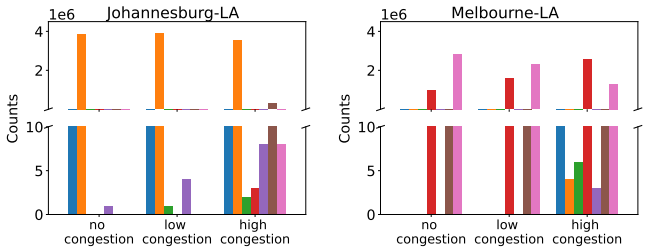


Figure 21: *Vultr LA Results: Number of times different paths contributed to being "best path" for a given congestion condition, per node pair. While one path might dominantly be best under no/low congestion, during high congestion more paths emerge as the best.*

fault? Of the 23 pairs in our Vultr Los Angeles (LA) measurements, *all 23 pairs had at least one alternative path that outperformed the default BGP path for a significant amount of time*: 100% of the time for 20 pairs, and 58-81% of the time for the remaining 3 pairs (Amsterdam-LA, Sao Paulo-LA, Singapore-LA). We also note that there were many pairs with *more than one* alternative path which outperformed the default, providing further path diversity and potential performance resilience: 18 pairs had two or more alternatives and 4 pairs had three or more alternatives. In Figs. 20 and 21, we visualize how often other paths emerged as best for a subset of pairs, chosen for their interesting breakdown. Distribution charts for all 23 pairs are included in Figs. 22 and 24.

How long is one path the best? We measured the window of time one path remained the best for each of the 23 Vultr pairs (Fig. 18). Many pairs showed path longevity, and for 17 pairs, the median window duration was 235*ms*-15*s*, with average durations of 1.8-10.8*s* (from Singapore, Sao Paulo, Dallas, Sydney, Atlanta, Miami, Chicago, Toronto), 20.1-59*s* (from Madrid, Seattle), and 100-107*s* (from Mumbai, Amsterdam, Mexico City, Frankfurt, London, Stockholm). For 3 pairs, the median duration was shorter at 20 *ms*, most likely due to ECMP behavior between the nodes. The average duration for these 3 pairs ranged between 105-106*s* (from Bangalore, Johannesburg, and Paris to LA). For the last 3 pairs out of all 23, the median and average duration were both much longer: 31*s* median and 104*s* average from New Jersey and 105*s* median with 106-107*s* average from both Seoul and Tokyo.

TANGO performs dynamic reroutes when a new path emerges as the best for longer than a given window threshold (e.g., 100ms), and it remains on the old best path for shorter windows. Thus, windows in the range of 20ms would not trigger a dynamic route update, while longer window lengths on the order of minutes would benefit from static path optimizations, instead of on-demand route updates. TANGO's on-demand reroutes would be most beneficial for the 17 pairs with median window durations between 235*ms*-15*s*.

By how much does the best path beat the BGP default? We measured the difference in latency between the best path and the default path, for each of the 23 Vultr pairs (Fig. 19). Of all 23 pairs, *the non-default best path outperformed the default by an average of 1-29% per pair*, for durations of 1.03-26.3 *hours*. The median improvement values also ranged from 0-22%. We saw even more opportunity for optimizing routing performance at the 95th percentile, where the TANGO-discovered best path *improved latency over the BGP default by 25-32%* for a duration of over half an hour (from Toronto) and between 1.3-1.6 hours (from New Jersey, Melbourne).

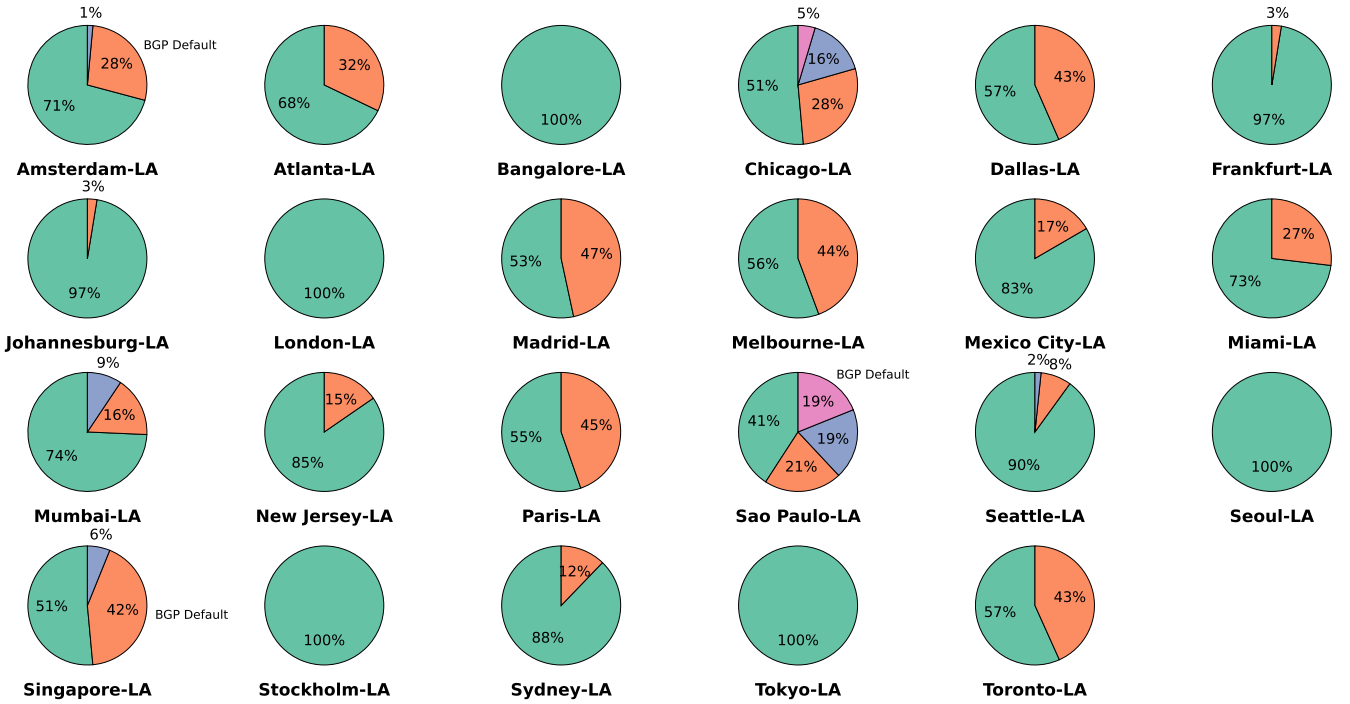


Figure 22: *Vultr LA Results: The BGP default path is beaten by one or more* TANGO*-exposed paths for all 23 node pairs, and for 4 out of 23 pairs it is beaten by 3 or more paths. On average, measurements from Vultr LA exposed richer path diversity across more nodes.*

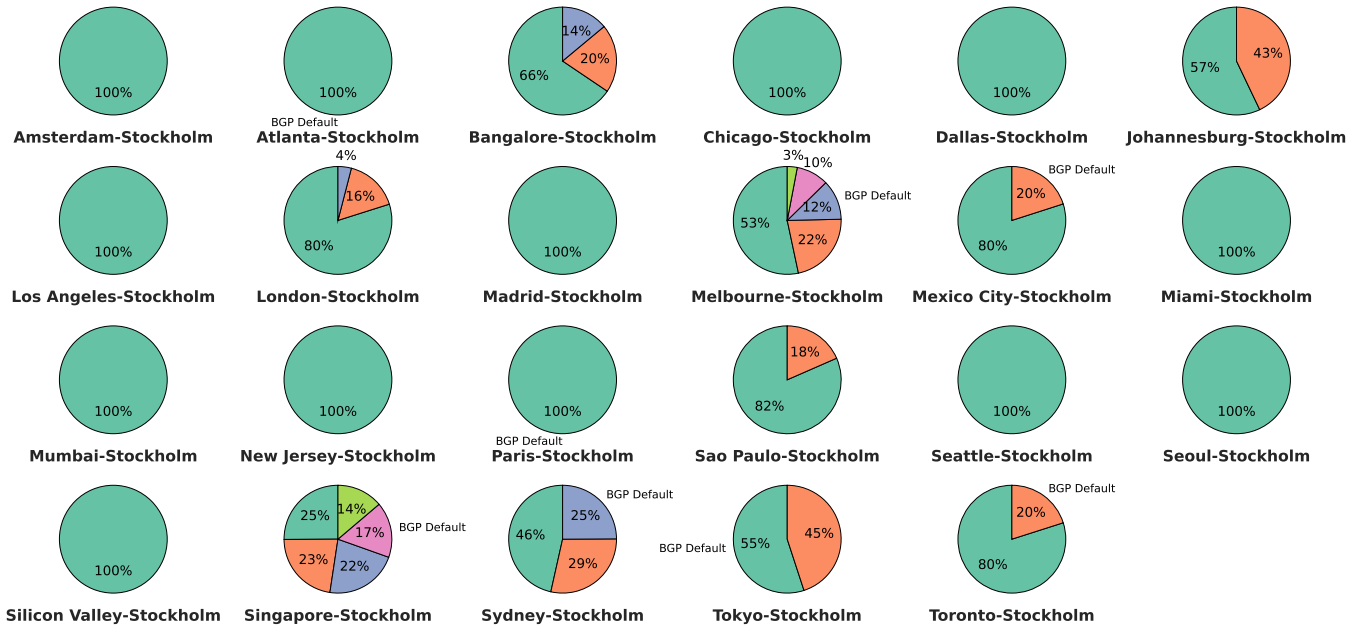


Figure 23: *Vultr Stockholm Results: The BGP default path is beaten by one or more* TANGO*-exposed paths for 20 out of 23 node pairs, for 4 out of 23 pairs it is beaten by 3 or more paths, while for 2 out of 23 pairs it is beaten by 4 alternative paths. On average, measurements from Vultr Stockholm exposed less path diversity across all nodes as compared to results from Vultr LA measurements.*

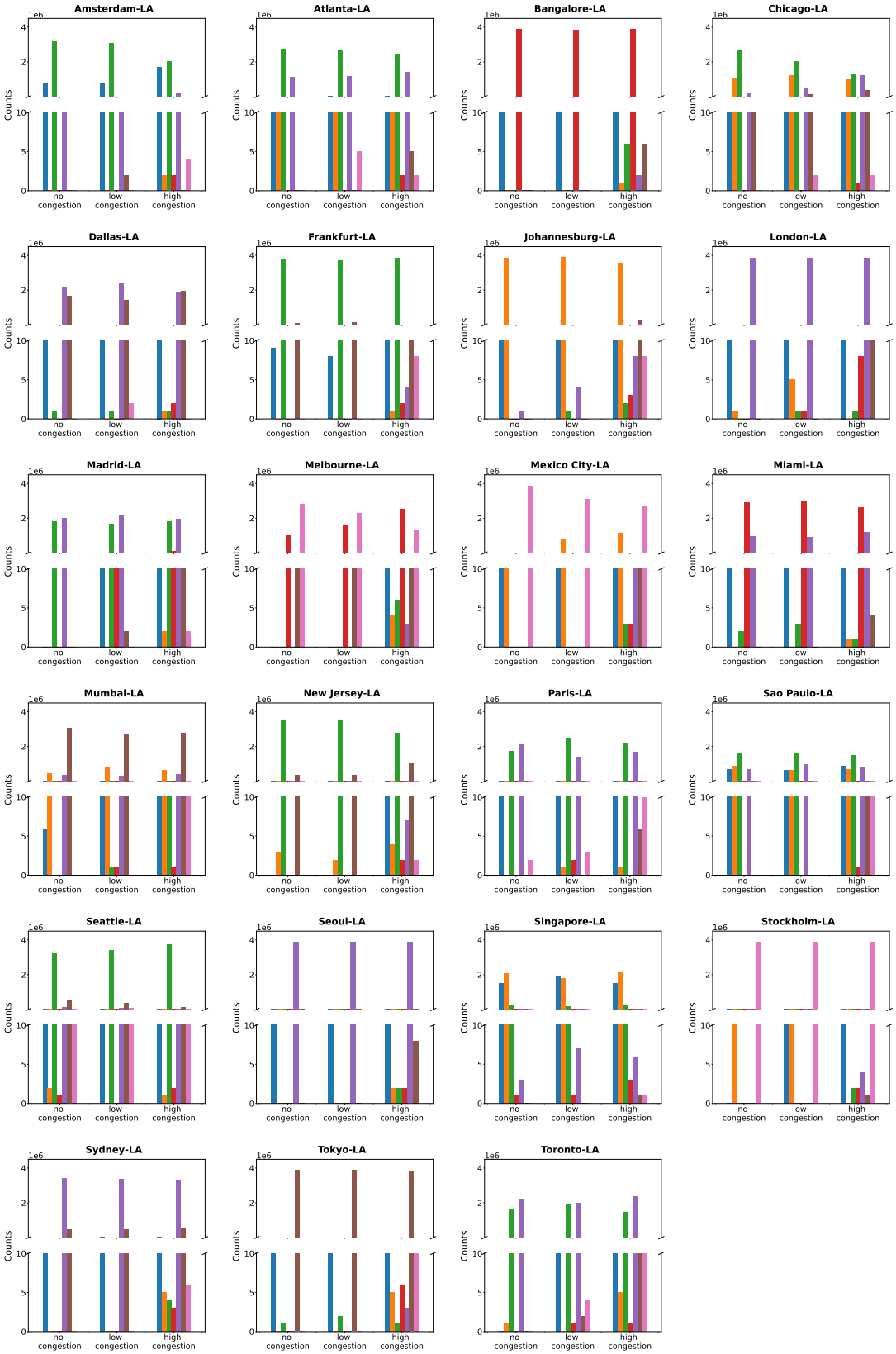


Figure 24: *Vultr LA Results: Number of times different paths contributed to being "best path" for a given congestion condition, per node pair. While one path might dominantly be best under no/low congestion, during high congestion more paths emerge as the best path.*

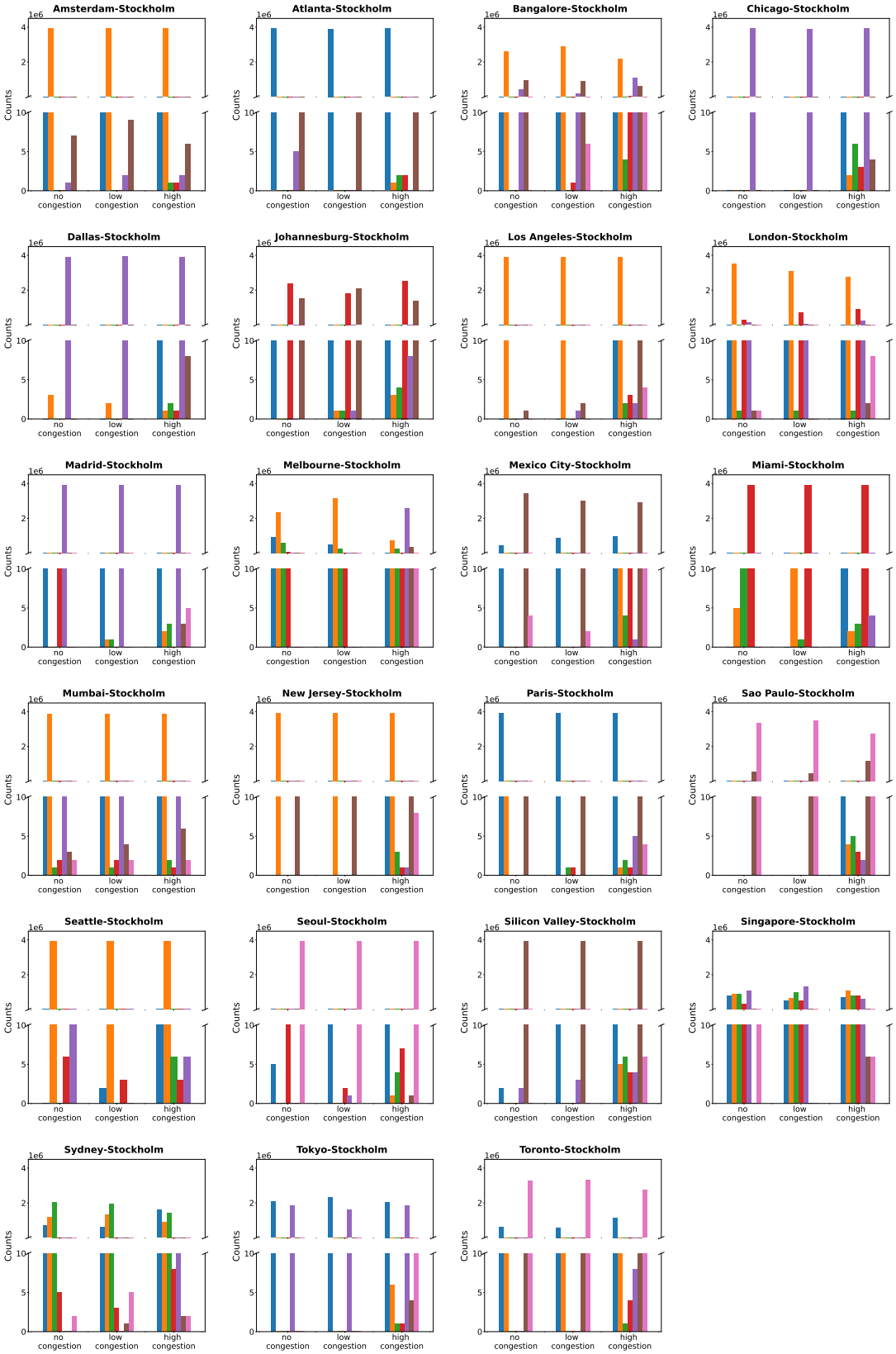


Figure 25: *Vultr Stockholm Results: Number of times different paths contributed to being "best path" for a given congestion condition, per node pair. While one path might dominantly be best under no/low congestion, during high congestion more paths emerge as the best path.*

Figure 26: *Geomap of 25 Vultr data centers running* TANGO *globally.*

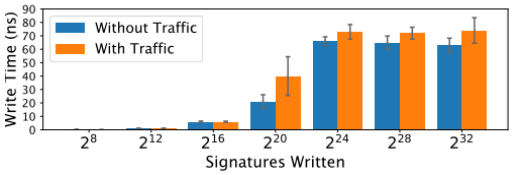
Errata Slip #2

Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation

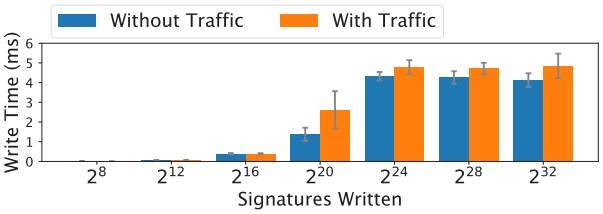
In the paper "TANGO: Secure Collaborative Route Control across the Public Internet" by Henry Birge-Lee, Sophia Yoo, Benjamin Herber, Jennifer Rexford, and Maria Apostolaki, *Princeton University* (Thursday session, "Security," pp. 1791–1811 of the Proceedings), the authors have provided the following correction:

On page 1802, Figure 13 at the top of the page, the graph on the right has been updated:

Original figure:



Revised figure:



On page 1802, left column, end of first paragraph the following text has been updated to match the revised graph:

Original text:

As shown in Fig. 13, at a refresh period of 8ms at maximum port speed, the control plane can write 2^{16} signatures in less than 10ns over the data channel. Meanwhile, the required 2^{20} signatures for all blocks across all books can be written in approximately 20ns without background traffic and 40ns with it, while being well below the 8ms refresh limit.

Revised text:

As shown in Fig. 13, at a refresh period of 8ms at maximum port speed, the control plane can write 2^{16} signatures in less than .6ms over the data channel. Meanwhile, the required 2^{20} signatures for all blocks across all books can be written in approximately 1.3ms without background traffic and 2.6ms with it, while being well below the 8ms refresh limit.