Optimized Cross-Path Attacks via Adversarial Reconnaissance

YUDI HUANG*, Pennsylvania State University, USA
YILEI LIN*, Meta, USA
TING HE, Pennsylvania State University, USA

While softwarization and virtualization technologies make modern communication networks appear easier to manage, they also introduce highly complex interactions within the networks that can cause unexpected security threats. In this work, we study a particular security threat due to the sharing of links between high-security paths and low-security paths, which enables a new type of DoS attacks, called cross-path attacks, that indirectly attack a set of targeted high-security paths (target paths) by congesting the shared links through a set of attacker-controlled low-security paths (attack paths). While the feasibility of such attacks has been recently demonstrated in the context of SDN, their potential performance impact has not been characterized. To this end, we develop an approach for designing an optimized cross-path attack under a constrained total attack rate, consisting of (i) novel reconnaissance algorithms that can provide consistent estimates of the locations and parameters of the shared links via network tomography, and (ii) efficient optimization methods to design the optimal allocation of attack rate over the attack paths to maximally degrade the performance of the target paths. The proposed attack has achieved a significantly larger performance impact than its non-optimized counterparts in extensive evaluations based on multiple network settings, signaling the importance of addressing such intelligent attacks in network design.

CCS Concepts: • **Networks** → **Network measurement**; **Network security**; *Network management*.

Additional Key Words and Phrases: Denial-of-service attack, topology inference, queueing analysis, adversarial reconnaissance, attack modeling.

ACM Reference Format:
Yudi Huang, Yilei Lin, and Ting He. 2023. Optimized Cross-Path Attacks via Adversarial Reconnaissance. *Proc. ACM Meas. Anal. Comput. Syst.* 7, 3, Article 58 (December 2023), 30 pages. https://doi.org/10.1145/3626789

1 INTRODUCTION

The trend of network softwarization and virtualization has fundamentally altered the way we build network systems. While the logically centralized control plane provides convenient ways to manage the network resources through various abstractions, such abstractions also hide the complex interactions within the network, which can cause unexpected security threats. In this work, we focus on a particular security threat due to the sharing of links between high-security paths and low-security paths, which enables a new type of *denial-of-service (DoS)* attacks called *cross-path attacks*.

*Both authors contributed equally to this research.

Authors' addresses: Yudi Huang, Pennsylvania State University, University Park, Pennsylvania, USA, yxh5389@psu.edu; Yilei Lin, Meta, Santa Clara, California, USA, yileilin1997@gmail.com; Ting He, tinghe@psu.edu, Pennsylvania State University, University Park, Pennsylvania, USA.

2476-1249/2023/12-ART58 $15.00
https://doi.org/10.1145/3626789

Intuitively, cross-path attacks are indirect DoS attacks, where instead of directly attacking the paths of interest (*target paths*), the attacker sends attack traffic on some other paths (*attack paths*) sharing resources (e.g., link bandwidth) with the target paths, so as to degrade the performance of the target paths by consuming the shared resources. Such attacks are of interest when the target paths are difficult to attack directly, but share network resources with some low-security paths that are more susceptible to attacks.

One scenario for cross-path attacks is in the context of a *Software Defined Network (SDN)* [9], where the target paths are control-plane paths connecting switches to the controller and the attack paths are data-plane paths originating from attacker-controlled hosts that share links with some of the control-plane paths. Instead of directly triggering a flood of control messages to attack the control-plane paths as in earlier attacks [45], a cross-path attack only floods selected paths in the data plane, which makes it both stealthier and more resilient to state-of-the-art control plane defenses such as FloodGuard [47], FloodDefender [44], and SPHINX [14]. Another scenario for cross-path attacks is in the context of network slicing [51], which is a technology in 5G networks that allows the network provider to set up multiple virtual networks over a shared infrastructure. To improve resource utilization and support elasticity, different slices may share network and computing resources [51]. Meanwhile, slices created for different applications can follow different security standards [27, 28], and some slices may even be managed by less trusted third parties [2]. These practices create opportunities for an attacker to attack paths in a high-security slice (target paths) by consuming resources shared with some paths the attacker can access in a low-security slice (attack paths), while remaining stealthy to intrusion detection systems in the high-security slice.

Despite the demonstration of feasibility in [9], there is little quantitative understanding about cross-path attacks. In this work, we will address this gap by designing an optimized attack strategy that can achieve the maximum impact with a constrained total attack rate. At a high level, our strategy works by (i) inferring the locations and parameters of network elements shared between the target paths and the attack paths, and then (ii) optimally allocating the total attack rate over the attack paths to maximize the performance degradation of the target paths. By analyzing the optimal attack strategy, we not only quantify the maximum damage due to cross-path attacks as a function of the attack rate, but also shed light on possible defenses.

1.1 Related Work

As a newly identified attack, cross-path attack has not been extensively studied previously. Therefore, we will provide background by reviewing some relevant security problems and solution techniques.

Security vulnerabilities in SDN: As the architectures and protocols of SDN are designed to facilitate performance and programmability, there are many security vulnerabilities, mainly due to the interdependency between the controller and the switches. In particular, the switch→controller dependency that arises due to the need for the data plane to obtain instructions from the control plane can create a communication bottleneck, which has been exploited in active attacks [45], adversarial reconnaissance [4], and joint reconnaissance and attack [9, 50]. The cross-path attack in this context [9] is a reconnaissance-based attack that exploits the switch→controller dependency to infer which attacker-controlled data-plane paths share links with at least one control-plane path, in order to identify the data-plane paths that can be used to launch a cross-path attack on the control plane. The reconnaissance strategy in [9] only infers whether there exists at least one shared link between a data-plane path and the targeted control-plane paths, and thus can only support a non-optimized cross-path attack. In contrast, we will provide a way to design an optimized cross-path attack through fine-grained reconnaissance based on network tomography.

Security vulnerabilities in network slicing: Network slicing introduces both *content-level threats* such as unauthorized access, compromise of functions/devices, and side-channels across

slices [12, 34], and *performance-level threats* due to malicious abuse of resource quotas [12, 43]. The cross-path attack in this context belongs to performance-level threats. While cross-path attack under network slicing can be defended by detecting the attack through the cooperation of its control plane [32] or completely isolating different slices [41], both approaches have severe limitations: the former will fail if the attack slice's control plane is compromised, and the latter will cause poor resource utilization. In this regard, our work helps to strike a balance between resource utilization and security in network slicing by quantifying the maximum impact of cross-path attacks.

Network interdiction: Traditional network interdiction refers to a problem where an interdictor tries to reduce the throughput of network users by removing selected links under a budget constraint [36]. A variation of this problem, recently proposed in [17], is conceptually similar to cross-path attack in that instead of removing links, the interdictor tries to reduce the available capacities of links traversed by target paths by injecting flows on selected paths. Despite the conceptual similarity, [17] addressed a fundamentally different problem of optimizing the (possibly multi-path) routing of injected traffic in a clairvoyant setting where the network topology and the links traversed by each target path are known to the interdictor, and the routing of injected traffic is controllable, which generally requires the cooperation of the network. In contrast, cross-path attack is based on a much weaker threat model where the attacker is an outsider of the network without internal support or information. Therefore, how to learn the internal information required for effective attack design is a critical question in cross-path attack, which is the focus of this work.

Network tomography: The optimal design of cross-path attacks requires the attacker to infer the network elements shared between two sets of paths from end-to-end measurements, which is similar to the problem addressed by network topology tomography/inference [21]. With few exceptions (e.g., [29, 49]), topology inference algorithms generally require active probing on all the paths; see [31] and references therein. This can be used to generate carefully crafted probes, such as "packet strings" [33] or "packet sandwiches" [10], which produce correlated measurements that can reveal the existence and parameters of the links shared by different paths. Our problem is different in that the attacker can only probe a subset of paths (i.e., the attack paths) but wants to infer the elements they share with the other paths (i.e., the target paths).

1.2 Summary of Contributions

Our goal is to understand the strategy and impact of the optimal cross-path attack under a constrained total rate, with the following contributions:

1) We develop novel inference algorithms that can consistently estimate the locations and parameters of the links shared between attack paths and target paths via active probing on the attack paths and passive monitoring on the target paths.

2) Under the assumption that each shared link can be modeled as an M/M/1, M/D/1, or G/G/1 queue, we derive the optimal attack design that can maximally degrade the performance of the target paths under a bounded total attack rate.

3) We evaluate the proposed algorithms by high-fidelity packet-level simulations under various settings, which show that (i) our inference algorithms can estimate the (topological) locations of shared links with good accuracy but not their detailed parameters, but (ii) our attack strategy designed based on these estimates can still cause substantially more performance degradation than some intuitive ways of launching cross-path attacks, which signals the importance of considering such intelligent attack strategies in the design of defenses.

Roadmap. We formulate our problem in Section 2, present the algorithms to infer the locations and parameters of shared links in Section 3, and present the corresponding attack design in Section 4. We then evaluate our solutions in Section 5 and conclude the paper in Section 6. **All the proofs are provided in Appendix A.**

2 PROBLEM FORMULATION

2.1 Network and Threat Model

Consider two sets of paths in a network, referred to as the *attack paths* $P_A := \{s_{Ai} \to t_{Ai}\}_{i=1}^{N_A}$ and the *target paths* $P_B := \{s_{Bi} \to t_{Bi}\}_{i=1}^{N_B}$, where $s \to t$ denotes the routing path from source s to destination t and N_A/N_B the number of paths in P_A/P_B. Suppose that an attacker is interested in attacking the target paths in P_B, but can only passively monitor the end-to-end performance (e.g., delays) on these paths. Meanwhile, the attacker can actively send packets on the attack paths in P_A. We will focus on the important special case of $s_{Ai} \equiv s_A$ ($i = 1, \ldots, N_A$), as it represents a most easily-deployable attack with only one active malicious node (i.e., s_A). Let $T_A := \{t_{Ai} | i = 1, \cdots, N_A\}$ denote the set of destinations of the attack paths. We note that the multi-source case, where the attacker controls multiple active malicious nodes, is not a trivial extension of the single-source case. We leave the study of the multi-source case to future work.

The two sets of paths may share some network elements. For clarity, we will model all the shared elements as "shared links", which can represent any shared resources (e.g., communication links, network functions, and other services). Here, "shared" means shared by attack traffic and target traffic without isolation. While traffic isolation technologies exist, applying them will lower resource utilization and hence the revenue of the network provider [40, 51]. In this regard, our work aims at quantifying the risk due to lack of isolation to inform a proper tradeoff. We model such link sharing by a (logical) routing topology $G = (V, E)$, which is a graph formed by all the paths in $P_A \cup P_B$. According to [33, 42], V is a set of vertices representing sources, destinations, and branching/joining points between paths, and E is a set of edges representing the connections between the vertices, where a sequence of consecutive links without branching/joining points is represented by a single edge. We assume that the attacker does not have access to the control plane, i.e., he does not know the ground truth of G. Instead, the attacker can infer information about G from end-to-end measurements on P_A and P_B. We will use "link" to refer to a communication link in the underlying network and "edge" to refer to a point-to-point connection in the routing topology. Similarly, we will use "node" to refer to a physical node in the underlying network and "vertex" to refer to a logical node in the routing topology. As commonly assumed in the literature [33, 42], we assume that during the inference and attack, there is a fixed and unique routing path from each node to every other node.

Remark: While there may be other paths carrying co-existing flows in the network, it suffices to focus on the target and the attack paths for the purpose of modeling the cross-path attack. The impact of co-existing flows will be captured as background traffic on the links traversed by $P_A \cup P_B$. Our threat model depicts a *pure* cross-path attack where the attacker can only actively send packets on the attack paths and thus can only attack the target paths through "cross-path" influence. In some scenarios such as the cross-path attack between the data plane and the control plane in SDN [9], it is possible for the attacker to generate packets on the target paths (e.g., by triggering "packet-in" messages). However, to evade existing defenses against direct attacks, such attacker-triggered traffic on the target paths must resemble the normal traffic on these paths, which is usually insufficient to cause notable performance degradation. Intuitively, the ability to passively monitor the performance of the target paths is the minimum requirement for designing a nontrivial cross-path attack. We thus adopt this threat model to maximize the applicability of our result.

We assume the following capabilities of the attacker. **First,** the attacker can observe packets on each target path $s_{Bi} \to t_{Bi}$ as soon as they are transmitted, even if s_{Bi} differs from s_A. For example, the attacker may intersect target traffic at locations $(s_{Bi})_{i=1}^{N_B}$ (i.e., s_{Bi} denotes the starting point of intersection for a path of interest). Instead of directly attacking target traffic at these points of intersection, the attacker only uses them to passively monitor the target traffic to launch a stealthier attack. **Second**, the attacker can measure the end-to-end one-way delays of packets on both the

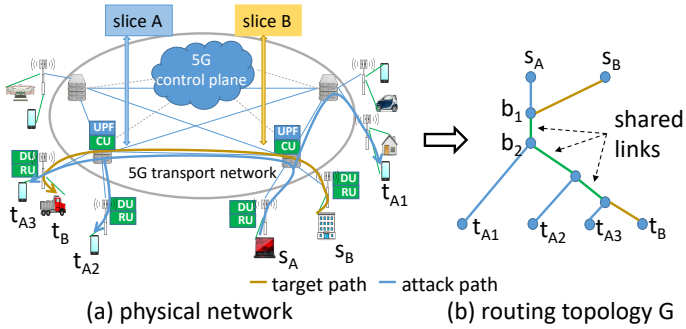


Fig. 1. Cross-path attack in the context of network slicing.

attack paths P_A and the target paths P_B. **Third**, the attacker will not be exposed by sending traffic on the attack paths. For example, the attacker may control many edge devices, which connect to the same ingress point s_A or egress point t_{Ai} (i.e., s_A and t_{Ai} may each represent a set of devices), and thus the attack flows can evade detection by the network operator even if the aggregate flow rate is high.

2.2 Problem Statement

While the sharing of links makes the paths in P_B vulnerable to cross-path attacks launched from the paths in P_A, the impact of such attacks greatly depends on the attack strategy. To understand the maximum impact of cross-path attacks, we develop an intelligent attack strategy by combining fine-grained adversarial reconnaissance with optimized attack design, by solving the following problems:

1) Adversarial Reconnaissance. We investigate to what extent the attacker can learn about the shared links based on active probing on P_A and passive monitoring on P_B.

2) Optimized Attack Design. Based on the inferred information, we investigate the optimal allocation of attack traffic over the attack paths to maximize the performance degradation (e.g., increase in average delay) inflicted on the target paths.

Remark: Our threat model requires the attacker to monitor end-to-end performance on the target paths. While this is arguably the minimum information needed for any nontrivial attack design, it does impose limitations on which paths can be set as the target. For example, in the context of SDN [9], only the control-plane paths for switches traversed by attacker-controlled data-plane paths can be the target paths. In the context of network slicing, the target paths can be the backhaul paths to cells containing attacker-controlled user equipments (UEs), which are likely to share the same paths with other UEs in the same cell and can thus be used as their proxies in collecting measurements. Instead of directly launching attacks from these attacker-controlled UEs, a cross-path attack only uses them to passively collect measurements so as to launch an effective attack from elsewhere in the network, and is thus stealthier.

2.3 Illustrative Example

Example in network slicing: Consider the scenario in Fig. 1 (a), where the attacker controls a malicious node s_A that can send traffic on a set of paths $s_A \rightarrow t_{Ai}$ $(i = 1, 2, 3)$ in slice A but wants to attack another path $s_B \rightarrow t_B$ in slice B. The attack paths share the following network elements with the target path: $s_A \rightarrow t_{A1}$ only shares the source-side central unit (CU) and user plane function (UPF); $s_A \rightarrow t_{A2}$ also shares backhaul links between the cell sites and the destination-side CU and UPF; $s_A \rightarrow t_{A3}$ further shares midhaul links and the destination-side radio unit (RU) and distributed unit (DU). These relationships can be modeled by the routing topology in Fig. 1 (b).

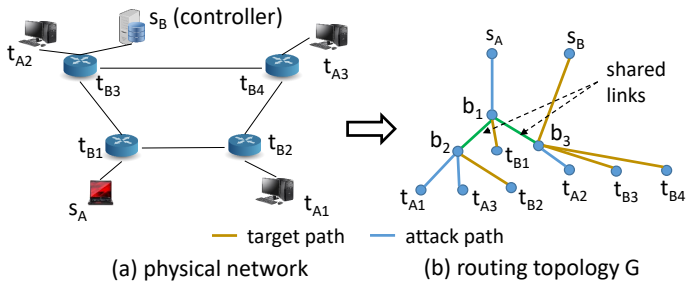


Fig. 2. Cross-path attack in the context of SDN.

Example in SDN: Consider the scenario in Fig. 2 (a), where the attacker wants to attack the control paths between the controller s_B and the switches t_{Bi} ($i = 1, \ldots, 4$) by sending traffic on the data paths $s_A \rightarrow t_{Ai}$ ($i = 1, 2, 3$). Assume shortest path routing for both data and control paths (assuming that t_{B2} connects to the controller via t_{B1} and s_A connects to t_{A3} via t_{B2}). Each data path shares some links with the control paths: $s_A \rightarrow t_{A1}$ shares a link with $s_B \rightarrow t_{B2}$, $s_A \rightarrow t_{A2}$ shares a link with $s_B \rightarrow t_{B1}$ and $s_B \rightarrow t_{B2}$, and $s_A \rightarrow t_{A3}$ shares a link with $s_B \rightarrow t_{B2}$. Meanwhile, due to the separate processing of data and control packets within a switch, the shared nodes (i.e., switches) will not cause performance correlation between data and control paths and can thus be ignored. These relationships can be modeled by the routing topology in Fig. 2 (b), where we have inserted zero-delay edges (b_1, t_{B1}), (b_2, t_{B2}), and (b_3, t_{B3}) to make each source/destination have degree one.

While it is relatively easy to identify which attack paths share at least one link with the target paths (e.g., by measuring the target path delays with/without traffic on each attack path as in [9]), different attack paths can influence the target paths to different extents. Given the routing topology G, one can intuitively identify the most useful attack path, e.g., $s_A \rightarrow t_{A3}$ in Fig. 1, but the attacker cannot directly observe such internal information. Therefore, when the attacker has access to multiple attack paths but only resources to generate a limited amount of traffic, it is unclear how he can attack most effectively. Below, we will show that the attacker can actually infer sufficient information about the routing topology to design the optimal attack that causes the maximum performance degradation to the target paths, by only passively monitoring the target paths.

3 ADVERSARIAL RECONNAISSANCE

We will show that under mild assumptions, the attacker can consistently infer both the locations and the parameters of the links shared between the attack paths and the target paths.

3.1 Preliminaries

The problem of inferring the relationship between paths from end-to-end measurements belongs to a branch of network tomography focusing on topology inference, for which many algorithms have been proposed (see Section 1.1). However, these algorithms typically require active probing on all the paths and hence are not applicable to our problem. Nevertheless, there are some results we can leverage, as summarized below.

The foundation of topology inference is using end-to-end measurements to infer the "lengths" of links defined by certain additive performance metrics. As a concrete example, we will adopt a canonical metric that can be inferred from delay measurements, but our reconnaissance algorithm can work with any additive metric for which the so-called "category weight" (see Definition 1) can be inferred from end-to-end measurements.

The metric we adopt is called *utilization-based metric* [16, 33], which is a classical additive metric used in topology inference. Let γ_e denote the probability that a packet traversing link e does not

experience any queueing delay. The utilization-based metric for link e is defined as $u_e := -log\gamma_e$, which is additive across independent links. By comparing the end-to-end delay of a packet with the minimum delay on the measured path, we can infer whether the packet incurs any queueing delay and hence estimate the utilization-based metric for the path. It was shown in [30] that with simultaneous measurements from multiple paths (obtained via multicast or back-to-back unicast), we can uniquely identify the utilization-based metric[1] at a certain granularity as follows.

Definition 1. Given a set C of paths, we define the following:

(1) the *cast weight* ϕ_C is the sum of metrics for all the links traversed by any of the paths in C;
(2) a *category* $\Gamma_{\frac{C'}{C}}$ for $C' \subseteq C$ and $C' \neq \emptyset$ is the set of links traversed by every path in C' but none of the paths in $C \setminus C'$;
(3) the *category weight* for a category $\Gamma_{\frac{C'}{C}}$, denoted by $w_{\frac{C'}{C}}$, is the sum of the metrics for all the links in $\Gamma_{\frac{C'}{C}}$.

For example, consider the set of paths $C := \{s_A \to t_{A1},\ s_A \to t_{A2},\ s_A \to t_{A3},\ s_B \to t_B\}$ in Fig. 1 (b). The cast weight $\phi_{s_A\to t_{A1},\ s_A\to t_{A2},s_A\to t_{A3}}$ is the sum metric for all the blue and green links. Category $\Gamma_{\frac{s_A\to t_{A1},s_A\to t_{A2},s_A\to t_{A3}}{s_A\to t_{A1},s_A\to t_{A2},s_A\to t_{A3},s_B\to t_B}}$ only contains link (s_A, b_1), and category $\Gamma_{\frac{s_A\to t_{A1},s_A\to t_{A2},s_A\to t_{A3},s_B\to t_B}{s_A\to t_{A1},s_A\to t_{A2},s_A\to t_{A3},s_B\to t_B}}$ only contains link (b_1, b_2).

Let $\mathcal{C} := 2^C \setminus \{\emptyset\}$ denote all the nonempty subsets of C and $U_{C'}$ ($C' \in \mathcal{C}$) denote a Bernoulli variable that equals 1 if and only if a multicast probe on C' does not incur any queueing delay. Under the assumption that different links have independent queue states as in [16, 30, 33] (which holds approximately under heavy independent cross-traffic), we have

$$-\log(\Pr\{U_{C'}=1\}) = -\log(\prod_{e\in\bigcup_{p\in C'} p} \gamma_e) = \sum_{e\in\bigcup_{p\in C'} p} u_e =: \phi_{C'}. \tag{1}$$

This, together with Definition 1, implies the following relationship between the cast weights and the category weights:

$$\sum_{C_1\in C: C_1\cap C_2\neq\emptyset} w_{\frac{C_1}{C}} = \phi_{C_2},\ \forall C_2 \in \mathcal{C}. \tag{2}$$

Using (approximated) multicast on C, we can infer all the cast weights in $(\phi_{C'})_{C'\in\mathcal{C}}$, which can then be used to uniquely identify the category weights as shown below.

Theorem 3.1 (Theorem III.1 in [30])**.** Given the cast weights $(\phi_{C'})_{C'\in\mathcal{C}}$, all the category weights $(w_{\frac{C'}{C}})_{C'\in\mathcal{C}}$ are uniquely determined by (2).

Below, we will show how to use this existing result to detect the links shared between the target paths and the attack paths.

3.2 Shared Weight Inference

Category weights provide valuable information about the relationship between paths. Specifically, if $w_{\frac{C'}{C}} > 0$, then we know that $\Gamma_{\frac{C'}{C}} \neq \emptyset$, i.e., there is at least one link shared by the paths in C' but not those in $C \setminus C'$. Moreover, under the assumption that every link has a non-zero metric (i.e., non-zero queueing probability), $w_{\frac{C'}{C}} = 0$ implies that $\Gamma_{\frac{C'}{C}} = \emptyset$. However, applying this idea directly to the paths in $P_A \cup P_B$ will require active probing on all the paths. Nevertheless, with active probing only on P_A, we can still infer the relationship between each target path $p \in P_B$ and

[1]The empirical evaluations in [30] were based on *loss-based metric* defined as $u'_e := -\log\gamma'_e$, where γ'_e denotes the no-loss probability at link e, but the theoretical result in Theorem III.1 in [30] held for any additive metric for which the cast weights can be estimated from end-to-end measurements.

all the attack paths in P_A, which turns out to be sufficient for the design of the optimal cross-path attack as explained in Section 4. The idea is to mimic a multicast on $P_A \cup \{p\}$ by monitoring path p and sending a multicast probe (or back-to-back unicast probes) on P_A when a packet is transmitted on p. Because these packets are sent very close to each other, they will observe similar queue states at shared links [33], thus mimicking a multicast on $P_A \cup \{p\}$.

Using such mimicked multicast, one may try to apply existing topology inference algorithms. However, most of such algorithms are heuristics without guaranteed accuracy, and the existing algorithms with performance guarantee all address scenarios different from ours. For example, [33] requires all the paths to share a single source, [37, 42] require all the sources to share the same set of destinations, and [7] requires the ability to probe the path between any pair of boundary vertices (in our case, these are the endpoints of all the paths in $P_A \cup \{p\}$). These differences make the existing algorithms inapplicable to our problem. Below, we will show an algorithm that can infer the shared links between the attack paths in P_A and a given target path p with guaranteed accuracy, which is then repeated for each $p \in P_B$. To our knowledge, this is the *first* algorithm that can infer the routing topology formed by a set of single-source paths and another path with arbitrary source and destination by only measuring these paths.

3.2.1 Algorithm. Under the assumptions in Section 2.1, the paths in P_A form a (logical) routing tree $\mathcal{T}$ with the source s_A as root and the destinations in T_A as leaves. Since the attacker can send active probes on P_A, he can infer $\mathcal{T}$ using existing topology inference algorithms such as Rooted Neighbor Joining (RNJ) [33]. Without loss of generality, we assume that $\mathcal{T}$ is a binary tree, as non-binary trees can be represented as binary trees by inserting zero-weight edges. Our focus is thus on inferring the relationship between $\mathcal{T}$ and a given target path $p := s_B \to t_B$. We model this relationship by a vector $\boldsymbol{W} := (W_e)_{e\in\mathcal{T}}$, where W_e denotes the sum metric of the links shared between edge $e \in \mathcal{T}$ and the target path ($W_e := 0$ if they do not share any link). We will refer to W_e as the *shared weight* on e for simplicity.

We define a few notations for the ease of presentation. Given a binary tree $\mathcal{T}$, $s_{\mathcal{T}}$ denotes the root, $b_{\mathcal{T}}$ denotes the first branching point from the root, $\delta_{l\mathcal{T}}$ is the set of leaves located in the left subtree of $\mathcal{T}$, and $\delta_{r\mathcal{T}}$ is the set of leaves in the right subtree of $\mathcal{T}$. If $\mathcal{T}$ only has one leaf t, then $\delta_{l\mathcal{T}} = \delta_{r\mathcal{T}} = \{t\}$. We denote the shared weight on a subpath $v_1 \to v_2$ in $\mathcal{T}$ by $W_{v_1\to v_2} := \sum_{e\in v_1\to v_2} W_e$.

The overall algorithm is given in Alg. 1, which prepares the routing tree $\mathcal{T}$ formed by the attack paths and then invokes Alg. 2. Alg. 2 is a recursive algorithm. Given a binary tree $\mathcal{T}'$ (initially $\mathcal{T}' = \mathcal{T}$), each recursion estimates the shared weight on the stem of $\mathcal{T}'$, i.e., edge $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$. To this end, the attacker mimics tri-cast by sending two back-to-back probes from s_A to two destinations τ_1, τ_2 from different subtrees of $\mathcal{T}'$ whenever observing a packet on the target path $s_B \to t_B$ (lines 1–2). The measured delays are used to estimate a subset of the category weights, stored in variables ρ_l, ρ_r, and ρ_s as in line 3. If we measure the category weights by the utilization-based metric, then the category weights can be inferred by first using the measured delays to estimate the no-queueing probability on each subset of $C := \{s_A \to \tau_1, s_A \to \tau_2, s_B \to t_B\}$ and compute the cast weights $(\phi_{C'})_{C'\subseteq C}$, which are then plugged into (2) to solve for the category weights. We can adopt other metrics by modifying the implementation of line 3, as long as the corresponding category weights can be inferred from end-to-end measurements. The shared weight on edge $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$ is estimated as ρ_s minus the shared weight on $s_A \to s_{\mathcal{T}'}$, which has been estimated in previous recursions (line 4). The recursion is then repeated for each subtree of $\mathcal{T}'$. The recursion stops when either (i) $\mathcal{T}'$ has no subtree (line 5), or there is no overlap between the target path and either subtree (line 8).

[2]For simplicity, here we assume that the attacker can observe packets on the target path as soon as they are transmitted, and s_A, s_B have similar distances to the shared links. This assumption will be relaxed by aligning the measurements via correlation maximization as discussed in Appendix B.1.

Algorithm 1: Shared_Weight_Inference

input : s_A, $T_A := \{t_{Ai}\}_{i=1,\cdots,N_A}$, s_B, t_B
output: shared weight vector $\boldsymbol{W}$

$\mathcal{T} \leftarrow \bigcup_{i=1,\cdots,N_A}(s_A \rightarrow t_{Ai})$; // inferred by RNJ
$\boldsymbol{W} \leftarrow \boldsymbol{0}$;
$\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}, s_A, T_A, s_B, t_B, \boldsymbol{W})$;

Algorithm 2: Recursive_Inference

input : $\mathcal{T}'$, s_A, T_A, s_B, t_B, previously inferred $\boldsymbol{W}$
output: updated shared weight vector $\boldsymbol{W}$

randomly pick τ_1 from $\delta_{l\mathcal{T}'}$ and τ_2 from $\delta_{r\mathcal{T}'}$;
send probes on $s_A \rightarrow \tau_1$, $s_A \rightarrow \tau_2$ concurrently[2] with packets monitored on $s_B \rightarrow t_B$;
use the measured delays to infer the category weights: $\rho_l \leftarrow w_{\frac{s_A\rightarrow\tau_1, s_B\rightarrow t_B}{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}}$,
$\rho_r \leftarrow w_{\frac{s_A\rightarrow\tau_2, s_B\rightarrow t_B}{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}}$, $\rho_s \leftarrow w_{\frac{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}}$;
$W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} \leftarrow \rho_s - W_{s_A\rightarrow s_{\mathcal{T}'}}$;
if $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ **then**
  return;
if $\rho_s \neq 0$ **then**
  if $\rho_l = \rho_r = 0$ **then**
    return;
  if $\rho_l > 0$ **then**
    $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{l\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
    $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;
  if $\rho_r > 0$ **then**
    $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{r\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
    $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;
else
  $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{l\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
  $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;
  $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{r\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
  $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;

3.2.2 Illustrative Example. Fig. 3 illustrates the steps of Alg. 2. On the left is the ground truth topology containing the attack paths from s_A to destinations t_{A1}, t_{A2}, t_{A3} and a target path $s_B \rightarrow t_B$, where the shared links are marked in green. Alg. 2 infers the locations and weights of these shared links in 4 steps. In each step, we mark the tree $\mathcal{T}'$ in red and label nodes $s_{\mathcal{T}'}$ and $b_{\mathcal{T}'}$ (if any). In step 1, we mimic tri-cast probes on $s_A \rightarrow t_{A1}$, $s_A \rightarrow t_{A2}$ (or t_{A3}), and $s_B \rightarrow t_B$. The results should show that $\rho_s = 0$, indicating that $s_B \rightarrow t_B$ has no overlap with $s_A \rightarrow b_{\mathcal{T}'}$. Then we search both subtrees. In the left subtree (step 2), we mimic bi-cast on $s_A \rightarrow t_{A1}$ and $s_B \rightarrow t_B$ to find out the shared weight between $s_B \rightarrow t_B$ and $s_{\mathcal{T}'} \rightarrow t_{A1}$. In the right subtree (steps 3–4), we first mimic tri-cast on $s_A \rightarrow t_{A2}$, $s_A \rightarrow t_{A3}$, and $s_B \rightarrow t_B$ to find out the shared weight on $s_{\mathcal{T}'} \rightarrow b_{\mathcal{T}'}$ (step 3), and then since $\rho_l > 0$, we will search the left subtree (step 4) to obtain all the shared weights.

3.2.3 Correctness. Alg. 1 gives consistent estimates of the shared weights in the following sense.

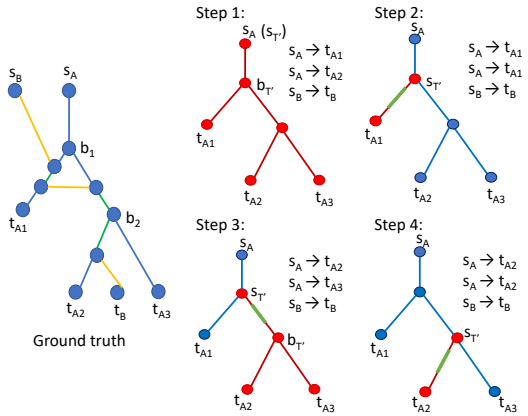


Fig. 3. Illustration for Alg. 2 (shared links are marked in green).

Theorem 3.2. If all the shared links have non-zero metrics and the category weights are accurately inferred in line 3 of Alg. 2, then Alg. 1 will accurately infer the shared weight vector $\boldsymbol{W}$.

As the number of probes increases, the estimated path-level statistics (i.e., no-queueing probabilities) will converge to their true values, so will the estimated category weights by Theorem 3.1. Thus, Alg. 1 provides consistent estimates of the shared weights.

3.2.4 Complexity. Each recursion of Alg. 2 takes $O(1)$ time (excluding probing time) as it only estimates a constant number of cast/category weights. For the number of recursions, the worst case is when all the non-zero shared weights are associated with the last edges to the destinations in $\mathcal{T}$, in which case Alg. 2 needs to perform a recursion for each edge. As a tree with N_A leaves (N_A: #attack paths) and no degree-2 vertices (implied by RNJ [33]) has at most $2N_A - 2$ edges, the complexity of Alg. 2 is $O(N_A)$. The overall complexity of Alg. 1 is $O(N_A^2 \log N_A)$, dominated by the complexity of RNJ [33].

3.3 Parameter Inference

For simplicity, we will refer to the shared portion between each edge $e \in \mathcal{T}$ and the target path as a *shared link* (although it can correspond to a sequence of links in the underlying network). Although the shared weight vector $\boldsymbol{W}$ provides both the locations and the metrics of the shared links, this information is not sufficient for optimal attack design. Specifically, by Alg. 2, each W_e is inferred under a probing rate that is only twice of the traffic rate on the target path, which is generally not enough to cause congestion. To design an effective attack, the attacker needs to predict the impact of higher attack rates on the shared links. Our idea for addressing this challenge is to model each shared link (detected by $W_e > 0$) as a queue with unknown parameters, and conduct further probing experiments with varying rates to infer these parameters.

Algorithm 3: Parameter_Inference

input : $\mathcal{T}, \boldsymbol{W}, s_A, T_A, s_B, t_B$

output: parameters $\boldsymbol{\xi} := (\xi_e)_{e\in\mathcal{T}}$ of shared links

1 $\boldsymbol{\xi} \leftarrow \mathbf{0}$;

2 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}, \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;

3.3.1 Algorithm. Let ξ_e denote the unknown parameter (or parameter vector) of the shared link on edge $e \in \mathcal{T}$. We infer $\boldsymbol{\xi} := (\xi_e)_{e\in\mathcal{T}}$ through a recursive procedure similar to Alg. 1–2, as shown in Alg. 3–4.

Algorithm 4: Parameter_Update

input : $\mathcal{T}'$, $\boldsymbol{W}$, s_A, T_A,s_B,t_B, previous $\boldsymbol{\xi}$
output: updated $\boldsymbol{\xi}$

1 **if** $W_{(s_{\mathcal{T}'},b_{\mathcal{T}'})} > 0$ **then**
2 **if** $\delta_{l\mathcal{T}'} \neq \delta_{r\mathcal{T}'}$ **then**
3 randomly choose a destination τ^* from the subtree of $\mathcal{T}'$ not sharing any link with $s_B \to t_B$;
4 **else**
5 set τ^* to the only destination in $\mathcal{T}'$;
6 vary the probing rate on path $s_A \to \tau^*$ among $\bar{\lambda}_k$ $(k = 1, \ldots, K)$ and measure the corresponding average delay ψ_k of path $s_B \to t_B$;
7 $\xi_{(s_{\mathcal{T}'},b_{\mathcal{T}'})} \leftarrow \text{argmin}_{\xi_{(s_{\mathcal{T}'},b_{\mathcal{T}'})}} \sum_{k=1}^{K} \left(\psi_k - D_{\tau^*}(\boldsymbol{\xi}; \bar{\lambda}_k)\right)^2$;
8 **if** $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ **then**
9 return
10 **if** $\tau^* \in \delta_{l\mathcal{T}'}$ **then**
11 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{r\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
12 **if** $\tau^* \in \delta_{r\mathcal{T}'}$ **then**
13 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{l\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
14 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}', \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;
15 **else**
16 **if** $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ **then**
17 return
18 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{l\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
19 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}', \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;
20 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{r\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
21 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}', \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;

Specifically, given a binary tree $\mathcal{T}'$ (initially $\mathcal{T}' = \mathcal{T}$), each recursion of Alg. 4 estimates the parameter of the shared link on the stem of $\mathcal{T}'$, if any. If the shared link exists (i.e., $W_{(s_{\mathcal{T}'},b_{\mathcal{T}'})} > 0$), then either the left or the right subtree does not share any link with the target path as explained in the proof of Theorem 3.2 (e.g., if the stem $(b_{\mathcal{T}'}, b_{l\mathcal{T}'})$ of the left subtree has $W_{(b_{\mathcal{T}'},b_{l\mathcal{T}'})} = 0$, then the left subtree contains no shared link). Therefore, we can pick a destination τ^* from the subtree without any shared link (line 3). We then conduct a number of probing experiments on $s_A \to \tau^*$ with varying rates, while measuring the average delay of the target path (line 6). Under probing rate $\bar{\lambda}_k$, the true average delay of $s_B \to t_B$ is given by

$$D_{\tau^*}(\boldsymbol{\xi}; \bar{\lambda}_k) := c_{\tau^*} + \sum_{e \in s_A \to \tau^*: W_e > 0} d(\xi_e; \bar{\lambda}_k), \tag{3}$$

where c_{τ^*} denotes the average queueing and transmission delay on the links of $s_B \to t_B$ that are not shared with $s_A \to \tau^*$ plus the propagation delay on $s_B \to t_B$, and $d(\xi_e; \bar{\lambda}_k)$ denotes the average queueing and transmission delay of the shared link on edge e, which is a function of the link parameter ξ_e and the probing rate $\bar{\lambda}_k$. Using (3) and the measured average delays, we can estimate the parameter for $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$ through least square fitting (line 7). The process is then repeated for other edges of $\mathcal{T}'$ through recursions. Note that the selection of τ^* and the top-down approach ensure that the parameters of other shared links on $s_A \to \tau^*$ would have been estimated, leaving $\xi_{(s_{\mathcal{T}'},b_{\mathcal{T}'})}$ (and possibly c_{τ^*}) as the only unknown parameter to estimate in line 7.

3.3.2 Queueing Models. As concrete examples, we consider the following queueing models ($\bar{\lambda}$ denotes probing rate):

• *M/M/1:* If each shared link is modeled as an M/M/1 queue with residual capacity $r_e - \bar{\lambda}$, its average delay equals [20]

$$d(r_e; \bar{\lambda}) = \frac{1}{r_e - \bar{\lambda}}, \tag{4}$$

where r_e is the residual capacity before probing, which is the unknown parameter to infer.

• *M/D/1:* If each shared link is modeled as an M/D/1 queue, the average delay depends on two unknown parameters [20]

$$d(\lambda_e, \mu_e; \bar{\lambda}) = \frac{2\mu_e - \lambda_e - \bar{\lambda}}{2\mu_e(\mu_e - \lambda_e - \bar{\lambda})}, \tag{5}$$

where λ_e is the background arrival rate (excluding probing traffic) and μ_e is the service rate.

• *G/G/1:* In general, we can model the shared link as a G/G/1 queue. By Kingman's formula [20], the average delay (including service time) can be approximated by

$$d(\lambda_e, \mu_e, \sigma_{ae}, \sigma_{se}; \bar{\lambda}) \approx \frac{1}{2\mu_e} \frac{\lambda_e + \bar{\lambda}}{\mu_e - \lambda_e - \bar{\lambda}} \left(\sigma_{ae}^2 (\lambda_e + \bar{\lambda})^2 + \sigma_{se}^2 \mu_e^2\right) + \frac{1}{\mu_e}, \tag{6}$$

which requires four unknown parameters: the background arrival rate λ_e, the service rate μ_e, the variance of the interarrival time σ_{ae}^2, and the variance of the service time σ_{se}^2. Note that treating σ_{ae}^2 as a constant is an approximation as it generally depends on the probing traffic.

Discussion: It is known that the delay in traversing an IP network can be modeled as a deterministic propagation delay (incorporated into c_{τ^*}) plus random delays to traverse a series of single-server FIFO queues [6]. The main restrictive assumptions here are that the background traffic is Poisson (for M/M/1 and M/D/1), and packet sizes are exponentially distributed (for M/M/1) or constant (for M/D/1). While these assumptions are not satisfied exactly in practice, studies have shown that when multiplexing a large number of independent flows as in the case of heavy background traffic, the packet arrivals tend to a Poisson process, and the queue length distribution tends to that of a M/D/1 queue [8]. In our evaluations (see Section 5.2), we will stress-test our algorithms derived from these queueing models in a realistic setting which does not follow these models exactly.

3.3.3 Correctness. Alg. 3 provides consistent estimates of the parameters of the shared links in the following sense.

Theorem 3.3. Given an accurate estimate of the shared weight vector $\boldsymbol{W}$, if all the shared links have non-zero metrics, and the estimated average delay ψ_k in line 6 of Alg. 4 is accurate and consistent with the model in (3), then Alg. 3 will accurately estimate the parameters of all the shared links as long as (i) $K > 2$ under the M/M/1 or M/D/1 model, and (ii) $K > 4$ under the G/G/1 model.

3.3.4 Complexity. The number of recursions of Alg. 4 is $O(N_A)$ (N_A: #attack paths) as $\mathcal{T}$ has $O(N_A)$ edges, i.e., the parameter estimation (lines 2–7) is repeated for $O(N_A)$ times. For $K = O(1)$, solving the least square fitting problem (line 7) takes $O(1)$ time as it fits an $O(1)$-variable function at $O(1)$ points. Thus, excluding the measurement time (which is independent of N_A), the complexity of Alg. 3 is $O(N_A)$.

4 OPTIMIZED ATTACK DESIGN

Given the locations and parameters of the shared links, the attacker can use this information to design optimized attacks. To quantify the potential impact of such attacks, we investigate attack strategies that can cause the maximum performance degradation on the target paths at a bounded cost.

4.1 Attacker's Optimization

As a concrete example, we consider the attacker's objective as maximally increasing the total average delay of the target paths. Our approach is extensible to other objectives, as demonstrated in Section 4.3.

Specifically, let W_{ie} denote the shared weight between target path $s_{Bi} \to t_{Bi}$ and edge $e \in \mathcal{T}$ (recall $\mathcal{T}$ denotes the routing tree formed by attack paths) and ξ_{ie} the corresponding queueing parameter (if $W_{ie} > 0$), both inferred as in Section 3. Let $h_{ek} \in \{0, 1\}$ indicate whether attack path $s_A \to t_{Ak}$ traverses edge e, $\beta_i > 0$ denote the importance of target path $s_{Bi} \to t_{Bi}$, and $\tilde{r}_e$ denote the minimum residual capacity of links from s_A to e (excluding e) before attack. Given a total attack rate λ, the attacker wants to find the rate allocation $\bar{\boldsymbol{\lambda}} := (\bar{\lambda}_k)_{k=1}^{N_A}$ that maximizes the weighted sum average delay of all the target paths, i.e.,

$$\max f(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e \in \mathcal{T}: W_{ie} > 0} d(\xi_{ie}; \sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k) \tag{7a}$$

$$\text{s.t.} \sum_{k=1}^{N_A} \bar{\lambda}_k \le \lambda, \tag{7b}$$

$$\sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k \le \tilde{r}_e, \ \forall e \in \mathcal{T}, \tag{7c}$$

$$\bar{\lambda}_k \ge 0, \ k = 1, \ldots, N_A, \tag{7d}$$

where $d(\xi_{ie}; \bar{\lambda})$ represents the average queueing and transmission delay of the link shared between edge $e \in \mathcal{T}$ and target path $s_{Bi} \to t_{Bi}$.

Remark: First, (7a) excludes both the propagation delays and the queueing and transmission delays at links on the target paths that are not shared with any attack path, because these delays are not affected by the attack traffic and thus only contribute a constant shift. *Moreover*, the attacker does not need to know the exact locations of the shared links and their relationships. To explain this, let e_i denote the link shared between edge $e \in \mathcal{T}$ and path $s_{Bi} \to t_{Bi}$. We observe that: (i) e_i will experience the same load $\sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k$ from attack traffic regardless of its exact location on e, and (ii) even if e_i and e_j for $i \ne j$ have some overlap (i.e., sharing links in the underlying network), the load imposed by $s_{Bj} \to t_{Bj}$ on e_i is part of the background traffic that has been incorporated into the parameter ξ_{ie} and vice versa.

4.2 Attack Design

We now derive explicit solutions to (7) under each of the queueing models considered in Section 3.3.2. When the attacker can destabilize the queue at some shared link, i.e., $\exists i \in \{1, \ldots, N_B\}$ and $e \in \mathcal{T}$ with $W_{ie} > 0$ such that $\sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k \ge r_{ie}$ for some $\bar{\boldsymbol{\lambda}}$ satisfying (7b)–(7d) (r_{ie}: residual capacity of the shared link e_i excluding attack traffic), then the attacker should simply allocate sufficient traffic to the attack paths traversing e to congest the shared link e_i and drive the average delay of path $s_{Bi} \to t_{Bi}$ (and hence (7a)) to infinity. Thus, below we will focus on the nontrivial case when

$$\max_{\text{s.t. (7b)–(7d)}} \sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k < \min_{i \in \{1, \ldots, N_B\}: W_{ie} > 0} r_{ie}, \ \forall e \in \mathcal{T}. \tag{8}$$

We will show that in this case, the optimal attack strategy is similar under all the considered queueing models.

4.2.1 Attack under M/M/1. When modeling each shared link as an M/M/1 queue, plugging (4) into (7a) yields

$$f_{\mathrm{M/M/1}}(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e\in\mathcal{T}:W_{ie}>0} \frac{1}{r_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k}, \tag{9}$$

which has the following property:

Lemma 4.1. Under (8), $f_{\mathrm{M/M/1}}(\bar{\boldsymbol{\lambda}})$ is convex in the feasible region of (7).

The convexity of the objective function implies the following property of the optimal solution:

Theorem 4.2. Under (8), the solution $\bar{\boldsymbol{\lambda}}^*$ that maximizes $f_{\mathrm{M/M/1}}(\bar{\boldsymbol{\lambda}})$ s.t. (7b)–(7d) must achieve "=" for N_A of the constraints.

Corollary 4.3. Under $\lambda \le \min_{e\in\mathcal{T}} \tilde{r}_e$ and (8), the solution $\bar{\boldsymbol{\lambda}}^*$ that maximizes $f_{\mathrm{M/M/1}}(\bar{\boldsymbol{\lambda}})$ s.t. (7b)–(7d) must satisfy $\bar{\lambda}_k^* = \lambda$ for some $k \in \{1,\dots,N_A\}$ and $\bar{\lambda}_{k'}^* = 0$ for all $k' \in \{1,\dots,N_A\} \setminus \{k\}$.

For a resource-constrained attacker that faces the case in Corollary 4.3, our analysis shows that the optimal attack strategy is to enumerate all the N_A candidate solutions, each allocating all the attack rate onto a single attack path, and pick the solution maximizing $f_{\mathrm{M/M/1}}(\bar{\boldsymbol{\lambda}})$.

4.2.2 Attack under M/D/1. When modeling each shared link as an M/D/1 queue, plugging (5) into (7a) yields

$$f_{\mathrm{M/D/1}}(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e\in\mathcal{T}:W_{ie}>0} \frac{2\mu_{ie} - \lambda_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k}{2\mu_{ie}(\mu_{ie} - \lambda_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k)}, \tag{10}$$

where μ_{ie} and λ_{ie} are the service/arrival rate at the link shared between $s_{Bi} \to t_{Bi}$ and $e \in \mathcal{T}$ before attack. This objective function has a property similar to $f_{\mathrm{M/M/1}}$:

Lemma 4.4. Under (8) (where $r_{ie} := \mu_{ie} - \lambda_{ie}$), $f_{\mathrm{M/D/1}}(\bar{\boldsymbol{\lambda}})$ is convex in the feasible region of (7).

The same argument as in the proofs of Theorem 4.2 and Corollary 4.3 leads to a similar attack design under M/D/1:

Theorem 4.5. Under (8), the solution $\bar{\boldsymbol{\lambda}}^*$ that maximizes $f_{\mathrm{M/D/1}}(\bar{\boldsymbol{\lambda}})$ s.t. (7b)–(7d) must achieve "=" for N_A of the constraints. Furthermore, if $\lambda \le \min_{e\in\mathcal{T}} \tilde{r}_e$, then $\bar{\boldsymbol{\lambda}}^*$ must satisfy $\bar{\lambda}_k^* = \lambda$ for some $k \in \{1,\dots,N_A\}$ and $\bar{\lambda}_{k'}^* = 0$ for all $k' \in \{1,\dots,N_A\} \setminus \{k\}$.

4.2.3 Attack under G/G/1. When each shared link is modeled as a G/G/1 queue, plugging (6) into (7a) yields

$$f_{\mathrm{G/G/1}}(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e\in\mathcal{T}:W_{ie}>0} \frac{\lambda_{ie} + \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k}{2\mu_{ie}(\mu_{ie} - \lambda_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k)} \cdot \left(\sigma_{aie}^2\big(\lambda_{ie} + \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k\big)^2 + \sigma_{sie}^2\mu_{ie}^2\right), \tag{11}$$

where we have omitted the average service time (i.e., transmission delay) $1/\mu_{ie}$ as it does not depend on the attack traffic. This function is again convex as stated below:

Lemma 4.6. Under (8), $f_{\mathrm{G/G/1}}(\bar{\boldsymbol{\lambda}})$ is convex in the feasible region of (7).

By the same argument as in Theorem 4.2 and Corollary 4.3, Lemma 4.6 implies the following attack design under G/G/1:

Theorem 4.7. Under (8), the solution $\bar{\boldsymbol{\lambda}}^*$ that maximizes $f_{\mathrm{G/G/1}}(\bar{\boldsymbol{\lambda}})$ s.t. (7b)–(7d) must achieve "=" for N_A of the constraints. Furthermore, if $\lambda \le \min_{e\in\mathcal{T}} \tilde{r}_e$, then $\bar{\boldsymbol{\lambda}}^*$ must satisfy $\bar{\lambda}_k^* = \lambda$ for some $k \in \{1,\dots,N_A\}$ and $\bar{\lambda}_{k'}^* = 0$ for all $k' \in \{1,\dots,N_A\} \setminus \{k\}$.

Remark: The above analysis shows that the optimal strategy for a resource-constrained attacker is to focus all the attack traffic on a single attack path, selected based on the locations and parameters of the shared links learned through the reconnaissance techniques presented in Section 3.

4.3 Other Attack Objectives

If the total attack rate violates (8), i.e., the attacker can destabilize the queue for at least one shared link, an objective that can differentiate the different ways of destabilizing queues is needed. As a concrete example, if the attacker wants to cause the worst congestion on any of the shared links, he can maximize the following objective function:

$$\max_{e\in\mathcal{T}:\exists W_{ie}>0}\left(\sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k - \min_{i\in\{1,\ldots,N_B\}:W_{ie}>0} r_{ie}\right), \tag{12}$$

subject to constraints (7b)–(7d), which will maximize the *maximum excess load* on a shared link. Maximizing (12) s.t. (7b)–(7d) is a maximization of a piece-wise linear convex function under linear constraints, for which the optimal solution must be achieved at an extreme point of the feasible region [23]. In our context, this will be a vertex of the polytope defined by (7b)–(7d), where "=" is achieved for N_A of the constraints. In the special case of $\lambda \le \min_{e\in\mathcal{T}} \tilde{r}_e$, (7c) is redundant, and thus the optimal attack must allocate all the attack traffic onto a single path as in the case of optimizing the objective (7a). Note that the new objective (12) is invariant to the queueing model. The above result together with the results of Section 4.2 suggests the efficacy of the generic attack strategy that focuses resources on an attack path selected based on the information learned through reconnaissance. When the objectives in (7a) and (12) are both applicable, (7a) is usually a more meaningful objective for the attacker as it represents the end-to-end performance impact across all the target paths. Nevertheless, these are just concrete examples of the attacker's objectives to illustrate the impact of adversarial reconnaissance. What objective is most suitable will depend on the application scenario and is left to future work.

5 PERFORMANCE EVALUATION

In this study, we evaluate the performance of our algorithms under **two types** of networks using NS3 [22], a widely-used discrete-event network simulator. First, we conduct simulations in the context of an IP-based backbone network (Section 5.1). Then, we validate our results by repeating the experiments in the context of a 5G Integrated Access and Backhaul (IAB) network (Section 5.2), leveraging the 5G-LENA module [35] for the radio access network (RAN).

5.1 NS3-based Simulation of Backbone Network

5.1.1 Simulation Setup. We simulate an IP-based backbone network based on GtsCe (GTS Central Europe) from the Internet Topology Zoo [26], which is a network with 149 nodes and 193 links. Following [19], we set the link capacities and delays using the dataset from [18], in which all link capacities are treated as 1 Gbps. In Appendix B.4, we additionally study a case with higher link capacities, which yields similar results. We generate attack paths by randomly picking a source s_A and N_A destinations $\{t_{Ai}\}_{i=1}^{N_A}$ from the network and computing the shortest paths (in hop count). We generate target paths $\{s_B \to t_{Bi}\}_{i=1}^{N_B}$ similarly, while ensuring that each target path shares at least one link with the attack paths. Here each node in GtsCe represents a point of presence (PoP) so that multiple source/destination hosts can attach to the same node (through 'other links' outside the simulated network). Fig. 4 shows an example topology formed by the generated paths.

To evaluate the robustness of our approach, we have examined its performance under two types of background traffic. In the experiments presented here, we generate background traffic by a

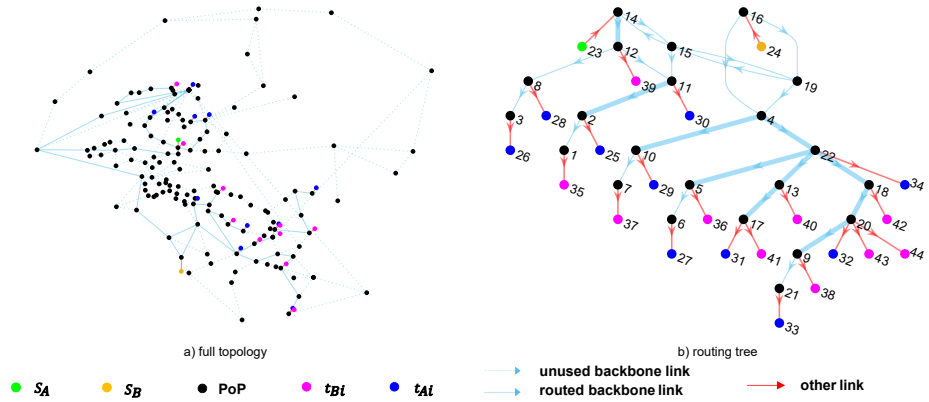


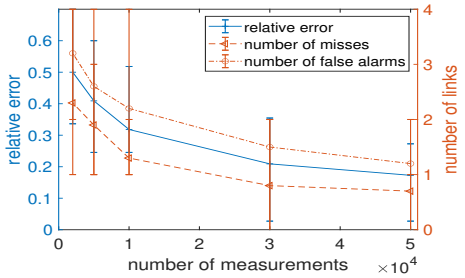
Fig. 4. Sample topology in the simulation of backbone network ($N_A = N_B = 10$), with shared links highlighted as thick lines.

recently proposed methodology from [5], wherein the background traffic rate for each link is periodically sampled from a log-normal distribution characterized by parameters (μ, σ). In this study, we regenerate the rate every 0.5 ms and set σ as 1. Following [25], each background packet has a size randomly selected from 50, 576 and 1460 bytes with probabilities 0.4, 0.2 and 0.4, respectively. The μ-parameter of background traffic is designed to achieve a total utilization that is randomly distributed in [10%, 50%] prior to attack. In Appendix B.2, we provide additional simulation results under background traffic generated according to ON-OFF processes as in Section 5.2. Here, we set $N_A = N_B = 10$, while in Appendix B.3, we additionally study the case of $N_A = 20$. All the additional studies yield qualitatively similar results.

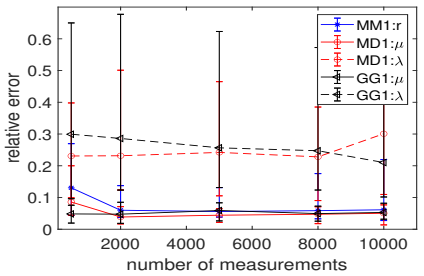
We configure each link to have a FIFO queue with a large buffer to guarantee no packet loss during the simulation. We set the rate on each target path to 50 Mbps with a constant packet size of 1000 bytes. The packet size on each attack path is 50 bytes for shared link detection and 1000 bytes for parameter inference and attack. We set the importance of target paths to $\beta_i = 1$ for $i = 1, \ldots, N_B$. All our results are based on 20 Monte Carlo runs.

In shared weight inference (Alg. 1), we consider a packet as not incurring queueing on a path if its end-to-end delay is smaller than the mean of the 10 smallest delays on this path plus 0.1 ms. We detect a shared link exists between a target path $s_B \to t_{Bi}$ and an edge $e \in \mathcal{T}$ if the inferred value of W_{ie} exceeds 0.005. To reduce correlation across measurements, we maintain a spacing of at least 2 ms between consecutive measurements. Since the distances from s_A and s_B to the shared links may be different, we find an offset κ by correlation maximization to identify measurements forming a mimicked multicast, as detailed in Appendix B.1. In parameter inference, we vary the probing rate among $K = 20$ values evenly distributed between 0 and 80% of the minimum residual capacity at shared links, and solve the least square fitting problem (line 7 in Alg. 4) by the trust-region-reflective least squares algorithm [11].

5.1.2 Results on Reconnaissance. Fig. 5 (a) shows the accuracy in detecting shared links, measured by the fraction of errors in inferring whether W_{ie} is non-zero for all $i = 1, \ldots, N_B$ and $e \in \mathcal{T}$. In addition, we also evaluate the number of false alarms (detected shared links that do not exist) and the number of misses (shared links that are not detected) averaged over all the target paths. Each measurement here corresponds to a mimicked tri-cast. The results show that our algorithms (Alg. 1–2) can detect the majority of the shared links with some errors (around 20% error if we collect 5×10^4 measurements for each tri-cast for both calibration and detection). Among the errors, there are more false alarms than misses.

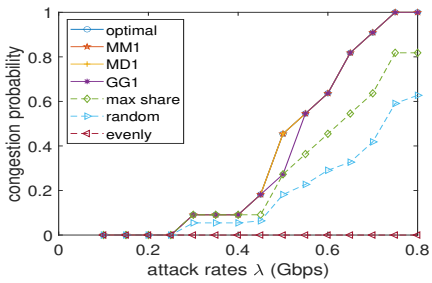


(a) Performance in detecting shared links

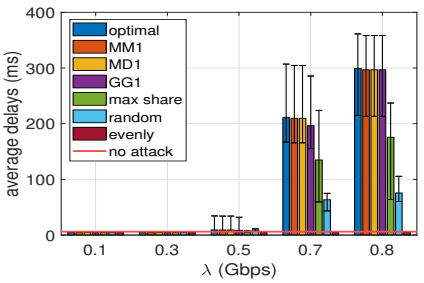


(b) Performance in inferring shared link parameters

Fig. 5. Performance of reconnaissance in backbone network simulation ($N_A = N_B = 10$).



(a) Probability of congesting at least one shared link



(b) Average delay over all the target paths

Fig. 6. Performance of attack design in backbone network simulation ($N_A = N_B = 10$).

Fig. 5 (b) shows the accuracy of inferring the parameters of the shared links, measured by the relative error $\|\hat{\xi} - \xi^*\|_1 / \|\xi^*\|_1$ ($\hat{\xi}$: estimate, ξ^*: ground truth). Although the queues at the links do not exactly follow any of the assumed queueing models, we can still compare the estimated parameters with the best-fitting parameters based on per-link measurements. The results show that: (i) although the real queueing behavior does not exactly fit any of the assumed queueing models, the inference results based on these models are reasonable, (ii) while the link capacities ('μ') and the residual capacities ('r') can be inferred with good accuracy (< 10% of error), there is notable error in estimating the background traffic loads ('λ'), and (iii) G/G/1-based estimation performs slightly worse due to the difficulty of jointly estimating more parameters. There are two other parameters (variance of interarrival/service time) under G/G/1, for which the trend is similar. Although the inference process involves active probing, each probing experiment only lasts for a short period (e.g., 0.8 seconds for 5000 measurements, each corresponding to a packet on a target path).

5.1.3 Results on Attack Design. Since the original design of cross-path attack [9] only ensures to use some attack paths that share at least one link with the target paths, we compare the proposed attack design with the following intuitive rate allocation strategies over such attack paths[3]:

(1) 'Evenly': A natural strategy is to evenly split the total attack rate λ among all the attack paths that share at least one link with the target paths.
(2) 'Random': Given that the optimal strategy is usually to focus on one path (see Section 4.2), the attacker may also allocate all the rate to a randomly selected attack path.
(3) 'Max share': The attacker chooses the attack path traversing the maximum number of shared tree links, i.e., $t_A^* = \max_{t_{Ak} \in T_A} \{\sum_{e \in \mathcal{T}} h_{ek} \mathbb{I}(\sum_{i=1}^{N_B} W_{i,e} > 0)\}$ ($\mathbb{I}(\cdot)$: indicator function).

[3]The set of attack paths sharing at least one link with the target paths can be inferred by a simple reconnaissance method proposed in [9]. Here, we use the true set for a conservative comparison.

The results presented in Fig. 6 show that despite containing notable error, the information obtained by our reconnaissance algorithms is still useful for attack design. Here, 'optimal' denotes the optimal attack designed based on the true locations/parameters of the shared links, and 'MM1'/'MD1'/'GG1' denotes the proposed attack design based on the parameters inferred from 10,000 measurements under the model of M/M/1 or M/D/1 or G/G/1. Fig. 6 (a) shows the probability that the attack can congest (i.e., destabilize) at least one shared link. The results show that: (i) the proposed reconnaissance-based optimized attack design ('MM1', 'MD1', 'GG1') can achieve near-optimal impact despite the notable estimation errors, (ii) the non-optimized attack strategies based on [9] ('random', 'even') are much less effective, and (iii) knowing the locations of shared links ('max share') helps but is not enough. A closer examination shows that the estimated parameters can reveal which attack paths traverse the weakest shared link (the one with the minimum residual capacity), even if the estimated parameters are inaccurate. In Fig. 6 (b), we evaluate the impact of attacks on the delays of the target paths, computed over 10,000 measurements. As the objective of delay maximization is only meaningful at attack rates that are within the stability region, we combine multiple attack designs as follows: when $\lambda \leq \min_{e\in\mathcal{T}} \tilde{r}_e$ which satisfies the condition of Corollary 4.3, the attacker will send all the attack traffic on the attack path predicted to maximize the delay increase over all the target paths; when $\lambda > \min_{e\in\mathcal{T}} \tilde{r}_e$, the attacker will maximize the maximum excess load (12) as in Section 4.3. We observe that (i) the proposed attack designs produce near-optimal delay increase regardless of the assumed queueing model, and (ii) there is a wide variation among the impacts of different cross-path attacks, where the carefully-designed attacks can generate a substantially higher performance impact than the straightforward attacks. These observations signal the importance of considering intelligent attack models in security analysis.

5.2 NS3-based Simulation of Integrated Access and Backhaul (IAB) Network

5.2.1 Simulation Setup. To test the generalizability of our observations, we repeat our experiments in the scenario of an IAB network with multiple slices. IAB network is a form of backhaul for 5G [3], where base stations (BS) are implemented as IAB nodes, among which only a subset of nodes (called IAB donors) are connected to the 5G core through fiber. An IAB node has both a DU and a mobile termination function. Thus, it can function not only as a traditional BS for UEs, but also as a relay for other IAB nodes through millimeter wave. In the process of downlink transmission, parent IAB nodes relay traffic to their child IAB nodes, and the process is reversed for uplink transmission. We simulate the IAB-UE links by 5G-LENA [35], which is a pluggable module in NS3 for simulating 5G RAN, and the rest of the links by point-to-point links[4].

Following [3], we consider an IAB network with 19 BSs in a hexagonal topology with one IAB donor at the center as illustrated in Fig. 7 (a). The network is shared by a slice A containing attack paths, a slice B containing target paths, and other slices treated as background traffic. We focus on downlink communication, where packets enter the IAB network through the IAB donor (node 2) and are then routed towards their destination UEs along a routing tree rooted at the donor. The links in the routing tree are highlighted as thick lines in Fig. 7 (a) and also depicted in Fig. 7 (b). We assume that there is at least one UE in slice A in each cell, and the UEs in slice B are randomly distributed among the cells. According to [3], we assign each slice a separate Bandwidth Part (BWP) for the IAB-UE links. We set the numerology in 5G-LENA to 5.

Following [3, 38], we set the capacity of IAB-IAB links to 2 Gbps, IAB-UE links to 0.5 Gbps, and fiber links to 100 Gbps. We limit the total flow rate to each cell in slice A to 1 Gbps. We set the flow rate for each UE in slice B to 0.1 Gbps to represent emerging applications like panoramic

[4]Although the IAB-IAB links are supposed to be through millimeter wave [3], this feature is not officially supported in NS3 to our knowledge, and hence we mimic them by point-to-point links with lower capacities than the fiber links.

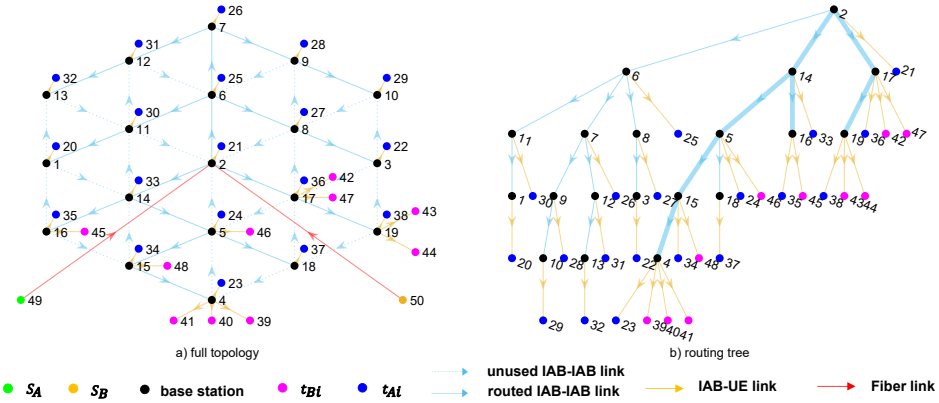
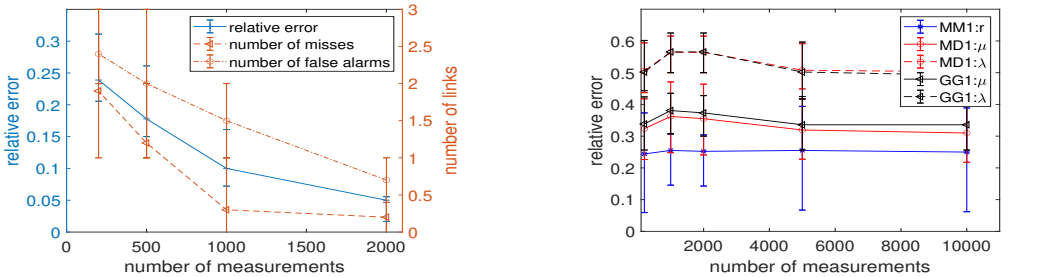


Fig. 7. Sample topology in the simulation of IAB network ($N_A = 19, N_B = 10$), with shared links highlighted as thick lines.



(a) Performance in detecting shared links

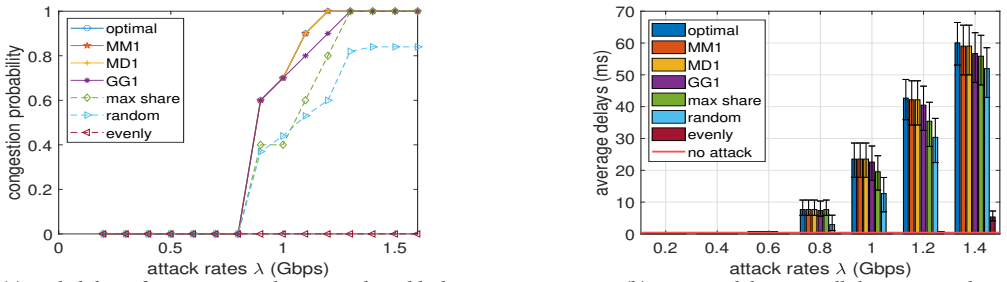
(b) Performance in inferring shared link parameters

Fig. 8. Performance of reconnaissance in IAB network simulation ($N_A = 19, N_B = 10$).

video streaming [1, 13]. Following [24, 39], we independently generate background traffic on each IAB-IAB link according to an ON-OFF process. The duration of each ON period follows the Pareto distribution with shape parameter set to 2.04 and scale parameter ζ_{ON} set to the average length of 13 packets. The duration of each OFF period follows the same distribution with a different scale parameter ζ_{OFF}, tuned to yield a link utilization randomly drawn from $[15\%, 35\%]$. To detect no queueing events for shared weight inference, we measure the delays during light traffic and set a threshold based on the 3σ rule. The rest of the setup is the same as that in Section 5.1. In the sequel, we will present our results in the case of $N_B = 10$. More results are given in Appendix C.

5.2.2 Results on Reconnaissance. First, we evaluate the accuracy of shared link detection as in Fig. 5 (a). The error in shared link detection is shown in Fig. 8 (a). The results show similar observations as Fig. 5 (a): the proposed algorithms (Alg. 1–2) can detect the shared links with good accuracy ($< 5\%$ error), and the errors are mostly due to false alarms.

In Fig. 8 (b), we evaluate the accuracy of parameter inference under each of the queueing models as in Fig. 5 (b). Similar to Fig. 5 (b), we observe that (i) the residual capacity ('r') and the capacity ('μ') can be estimated more accurately than the load ('λ'), and (ii) G/G/1-based estimation performs slightly worse. The main difference from Fig. 5 (b) is that the errors become larger. This is because the delays in the IAB network are affected by not only queueing in the backhaul but also MAC scheduling at the IAB-UE links. We also notice that the proposed parameter estimation method can help detect false alarms in shared link detection, as detailed in Appendix D.



(a) Probability of congesting at least one shared link

(b) Average delay over all the target paths

Fig. 9. Performance of attack design in IAB network simulation ($N_A = 19, N_B = 10$).

5.2.3 Results on Attack Design. We evaluate our attack design in comparison with the same benchmarks as in Section 5.1.3. The results, presented in Fig. 9, show similar observations as Fig. 6: (i) the attacks designed based on the results of our reconnaissance algorithms ('MM1', 'MD1', 'GG1') perform close to the optimal in terms of both the probability of congestion and the delay increase, and (ii) the proposed optimized attacks generate a higher performance impact than the straightforward attacks according to [9], especially under a limited total attack rate. Compared to Fig. 6, the gap between the optimized attacks and the baselines is smaller in Fig. 9. This is because on the average more links are shared between the attack paths and the target paths in the IAB network due to the single ingress point (the IAB donor), as shown in Fig. 7, making it easier to impact the target paths by launching attack on randomly selected attack paths.

6 CONCLUDING DISCUSSION

We studied a newly identified stealthy DoS attack called cross-path attack, with focus on quantifying the maximum impact of such attacks through optimized attack design. To this end, we developed a novel extension to network topology inference that allows the attacker to consistently estimate the locations and parameters of the links shared between the attack paths and the target paths by only passively monitoring the target paths, and provided an efficient method to compute the optimal attack rate allocation based on the estimated information. Our optimized attack achieved a much greater performance impact than its non-optimized counterparts in high-fidelity simulations. Besides quantifying the maximum impact of cross-path attacks, our work also sheds light on possible defenses. The root cause of such attacks is the sharing of network resources across flows of different levels of security. Although completely isolating flows (e.g., by assigning each flow a fixed share of bandwidth) can prevent cross-path attacks, it also sacrifices the benefits of resource sharing such as throughput elasticity and higher resource utilization. Meanwhile, allowing unlimited resource sharing will make the network vulnerable to malicious abuses of the shared resources as demonstrated in our work. Intuitively, an effective network design should strike a balance between the benefit of elastic resource allocation and the risk of abused elasticity. Determining the right balance will depend on a variety of factors, such as the capacity of the resource, the criticality of the supported application, and the perceived level of threat, which may vary over time. Due to the inherent ambiguity between attack traffic maliciously consuming resources and normal bursty traffic genuinely in need of more resources, the network will face an inevitable tradeoff between performance and security, the detailed investigation of which is left to future work.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under awards CNS-2106294 and CNS-1946022.

REFERENCES

[1] 2017. Preparing for a Cloud AR/VR Future. Huawei public white paper. https://www-file.huawei.com/-/media/corporate/pdf/x-lab/cloud_vr_ar_white_paper_en.pdf.

[2] 3GPP. 2016. Feasibility study on new services and markets technology enablers for network operation; Stage 1. TR 22.864. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3016

[3] 3GPP. 2018. NR; Study on integrated access and backhaul. TR 38.874. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3232

[4] Stefan Achleitner, Thomas La Porta, Trent Jaeger, and Patrick McDaniel. 2017. Adversarial Network Forensics in Software Defined Networking. In *ACM Symposium on SDN Research (SOSR)*. ACM.

[5] Mohammed Alasmar, Richard Clegg, Nickolay Zakhleniuk, and George Parisis. 2021. Internet traffic volumes are not Gaussian—They are log-normal: An 18-year longitudinal study with implications for modelling and prediction. *IEEE/ACM Transactions on Networking* 29, 3 (2021), 1266–1279.

[6] F. Baccelli, B. Kauffmann, and D. Veitch. 2009. Inverse problems in queueing theory and Internet probing. *Queueing Systems* 63 (2009), 59â€“107.

[7] Gregory Berkolaiko, Nick Duffield, Mahmood Ettehad, and Kyriakos Manousakis. 2018. Graph reconstruction from path correlation data. *Inverse Problems* 35, 1 (November 2018), 015001.

[8] Jin Cao, William S. Cleveland, Dong Lin, and Don X. Sun. 2001. On the Nonstationarity of Internet Traffic. *SIGMETRICS Performance Evaluation Review* 29, 1 (June 2001), 102–112.

[9] Jiahao Cao, Qi Li, Renjie Xie, Kun Sun, Guofei Gu, Mingwei Xu, and Yuan Yang. 2019. The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links. In *USENIX Security*.

[10] M. Coates, R. Castro, M. Gadhiok, R. King, Y. Tsang, and R. Nowak. 2002. Maximum Likelihood Network Topology Identification from Edge-based Unicast Measurements. In *ACM SIGMETRICS*.

[11] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. 2009. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics.

[12] Vitor A Cunha, Eduardo da Silva, Marcio B de Carvalho, Daniel Corujo, Joao P Barraca, Diogo Gomes, Lisandro Z Granville, and Rui L Aguiar. 2019. Network slicing security: Challenges and directions. *Internet Technology Letters* 2, 5 (2019), e125.

[13] Luca De Cicco, Saverio Mascolo, Vittorio Palmisano, and Giuseppe Ribezzo. 2019. Reducing the network bandwidth requirements for 360° immersive video streaming. *Internet Technology Letters* 2, 4 (2019), e118.

[14] Mohan Dhawan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. 2015. SPHINX: Detecting Security Attacks in Software-Defined Networks. In *Network and Distributed System Security Symposium*. https://doi.org/10.14722/ndss.2015.23064

[15] William F Donoghue. 2014. *Distributions and Fourier transforms*. Academic Press.

[16] Nick G Duffield, Joseph Horowitz, and F Lo Prestis. 2001. Adaptive multicast topology inference. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, Vol. 3. IEEE, 1636–1645.

[17] Xinzhe Fu and Eytan Modiano. 2019. Network Interdiction Using Adversarial Traffic Flows. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1765–1773. https://doi.org/10.1109/INFOCOM.2019.8737475

[18] Steven Gay, Pierre Schaus, and Stefano Vissicchio. 2017. Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms. *arXiv preprint arXiv:1710.08665* (2017).

[19] Nikola Gvozdiev, Stefano Vissicchio, Brad Karp, and Mark Handley. 2018. On low-latency-capable topologies, and their impact on the design of intra-domain routing. In *SIGCOMM*. 88–102.

[20] Peter Harrison and Naresh M. Patel. 1992. *Performance Modelling of Communication Networks and Computer Architectures*. Addison–Wesley.

[21] Ting He, Liang Ma, Ananthram Swami, and Don Towsley. 2021. *Network Tomography: Identifiability, Measurement Design, and Network State Inference*. Cambridge University Press.

[22] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. 2008. Network simulations with the ns-3 simulator. *SIGCOMM demonstration* 14, 14 (2008), 527.

[23] Karla Leigh Hoffman. 1981. A METHOD FOR GLOBALLY MINIMIZING CONCAVE FUNCTIONS OVER CONVEX SETS. *Mathematical Programming* 20 (1981), 22–32.

[24] Hao Jiang and Constantinos Dovrolis. 2005. Why is the internet traffic bursty in short time scales?. In *SIGMETRICS*. 241–252.

[25] Wolfgang John and Sven Tafvelin. 2007. Analysis of internet backbone traffic and header anomalies observed. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 111–116.

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

[27] Zbigniew Kotulski, Tomasz Wojciech Nowak, Mariusz Sepczuk, Marcin Tunia, Rafal Artych, Krzysztof Bocianiak, Tomasz Osko, and Jean-Philippe Wary. 2018. Towards constructive approach to end-to-end slice isolation in 5G networks. *EURASIP Journal on Information Security* 2018, 1 (2018), 1–23.

[28] Xin Li, Mohammed Samaka, H Anthony Chan, Deval Bhamare, Lav Gupta, Chengcheng Guo, and Raj Jain. 2017. Network slicing for 5G: Challenges and opportunities. *IEEE Internet Computing* 21, 5 (2017), 20–27.

[29] Yilei Lin, Ting He, and Guodong Pang. 2021. Queuing Network Topology Inference Using Passive Measurements.

[30] Y. Lin, T. He, S. Wang, K. Chan, and S. Pasteris. 2019. Multicast-Based Weight Inference in General Network Topologies. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC.2019.8761099

[31] Yilei Lin, Ting He, Shiqiang Wang, Kevin Chan, and Stephen Pasteris. 2020. Looking Glass of NFV: Inferring the Structure and State of NFV Network From External Observations. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1477–1490. https://doi.org/10.1109/TNET.2020.2985908

[32] Qiang Liu, Tao Han, and Nirwan Ansari. 2020. Learning-assisted secure end-to-end network slicing for cyber-physical systems. *IEEE Network* 34, 3 (2020), 37–43.

[33] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang. 2010. Efficient and Dynamic Routing Topology Inference From End-to-End Measurements. *IEEE/ACM Transactions on Networking* 18, 1 (2010), 123–135. https://doi.org/10.1109/TNET.2009.2022538

[34] Ruxandra F Olimid and Gianfranco Nencioni. 2020. 5G network slicing: a security overview. *IEEE Access* 8 (2020), 99999–100009.

[35] Natale Patriciello, Sandra Lagen, Biljana Bojovic, and Lorenza Giupponi. 2019. An E2E simulator for 5G NR networks. *Simulation Modelling Practice and Theory* 96 (2019), 101933.

[36] Cynthia A. Phillips. 1993. The Network Inhibition Problem. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*. 776–785.

[37] M. Rabbat, M. Coates, and R. Nowak. 2006. Multiple Source Internet Tomography. *IEEE Journal on Selected Areas in Communications* 24, 12 (December 2006), 2221–2234.

[38] Henrik Ronkainen, Jonas Edstam, Anders Ericsson, and Christer Östberg. 2020. Integrated access and backhaul – a new type of wireless backhaul in 5G. Ericsson Technology Review. https://www.ericsson.com/4ac691/assets/local/reports-papers/ericsson-technology-review/docs/2020/introducing-integrated-access-and-backhaul.pdf

[39] Matthew Roughan and Charles Kalmanek. 2003. Pragmatic modeling of broadband access traffic. *Computer Communications* 26, 8 (2003), 804–816.

[40] Josep Xavier Salvat, Lanfranco Zanzi, Andres Garcia-Saavedra, Vincenzo Sciancalepore, and Xavier Costa-Perez. 2018. Overbooking Network Slices through Yield-Driven End-to-End Orchestration. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (Heraklion, Greece) *(CoNEXT '18)*. Association for Computing Machinery, New York, NY, USA, 353–365. https://doi.org/10.1145/3281411.3281435

[41] Danish Sattar and Ashraf Matrawy. 2019. Towards secure slicing: Using slice isolation to mitigate DDoS attacks on 5G core network slices. In *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 82–90.

[42] Pegah Sattari, Maciej Kurant, Animashree Anandkumar, Athina Markopoulou, and Michael G. Rabbat. 2014. Active Learning of Multiple Source Multiple Destination Topologies. *IEEE Transactions on Signal Processing* 62, 8 (2014), 1926–1937. https://doi.org/10.1109/TSP.2014.2304431

[43] NGMN 5G security group. 2016. 5G security recommendations Package 2: Network Slicing. NGMN Alliance. https://ngmn.org/wp-content/uploads/Publications/2016/160429_NGMN_5G_Security_Network_Slicing_v1_0.pdf

[44] Gao Shang, Peng Zhe, Xiao Bin, Hu Aiqun, and Ren Kui. 2017. FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2017.8057009

[45] S. Shin, V. Yegneswaran, P. Porras, and G. Gu. 2013. AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks. In *ACM CCS*.

[46] Kyu-Seek Sohn, Seung Yeob Nam, and Dan Keun Sung. 2006. A distributed LSP scheme to reduce spare bandwidth demand in MPLS networks. *IEEE transactions on communications* 54, 7 (2006), 1277–1288.

[47] Haopei Wang, Lei Xu, and Guofei Gu. 2015. FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks. In *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 239–250. https://doi.org/10.1109/DSN.2015.27

[48] Xiaowei Yang. 2002. Designing traffic profiles for bursty internet traffic. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, Vol. 3. IEEE, 2149–2154.

[49] H. Yao, S. Jaggi, and M. Chen. 2012. Passive Network Tomography for Erroneous Networks: A Network Coding Approach. *IEEE Transactions on Information Theory* 58, 9 (September 2012), 5922–5940.

[50] Mingli Yu, Tian Xie, Ting He, Patrick McDaniel, and Quinn K. Burke. 2021. Flow Table Security in SDN: Adversarial Reconnaissance and Intelligent Attacks. *IEEE/ACM Transactions on Networking* 29, 6 (2021), 2793–2806. https://doi.org/10.1109/TNET.2021.3099717

[51] Shunliang Zhang. 2019. An overview of network slicing for 5G. *IEEE Wireless Communications* 26, 3 (2019), 111–117.

A PROOFS OF THEOREMS

Proof of Theorem 3.2. First, we prove that given accurate estimates of the category weights, the shared weight on the stem of each tree considered by Alg. 2 will be accurately inferred. Since $s_A \to \tau_1$ and $s_A \to \tau_2$ branches at $b_{\mathcal{T}'}$, ρ_s is the shared weight on $s_A \to b_{\mathcal{T}'}$. Moreover, as Alg. 2 works in a top-down manner, the shared weights on edges above $s_{\mathcal{T}'}$ must have been inferred before considering $\mathcal{T}'$. Thus, $\rho_s - W_{s_A \to s_{\mathcal{T}'}}$ (line 4) must be the true shared weight on edge $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$.

Moreover, we prove that every edge with non-zero shared weight will be the stem of a tree considered by Alg. 2. After inferring the shared weight on the stem of $\mathcal{T}'$, Alg. 2 will perform recursion for both subtrees of $\mathcal{T}'$ to consider the remaining edges except for two cases. The first case is when $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ (line 5), in which case $\mathcal{T}'$ has no other edge. The second case is when $\rho_s \neq 0$ and $\rho_l = 0$ (or $\rho_r = 0$), in which case we can skip the left (or right) subtree of $\mathcal{T}'$ as all its edges have zero shared weight. To see this, suppose that $\rho_s \neq 0$ and $\rho_l = 0$, but $\exists$ edge e in the left subtree of $\mathcal{T}'$ with $W_e \neq 0$. Let e' be the stem of the left subtree. Suppose that $s_B \to t_B$ intersects with $s_A \to b_{\mathcal{T}'}$ at node v_1 (which exists because $\rho_s \neq 0$), and intersects with e at node v_2 (which exists because $W_e \neq 0$). Then there exist two routing paths between v_1 and v_2, one follows $\mathcal{T}$ and traverses e', and the other follows $s_B \to t_B$ without traversing e' (as $\rho_l = 0$), which contradicts with the unique route assumption in Section 2.1. Similar argument holds for $\rho_s \neq 0$ and $\rho_r = 0$. □

Proof of Theorem 3.3. First, we argue that all the shared inks will be considered in parameter estimation. As the given weight vector $\boldsymbol{W}$ is accurate and all the shared links have non-zero metrics, every edge e containing a shared link will have $W_e > 0$, and thus will be considered in the parameter estimation when e is the stem of the tree $\mathcal{T}'$ under consideration. As the recursion examines the edges of $\mathcal{T}$ in a top-down manner, it remains to show that when $W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} > 0$, we can safely skip one of the subtrees of $\mathcal{T}'$ as long as $\delta_{l\mathcal{T}'} \neq \delta_{r\mathcal{T}'}$ (otherwise $\mathcal{T}'$ only has one edge, i.e., the stem). This is because conditioned on $W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} > 0$, $s_B \to t_B$ cannot intersect with both of the subtrees of $\mathcal{T}'$, or there will be a contradiction with the unique route assumption in Section 2.1.

Next, we argue that the parameter for each shared link considered in lines 2–7 of Alg. 4 will be estimated accurately. We start by considering the top-most shared link, assumed to reside on an edge $e \in \mathcal{T}$. Our selection of the probing destination τ^* ensures that it is the only shared link between $s_B \to t_B$ and $s_A \to \tau^*$, for which the objective of the least square fitting in line 7 is reduced to $\sum_{k=1}^{K} \left(\psi_k - c_{\tau^*} - d(\xi_e; \bar{\lambda}_k)\right)^2$. Let $(c^*_{\tau^*}, \xi^*_e)$ denote the ground truth parameters. By our assumption, $(c^*_{\tau^*}, \xi^*_e)$ achieves a zero fitting error. Suppose that the estimated parameters $(\hat{c}_{\tau^*}, \hat{\xi}_e) \neq (c^*_{\tau^*}, \xi^*_e)$. Then $(\hat{c}_{\tau^*}, \hat{\xi}_e)$ must also achieve a zero fitting error, i.e.,

$$\hat{c}_{\tau^*} + d(\hat{\xi}_e; \bar{\lambda}_k) = c^*_{\tau^*} + d(\xi^*_e; \bar{\lambda}_k),\ k = 1, \ldots, K. \tag{13}$$

Under M/M/1, plugging (4) into (13) implies that $\bar{\lambda}_k$ $(k = 1, \ldots, K)$ must all satisfy

$$\hat{c}_{\tau^*} + \frac{1}{\hat{r}_e - \bar{\lambda}} = c^*_{\tau^*} + \frac{1}{r^*_e - \bar{\lambda}}. \tag{14}$$

For $K > 2$, this leads to a contradiction as (14) is a quadratic equation in $\bar{\lambda}$ with at most two distinct solutions. Similarly, under M/D/1, plugging (5) into (13) gives a quadratic equation of $\bar{\lambda}$ with at most two distinct solutions, contradicting with $K > 2$; under G/G/1, plugging (6) into (13) gives a quartic equation of $\bar{\lambda}$ with at most four distinct solutions, contradicting with $K > 4$. The same argument applies to every other shared link, as our selection of the probing destination ensures that when estimating ξ_e, all the other shared links between the target path and the probing path are above e, whose parameters should already be accurately inferred by induction. □

Proof of Lemma 4.1. As $f_{\text{M/M/1}}$ is a non-negative linear combination of functions of the form $g_1(\bar{\boldsymbol{\lambda}}) := \frac{1}{p-\sum_{i=1}^{N_A} q_i\bar{\lambda}_i}$, where $p - \sum_{i=1}^{N_A} q_i\bar{\lambda}_i > 0$, it suffices to prove that $g_1(\bar{\boldsymbol{\lambda}})$ is convex.

To this end, it suffices to show that for any $\bar{\boldsymbol{\lambda}}_j$ $(j = 1, 2)$ satisfying $p-\sum_{i=1}^{N_A} q_i\bar{\lambda}_{ji} > 0$, $g_1(\bar{\boldsymbol{\lambda}}_1)+g_1(\bar{\boldsymbol{\lambda}}_2) \geq 2g_1(\frac{\bar{\boldsymbol{\lambda}}_1+\bar{\boldsymbol{\lambda}}_2}{2})$, since a continuous function that is midpoint convex must be convex [15]. The proof completes by ($\Leftrightarrow$ means equivalence):

$$
\begin{aligned}
&\frac{1}{p - \sum_{i=1}^{N_A} q_i\bar{\lambda}_{1i}} + \frac{1}{p - \sum_{i=1}^{N_A} q_i\bar{\lambda}_{2i}} \geq \frac{2}{p - \sum_{i=1}^{N_A} q_i\frac{\bar{\lambda}_{1i}+\bar{\lambda}_{2i}}{2}} \\
&\Leftrightarrow (\sum_{i=1}^{N_A} q_i\bar{\lambda}_{1i})(\sum_{i=1}^{N_A} q_i\frac{\bar{\lambda}_{1i} + \bar{\lambda}_{2i}}{2}) \\
&+ (\sum_{i=1}^{N_A} q_i\bar{\lambda}_{2i})(\sum_{i=1}^{N_A} q_i\frac{\bar{\lambda}_{1i} + \bar{\lambda}_{2i}}{2}) \geq 2(\sum_{i=1}^{N_A} q_i\bar{\lambda}_{1i})(\sum_{i=1}^{N_A} q_i\bar{\lambda}_{2i}) \\
&\Leftrightarrow (\sum_{i=1}^{N_A} q_i\bar{\lambda}_{1i})^2 + (\sum_{i=1}^{N_A} q_i\bar{\lambda}_{2i})^2 \geq 2(\sum_{i=1}^{N_A} q_i\bar{\lambda}_{1i})(\sum_{i=1}^{N_A} q_i\bar{\lambda}_{2i}) \\
&\Leftrightarrow \left(\sum_{i=1}^{N_A} q_i\bar{\lambda}_{1i} - \sum_{i=1}^{N_A} q_i\bar{\lambda}_{2i}\right)^2 \geq 0. \qquad (15)
\end{aligned}
$$

□

Proof of Theorem 4.2. By Lemma 4.1, the attacker's optimization is a maximization of a convex function over a polytope defined by (7b)–(7d), for which the optimal solution must be achieved at an extreme point of the feasible region [23]. In our context, this will be a vertex of the polytope, which achieves "=" for N_A of the constraints in (7b)–(7d). □

Proof of Corollary 4.3. Under $\lambda \leq \min_{e\in\mathcal{T}} \tilde{r}_e$, the constraint in (7c) can be ignored. The remaining constraints define a polytope with only N_A non-zero vertices, each in the form of $\bar{\lambda}_k^* = \lambda$ and $\bar{\lambda}_{k'}^* = 0$ for all $k' \in \{1, \ldots, N_A\} \setminus \{k\}$. The optimal solution must be one of them by Theorem 4.2. □

Proof of Lemma 4.4. As $f_{\text{M/D/1}}$ is a non-negative linear combination of functions of the form

$$
g_2(\bar{\boldsymbol{\lambda}}) := \frac{2\mu - \lambda - \sum_{i=1}^{N_A} q_i\bar{\lambda}_i}{2\mu(\mu - \lambda - \sum_{i=1}^{N_A} q_i\bar{\lambda}_i)}, \qquad (16)
$$

where $\mu - \lambda - \sum_{i=1}^{N_A} q_i\bar{\lambda}_i > 0$, it suffices to prove that $g_2(\bar{\boldsymbol{\lambda}})$ is convex. To this end, note that

$$
g_2(\bar{\boldsymbol{\lambda}}) = \frac{1}{2\mu} + \frac{1}{2(\mu - \lambda - \sum_{i=1}^{N_A} q_i\bar{\lambda}_i)} = \frac{1}{2\mu} + \frac{1}{2}g_1(\bar{\boldsymbol{\lambda}}), \qquad (17)
$$

where $g_1(\bar{\boldsymbol{\lambda}})$ is defined as in the proof of Lemma 4.1 with $p := \mu - \lambda$. Since $g_1(\bar{\boldsymbol{\lambda}})$ is convex, $g_2(\bar{\boldsymbol{\lambda}})$ is convex. □

Proof of Lemma 4.6. Function $f_{\mathrm{G/G/1}}$ is a non-negative linear combination of functions of the form

$$g_3(\bar{\boldsymbol{\lambda}}) := \frac{\theta}{t-\theta}(\theta^2 + s) \tag{18}$$

with $\theta := p + \sum_{i=1}^{N_A} q_i \bar{\lambda}_i$, where $\theta \geq 0$, $t - \theta > 0$, and $s > 0$. Thus, it suffices to prove that $g_3(\bar{\boldsymbol{\lambda}})$ is convex.

To this end, note that

$$\frac{\partial g_3}{\partial \theta} = \frac{3\theta^2(t-\theta) + \theta^3}{(t-\theta)^2} + \frac{st}{(t-\theta)^2} > 0, \tag{19}$$

$$\frac{\partial^2 g_3}{\partial \theta^2} = \frac{6t\theta}{(t-\theta)^2} + \frac{2\theta^3}{(t-\theta)^3} + \frac{2st}{(t-\theta)^3} > 0, \tag{20}$$

i.e., g_3 is an increasing convex function of θ. Since θ is a linear function of $\bar{\boldsymbol{\lambda}}$, g_3 is a convex function of $\bar{\boldsymbol{\lambda}}$. □

B SUPPLEMENTARY EVALUATION RESULTS FOR BACKBONE NETWORK

B.1 Measurement Calibration in NS3 Simulation of Backbone Network

As discussed in Section 5.1.1, during shared weight inference, we need to estimate an offset κ between measurements on a pair of probed attack paths (p_{A1}, p_{A2}) and measurements on a target path p_B to mimic tri-cast, as the delays from s_A and s_B to the links shared by all these paths (if any) may be different. We use the following heuristic to estimate κ.

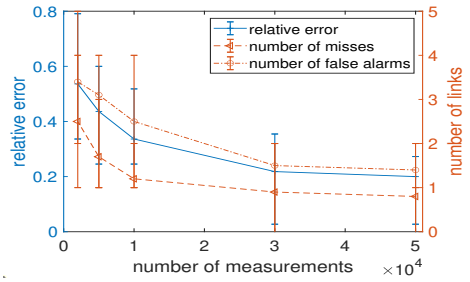
We send a flow on each probed attack path to collect a sequence of end-to-end delay measurements. We also collect end-to-end delays on the target path in the meanwhile. For the target path, we directly transform the delay measurements into a binary sequence of queueing indicators using the threshold given in Section 5.1.1, denoted as $\{q_B^t\}_{t=1}^T$, where $q_B^t = 1$ if the t-th measurement is detected to experience queueing and $q_B^t = 0$ otherwise. Since p_{A1}, p_{A2} share the same source s_A, we combine the delay measurements on p_{A1}, p_{A2} by adding the delays of the i-th packets from both paths, and then transform the combined delay measurements into a binary sequence $\{q_A^t\}_{t=1}^T$ as for $\{q_B^t\}_{t=1}^T$. To find κ so that the i-th packet on p_B and the $(i+\kappa)$-th packet pair on (p_{A1}, p_{A2}) traverse the shared links (if any) at approximately the same time, we maximize the correlation between $\{q_B^t\}_{t=1}^T$ and $\{q_A^t\}_{t=1}^T$ by solving

$$\kappa^* = \underset{1-T \leq \kappa \leq T-1}{\arg\max} \frac{1}{\min(T, T-\kappa) - \max(1, 1-\kappa) + 1} \sum_{i=\max(1,1-\kappa)}^{\min(T,T-\kappa)} q_B^i q_A^{i+\kappa}. \tag{21}$$

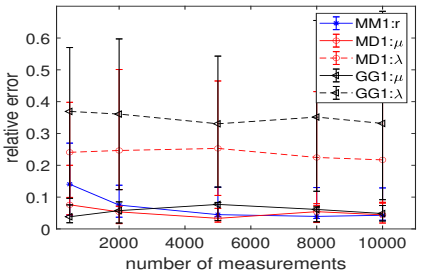
We then identify the i-th packet on p_B and the $(i+\kappa^*)$-th packet pair on (p_{A1}, p_{A2}) as a mimicked tri-cast.

B.2 NS3 Simulation of Backbone Network under an Alternative Background Traffic Model

In Section 5.1, we showed the results in the scenario where the background traffic follows log-normal distribution. In this section, we validate our algorithms under background traffic generated according to ON-OFF process [24, 46, 48]. More specifically, the duration of each ON period is sampled from a Pareto distribution with the shape parameter as 2.04 and the scale parameter as the average length of 13 packets. The duration of each OFF period is sampled from the same distribution with a different scale parameter, configured to result in the same utilization of each link as the values used in Section 5.1 for log-normally distributed background traffic. The results

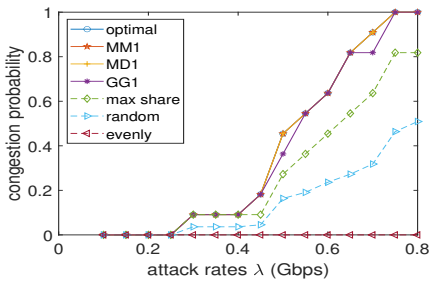


(a) Performance in detecting shared links

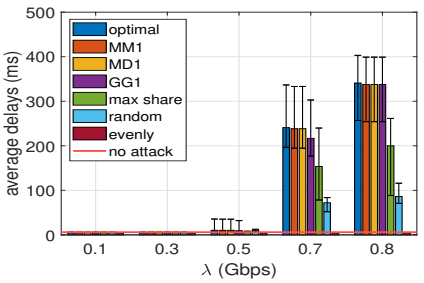


(b) Performance in inferring shared link parameters

Fig. 10. Performance of reconnaissance in backbone network simulation under ON-OFF background traffic ($N_A = N_B = 10$).



(a) Probability of congesting at least one shared link



(b) Average delay over all the target paths

Fig. 11. Performance of attack design in backbone network simulation under ON-OFF background traffic ($N_A = N_B = 10$).

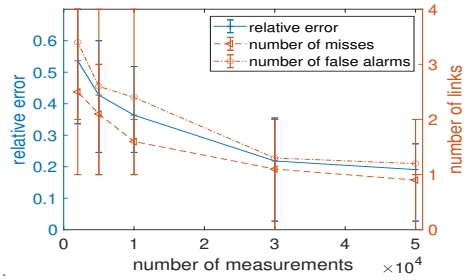
for reconnaissance are shown in Fig. 10 as the counterpart of Fig. 5, while the results for attack design are given in Fig. 11 as the counterpart of Fig. 6. We observe that the results under ON-OFF background traffic are similar to those in Section 5.1, which confirms the robustness of the proposed methods under different background traffic patterns.

B.3 Evaluation Results for NS3 Simulation of Backbone Network with $N_A = 20$

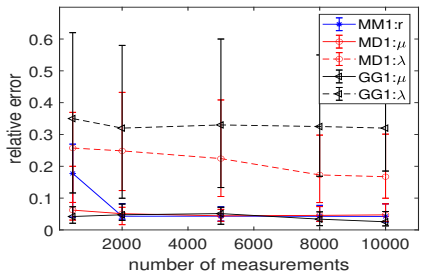
In Section 5.1, we evaluate our algorithms with $N_A = 10$. A larger N_A will result in fewer links on each edge in the routing tree $\mathcal{T}$, which makes it harder for Alg. 1 to accurately detect the shared links. To test its impact, we evaluate our algorithms with $N_A = 20$ in the same scenario as in Section 5. The results are given in Fig. 12-13, as the counterparts of Fig. 5-6. We observe that (i) the performance of reconnaissance slightly degraded, but (ii) the attack design still achieved significantly better performance than the baselines (i.e., "max share", "random", and "evenly"). This result demonstrates the robustness of our methods to the number of attack paths. We have also verified that the performance of our methods is not sensitive to the number of target paths.

B.4 Evaluation Results for NS3 Simulation of Backbone Network with 50 Gbps Link Capacity

Building on Section 5.1.1, where the link capacity is normalized to 1 Gbps, this section validates those results under increased link capacities. Specifically, we repeat the NS3 simulation for the backbone network GtsCe with a link capacity of 50 Gbps. To accommodate this, the rate for background traffic is regenerated every 0.05 ms, compared to the previous 0.5 ms in Section 5.1.1. Moreover, the flow rate on the target paths has been adjusted from 50 Mbps to 2500 Mbps, and the

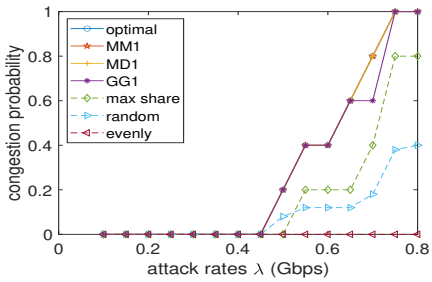


(a) Performance in detecting shared links

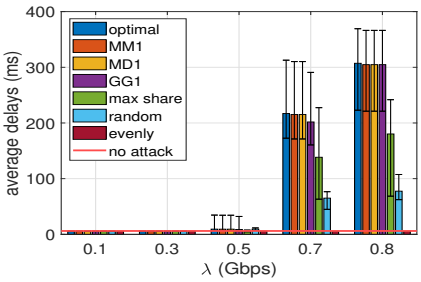


(b) Performance in inferring shared link parameters

Fig. 12. Performance of reconnaissance in backbone network simulation ($N_A = 20, N_B = 10$).

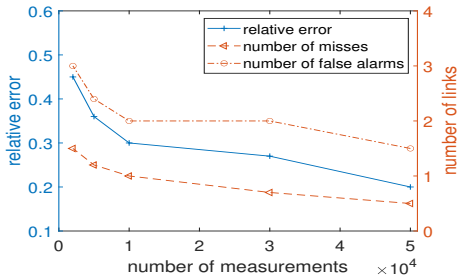


(a) Probability of congesting at least one shared link

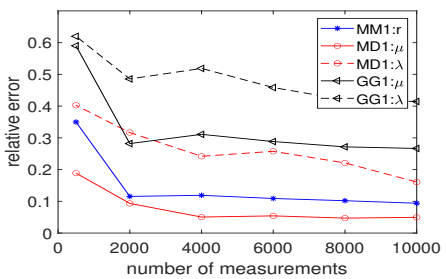


(b) Average delay over all the target paths

Fig. 13. Performance of attack design in backbone network simulation ($N_A = 20, N_B = 10$).



(a) Performance in detecting shared links



(b) Performance in inferring shared link parameters

Fig. 14. Performance of reconnaissance in backbone network simulation ($N_A = N_B = 10$).

background traffic rates have been increased by 50 times too, while all other settings remain the same as Section 5.1.1. Results from a single Monte Carlo run are presented below.

The reconnaissance results as the counterpart of Fig. 5 are given in Fig. 14, in which we observe similar trends as in Fig. 5. We then assess the rate at which each attack method induces congestion on at least one shared link, as a counterpart to Fig. 6 (a). For this specific Monte Carlo run, the benchmarks "optimal" and "max share", and all the proposed methods (i.e., "MM1"/"MD1"/"GG1") begin to induce congestion when the total attack rate exceeds 48.7% of the link capacity. At this rate, "random" starts exhibiting a non-zero (0.3) probability of causing congestion. Moreover, "random" only reaches a 0.5 congestion probability even when the attack rate surpasses 70% of the link capacity. In contrast, "evenly" fails to induce congestion even when the attack rate reaches 80% of the link capacity.

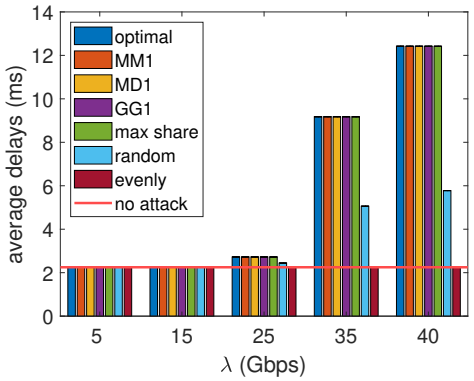
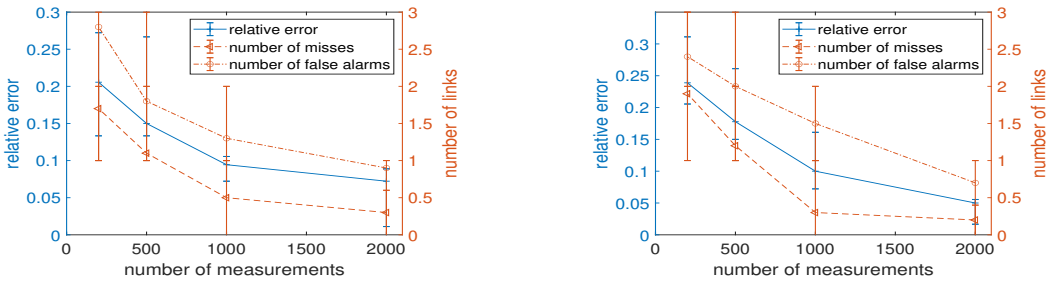


Fig. 15. Average delay over all the target paths ($N_A = N_B = 10$)).



(a) $N_B = 5$ (b) $N_B = 10$

Fig. 16. Performance in detecting shared links in IAB network simulation.

As the counterpart of Fig. 6 (b), we analyze the average delay induced by various attack designs in Fig. 15, computed over 30,000 packets on the target paths. We observe that "max share" and all the proposed methods (i.e., "MM1"/"MD1"/"GG1") achieve the same performance as "optimal" since they all correctly identify the attack path traversing the weakest shared link[5]. Notably, the proposed methods markedly outperform the non-optimized benchmarks "random" and "evenly". These findings, which are consistent with Fig. 6 (b), underscore the efficacy of our proposed methods.

C SUPPLEMENTARY EVALUATION RESULTS FOR INTEGRATED ACCESS AND BACKHAUL (IAB) NETWORK

In this section, we will present the supplementary experimental results for Section 5.2 in the case of $N_B = 5$. The previously presented results under $N_B = 10$ are also shown here for comparison.

C.1 Results on Reconnaissance

In Fig. 16, we present the performance of shared link detection for different numbers of target paths. We observe that the results are insensitive to the number of target paths N_B. Next, we show the results of parameter estimation for the detected shared links, as given in Fig. 17. Again, the observations under different values of N_B are qualitatively similar.

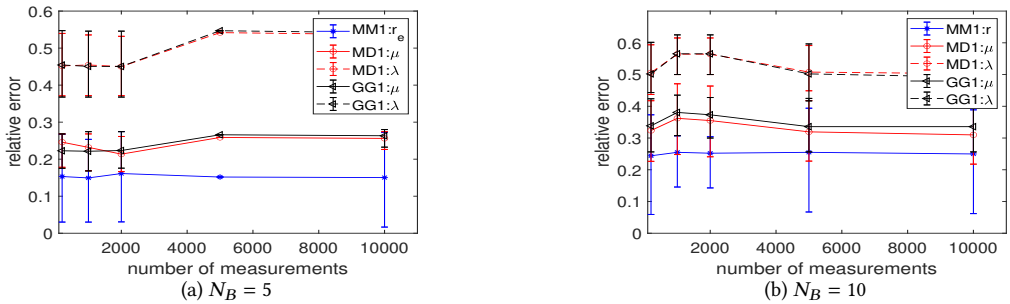


Fig. 17. Performance in inferring parameters of shared links in IAB network simulation.

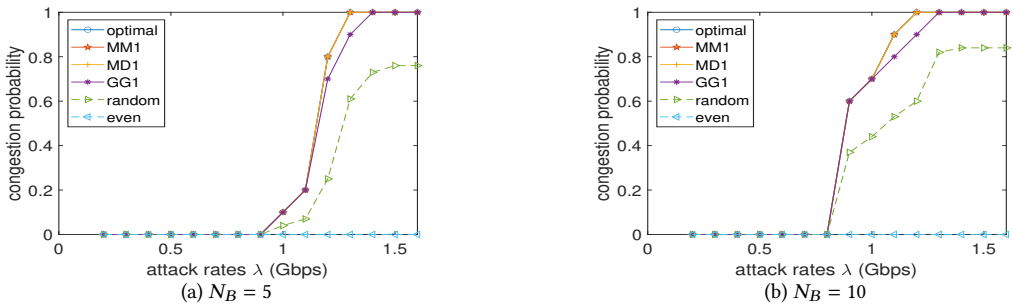


Fig. 18. Probability that the attack can destabilize the queue for at least one shared link in IAB network simulation.

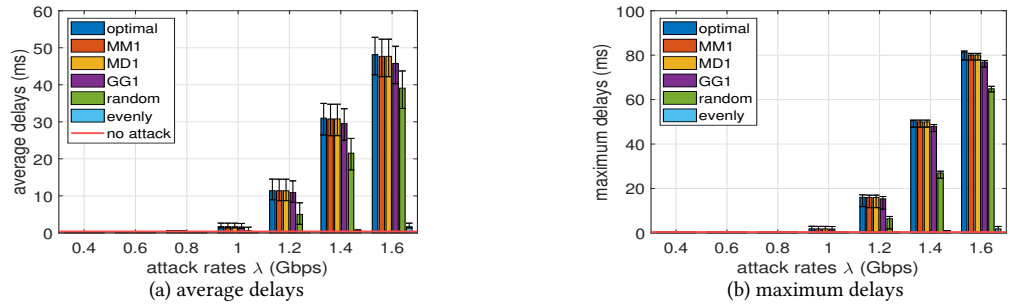


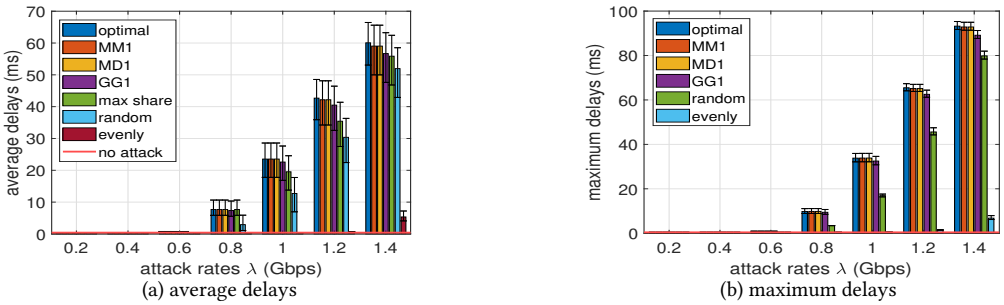
Fig. 19. Delay increase under different λ in IAB network simulation ($N_B = 5$).

C.2 Results on Attack Design

We first evaluate the probability that the proposed design with objective (12) can destabilize at least one queue, as shown in Fig. 18. As before, the results for $N_B = 5$ and $N_B = 10$ show the same trend.

Finally, we compare the delays of target paths under various attack designs in Fig. 19-20, where Fig. 19 (a) and Fig. 20 (a) show the overall average delay (averaged over all the target paths), while Fig. 19 (b) and Fig. 20 (b) show the maximum average delay (maximized over all the target paths). Similar to the results discussed in Section 5.2.3, we observe that the proposed attack designs generate higher impacts than the benchmarks, regardless of the number of target paths and the

[5]The absolute delays in Fig. 15 are smaller than those in Fig. 6 (b) due to the increased link capacity.



(a) average delays (b) maximum delays

Fig. 20. Delay increase under different λ in IAB network simulation ($N_B = 10$).

performance metric (either the average delay over all the target paths or the average delay of the worst-performing target path).

D DISCUSSION ON DETECTING FALSE ALARMS THROUGH PARAMETER ESTIMATION

In this section, we will discuss an observation that the proposed parameter estimation method (Alg. 3–4) can help detect the false alarms in shared link detection (based on Alg. 1–2).

In the case of a false alarm, the "shared link" under consideration does not actually exist, and thus varying probing rate (line 6 in Alg. 4) will not impact the average delay of the target path under consideration as expected. This will manifest as an abnormally large estimated link capacity, which can then be used to detect that this "shared link" does not exist. To see the reason, let us consider the example in Fig. 7. If Alg. 1 falsely detects $(2, 6)$ to be a shared link for the target path $2 \longrightarrow 43$ and Alg. 4 tries to estimate its capacity by varying probing rate on the path $2 \longrightarrow 25$, then the best-fitting capacity will be infinity as the average delay on $2 \longrightarrow 43$ will not increase with the probing rate on $2 \longrightarrow 25$. Even if the probing path and the target path have shared links, false alarms may still be detected. For example, suppose that link $(5, 18)$ in Fig. 7 is falsely detected as a shared link for the target path $2 \longrightarrow 46$, and $2 \longrightarrow 37$ is selected as the probing path for estimating its parameters, then the delay increase on $2 \longrightarrow 46$ caused by the probing on $2 \longrightarrow 37$ will be captured by the delay increase on the truly shared links $(2, 14)$ and $(14, 5)$ (if they are detected), still making the best-fitting capacity of link $(5, 18)$ infinity. This observation together with the fact that there are fewer misses than false alarms (see Fig. 8 (a)) allows our solution to detect the shared links with high accuracy.

Received February 2023; revised October 2023; accepted October 2023