Analyzing the Benefits of Optical Topology Programming for Mitigating Link-flood DDoS Attacks

Matthew Nance-Hall[†], Zaoxing Liu[‡], Vyas Sekar[◇], Ramakrishnan Durairajan[†]
[†]University of Oregon, [‡]University of Maryland, [◇]Carnegie Mellon University

Abstract—Link-flood attacks (LFAs) overwhelm bandwidth on links in a network using traffic from many sources, which is indistinguishable from benign traffic. Unfortunately, traditional DDoS defenses are incapable of stopping such attacks and recently proposed software-defined solutions are ineffective.

In this work, we observe a new opportunity for mitigating LFAs using optical networking advances. In essence, we envision new capabilities for *topology programming*, to scale capacity on-demand to avoid congestion and add new links to the network to create new paths for traffic during LFA incidents. Realizing these benefits of optical topology programming raises unique challenges; the search space for candidate topology configurations is very large and joint optimization of topology and routing is NP-hard.

We present ONSET—a framework that tackles these challenges to lay a practical foundation for topology programming-based defenses against LFAs. We show that ONSET complements existing programmable network defenses and amplifies their benefits. We perform a *what-if* style analysis of ONSET by simulating a wide-ranging set of attacks, including terabit-scale attacks against every single link, on five networks with two different routing capabilities and observe that ONSET provides the means to mitigate congestion loss in more than 90% of the hundreds of diverse attack scenarios considered.

Index Terms—DDoS Defense, Optical Networks, Wide Area Networks, Computer Simulation Experiments,

1 Introduction

Distributed denial-of-service attacks (DDoS) that overwhelm a network's bandwidth are on the rise [1], [2]. The immense attack volumes, attack diversity, sophisticated attack strategies, and low cost to launch attacks make them long-term cybersecurity issues. In 2021, 9.7 million DDoS attacks occurred. This number marked a 14% increase over 2019 [3].

Of particular concern within this broader class of threats are *link-flood attacks* (LFAs) which are also known as network-layer DDoS attacks. While this attack variant has been a scholarly curiosity in years past, it is now a legitimate threat to networks; according to a CloudFlare report, the number of LFAs recorded in their network increased by 109% in the second quarter of 2022 year-on-year. They also recorded an 8% increase in the number of LFAs with 100 Gbps of attack traffic during the previous quarter [4]. LFAs are more effective than conventional volumetric attacks as they are targeting on shared links instead of victim hosts. In this context, we observe a few key trends in the LFA landscape. First, the attacks are adapting to defenses by changing their traffic characteristics frequently. Static mechanisms to defend them become ineffective and treat attack and benign flows equally, affecting the performance of benign flows. Second, the attackers are generating terabits per second (Tbps) of malicious traffic [2], [5], [6], [7].

In light of this increasing LFA sophistication, existing defense capabilities (e.g., packet scrubbing [8], [9], [10], [11], [12], in-network filtering [13], [14], [15], [16], routing around congestion [17], overlays for tracking [18], and more recent software-defined defenses [19], [20], [21]) can be improved. For example, since simple network layer filters are ineffective (the indistinguishable nature of the benign and malicious traffic), we need to reroute traffic to sophisticated packet scrubbers for deeper inspection. This inevitably impacts benign traffic and/or worsens network congestion. Similarly, even programmable defenses (e.g., [19], [21]) are ineffective. As we show empirically in our results, LFAs can induce a substantial penalty for legitimate traffic as programmable defenses simply shift the attack-induced congestion elsewhere in the network.

In this work, we observe a new opportunity to bolster LFA defenses in wide area networks (WANs) by leveraging recent advances in optical networking called *topology programming*. Topology programming scales/augments existing LFA defenses by dynamically adding new optical wavelengths to scale the network capacity and alleviate network congestion [22]. Similarly, using reconfigurable add-drop multiplexers (ROADMs) [23], topology programming allows steering of wavelengths at finer granularities and enables fast traffic rerouting [24] in the face of congestion. Optical topology programming has been adopted for classical networking tasks (e.g., traffic engineering [24], [25]) and is increasingly being commoditized [26], [27], [28], but its

Manuscript received February 28, 2023. This work was funded in part by the National Science Foundation under Grants CNS-2212590 and SaTC-2132651, and in part by the University of Oregon Doctoral Research Fellowship.
Corresponding author: M. Nance-Hall (e-mail: mhall@cs.uoregon.edu).

benefits have not been explored in depth for LFA defenses.

Leveraging optical topology programming leads to two novel opportunities in combating terabit LFAs. (1) Scaling capacity on demand to avoid congestion. By dynamically scaling the capacity of optical paths on-demand, we can potentially reduce network congestion in targeted links. (2) New capabilities for advanced/future LFAs. With novel optical features such as rapid wavelength reconfiguration, we can improve defenses for LFAs by providing new *links and paths* to route around congested links.

Realizing these benefits in practice, however, requires addressing two key challenges. The first challenge is to dynamically identify the optimal topology, out of $\mathcal{O}(2^{n^2})$ possibilities for a network with n nodes, achievable using topology programming[1]. We refer to this as topology enumeration. Here, optimal is with respect to the maximum reduction in attack-induced congestion. This is key because a sub-optimal topology configuration can shift attack-induced congestion to a different link. Second, there is a challenge for managing network performance with dynamic topology changes; i.e., given a set of links and paths we can activate, we need to choose a routing configuration that is optimal with respect to network demand from both legitimate and attack traffic. In other words, we must jointly optimize routing and topology to effectively address attack-induced congestion. Joint optimization of topology and routing is an NP-hard problem [29].

To address these challenges, we propose ONSET (**O**ptics-enabled **N**etwork defen**S**e for **E**xtreme **T**erabit LFAs). ONSET is a framework for augmenting existing link-flood defenses with topology programming and consists of two components. First is the topology pruning algorithm that addresses the enumeration challenge. This algorithm computes a subset of the potential network topology link sets considered by introducing k new links. Subsequently, the algorithm computes a set of shortest paths based on the augmented topology with k new links. Second is the optimization component that tackles the NP-hard problem by formulating a mixed-integer linear program to optimally map and forward traffic atop the augmented links, minimizing attack-induced congestion. This formulation is agnostic to any packet-processing logic or mitigation methods and can be augmented to any existing defenses and network controllers. The links added to the network to mitigate an attack can persist on the network, providing stability for repeat attacks, or be relinquished to the system and used to defend subsequent attacks. Our integer linear programming solution provides support for both of these cases, which the network operator may choose between.

Given the limitations of existing network simulation and emulation tools to study topology programming, we use a custom discrete-event network simulator to analyze the benefits of ONSET.[2] Our simulator models how different topologies forward the same traffic, allowing us to see how this change affects link utilization across the network. We use it to test ONSET using different terabit LFA scenarios for a diverse set of networks.

Using the simulator, we explore *what-if* questions regarding the value added by our framework for topology programming with analysis on the merit of defenses that can employ it. We approach these questions by processing traffic on simulated networks for which the topology can change subject to a set of real-world limitations. To this end, we simulated a wide variety of LFAs against several networks, where each attack targets a specific link or set of links. Full details about our attack generation and assumptions are discussed in § 6. We observe that defenses *with topology programming* offer traffic performance that is always at least as good as a defense without it. Concretely, in 93% of the hundreds of attacks simulated, topology programming helped mitigate all congestion loss from the attack. In every case, ONSET yields its solution in under 1 minute.

In summary, we make the following contributions:

- The first optical topology programming-based defense framework for terabit LFAs. In our prior work [30], we introduced the idea of a topology programming-based defense for DDoS and presented simple numerical models to demonstrate its benefit for simple topologies and attacks. In this work we go much further in our analysis, presenting a complete framework for applying an optical topology defense for LFAs for any topology, considering various instances of attacks.
- A topology pruning algorithm to tame the complexity associated with modeling the exponential number of possible network topologies and paths on each of them.
- A formal mixed-integer linear programming model for the optimal mapping of traffic to new optical links.
- A simulator for what-if analysis, demonstrating our approach under diverse attack scenarios and for a diverse set of networks.
- A decision-support capability for incremental deployment of ONSET and measure the cost-benefit trade-off for enabling the optical topology programming-based defense at different locations in a network.

2 Background and Motivation

In this section, we first provide a brief background on LFAs and their threat model. We then present the limitations of state-of-the-art LFA defenses, followed by the requirements for an ideal LFA defense.

2.1 Link-Flood Attacks

An LFA is a DDoS attack that overwhelms the network bandwidth for a victim [1]. The attacker presupposes knowledge of the topology, which may be gained through active measurement campaigns (e.g., utilizing `traceroute`) or by stealing insider knowledge. Given the topology information, pairs of nodes that share common intermediate hops (e.g., bottleneck links) are targeted by the attacker with an overwhelming volume of traffic.

There are three types of LFAs we briefly scope. These types differ based on their methods for selecting a targeted link (or set of links). First is the Coremelt attack [31] which targets a network link in an attempt to remove it from the

1. Because a complete graph of n nodes has $\mathcal{O}(n^2)$ links, the number of topologies possible is the power set of these links $|\mathcal{P}(n^2)| = 2^{n^2}$

2. Source code of the simulator is at github.com/mattall/topology-programming and datasets will be released to the community upon publication.

network with precision. Second is the Crossfire attack [32] which targets an entire region of a network by flooding all links surrounding the region in an attempt to partition it from the rest of the network. Finally, we consider the attack studied in the Spiffy paper [33], which assumes a cost-sensitive attacker who may leverage either the Crossfire or Coremelt attack strategy, but behaves rationally with the goal of maximizing impact with the minimal amount of attack traffic.

Threat Model: The attacker has access to a *botnet* or large number of compromised hosts and services. The attacker uses the botnet to send terabytes of traffic to a network. We are primarily concerned with sustained attacks where the duration is at least five minutes or longer. The attacker can flood either or both directions of a bidirectional network link that is critical for the service of targeted hosts. The attacker measures their success based on whether they can induce network congestion and thereby degrade the performance of legitimate traffic intended for the network. We assume that all attack traffic comes from legitimate non-spoofed senders, is protocol-conforming, and is indistinguishable from benign network traffic. We assume that an attacker has obtained an accurate map of a network's link-layer topology, with which it determines which bots to activate and what destinations they will send traffic to. How the attacker acquires the network topology information is beyond the scope of this paper. In this work we do not consider a "smart attacker" who updates their network reconnaissance—the attacker has a one-time snapshot of the network topology and deploys bot traffic strategically based on that snapshot. This assumption keeps the work grounded in LFA defense specifically, rather than entering the network reconnaissance and obfuscation space which is out of scope for this paper. Note that this assumption in our threat model is consistent with prior efforts (e.g. Ripple [20]). Finally, we assume there exists a mechanism to detect an LFA. This is reasonable because the bandwidth utilization of links is easy to monitor. We do not assume that detecting an attack implies easily flagging/dropping attack traffic with high accuracy.

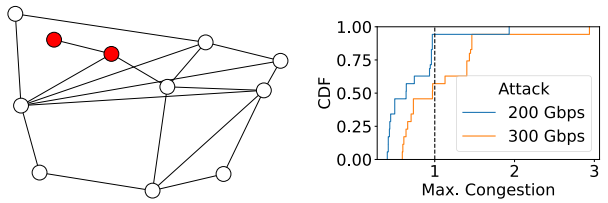
2.2 Prior Efforts and Their Limitations

State-of-the-art defenses against LFAs use software-defined networking (SDN), thus altering only the network's *forwarding behavior* to mitigate attacks [20], [33]. The SDN-based approaches for LFA defense can broadly fit into three categories: programmable control plane, programmable data plane, or hybrid. Control plane programmable defenses (e.g., Spiffy [33]) use a central network controller to monitor traffic. The controller issues commands to network forwarding devices and updates their forwarding paths when an attack is detected. Data plane programmable defenses (e.g., Ripple [20]) cut out the centralized aspect from prior work, and implement monitoring and mitigation within the network switches themselves. Hybrid approaches (e.g., Jaqen [34]) use a mix of both data plane and control plane programmability; they can adapt the forwarding paths for suspicious traffic, thereby sending it to specialized switches that will run defense programs and drop malicious traffic or allow benign traffic to traverse the network further. We note that Jaqen has not been applied to LFAs, but include it in our taxonomy of prior work to show that the hybrid approach has been applied to DDoS mitigation in a general sense. Another recent defense, ACC-Turbo [35], employs a fast clustering technique to flag and deprioritize suspicious traffic at line rate.

State-of-the-art LFA defenses treat topology as a static entity and thus overlook an opportunity to remove the network bottlenecks created by LFAs. There are no defenses that optimize or change the underlying topology, and therefore prior efforts are forced to filter and drop traffic during an attack on a highly congested link shared by multiple hosts. As we will see in the empirical example below (in § 2.2.1), the lack of topology flexibility implies inevitable congestion loss for high-volume LFAs. We discuss related work in more detail in § 8.

2.2.1 Empirical Example

Figure 1 illustrates the limitations inherent in adapting forwarding behavior to defend against LFAs for a real-world network, Sprint, from the Internet Topology Zoo [36]. In this quantitative demonstration, we created a set of Coremelt attack traffic matrices, each targeting an individual link in the network; see § 6 for details. We also varied the attack strength from 200 Gbps to 300 Gbps. We gave the network SDN routing defense capabilities, whereby the network optimally routes traffic and minimizes max link congestion. The routing strategy used in this example is more optimal than the current generation of traffic engineering systems (e.g., B4 [37], Orion [38]) because those systems rely on heuristics to scale and compute allocations quickly. In our example, this routing system has unlimited time to find the optimal set of paths for each flow. Figure 1b shows CDFs of max link congestion for both of these attacks. We observe congestion loss with 200 Gbps of attack traffic in only one instance, where a link to a leaf node was flooded (this link is identified with the highlighted nodes in Figure 1a). When we increase the attack to 300 Gbps we see that SDN-based rerouting has significantly greater difficulty mitigating loss. In fact, about 50% of all links targeted with this attack incurred a loss.



(a) Sprint Network.

(b) CDF of Max Congestion. Dashed line shows congestion-loss threshold.

Fig. 1: Every link in the network was targeted individually with a 200 and 300 Gbps Coremelt attack. At 200 Gbps, it was impossible to guard one link from congestion loss. At 300 Gbps, ∼50% of links experienced loss.

This result illustrates that defenses that only adapt forwarding behavior have a finite breaking point at which congestion loss is unavoidable, even when routing choices are optimal. Observing these factors, we raise our motivating question: how can we enable capacity on demand

and topology flexibility without th
collateral loss?

Summary: *We observe that state-of-t
from a key limitation that network top
entity. It is often impossible to rero
insulate benign traffic from loss in th
overwhelm a link's bandwidth multiple*

3 Approach: Optical Tor ming for LFA Defenses

We observe a new opportunity t
by leveraging a recent advancemen
called *topology programming* to achie
Using topology programming, an
network's topological structure via
configuration in addition to the traffic forwarding behavior.

Combining topology programming, provided by optics, and adaptive forwarding behavior, provided by SDN and programmable switches, leads to two new opportunities in combating terabit LFAs. First, it allows a network's underlying topology to scale capacity on demand to avoid congestion. Second, topology programming enables a defender to amplify the benefits of traditional programmable defenses. Improved general network performance is possible because changes made at the optical layer give us increased possibilities to forward traffic on new paths in the face of attacks. Note that we do not claim that topology programming offers a panacea for all DDoS-related concerns—we claim that it provides a novel means to bolster existing programmable defenses for LFAs as described above, and investigate that means more deeply than any prior work to date.

While topology programming is compelling, it has not received a great deal of attention for DDoS. Hence, we provide a brief background of optics and topology programming in § 3.1. In § 3.2, we outline the challenges of using topology programming for LFA defenses.

3.1 Primer on Optics

To introduce a new link or scale capacity in a network with topology programming, we must establish a *circuit* or wavelength between two network elements (e.g., switches or routers). Modern WANs have optical cross-connects (OXCs) where electrical or optical signals from switches are multiplexed and routed onto a shared fiber [39]. A ROADM is a specialized OXC that can dynamically route wavelengths. That is, it can add circuits to a fiber, drop circuits traversing it from the fiber, or enable transit wavelengths to pass through without adding or dropping them. A pair of *transponders* tuned to the same wavelength is required at the two ends of the circuit to route a wavelength at an OXC. Figure 2 shows how a linear physical topology can be configured into a mesh logical topology with wavelength routing at OXCs.

To allow the dynamic adding or dropping of new links in a network, there must be resources at the network endpoints (henceforth, fallow transponders) and optical switching capabilities in the core (ROADMs). While ROADMs are commonly deployed in WANs [40], [41], [42], fallow transponders are not. In this study, we posit that this cost

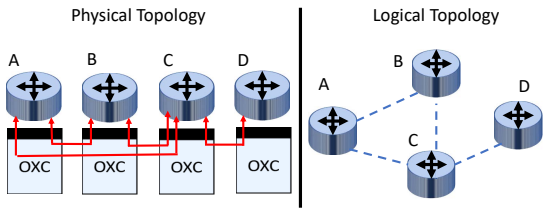


Fig. 2: Four routers (A, B, C, D) are connected with OXCs in a linear topology. The OXC at router B is allowing a bypass signal from A to C.

is worth exploring, and we take measures in our evaluation to succinctly present the cost of deploying this defense with respect to the number of fallow transponders required to defend against attacks of various magnitudes and against different sets of links.

From a deployment perspective, new advances in terms of the timescales and capabilities for *topology programming* are increasingly being commoditized [28]. For example, efforts have shown the benefits of topology programming including wavelength steering at finer granularities to enable fast traffic rerouting [43]. Researchers have also demonstrated these capabilities for classical networking tasks (e.g., traffic engineering [24], [25], [26], [28], [44]). Preliminary studies have also explored the security benefits enabled by programmable optics in limited DDoS scenarios (e.g., for simple, 3-node topologies [45] or data centers [46]), however, an in-depth study of optical defenses against LFAs is critically needed.

3.2 Challenges

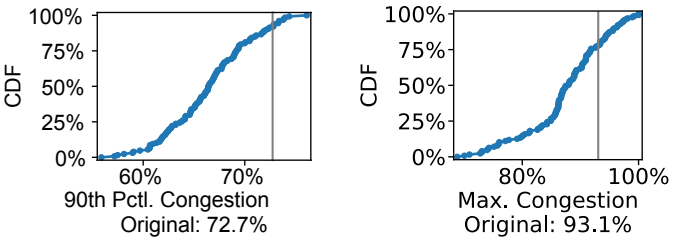
To use topology programming for LFA defense, we need to solve two unique challenges (**C**s).

C1: Topology Enumeration. When we open the door to topology programming, we are immediately confronted with an exponential number of network link-layer configurations to choose from. This size is further compounded with every path on each of those topologies and the number of ways to split traffic among a set of paths. The state space for network topologies wherein the set of active links can change is $\mathcal{O}(2^{n^2})$ where n is the number of nodes. Considering these topologies and their relative benefit for attack-specific demand introduces the challenge of topology enumeration.

To illustrate, consider a network with 30 nodes has 435 possible links ($30*29/2$). The set of all combinations of these links is 2^{435}. Therefore the number of different topology instances that we might create is $\sim$50 orders of magnitude larger than the number of atoms in the known universe. The runtime complexity for enumerating all shortest paths on every instance of the network topology, therefore, is $\mathcal{O}(2^{n^2}n^3)$[3]. Clearly, enumerating each potential network state (the paired sets of active links and available paths on those links) and storing these states is a daunting task, but it will enable us to quickly instantiate the most opportune configuration of links given the shifting behavior of an attacker.

3. The Floyd-Warshall all-pairs shortest path algorithm is $\mathcal{O}(n^3)$

C2: Managing Network Performance. Any addition or removal of a link from the network can have a unique effect on traffic performance across the entire network as seen in Figure 3; these plots show that arbitrarily adding a new link on the ANS network (from the network topology zoo [36]), while employing ECMP routing, can occasionally increase link utilization and induce network congestion. Figure 3a shows that the original 90^{th} percentile congestion was 72.7%, indicated by the vertical bar, and roughly 15% of all links that were added increased the 90^{th} percentile congestion. Similarly, Figure 3b shows an increase in maximum link congestion for roughly 25% of all possible single-link additions. This observation holds on any network that uses a link-state routing strategy such as ECMP because when we add a link we change the set of paths favored for routing traffic between some pairs of hosts. If a new link creates a new shortest path for every pair of nodes, then that new link will quickly become congested. This is an instance of Braess's paradox [47]. Therefore, we must have an optimization method to ensure that the changes that we introduce to a network by adding or removing links has a net-positive benefit for all traffic across the network.



(a) CDF of 90th percentile congestion after adding different links.

(b) CDF of maximum congestion after adding different links.

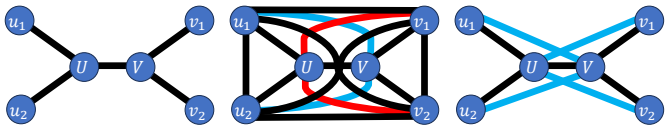
Fig. 3: Effect on network congestion in ANS from adding different links with ECMP routing.

In SDN-based networks wherein routing paths can be centrally defined and controlled, we must also choose to tread carefully between the trade-off of congestion avoidance and topology adaptation. We seek to optimally choos a topology and the set of routes based on it, but choosing set of new links to activate in a network while considerin the different routing choices available is NP-Hard [29].

As adding and removing links affects traffic paths, it ma be that the frequency with which those paths are change can lead to performance impacts. Therefore, we must verif that the frequency with which optical topology changes ar made is not a cause for a performance error in § 6.4.

3.3 Post-attack Decisions

The decision to revert the adapted links is an implemer tation detail within the network operator's purview, an trade-offs are involved in such decisions. For example, if the adapted links are not reclaimed, they serve as insurance against the impact of a repeat attack. Conversely, if they are reclaimed, they become available to defend against attacks on previously unaffected paths. Regardless, in the event of a new attack, a combination of unused (fallow) transponders and those with low utilization can be employed—this includes transponders that were not reclaimed, if applicable. In § 6.4 we evaluate the performance of the defense for rolling attacks.



(a) (b) (c)

Fig. 4: (a) Nodes U and V represent a bottleneck link between their neighbors, u_1, u_2, v_1, and v_2. (b) Set off all possible candidate links around U and V. (c) Illustration of the topology programming idea. ONSET considers a pruned down set of candidate links, containing. For each (U, V) link in the top 10% of ranked links, it chooses $(U, v*)$ and $(V, u*)$ where $v*$ and $u*$ are mutually exclusive neighbors of U and V respectively.

4 ONSET: An LFA Defense Framework Using Optical Topology Programming

We present ONSET (**O**ptics-enabled **N**etwork defen**S**e for **E**xtreme **T**erabit LFAs)—a defense framework for augmenting existing programmable defenses with optical topology programming to defend terabit LFAs. ONSET consists of a model and algorithm that address the major challenges for an LFA defense. Figure 5 outlines the two major algorithmic and modeling components of our framework. The first component, *Topology Pruning*, is an algorithmic step that (1) catalogues the different topologies that we may instance by activating a set of links and paths and (2) finds the set of shortest paths available under these topologies. The second component, *Joint Topology and Routing Optimization*, is an optimization model that runs when an ongoing LFA is detected (the instrument for detection is beyond the scope of this work). This component accommodates the demand present in the network using the topologies found during Topology Pruning. The result of this optimization is a set

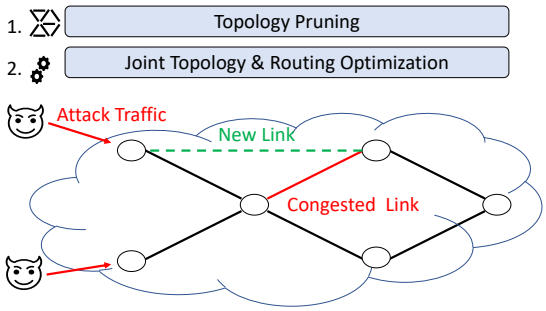


Fig. 5: Overview of the ONSET defense framework.

Hardware Requirements: The hardware requirements for ONSET under this framework are (1) optical fiber and transponders for establishing new links, and (2) ROADMs at nodes where new links originate, terminate, and bypass other nodes. ONSET can be incrementally deployed with these resources deployed at a subset of the network.

4.1 Topology Pruning

We address **C1** as follows. Given topology, T, and budget, B, we first find a set of links, L, and the network paths on these links, P_L. The set is large but can be pruned down considerably. As a first-order pruning step, we eliminate the possibility for links that are longer than the maximum transmission distance supported by the transponders (e.g., 5,000 km for 100 Gbps circuits [48]). This pruning removes infeasible links in large networks but does not help reduce the number of topologies in networks for which all of the nodes are closer than the max transmission distance.

Link Rank: A striking observation allows us to trim the candidate set further and consider a smaller set of topologies. We observe that for a diverse set of attacks on a network, each targeting a different subset of links, there is a consistent set of links that are disproportionately affected. We introduce a metric, *link rank*, which captures this phenomena. Consider a set of possible LFAs on a network, each of which targets a different set of links. The link rank is the percentage of attacks in which a given link is congested. For example, when 100 attacks are considered on a network, and a given link experiences congestion loss in 12 of those cases, the link rank for that link is 12%.

Figure 6a shows the CDF of link rank for networks of different sizes. From this result, we observe that a majority of links (for all networks considered) have a small link rank (i.e., less than 10%). Only a minority of links experience congestion during a relatively high proportion of the total attacks. Concretely, in the network with 50 nodes, only two links are congested in 43% and 37% of the attacks, respectively. At the tail end of the distribution, 74 of the links were *only* congested for one attack, or not congested at all. This observation suggests that only a handful of vulnerable links are severely affected by LFAs. We leverage this insight to prune candidate topologies: more redundancy is granted to links with high rank by prioritizing the k highest ranked links when enumerating potential candidate links and topologies.

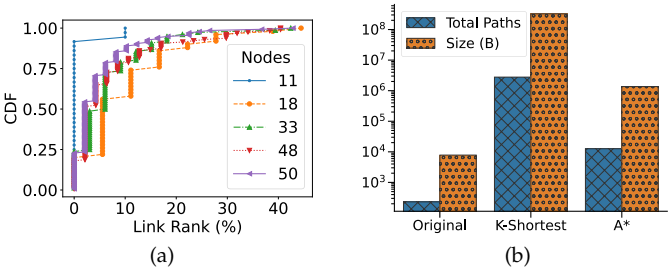


Fig. 6: (a) Link Rank for attacks on networks with different sizes, noted by the number of nodes. (b) Space complexity for comparison for path finding methods. "Original" represents the set of paths that would be stored in a traditional SDN system. "K-shortest" is the set of "K-Shortest" paths among the ranked links. "A*" is the pruned down selection of those paths.

In practice, we choose the top 10% of links and consider the potential to add new, *candidate links* links dynamically that bypass these bottlenecks. Figure 4 illustrates how the pruning process drastically decreases the search space for the reconfigurable topologies by honing in on bottleneck links and considering candidate links as those the provide potential for new paths that don't traverse the bottleneck. The sub-graph (4a) has a bottleneck (U, V). The complete graph induced by connecting all nodes in the neighborhood of (U, V) (4b) has 15 edges, 10 of which are not in the original topology. ONSET considers just 4 of the 10 links (4c) when enumerating topology and routing solutions.

Path Selection: While a general K-shortest paths search gives a small number of paths for a fixed topology, we must include paths with links that may or may not be members of the physical topology at any given time. Therefore, we must broaden the search. We might enumerate a set of paths for each pair of nodes that is exponentially greater in magnitude than the original set of K shortest paths. Therefore, we add a heuristic function to our graph searching process to mitigate the explosion in space complexity for our paths. Our path finding method is implementation of the A* algorithm [49]. Figure 6b shows the number of paths found with a A* versus a general all-pairs k-shortest paths. To ensure that the set of paths includes enough diversity with respect to candidate links, we populate that paths until the length of the path is greater than the length of the original path. Furthermore, to account for the potential of a link from the original graph to be removed, the original set of "shortest paths" for single-hop paths is expanded to include paths that are at least 3-hops long.

4.2 Joint Topology & Routing Optimization

We address challenge **C2** with a mixed-integer linear program (MILP), presented below. The intent of this optimization model is to find the set of links that will reduce network congestion by the greatest amount. The MILP is hosted by the ONSET controller, as shown in Figure 7. The model uses the enumerated topology and path data set to yield a set of optimal links to activate in a network in light of an ongoing LFA. These optimally chosen links come from the set of *candidate links* which are input to the system. Candidate links are links that don't exist in the network at the time the optimization solver is invoked but that can be quickly activated to augment the topology and allow data to travel directly between two nodes.

The controller periodically receives a traffic matrix and link utilization data with flow demands aggregated over a series of epochs. We assume an oracle for detecting the presence of an attack Note that this oracle does not identify attack traffic. It merely answers the yes or no question, "Is the network under attack?"; when an attack is detected the controller runs the optimization model and yields a set of links to add to the network. These links persist until network congestion falls below the levels that were seen before the attack. Table 1 shows the descriptions of variables used in this model.

The objective is to minimize the max link utilization ($\mathcal{U}$).

$$\texttt{minimize}\,\mathcal{U} \tag{1}$$

$$\forall (u,v) \in E, \texttt{cap}(u,v) = \texttt{cap}(v,u) \tag{2}$$

	Constants
G_0	Initial topology
E_0	Initial set of (directional) links
$\mathcal{E}$	Set of potential (directional) links (includes E
V	Set of all nodes
D	Set of all demands
$\texttt{txp}(v)$	Transponders at v
	Variables
b_e or $b_{(u,v)}$	Binary link-status variable
$\mathcal{F}^{s\to t}$	The set of links that are available to any potential path $s \to t$
$\texttt{flow}^{s\to t}_{(u,v)}$	Flow allocated from $D(s,t)$ onto edge (u,v) s.t. $(u,v) \in \mathcal{E}$
$\texttt{in}(n)$	Total flow from all demands going into node
$\texttt{out}(n)$	Total flow from all demands departing node n
$\texttt{cap}(u,v)$	Capacity of edge (u,v)

TABLE 1: Reference for notation, variables, and constants in equations 1–8.

capacity of each directional link, (u, v), is symmetrical. This ensures that a link is only active if it can be activated in each direction.

$$\forall n \in V, \texttt{txp}(n) \geq \sum_{u \in b_{(u,v)}} u \tag{3}$$

the total number of fallow transponders at a node, $\texttt{txp}(n)$ limits the total number of links in the topology that can start from n.

$$\forall e \in \mathcal{E}, \texttt{cap}(e) = b_e C_e \tag{4}$$

An edge's capacity is C_e or 0, $where C_e$ is capacity of a network link when edge e is active in the network.

$$\forall (s,t) \in D, \sum_{e \in \mathcal{F}^{s\to t}} \texttt{flow}^{s\to t}_{e} \leq \texttt{cap}(e) \tag{5}$$

That is, the sum of flows allocated from all demands allocated onto an edge must be bound by the capacity of that edge. Note that in the constraint, the only edges considered for a demand, $s \to t$, are limited to $\mathcal{F}^{s\to t}$ rather than the entire set of links $\mathcal{E}$. In practice, we employ the link selection and path finding strategies described in Section 4.1 to find the appropriate set of candidate links for each pair of nodes in the graph induced by all possible edges, $\mathcal{E}$.

$$\forall n \in V, \forall (s,t) \in D, \ \texttt{in}(n) = \sum_{(u,v) \in \mathcal{E} | n=u} \texttt{flow}^{s\to t}_{(u,v)} \tag{6}$$

$$\forall n \in V, \forall (s,t) \in D, \ \texttt{out}(n) = \sum_{(u,v) \in \mathcal{E} | n=v} \texttt{flow}^{s\to t}_{(u,v)} \tag{7}$$

$$\forall n \in V, \forall (s,t) \in D, \begin{cases} d(s,t) + \texttt{in}(n) = \texttt{out}(n) & \text{if } n = s \\ \texttt{in}(n) = d(s,t) + \texttt{out}(n) & \text{if } n = t \\ \texttt{in}(n) = \texttt{out}(n) & \text{otherwise.} \end{cases} \tag{8}$$

Constraints (6–8) are general multi-commodity flow optimization constraints [50] and ensure conservation of flow along paths through the network.

Having described the topology pruning and joint optimization components, we next focus on meaningfully assessing the efficacy of ONSET in the face of diverse terabit LFAs.

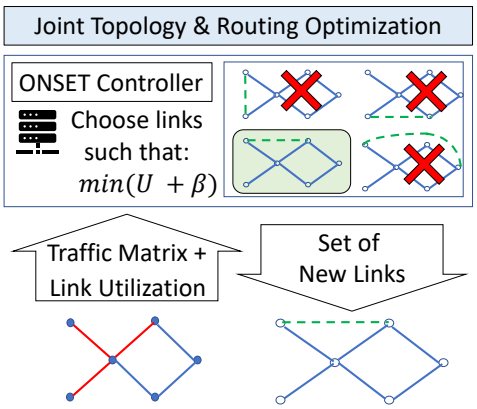


Fig. 7: Topology optimization process for ONSET.

5 ONSET Simulator for What-if Analysis

Evaluating the methods we have proposed in absence of a backbone network requires a simulator, the likes of which are not presently available. While there are traffic engineering simulators (e.g., Yates [51]) that let us process traffic through a network and see the effect of different routing methods on link utilization, to the best of our knowledge, there is no simulator that incorporates topology programming in the context of terabit LFAs. Furthermore, our design of a new simulator is motivated by the need to evaluate how different topological link configurations perform when they are forwarding the same attack traffic. The goal is to see how *topology* changes *link utilization* across the network in the face of terabit LFAs.

To this end, we build a cross-layer optical-network layer discrete-event simulator. This simulator enables us to ask valuable *what-if* questions about topology programming and its applicability for defending LFAs without access to a wide-area backbone optical network. Pertinent questions include how can the ONSET framework augment existing defenses to combat different types of LFAs, against attacks on different sets of links? What quantity of fallow transponders is required at the network's nodes to support the flexibility required to mitigate those threats using ONSET? How does the distribution of fallow transponders among nodes affect the ability of ONSET to mitigate traffic loss for a set of attacks?

Figure 8 shows a block diagram of the simulator, which models the way we envision ONSET to be used in a live deployment. The network operator defines optical constraints and traffic engineering system. Optical constraints include the number of simulated links available for adding, the max. link utilization thresholds which will trigger a topology-update event, and a target link utilization threshold which is used by the optimization method to find the best set of links to add. The ONSET controller, which controls SDN and optical components of the network, receives these input parameters from the operator and uses them along with the link utilization data to decide on a runtime defense strategy, whereby it adds links to the network and monitors their utilization. The traffic matrix processed by ONSET is a mixed bag of attack and benign traffic, the two of which are indistinguishable.

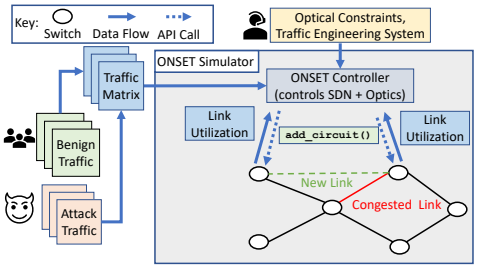


Fig. 8: Overview of ONSET simulator.

We wrote the simulator in Python 3 and implemented our defense optimization model in Gurobi [52] with Python. Our framework uses Yates [51] to implement traffic engineering and routing constraints with two methods: ECMP and multi-commodity flow (MCF); ECMP routing is commonly implemented in enterprise networks, as it is supported out-of-the-box by commodity switches and routers while MCF is seeing adoption in emerging SDN deployments [53] and is used in our analysis to emulate Ripple [20].

6 Evaluation

In this section, we evaluate the ONSET framework for defending LFAs. The key metric of success is link congestion (the aggregate traffic demand for the most-utilized link in the network). Traffic loss and reduced throughput occur when link congestion is greater than one. We compare ONSET against a baseline ECMP-routed network and an SDN-enabled network that optimizes traffic allocations to minimize max utilization across all network links. We denote the SDN defense as Ripple* throughout this section. We show that ONSET is an additive capability for network defense that can be applied to ECMP-routed networks or SDN-controlled networks employing the Ripple defense. We, therefore, compare network performance for both strategies with and without ONSET. To this end, we demonstrate the following key results.

(1) ONSET improves the capabilities of the Ripple defense for multi-target high volume attacks such as Coremelt (§ 6.2). We show ONSET can complement Ripple to mitigate terabit LFAs.

(2) Regional attacks, such as Crossfire, that target all links adjacent to a node, can be mitigated with the ONSET framework (§ 6.3). We show ONSET improves crossfire attack mitigation in over 90% of simulated attacks on 5 networks.

(3) ONSET can respond and mitigate rolling attacks, where a series of attacks with different volumes, numbers of links target targeted, and attack styles, vary in succession (§ 6.4). ONSET was effective at mitigating congestion loss in 64 out of 70 rolling attacks.

We conclude this section by presenting a cost-benefit analysis for ONSET vs. statically over-provisioning network links (§ 6.5), and taking a deep-dive into cost optimization with variable fallow transponder allocations where ONSET is enabled only for a subset of network links (§ 6.5).

6.1 Experiment Setup

Attack Matrices: We generate attack matrices using a custom tool written in Python to emulate three attacks: Coremelt [31], Crossfire [32], and Spiffy [33], which we refer to as $TM_{Coremelt}$, $TM_{Crossfire}$, and TM_{Spiffy} respectively. The TM tool takes in the topology of the network as an input, then finds the shortest paths between pairs of nodes, and creates demand between hosts that share a common link. $TM_{Coremelt}$ is made by choosing a random link (or links) in the network, and then choosing pairs of hosts for which their shortest paths use the chosen link(s). The Crossfire attack targets a region of the network. In our evaluation, we restrict a region to a single node. $TM_{Crossfire}$ floods all of the adjacent links to the target node. TM_{Spiffy} is constructed by finding that most-shared link(s) or node(s) and flooding them.

We emphasize that the attacker does not have control over the network routing. Our attacker assumes traffic is routed via the shortest path. The assumption does not hold for Ripple's defense due to its ability to optimize path and flow allocations, and we will see that Ripple, therefore, performs well enough for mild attacks. However, as an attacker increases their power with more traffic, Ripple's defense has a breaking point where ONSET improves the capability to defend.

An attack traffic matrix encapsulates two important attributes of the botnets, namely the size of the botnet (by proxy of its aggregate bandwidth), and the locations of the bots in the network (explicitly by the nodes from which their traffic originates).

Benign Traffic Matrices: Unless otherwise stated, we used TMGen [54] to create random gravity model traffic matrices for benign traffic in our experiments.

Routing: Our evaluations address two routing strategies, ECMP and Ripple*. ECMP is commonly implemented in service-provider networks. We pair it with ONSET to observe how legacy networks might benefit from the ONSET framework. On the other hand, modern enterprise and cloud backbone networks are increasingly looking to SDN to address network resource (e.g., bandwidth) management. Recent proposals for LFA defenses use SDN as a primary tool to insulate legitimate traffic from the effects of malicious traffic [20], [33]. SDN-based networks can use a central network controller to update forwarding paths and flow rates applied to these paths. Ripple attempts to drop malicious traffic before forwarding it, but when attack traffic cannot be detected, the Ripple defense reroutes traffic to avoid congestion on links. We emulate this capability by using a multi-commodity flow optimization to route traffic during attacks and denote this as Ripple*. The implementation of the ECMP+ONSET defense cannot tune traffic forwarding rates among paths by definition, and to model ECMP routing with binary links would introduce quadratic constraints to the model. However, to compute an ECMP routing assignment for a single topology is quick and efficient. Therefore, we elect to generate 100 sub-optimal solutions from our model and simulate the ECMP link utilization for all network links in all of the model's solution topologies in parallel. The network's topology is then configured based on the solution with the best ECMP congestion result.

Networks: Our evaluations consider five real-world network topologies, shown in Table 2. These networks are representative of enterprise optical backbone networks and have been used to investigate other LFA defenses in prior

work [20]. These networks range in size from 18 to 68 links. For each of these networks, we apply a similar series of tests where we vary the strength of an attack and the number of links targeted. In our experiments, every link in the network has a bandwidth of 100 Gbps unless otherwise stated. We gave every node in the network 10×100 Gbps fallow transponders; we revisit this allocation in § 6.6. Therefore, each node is capable of establishing a 100 Gbps link between itself and up to 10 other remote nodes. This bandwidth constraint per transponder is emulative of a 100 Gbps polarization multiplexed quadrature phase shift keying (PM-QPSK) transponder [55]; this type of transponder has been widely deployed in backbone networks for decades, and can reliably transmit 100 Gbps data channels approximately 5,000 km [56]. While higher-bandwidth transponders are also widely deployed, we only consider 100 Gbps QPSK transponders in this study. This is a conservative assumption for a lowest-common-denominator evaluation of the ONSET defense—we expect higher power/bandwidth transponders will improve the network performance further.

Network	Nodes	Links
Sprint	11	18
ANS	18	25
CRL	33	38
Bell Canada	48	65
SurfNet	50	68

TABLE 2: Networks used in our study.

Optimization Time: Our model implementation has a 1 minute cut-off window. Said differently, if the model does not find an optimal solution by then, it returns the best feasible solution. In cases where the solver finds a solution early, it may populate a set of alternative feasible solutions with the remaining time. We find that ONSET is able to dynamically derive topology configuration and routing settings for many attack scenarios presented to it. Figure 9 shows the time distribution for all of the ONSET models evaluated in this section. We observe that for the graphs Sprint, CRL, and ANS, all have a strong majority of evaluations where an optimal solution is found before the cut-off period at one minute. Bell Canada has 15 more nodes than CRL and nearly twice as many edges. ONSET found an optimal solution for attack on Bell Canada within the prescribed time in 38% of experiments. Surfnet, with only marginally more nodes and edges than Bell Canada, found optimal solutions in the allotted time in 25% experiments.

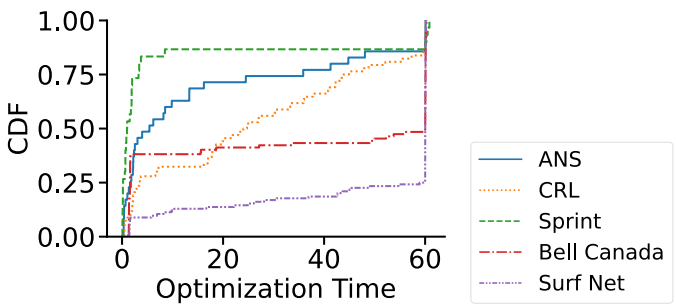


Fig. 9: CDF of optimization time for ONSET experiments by network.

Presentation of Results: Due to the different nature of the coremelt, Crossfire, and Rolling attacks, we plot the results for each test differently. Coremelt attacks target one or more network links, and the targets can be arbitrary and random. Therefore, we plot these results as grouped bar charts, where a group corresponds to a specific attack, and each bar represents the network performance of the different mitigation strategies (ECMP, ECMP+ONSET, Ripple*, and Ripple*+ONSET). The attack in each group of bars targets the same exact set of links with the same volume of traffic.

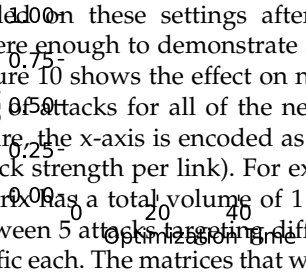
In many of the results we show network performance as it relates to maximum link congestion because LFAs, by purpose, attempt to maximally congest a link or set of network links. Therefore, we use this metric to determine the success or failure of an attack for each experiment. Related studies, e.g., Ripple [20], also narrow the scope of their evaluation specifically to links that are targeted by an attack. However, we include the complimentary results for total network throughput in the evaluation of the coremelt attacks in § 6.2, as it illustrates the relationship between maximum congestion and the overall performance of the network traffic. After establishing this relationship we omit throughput from the results as we are primarily interested in keeping maximum link utilization below 100% and keeping throughput at 100%.

The Crossfire attack targets a region of the network. To see how the different mitigation strategies perform, we launch crossfire attacks against each node in the network independently by targeting each link incident to each node targeted with an LFA. To view the performance of multiple attacks for each network, we present the results as CDF, where the X-axis shows max network congestion for each attack in the distribution and the Y-axis shows the CDF function for a given value of congestion.

The rolling attacks can be composed of crossfire and coremelt attacks, and we are interested in seeing how the network performance changes as the attacks change over time. Therefore, we plot network performance as a time series, where the X-value is a point in time, and the Y-value is the relative network performance metric at that time.

6.2 Coremelt Attack

To evaluate the performance of our framework against the coremelt attack, we consider a variety of attack strengths and attacks against a varying number of total links. In particular, we generate matrices composed of attack traffic with volumes of 100, 150, and 200 Gbps, each targeting 1 to 5 links simultaneously. These parameters are chosen in an attempt to get a broad-scope view of the impact of ONSET for a range of (multi)-attacks, each of which is capable of flooding a link with 1x to 2x its maximum capacity. We settled on these settings after discovering that they are severe enough to demonstrate a breaking point for Ripple*. Figure 10 shows the effect on network congestion from this suite of attacks for all of the networks in this study. In this figure, the x-axis is encoded as (number of links targeted × attack strength per link). For example, a 5×200 Gbps traffic matrix has a total volume of 1 Tbps; this volume is spread between 5 attacks targeting different links with 200 Gbps of traffic each. The matrices that we use in this test are made up



This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3391188

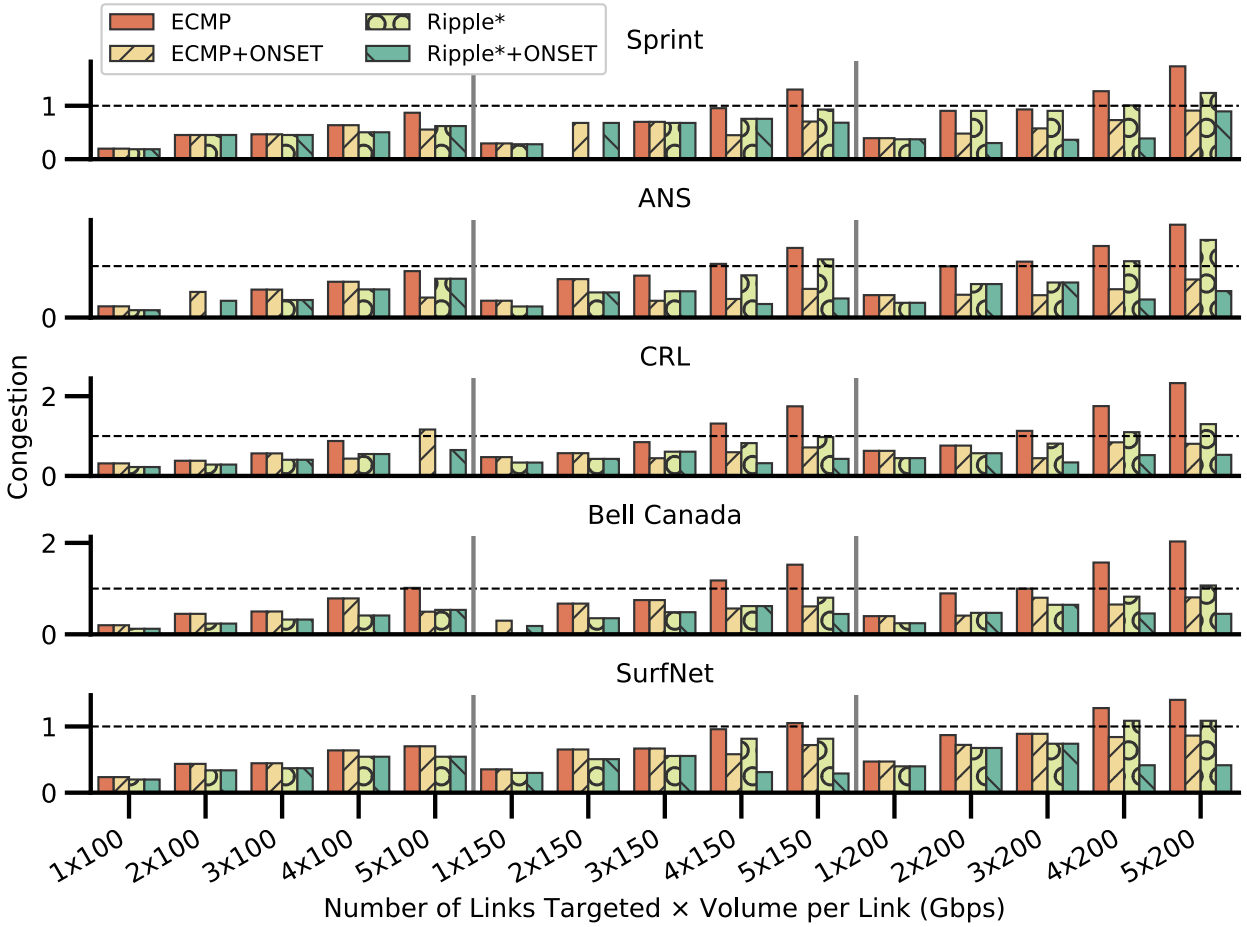


Fig. 10: Network congestion induced by coremelt attacks varying in strength and total targets on networks with different routing strategies and optical topology programming capabilities. The x-axis is encoded (links targeted × attack strength per link).

completely of attack traffic. The y-axis shows the maximum link congestion (max congestion) in the network. When max congestion is greater than 1 the attacker successfully induces traffic loss.

We separate these results based on routing strategy (ECMP or Ripple*) and whether or not the network employed an ONSET topology programming defense. We see that SDN-based routing with Ripple's defense can offer notable savings up to a point. For example, in the CRL network, when 5 links are targeted with a 100 Gbps attack each, this network experiences congestion loss. However, *Ripple*+ONSET is able to prevent congestion loss in every 100 Gbps attack against 5 or fewer links in every network.*

We also observe that link-state routing with ECMP has greater difficulty mitigating loss from adversarial traffic. A 100 Gbps attack is able to induce congestion when only two links are targeted in ANS. As the number of targets increases to three, all of the networks experienced congestion loss. In every attack shown, ONSET is able to find a topology and routing solution in under 1 minute that completely mitigates all congestion loss.

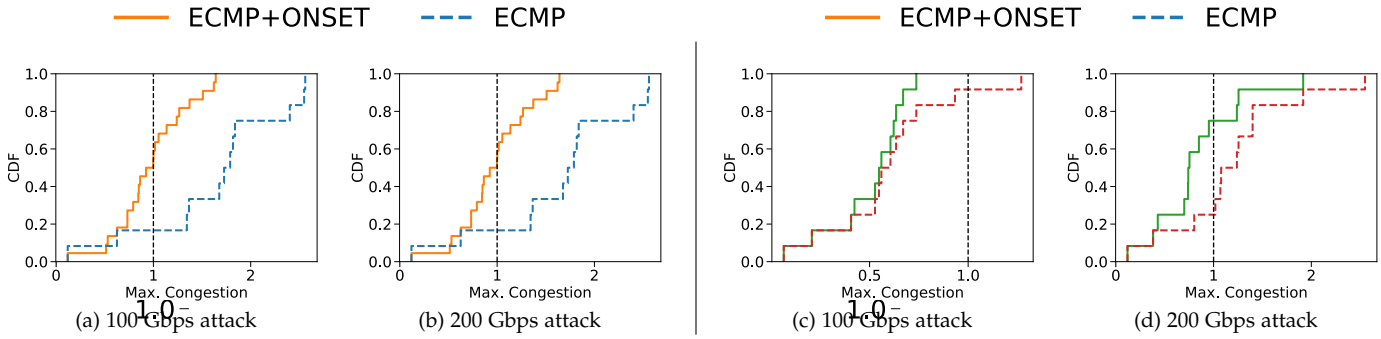
Summary: *Out of 94 crossfire attacks against 5 networks, only 15 attacks resulted in congestion loss with ONSET. Of the routing-based defense without ONSET (plain ECMP or Ripple*) 68 of the attacks resulted in traffic loss. ONSET reduced loss rates in the limited cases where it faced loss.*

6.3 Crossfire Attack

We evaluate the resilience of ECMP and the Ripple* defense against crossfire attacks, where each node in each network is targeted with a 100 Gbps attack on all incident links and a 200 Gbps attack on all incident links. Similar to our evaluation of ONSET's added benefit for Coremelt attacks, these parameters are chosen because they represent a range of moderate to strong attacks that are capable of inducing traffic loss under ECMP and Ripple* respectively. In this section, we highlight the results for both these attacks on Sprint, ANS, and CRL, Bell Canada and SurfNet. Figures 11–15 show the results for each network. Subfigures, (a) and (b), show the effect on max. congestion when the network uses ECMP routing with and without ONSET for a 100 Gbps attack (a) and a 200 Gbps attack (b). Subfigures (c) and (d) show the effect when the network uses the Ripple* defense.

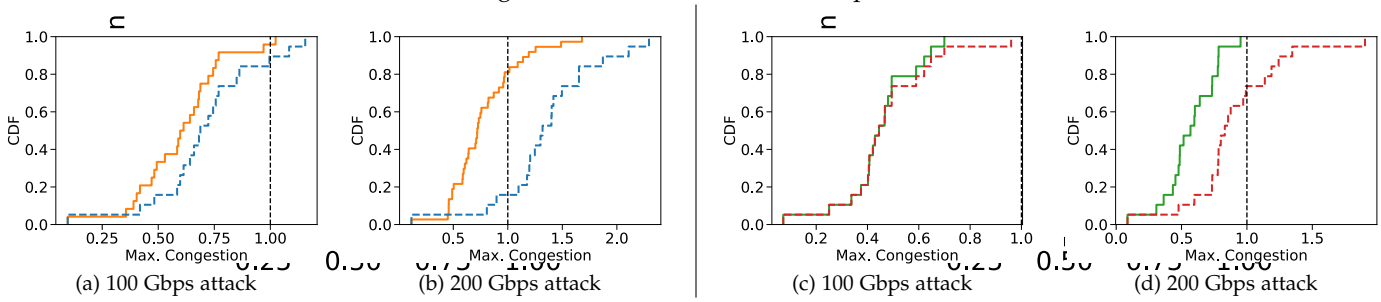
We find that the link-state routing protocol, ECMP, is highly vulnerable to crossfire attacks. An attack of 100 Gbps is enough to cause congestion for approximately 20% of the 100 Gbps attacks to induce traffic loss. In comparison, ONSET had congestion loss in less than 5% of all events at this volume.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3391188



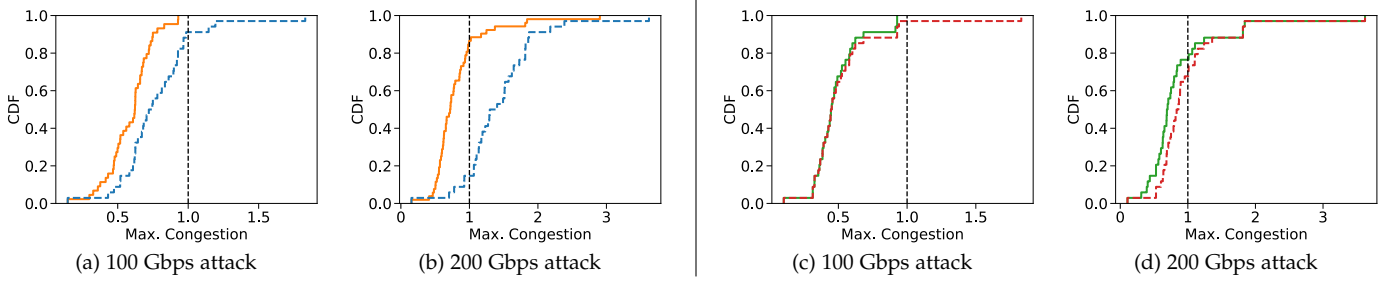
(a) 100 Gbps attack (b) 200 Gbps attack (c) 100 Gbps attack (d) 200 Gbps attack

Fig. 11: All Crossfire Attacks on Sprint.



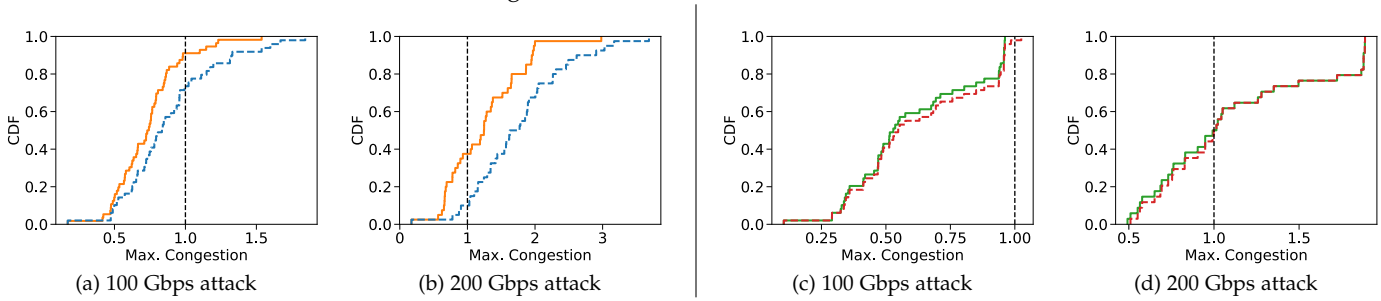
(a) 100 Gbps attack (b) 200 Gbps attack (c) 100 Gbps attack (d) 200 Gbps attack

Fig. 12: All Crossfire Attacks on ANS.



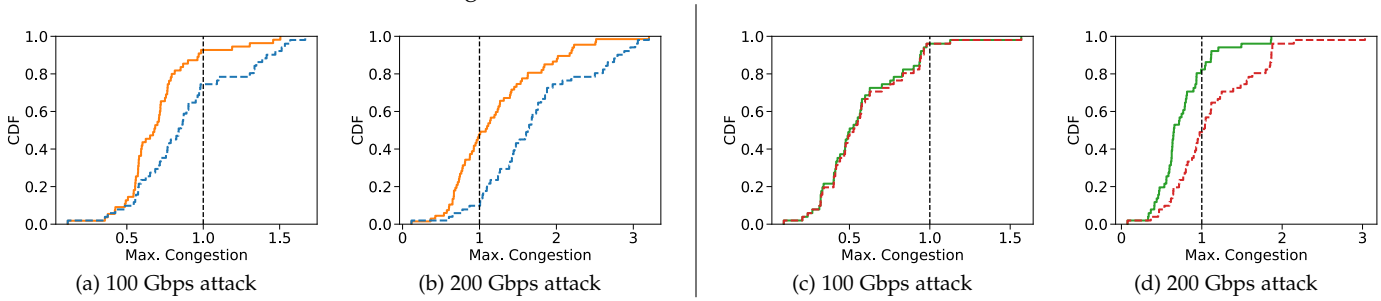
(a) 100 Gbps attack (b) 200 Gbps attack (c) 100 Gbps attack (d) 200 Gbps attack

Fig. 13: All Crossfire Attacks on CRL.



(a) 100 Gbps attack (b) 200 Gbps attack (c) 100 Gbps attack (d) 200 Gbps attack

Fig. 14: All Crossfire Attacks on Bell Canada.



(a) 100 Gbps attack (b) 200 Gbps attack (c) 100 Gbps attack (d) 200 Gbps attack

Fig. 15: All Crossfire Attacks on Surf Net.

When networks use the Ripple* defense, they can aptly mitigate the lower-rate, 100 Gbps attacks (Figures 11c, 12c, and 13c). However, for larger scale attacks, at the 200 Gbps level (Figures 11d, 12d, and 13d) 68 of the attacks are successful at inducing congestion. Ripple*+ONSET had 37 congestion loss events for the same set of attacks.

The reason that performance in Sprint is not perfect for every attack is due to the size of the network. It is the smallest network in the evaluation and therefore has the fewest possibilities for adding links dynamically with ONSET. Similarly, the aggregate bandwidth from the attacks is concentrated on fewer total links, magnifying their impact. This underscores the notion that bandwidth *is* limited—even if you can establish new links opportunistically. However, ONSET increases the amount of traffic needed by an attacker to induce congestion loss with an LFA.

Summary*: Defending Crossfire attacks with ONSET can greatly improve the defensive posture of a network and is complementary to SDN defenses that adapt the forwarding behavior of network traffic. In total, we simulated ONSET against 222 attacks on five networks. ECMP (without ONSET) led to traffic loss in 84 attacks (67%). With ONSET, ECMP led to traffic loss in 6 attacks (4%). The Ripple* defense without ONSET resulted in traffic loss for 50 of the 124 attacks (40%). With ONSET, the number of attacks that resulted in traffic loss fell to 3 (2%).*

6.4 Rolling Attack

Next, we evaluate ONSET's ability to adapt to an ongoing/rolling attack. We evaluate a series of traffic matrices constructed to model several attacks. We model the attack traffic matrices for Crossfire and Coremelt attacks as described above. We also included a Spiffy attack, where the attacker gradually increases their demand until a cost threshold and targets a link that is expected to be shared by the greatest number of paths.

We simulated seven attacks, sampling traffic metrics (throughput/loss/congestion) at 5-second intervals over a 60-minute period. The time between attacks varies from 5 seconds to 5 minutes. Figure 16 shows the network performance with respect to congestion during these attacks for a Ripple* routed network. Figure 17 shows simulated network congestion over an hour, sampled at 5-minute intervals for rolling attacks in an ECMP-routed network with and without ONSET. The black dashed line at Congestion = 1.0 marks the loss threshold; congestion beyond that point results in traffic loss. These results show that the ONSET framework can quickly adapt to dynamic attacks. In over 90 percent of instances, ONSET mitigates attack-induced congestion loss.

Figure 18 shows the total number of active network links during the rolling attacks. Our optimization is triggered whenever congestion is above the loss threshold. If congestion remains above that threshold, then we invoke the optimization again to find more links to add to the network. In every event, the optimizer yields a solution that the network can instantiate in under sixty seconds. The added links are released when congestion reduces back to a level seen before the attack started. Therefore, in the last two attacks, which happen in quick succession, the number of links drops as the attack ends and then quickly jumps up again after the next attack begins. The decision to release the added links after the attack or not is configurable by the network operator, but for this demonstration, we chose to release them.

Summary*: ONSET can be used with SDN and link-state routing to react and adapt to rolling attacks. When traffic demand falls after an attack is over, ONSET is able to detect the change in utilization and deactivate links that it had activated. These fallow transponders can then be used to respond to new attacks that target different sets of links.*

6.5 Cost Benefit Analysis

We now assess how the cost of provisioning ONSET (i.e., the capital expense for hardware required to realize optical topology programming) compares with defenses on a static topologies. To this end, we count the number of transponders required to insulate legitimate traffic from attack induced-congestion when an attack occurs leveraging 2x and 3x the bandwidth of one transponder. Table 3 shows that the cost-benefit of ONSET comes from scaling our defense with the number of nodes in the network, rather than links; to defend an arbitrary attack in a static topology, you must over-provision all of the links by a factor, e.g., 2 or 3x, depending on what the volume of the attacks you want to be protected from is. In ONSET, if you simply provision 1 or 2 fallow transponders per node, you can provide the same bandwidth guarantee for any link without over-provisioning them all. To see this intuitively, consider star graph with 5 spokes. To guarantee an attack threatening 2x bandwidth utilization on any link, you will need 20 transponders (4 per edge given by 2 per each end of each link). If you wanted to provide the same benefit with ONSET, you just need 16 (the original 10 and one more per each of the six nodes). This benefit is modest for the simple example but translates to hundreds of transponders in savings for real-world networks as seen in Table 3.

Network	2x Static	2x ONSET	3x Static	3x ONSET
Sprint	72	47	108	58
ANS	100	68	150	86
CRL	152	109	228	142
Bell Canada	256	176	384	224
SurfNet	272	186	408	236

TABLE 3: Cost to defend an attack threatening 2 or 3x Max Link Utilization on an arbitrary link with a Static Topology vs. ONSET.

6.6 Cost Reduction via Variable Fallow Transponder Allocation

Cost numbers in Table 3 and link ranks from Figure 6a together suggest that we may be able to further reduce the cost of provisioning ONSET by deploying more fallow transponders around critical links and fewer fallow transponders at other nodes in the network. We evaluate this prospect by starting with a naïve approach wherein we provision 10 fallow transponders to the top 10% ranked links (given by link rank metric defined in § 4.1) and then provision half as many fallow transponders at every other node in the network. We then simulate coremelt attacks on each single network link and compare the performance of ONSET with

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3391188

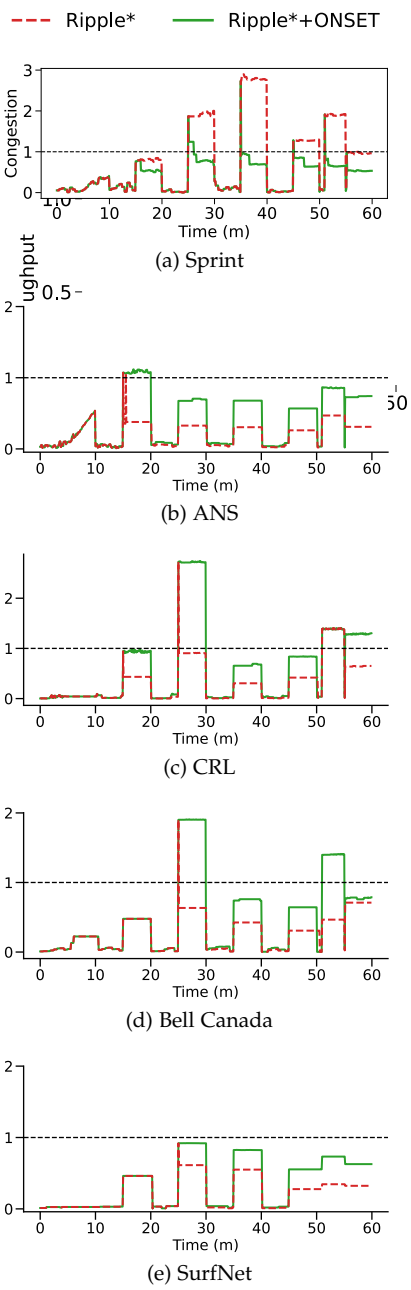


Fig. 16: Max. Link Congestion During Rolling Attacks on different networks, Ripple* vs. Ripple*+ONSET

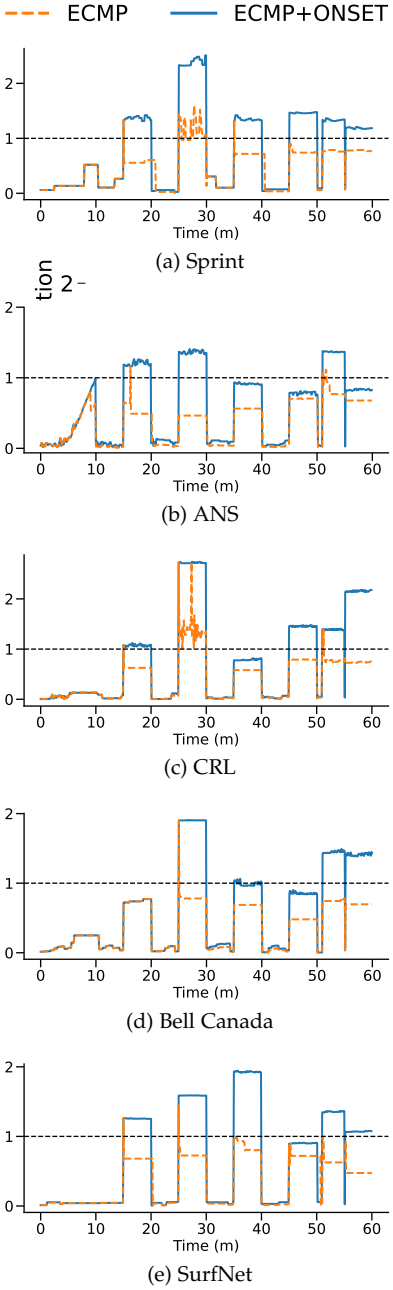


Fig. 17: Max. Link congestion During Rolling Attacks on different networks, ECMP vs. ECMP+ONSET

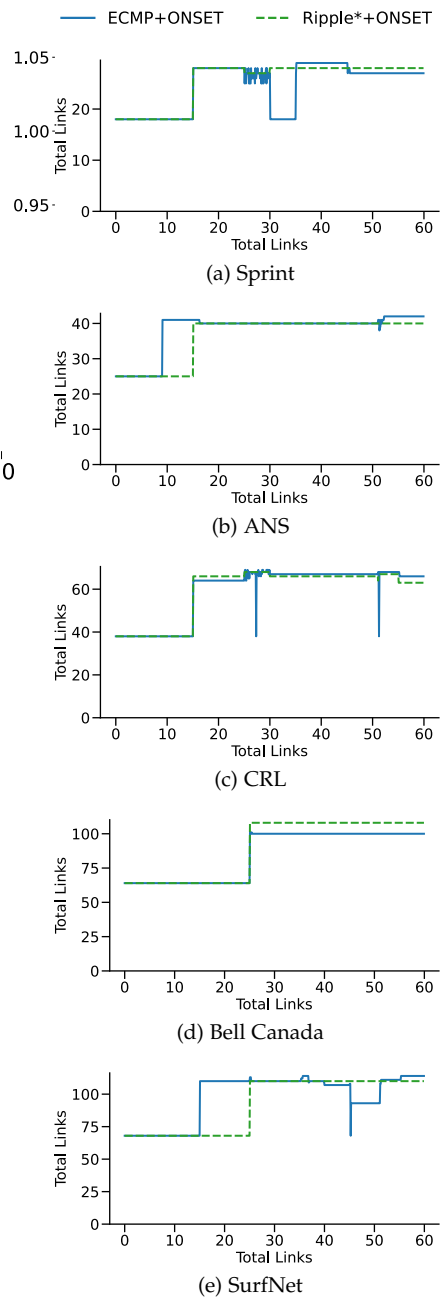


Fig. 18: Total Network Links Active During Rolling Attacks on different networks, ECMP+ONSET vs. Ripple*+ONSET.

the static and variable fallow transponder allocations. We reproduce this experiment for all of the networks considered in this study, both using ECMP routing and the Ripple* defense. Our results conclusively show that reducing the number of fallow transponders we provision for the bottom 90% of nodes does not reduce the performance of ONSET in defending single-link coremelt attacks—the results were identical to those seen in Figure 10.

Motivated by this result, we explore the effect of variable fallow transponder allocations on ONSET's performance more deeply. In this pursuit, we seek for a *decision-support capability* to determine the appropriate fallow transponder allocation strategy based on an operator's budget and the magnitude of loss they are willing to tolerate. We now allocate only *two* fallow transponders to each node if the node's *rank* is greater than or equal to a given rank, n. We vary n over all of the numeric *rank* values for nodes in the given network. We identify the *cost* of an allocation n as the total number of fallow transponders provisioned under that allocation. In practice, this cost can be swapped with the dollar value of that same number of transponders. Figure 19 shows the cost of each allocation strategy in ANS (19b) and

CRL (19a). The most costly solution is to deploy the fallow transponders at every node (where $n \geq 1$). As n grows, we restrict fallow transponders to more highly-ranked nodes. If we limit these to nodes with a rank of 3 or higher, we reduce the cost from 36 to 24 in ANS, and from 66 to 36 in CRL.

To gauge the relative *value* of each of these allocations, we enumerate a series of stressful attacks against every link in each network, repeating this series of attacks on the networks under each fallow transponder allocation, n. We plot the total number of *loss events* for this set of attacks against the cost of a given fallow transponder allocation. We conducted this experiment for both ECMP-based routing and Ripple*. The results, shown in Figure 20, show a Pareto front cost and loss events under each allocation n. In these graphs, better quality solutions fall closest to the origin of the graph, where Loss Events and Cost are both minimized.

Summary: *We provide a decision-support capability in ONSET with which operators can choose how to deploy fallow transponders based on their needs and budget. In practice, an operator can use this capability to deploy ONSET by leveraging data from historical attacks they have been exposed to and the existing routing and defense strategy they employ.*

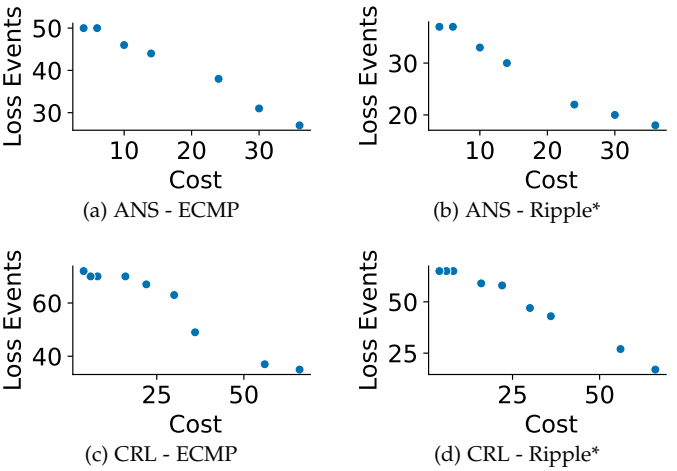


Fig. 20: Cost vs. Loss Events for various networks under ECMP or Ripple*. As cost increases and fallow transponders are deployed more liberally, the number of Loss Events for the set of attacks falls. An operator may use charts similar to these, with their own network and historical attack data sets, to determine which level of defense they would like to

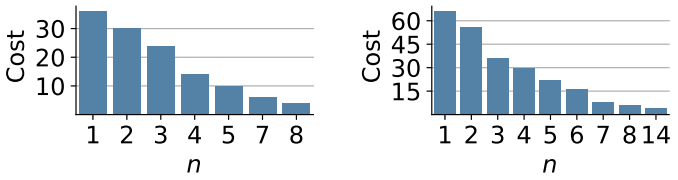


Fig. 19: Cost vs. n where cost is the number of fallow transponders allocated to the network for different values of n. n is defined as the minimum rank a node must have to be allocated fallow transponders. When n is equal to one all nodes receive fallow transponders.

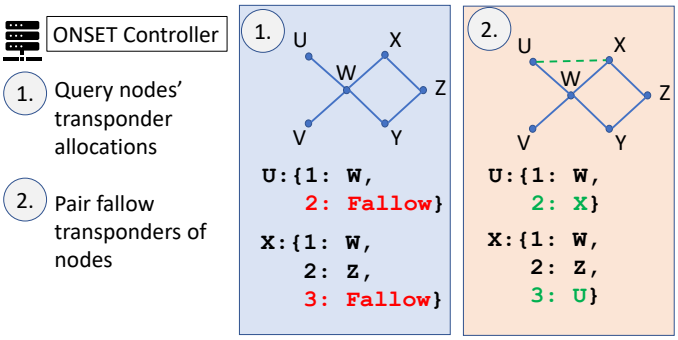


Fig. 21: The ONSET controller leverages its optical layer API to query the set of transponders at the two nodes, U and X. It finds that the pair of nodes each have a fallow transponder. It maps the fallow transponder at U to X and the fallow transponder at X to U. After the transponders are mutually paired the link is active and able to forward traffic.

7 Future Work

In this section, we discuss three opportunities that we plan to explore as part of future work.

(1) Topology Programming API: As ongoing work, we are considering methods to construct a high-level API that can be leveraged to programmatically control network topology and routing. Concretely, our envisioned list of API calls includes:

1) `get_available_transponder(`$node$`)` which returns an index to a fallow transponder at $node$
2) `add_circuit(`$node_u, node_v$`)` which queries fallow transponders at both nodes and pairs them.
3) `get_peer_transponder(`$node_u, node_v$`)` which returns an index to a $node_u$ transponder peered with $node_v$
4) `drop_circuit(`$node_u, node_v$`)` which queries peered transponders at both nodes and de-allocates them.

An example of how these API calls can be employed in coordination with the optimization model described in § 4.2 is shown in Algorithm 1. In this example, `SIG_LFA_DETECTED` and `SIG_LFA_OVER` are flags that are set by a network monitor. We assume that this signal is generated by a mechanism outside the scope of this work, e.g., from a programmable switch. This program is agnostic to the type of LFA occurring (e.g., crossfire or coremelt). It uses link utilization data to choose where to add one or more flux links to the network using the available fallow transponders.

Line 2 states the triggering condition for activating the optimization step. Line 3 invokes the optimized method from § 4.2. The solver returns a set of links that will minimize max link utilization in the network, and in lines 5–6 the links are added to the network with the link provisioning API call. When LFA is over, lines 9–10 remove the flux links from the network.

Other low-level optical hardware configuration requirements must be met to support this high-level API, e.g.,

Algorithm 1 Topology Programming API for LFAs

```
flux_links ← [ ]
if SIG_LFA_DETECTED then
    flux_links ← optimize_topology()
end if
for (u, v) ∈ flux_links do
    add_circuit(u, v)
end for
if SIG_LFA_OVER then
    for (u, v) ∈ flux_links do
        drop_circuit(u, v)
    end for
end if
```

configuring transponder power, amplifier gain adjustments on the optical path, and configuring paths with ROADMs. In this work, we are most interested in defining the requirements of our framework at a high level and evaluating the potential benefit of it for LFAs.

Figure 21 illustrates the controller's view of transponder allocations while using the API. The API enables the network operator programmatically query the set of *allocated* and *fallow* transponders at each node in the network. The API also has methods to pair fallow transponders together, thereby establishing a new link in the network. When a link is added to the topology, a pair of fallow transponders between the nodes is activated and those transponders become unavailable for future links until the pair is deactivated.

(2) Topology Jitter: Before launching infrastructure-centric attacks targeting specific network links such as Crossfire [57], attackers must obtain sufficient network topology information, usually through network reconnaissance. This is effective if the network topology is stable/static and attackers use path probing tools such as `traceroute`. Existing countermeasures [20], [58] on infrastructure attacks tend to distinguish between legitimate and attack traffic without handling network reconnaissance. However, these solutions make an unrealistic assumption that link flooding attack traffic is distinguishable from legitimate traffic while reconnaissance tools let attackers easily probe the network paths around the target link and access public services with "indistinguishable" traffic. Ideally, we should thwart attackers' reconnaissance to effectively mitigate the attacks from the root.

To tackle network reconnaissance, we plan to investigate "topology jitter" using ONSET. The idea is to employ a *moving-target defense* by dynamically changing the optical topology to combat network reconnaissance in two steps. (1) In the first step, we will enable *dynamic capacities* by invoking ONSET to allocate new wavelengths on-demand to physically isolate suspicious and malicious flows and steer away from the attack-induced congestion on a targeted link. (2) Second, we will write a defense application using the API calls described above to periodically reallocate wavelengths for suspicious traffic in the optical layer.

(3) Stress Testing and Adversarial Considerations: Our simulation-based analysis of ONSET only scratches the surface for evaluating a topology-programming defense against LFAs. In the previous section, we have attempted to deeply explore basic questions regarding the potential benefit of ONSET with some generous assumptions regarding the availability of optical resources while looking deeply at the effect of network throughput in the face of high-volume attacks. More work is yet to be done in expanding this analysis; for example, we have yet to consider the cross-traffic dynamics for legitimate and benign traffic as they compete with network services on a dynamic topology. Low-level implementation of the physical links concerning optical-grid spacing and the impact of bandwidth-variable transceivers on the defense framework is also a ripe area of exploration for future work. We hope that our open-source implementation of the framework aids researchers in exploring this area more deeply.

Inspired by [59], adversarial considerations including potential attacks against the ONSET system, overwhelming the compute capability of the network controller that runs the optimization to configure the network topology, among others, are also needed. We plan to consider these as part of future work.

8 Related Work

In addition to the work described in § 2, we refer the readers to recent surveys [60], [61] about LFAs and other DDoS attacks. We cover a few other related efforts here.

Software-based DDoS Defense: SDN and network function virtualization (NFV) enable a wide range of software solutions to detect and mitigate DDoS attacks. For instance, Bohatei [19] orchestrates available NFV resources dynamically to allocate sufficient defense capabilities towards various volumetric attack vectors. SPIFFY [33] leverages SDN capabilities to temporarily increase the bandwidth on a congested link by rerouting around the link and identify the potential attackers via sudden bandwidth augmentation. While software-based defenses bring highest flexibility, they do not scale to terabit LFAs. ACC-Turbo [35] presents a programmable switch based defense for pulse-wave DDoS attacks without dropping suspicious traffic, but rater, prioritizing it. However, ACC-Turbo is not suited towards sustained LFAs, and when congested, will drop traffic.

Switch-based DDoS Defense: Programmable switches have emerged as a promising platform to perform DDoS detection and mitigation. Unlike traditional switches that focus only on packet forwarding, programmable switches adopt a new type of programmable ASICs and can support additional computation (e.g., DDoS related computation like packet filtering, rate limiting, and hash tables) at a per-packet basis while retaining high line rate guarantees. For instance, Poseidon [21] uses programmable switches as a first-line defender to augment a DDoS scrubbing cluster. Jaqen [34] introduces a switch-native approach to detect and mitigate volumetric attacks. Their design includes a range of probabilistic data structures to efficiently utilize the switch resources for DDoS defense. However, switch-based DDoS defenses highly rely on accurate identification of malicious and benign traffic, which is fundamentally challenging in LFA scenarios where attack traffic may appear as legitimate.

Topology Obfuscation Techniques: There has been a concerted effort to stop attackers from gaining the information about topology required to launch an LFA. These efforts revolve around topology obfuscation, or techniques to hide

topological information from an adversary. Efforts include NetHide [62], BottleNet [63], EqualNet [64] and references therein. Topology obfuscation is an orthogonal goal to LFA mitigation. In this work we assume that the attacker has gained knowledge of the topology, and is able to use that knowledge to launch their attacks. We are concerned with finding ways to mitigate loss that may occur during such an attack.

Topology Reconfiguration Techniques: Optical layer topology programming has recently gained attention in several networking contexts. Its benefits have been demonstrated in the context of traffic engineering in WANs [26], [65], [66] and data centers [24], [25]. Prior work has posed topology reconfiguration to augment DDoS defense [45], [46]. Our paper moves beyond prior work by providing the first general framework for an optical defenses against LFAs and demonstrating its applicability to various networks.

9 Summary

LFAs present a particularly insidious and difficult-to-defend-against form of DDoS attacks. While some early work has proposed LFA defenses, the techniques treat the network topology as a static resource and only alter the forwarding behavior for traffic. Consequently, they incur fundamental limitations in terms of tackling attacks, or worse inducing collateral damage elsewhere in the network. Our vision is to leverage optical layer advancement called topology programming to augment existing LFA defense capabilities. Our framework, ONSET, paves the way for this feat. ONSET jointly optimizes topology and routing, using fallow transponders at nodes in the network to create opportunistic links. We show via *what-if* style analysis that ONSET amplifies the benefits of existing LFA defenses for diverse terabit attack scenarios and for a diverse set of networks.

References

[1] Mitre, "Network denial of service: Direct network flood," https://attack.mitre.org/versions/v10/techniques/T1498/001/, 2022.

[2] D. Warburton, E. Ojeda, and M. Heath, "2022 application protection report: Ddos attack trends," https://www.f5.com/labs/articles/threat-intelligence/2022-application-protection-report-ddos-attack-trends, 2022, f5.

[3] NETSCOUT, "Issue 8: Findings from 2nd half 2021," *Netscout Systems, Inc., Tech. Rep*, 2021.

[4] O. Yoachimik, "Ddos attack trends for 2022 q2," https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2/, 2022, cloudFlare.

[5] J. Russell, "The world's largest ddos attack took github offline for fewer than 10 minutes," https://techcrunch.com/2018/03/02/the-worlds-largest-ddos-attack-took-github-offline-for-less-than-tens-minutes/, 2018.

[6] A. Toh, "Azure ddos protection—2021 q3 and q4 ddos attack trends," https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/, 2021.

[7] O. Yoachimik, "Cloudflare ddos threat report for 2022 q4," https://blog.cloudflare.com/ddos-threat-report-2022-q4/, 2022, cloudFlare.

[8] Akamai, "Akamai security solutions," https://www.akamai.com/us/en/products/security/, 2019.

[9] AWS, "Aws shield: Managed ddos protection," https://aws.amazon.com/shield/, 2019.

[10] CenturyLink, "Centurylink ddos mitigation," http://www.centurylink.com/asset/business/enterprise/brochure/ddos-mitigation.pdf, 2019.

[11] Cloudflare, "Advanced ddos attack protection," https://www.cloudflare.com/ddos/, 2019.

[12] M. N. Kumar, P. Sujatha, V. Kalva, R. Nagori, A. K. Katukojwala, and M. Kumar, "Mitigating economic denial of sustainability (edos) in cloud computing using in-cloud scrubber service," in *2012 Fourth International Conference on Computational Intelligence and Communication Networks*. IEEE, 2012, pp. 535–539.

[13] F. Baker and P. Savola, "Ingress filtering for multihomed networks," Internet Requests for Comments, RFC Editor, RFC 3704, March 2004. [Online]. Available: https://tools.ietf.org/html/rfc3704

[14] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing," Internet Requests for Comments, RFC Editor, RFC 2827, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2827.txt

[15] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed ddos traffic," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, pp. 30–41.

[16] A. Yaar, A. Perrig, and D. Song, "Stackpi: New packet marking and filtering mechanisms for ddos and ip spoofing defense," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1853–1863, 2006.

[17] J. M. Smith and M. Schuchard, "Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive bgp routing," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 599–617.

[18] R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods," in *9th USENIX Security Symposium (USENIX Security 00)*, 2000.

[19] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 817–832. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/fayaz

[20] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *30th USENIX Security Symposium (USENIX Security 21)*. Vancouver, B.C.: USENIX Association, Aug. 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/xing

[21] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric ddos attacks with programmable switches," in *the 27th Network and Distributed System Security Symposium (NDSS 2020)*, 2020.

[22] "The history of Optical and Ethernet," https://www.ciena.com/insights/infographics/Packet-Optical-Convergence-Infographic-prx.html.

[23] M. Li, D. Zaccarin, and C. Barnard, "Reconfigurable optical add-drop multiplexer," Feb. 27 2007, uS Patent 7,184,666.

[24] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 447–458, 2013.

[25] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 216–229.

[26] R. Durairajan, P. Barford, J. Sommers, and W. Willinger, "Greyfiber: A system for providing flexible access to wide-area connectivity," *arXiv preprint arXiv:1807.05242*, 2018.

[27] M. Nance-Hall and R. Durairajan, "Bridging the optical-packet network chasm via secure enclaves," in *Proceedings of the ACM SIGCOMM 2020 Workshop on Optical Systems Design. ACM*, 2020.

[28] M. Nance-Hall, P. Barford, K.-T. Foerster, M. Ghobadi, W. Jensen, and R. Durairajan, "Are wans ready for optical topology programming?" in *Proceedings of the ACM SIGCOMM 2021 Workshop on Optical Systems*, ser. OptSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 28–33. [Online]. Available: https://doi.org/10.1145/3473938.3474510

[29] R. M. Krishnaswamy and K. N. Sivarajan, "Design of logical topologies: A linear formulation for wavelength-routed optical networks with no wavelength changers," *IEEE/ACM Transactions On Networking*, vol. 9, no. 2, pp. 186–198, 2001.

[30] M. Hall, R. Durairajan, and V. Sekar, "Fighting Fire with Light: A Case for Defending DDoS Attacks Using the Optical Layer," arXiv 2002.10009, 2020.

[31] A. Studer and A. Perrig, "The coremelt attack," in *Computer Security – ESORICS 2009*, M. Backes and P. Ning, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 37–52.

[32] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 127–141.

[33] M. S. Kang, V. D. Gligor, and V. Sekar, "Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks." in *NDSS*, 2016, pp. 53–55.

[34] Z. Liu, H. Namkung, G. Nikolaidis, J. Lee, C. Kim, X. Jin, V. Braverman, M. Yu, and V. Sekar, "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3829–3846.

[35] A. G. Alcoz, M. Strohmeier, V. Lenders, and L. Vanbever, "Aggregate-based congestion control for pulse-wave ddos defense," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 693–706.

[36] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[37] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: Experience with a globally-deployed software defined wan," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 3–14.

[38] A. D. Ferguson, S. D. Gribble, C.-Y. Hong, C. E. Killian, W. Mohsin, H. Muehe, J. Ong, L. Poutievski, A. Singh, L. Vicisano *et al.*, "Orion: Google's software-defined networking control plane." in *NSDI*, 2021, pp. 83–98.

[39] M. Filer, J. Gaudette, Y. Yin, D. Billor, Z. Bakhtiari, and J. L. Cox, "Low-margin optical networking at cloud scale," *Journal of Optical Communications and Networking*, vol. 11, no. 10, pp. C94–C108, 2019.

[40] M. Filer, J. Gaudette, M. Ghobadi, R. Mahajan, T. Issenhuth, B. Klinkers, and J. Cox, "Elastic optical networking in the microsoft cloud," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 8, no. 7, pp. A45–A54, 2016.

[41] S. Tse and G. Choudhury, "Real-time traffic management in at&t's sdn-enabled core ip/optical network," in *Optical Fiber Communication Conference*. Optica Publishing Group, 2018, pp. Tu3H–2.

[42] "Verizon optical wave service," https://www22.verizon.com/wholesale/solutions/solution/Optical+Wave+Service.html, 2023.

[43] M. Nance-Hall, K.-T. Foerster, S. Schmid, and R. Durairajan, "A survey of reconfigurable optical networks," *Optical Switching and Networking*, vol. 41, p. 100621, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1573427721000187

[44] Z. Zhong, M. Ghobadi, M. Balandat, S. Katti, A. Kazerouni, J. Leach, M. McKillop, and Y. Zhang, "Bow: First real-world demonstration of a bayesian optimization system for wavelength reconfiguration," in *Optical Fiber Communication Conference*. Optical Society of America, 2021, pp. F3B–1.

[45] M. Nance-Hall, G. Liu, R. Durairajan, and V. Sekar, "Fighting fire with light: Tackling extreme terabit ddos using programmable optics," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, ser. SPIN '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 42–48. [Online]. Available: https://doi.org/10.1145/3405669.3405824

[46] Y. Shen, R. Goodfellow, M. S. Glick, G. Bartlett, and K. Bergman, "Optical mitigation of ddos attacks using silicon photonic switches," in *Metro and Data Center Optical Networks and Short-Reach Links III*, vol. 11308. International Society for Optics and Photonics, 2020, p. 113080J.

[47] D. Braess, A. Nagurney, and T. Wakolbinger, "On a paradox of traffic planning," *Transportation science*, vol. 39, no. 4, pp. 446–450, 2005.

[48] R. Singh, N. Bjorner, S. Shoham, Y. Yin, J. Arnold, and J. Gaudette, "Cost-effective capacity provisioning in wide area networks with shoofly," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 534–546.

[49] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[50] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *16th annual symposium on foundations of computer science (sfcs 1975)*. IEEE, 1975, pp. 184–193.

[51] P. Kumar, C. Yu, Y. Yuan, N. Foster, R. Kleinberg, and R. Soulé, "Yates: Rapid prototyping for traffic engineering systems," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3185467.3185498

[52] "Gurobi optimizer," https://www.gurobi.com/.

[53] O. Michel and E. Keller, "Sdn in wide-area networks: A survey," in *2017 Fourth International Conference on Software Defined Systems (SDS)*, 2017, pp. 37–42.

[54] V. Heorhiadi, "TMgen," https://github.com/progwriter/tmgen, 2020.

[55] M. Birk, P. Gerard, R. Curto, L. Nelson, X. Zhou, P. Magill, T. Schmidt, C. Malouin, B. Zhang, E. Ibragimov, S. Khatana, M. Glavanovic, R. Lofland, R. Marcoccia, R. Saunders, G. Nicholl, M. Nowell, and F. Forghieri, "Coherent 100 gb/s pm-qpsk field trial," *Communications Magazine, IEEE*, vol. 48, pp. 52 – 60, 08 2010.

[56] M. Filer, J. Gaudette, Y. Yin, D. Billor, Z. Bakhtiari, and J. L. Cox, "Low-margin optical networking at cloud scale [invited]," *J. Opt. Commun. Netw.*, vol. 11, no. 10, pp. C94–C108, Oct 2019. [Online]. Available: http://opg.optica.org/jocn/abstract.cfm?URI=jocn-11-10-C94

[57] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 127–141.

[58] M. S. Kang, V. D. Gligor, and V. Sekar, "Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks." in *NDSS*, 2016.

[59] M. A. Aladaileh, M. Anbar, I. H. Hasbullah, Y.-W. Chong, and Y. K. Sanjalawe, "Detection techniques of distributed denial of service attacks on software-defined networking controller–a review," *IEEE Access*, vol. 8, pp. 143 985–143 995, 2020.

[60] R. ur Rasool, H. Wang, U. Ashraf, K. Ahmed, Z. Anwar, and W. Rafique, "A survey of link flooding attacks in software defined network ecosystems," *Journal of Network and Computer Applications*, vol. 172, p. 102803, 2020.

[61] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[62] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and practical network topology obfuscation," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 693–709.

[63] J. Kim, J. Nam, S. Lee, V. Yegneswaran, P. Porras, and S. Shin, "Bottlenet: Hiding network bottlenecks using sdn-based topology deception," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3138–3153, 2021.

[64] J. Kim, E. Marin, M. Conti, and S. Shin, "Equalnet: a secure and practical defense for long-term network topology obfuscation," *Proceedings of the USENIX NDSS*, 2022.

[65] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford, "Optimizing bulk transfers with software-defined optical wan," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 87–100.

[66] Z. Zhong, M. Ghobadi, A. Khaddaj, J. Leach, Y. Xia, and Y. Zhang, "Arrow: restoration-aware traffic engineering," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 560–579.

Matthew Nance-Hall is a Ph.D. candidate in computer science from the University of Oregon where he was recognized with the University of Oregon Dissertation Research fellowship. His research interests are in computer networking with a focus wide-area network performance and network security.

Zaoxing "Alan" Liu is an Assistant Professor of Computer Science at the University of Maryland, College Park. My research has won interdisciplinary recognitions, including USENIX FAST Best Paper Award, USENIX ATC "Best of Rest", and ACM STOC "Best of Rest".

Vyas Sekar is the Tan Family Professor of Electrical and Computer Engineering in the ECE Department at CMU with a courtesy appointment in the Computer Science Department. He is affiliated with Cylab and co-directs the Future of Enterprise Security initiative at Cylab.

Ramakrishnan Durairajan is an Associate Professor in the Department of Computer Science at the University of Oregon. His research has been recognized with multiple NSF awards including the NSF CAREER award, a Ripple faculty fellowship, a UO faculty research award, and several best paper awards, and has been covered in several fora.