# SDX-based security collaboration: Extending the security reach beyond network domains

Kristina Dzeparoska, Hadi Bannazadeh, and Alberto Leon-Garcia
Department of Electrical and Computer Engineering
University of Toronto, Toronto, ON, M5S 3G4, Canada
Email: {kristina.dzeparoska}@mail.utoronto.ca, {hadi.bannazadeh, alberto.leongarcia}@utoronto.ca

***Abstract*—Traditionally, internal networks rely on border firewalls as the first line of defence against untrusted external networks. However, customized security measures can begin further away. We have already noticed proliferating deployment of Software Defined Networking (SDN) in internal networks, and even further within the Internet core, at Internet Exchange Points (IXP). Software-Defined Internet Exchange Points (SDX) enable flexible and programmable control over the delivery of wide area network traffic. Therefore, SDX's are an appealing place to introduce security actions that span beyond the edge of internal networks. By extending security actions to SDX's, dedicated border security appliances would no longer be as overwhelmed and wide area network links would no longer deliver traffic only to be dropped at the edge of the destination networks.**

**In this paper we present a hierarchical, logically centralized architecture enabling SDX security policies to be expressed by the Autonomous Systems (AS) as intents. Through SDX collaboration, these security intents can be compiled and installed at the closest available SDX relative to the offending source. Moreover, parallel intent compilation over multiple selected SDX's can be simultaneously executed, thus enabling a distributed security response activated at the Internet core. This approach that relies on SDX's allows faster adoption in contrast to changing all routers, or the Internet architecture. This proposed security collaboration could be used to address massive internet blackouts caused by DDoS attacks with the capacity to match the distributed force of the attacks today and future ones of even greater scale.**

***Index Terms*—Security Collaboration, SDX, IDN, DDoS**

## I. Introduction

Security is becoming increasingly challenging, especially as we witness tremendous network growth producing terabytes of Internet-scale network traffic [1], [2]. This challenge originates from the very nature of the Internet having no security by design [3], [4]. This necessitates having to sift through enormous amounts of daily traffic to separate malicious from benign traffic [5]. In addition, security infrastructure is quite expensive and, unlike sensitive enterprise networks, smaller internal networks often cannot afford to have dedicated, high-performance security devices. Within this context, network carriers might not be motivated to invest in expensive security infrastructure either [6], as they assume the role of "carrier-only" and might apply only some of the minimal recommendations outlined in [7], [8]. Additionally, even if security equipment is available, the attack surface is quite large as there are many vulnerabilities in any network design or architecture that can be exploited [9].

A proposal towards source-based security measures was presented in [10], however it relies on networks deploying the D-WARD defense system per router basis. Rather than dealing with an attack once inside or at the doorstep of the network, it would be more effective to consider Internet-wide security actions to limit incoming attacks or any unwanted, suspicious traffic. This approach would greatly scale down the attack influence footprint. Therefore, we argue that the ideal place for Internet-wide security actions is at Internet Exchange Points [11], specifically, Software-Defined Internet Exchange Points (SDX). This stems from the fact that it is feasible to change IXPs to SDX's. Moreover, with SDX collaboration all parties benefit; members will have even higher incentives to join the SDX's (to utilize the additional security platform), whereas SDX's will leverage from members' participation (the more AS's participate, the greater the surface to limit attacks).

Since there are many possibilities in terms of use-cases for the proposed model, we focus on the most challenging and debilitating network attack. We hope that this proposal will change the outlook on DDoS defense strategies and mitigations and will inspire a next-generation security architecture. Currently, ISPs use blackholing [12] and DDoS scrubbers to deal with DDoS attacks which results in legitimate traffic being dropped as well. Instead, our proposal enables collaborative security policies deployed over multiple SDX's to address the distributed nature of the attack, while mitigating malicious flows only.

The motivation behind this work is to demonstrate that SDX collaboration can orchestrate Internet-wide security actions which then creates a three-fold benefit:

1) First, it allows AS's to have Internet-wide security control over traffic destined to their networks.
2) Second, by distributing the countermeasure it provides a mechanism to match the scale of distributed attacks.
3) Third, it decreases the link utilization in the Internet backbone and access.

These capabilities are realized through AS-level intents to simplify and abstract management and configuration burdens. Each AS (most likely an ISP) would coordinate with its users to receive and forward the intent requests to the local SDX where the AS participates on behalf of its users. The intent aggregation per AS allows easier management for both the AS's and the local SDX's.

978-3-903176-14-0 

## II. Background Overview and Related Work

Internet exchange points (IXPs) [13], [14] are an integral part of the Internet backbone where different networks interconnect to exchange wide area network traffic. However, current practises for traffic routing rely on the Border Gateway Protocol (BGP) which imposes constrained routes based on destination IP prefixes. Moreover, due to BGP's limitations and current inter-domain architecture [15], control over end-to-end paths and fine-grained actions are not available. The research literature has proposed many modifications to BGP to improve its behaviour and enable security, flexibility and traffic engineering properties. However, these modifications demand significant changes to the existing implementation and infrastructure that most IXPs and Autonomous Systems are reluctant to deploy. Instead, a shift towards SDN seems both inevitable and necessary, as it decouples the control from the data plane to enable flexibility and programmability.

### A. Related Work: A Software Defined Internet Exchange Point

SDX is the product of a traditional IXP complemented by SDN. As such, it inherits all the SDN benefits, while allowing backward compatibility to the default IXP behavior. Initially, SDX was introduced in [16], followed by an elaborate research proposal presented in [17]. The authors presented a more scalable version in [18] dubbed "An Industrial-scale SDX". By compiling SDN policies per participant and by adding encoding mechanisms, both control-plane computation and BGP update recompilation challenges are addressed. Additionally, the scalability challenge is tackled through the multi-table feature available in the later versions of OpenFlow. We proceed by briefly describing the general SDX structure and operation.

The SDX architecture follows the IXP architecture, whereby the traditional layer-2 fabric is replaced by a programmable SDN fabric and an SDN controller. The BGP route server (RS) was introduced at IXPs to centralize BGP connections and reduce computational load in participating routers; as such its role remains involved at the SDX as well. Participating Autonomous Systems at the SDX are able to express custom, fine-grained SDN policies that the controller pushes to the programmable SDN fabric as flow entries in the forwarding table. If traffic flow at the SDX does not match any of the custom rules, then the default BGP routing takes precedence. The default BGP routing behaviour is enabled by the RS which manages BGP sessions to exchange network reachability information among the SDX participants. Each AS at the SDX maintains a BGP session with the RS to advertise, withdraw or update BGP information. The RS collects this information, processes it through a best prefix route selection algorithm and determines the best prefix routes. Depending on the specific RS implementation, as well as specific export and import policies that each AS has, there could be variations as to how the RS output should be used to enable the default BGP routing behaviour in the SDN fabric. The SDN fabric is continuously updated to reflect any changes in best routes as computed by the RS.

## III. Proposal Overview

The proposed SDX collaboration enables steering security policies as close as possible to the offending domains. To achieve the above, we leverage a two-tier, hierarchical view to determine the best SDX placement for each policy.

The architecture involves a Global SDX Orchestrator (GSO) which communicates with multiple participating local SDX's (lo-SDX). Relations between the entities follow a tree structure, where GSO, lo-SDX and AS correspond to the root, parent and leaf nodes respectively, as shown in Figure 1. The GSO is assumed to be an independent authority which could be agreed upon by the participating SDX's, or alternatively it could be co-managed by the participating SDX entities.

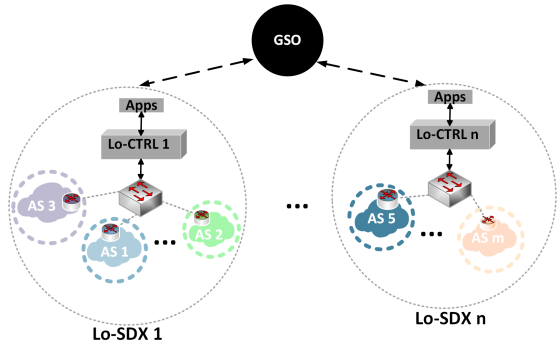



Fig. 1: Proposal Overview

### A. Requirements:

To enable the SDX collaboration we outline several challenging requirements that need to be addressed:

**Scalable Performance:** Hundreds of participating AS's at an SDX will result in hundreds of thousands of IP prefixes, as well as a multitude of AS-specific policies. Clearly, scalability is a major requirement to ensure efficiency and correctness. Even though recent progress in switch design has enabled larger rule size tables [19], at increased cost. Nevertheless, the challenge lies with successfully allocating all the rules from the participating AS's at the SDX within the available fixed rule table sizes. In [18] different techniques were combined and proposed to achieve better scalability. While this challenge is valid per lo-SDX, we additionally tackle the scalability challenge with parallel intent distribution and compilation at a set of elected lo-SDX's as orchestrated by the GSO.

**Intent Authorization:** As Resource Public Key Infrastructure (RPKI) [20] serves BGP to validate route announcements, AS intent requests need to be authenticated and authorized prior the intent compilation process as well.

**Conflict Detection and Resolution:** Before adding or modifying an intent at a lo-SDX we need to determine if it conflicts with any of the existing intents. To enable true intent-driven networking (IDN) [21], automation is key, which in turn requires conflicting intents to be properly addressed during their lifecycle.

### B. Contributions:

The paper's contributions are as follows:

- Autonomous Systems can activate security actions at the Internet core. A hierarchical, two-tier architecture distributes and executes security policies in parallel at lo-SDX's closest to the offending domains.
- A powerful abstraction through IDN permits AS's to declare security-based intents through the lo-SDX they participate in. An IDN framework automates the multi-tier intent lifecycle.
- The considered DDoS use-case is only the first of the opportunities offered by the proposed model; which to the best of our knowledge is the first model exploring the collaboration of Software-Defined IXPs.
- Consequently, decreased workload at network border devices, as well as decreased Internet-wide network link utilization is expected due to less contaminated traffic.

The rest of the paper is organized as follows. Section IV provides a DDoS use-case to showcase the applicability of the proposed model in terms of security applications. Section V provides an overview of the intent abstraction model, an integral part of the IDN framework. In section VI we present the hierarchical two-tier, logically-centralized model. For both tiers we describe the main components and outline the general work-flow. Section VII covers evaluation and results of a simulated DDoS attack experiment. Finally, conclusions and future directions are presented in section VIII.

## IV. Motivation: A DDoS use-case

DDoS attacks have evolved to become serious threats to any targeted system [22], [23]. The distributed nature of the attack renders its victims incapable to respond as resources are overwhelmed and services are degraded to the point of a complete operational halt. Even more, their power is set to rise with the ubiquitous and heterogeneous proliferation of IP-based systems, including IoT. As distribution is the key component fueling DDoS, it is logical to consider distribution as a measure of defense as well. In fact, if we can coordinate distributed defense points that are strategically selected based on proximity to the offending sources, we could potentially disturb the scale of the attack or even mitigate it completely. More specifically, the defense points should ideally be at the core of the Internet to ensure the most defensive impact, which leads us to consider Software-Defined Internet Exchange Points. However, having SDX's operate independently forgoes major benefits. Instead we view each SDX as a defense point on the backbone map and by coordinating these distributed points we can match the DDoS nature of the attack. It is worth noting that there are over 300 IXPs in the world with number of members ranging from 50 to 1000 per IXP [24].

A major DDoS challenge is scalability in terms of maximum number of rules allowed, and while the industrial SDX presented scalability optimizations, it still does not solve the problem completely. If we assume one or few SDX members to be under a DDoS attack, then we can easily approximate that the SDX will receive at least a few million intents from the members to block the offending sources; this will likely collide with the average fabric size (half a million) and fail.

With the proposed SDX collaboration we can split the load to multiple SDX's selected based on source proximity and defense distribution, thus solving the DDoS scalability challenge. Furthermore the DDoS will suffer from this interference causing its magnitude to be greatly decreased leading to the attack being essentially cut-off. Although DDoS attacks have many different attack vectors, each can be addressed as soon as a list of offending sources supplied as intents is compiled and distributed through the collaboration model. For instance with reflection attacks, the abused amplification server becomes the attack source, therefore intents to block traffic from such offending sources to the victim will be compiled and installed. The intents would be prioritized based on sources that originate the most traffic accordingly.

## V. Intent Abstraction

Intent-driven networking (IDN) is an effort to simplify and abstract the SDN north-bound interface (NBI) communications. Although a standard NBI definition is not available yet, the common approach is an intent-based interface used to declare high level policies as opposed to low-level, detailed networking instructions [21]. In simple terms, IDN's paradigm is: "what to do", rather than "how to do it".

We perceive an intent as the desired relation between an ingress and egress network traffic flow. According to a defined set of intent classes, a suitable function maps the intent input to an intent digraph which represents the desired network policy or set of policies. We align with RFC3060 [25] which views policies as a set of rules to administer, manage and control access to network resources, whereby network resources in our case correspond to IP prefixes or infrastructure ports. In the context of our proposed model, once an AS sends an intent to its lo-SDX, an IDN-capable framework takes over and automates the rest of the process. To enable intents to be expressed conveniently in human natural language we leverage the NLTK [26] platform to tokenize, tag and parse.

### A. Intent Specification Data Model

Each intent is processed (as described in section V) to form an intent object as shown in Figure 2, whereby the data model is JSON structured. The format of the intent object is defined by mandatory, optional and supplementary (system) attributes. The intent attributes have multiple roles in processes related to intent validity, flow space authorization, conflict and intent management. Based on specific attributes, intent classes are defined *a priori*. An intent belongs to a specific intent class, if there exists high correlation between certain intent attributes and the said intent class. The intent classes are defined based on the keywords ("DDoS/BL/WL/RL/VNF/modIP") in the JSON model. Based on these keywords the JSON model has applicability to other security scenarios, however in this paper the focus is solely on the DDoS use-case.

```
{intent_origin: {
   Intent_id: {SDX_id:"", AS_id:"", (curr_nb+1):""},
   SDX_request: {SDX_id:"", AS_id:""},
   Request_type: ["add/update/delete/query"],
   intent_class: ["DDoS/BL/WL/RL/VNF/modIP"] },
intent_attributes: {
   nodes: {src_ip:"", dest_ip:""},
   edge: {src_port:"", dst_port:"", ip_prot:"",
   IP_TOS:"", direction:""},
   intent_variables: {priority:"", timeout:""},
   action: ["block/allow/rate/VNF/mod_dest_IP"],
     action_attributes: {
       action_threshold: {min:"", max:""},
       action_VNF: {ip:"", mode:"IPS/tap/honeypot"},
       action_MOD: {action_mod_dest_IP:""}} },
intent_execute: {
   ports: {src_mac:"", dst_mac:"", in_port:""},
   management: {metadata:"", cookie:""},
   SDX_exec: {SDX_id:"", AS_id:""} } }
```

Fig. 2: Intent Specification JSON structured Data Model

The "Intent_id" attribute contains the requesting lo-SDX, AS and intent number (added by the system based on the current number of intents for the specific AS, augmented by one). The "request_type" determines the type of workflow to be followed, as described later. The "intent_class" designates the function to be used; depending on the class, additional "action_attributes" are supplied to the mapping function. The "intent_attributes" contains nested attributes to assist with flow matching parameters, vertices, edge functions, direction, prioritization, conflicts etc. Lastly the "intent_execute" contains system attributes that assist with flow and intent management. For instance the "cookie" value is used by the controller if an intent needs to be removed or updated; the "cookie" value matches the intent number to associate intents to their respective rules installed at the fabric.

### B. Mapping Functions and Intent Digraph:

Each intent class has a mapping function to enable uniform intent compilation. The function maps the intent attributes from Figure 2 to an intent digraph to represent the policy between the source and destination pair for that intent.

For each intent digraph, the vertices correspond to the source and destination accordingly, whereas a function over the edge specifies the desired flow between the pair. In case of complex policies that correspond to multiple rules, for each distinguished flow rule an edge is added between the vertices with the corresponding function over it. A mapping function could add intermediary nodes based on the intent class it supports. Figure 3 shows a sample digraph.

According to this model we do not anticipate misclassifications, unless the intent contains multiple intent class keywords. In such case we can determine if it should be multiple intents and separate them accordingly, or we can provide feedback to the requesting AS to clarify the intent request.

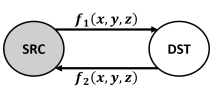



Fig. 3: Sample Intent Digraph

## VI. Design and Workflow

The proposed SDX collaboration relies on a two-tier architecture. The top tier is represented by a logically centralized entity called the Global SDX Orchestrator (GSO), whereas the bottom tier consists of individual, participating local SDX's (lo-SDX). To enforce secure communications between the entities involved, out of band communication is established. In addition, authentication and authorization of the participants at each tier is enabled through TLS. The logically centralized GSO can be physically distributed to address scalability. We describe the general workflow as per Figure 4:

1) An Autonomous System member submits an intent to a lo-SDX where it participates.
2) The lo-SDX starts compiling and analyzing the received intent to determine whether its location is suitable for the intent installation. **Note:** *Except a DDoS intent, which is forwarded to the GSO for proper distribution.*
3) If the lo-SDX is not deemed an adequate location; the intent is forwarded to the GSO. The GSO selects the executing lo-SDX and sends the intent accordingly. In case of multiple intents received, the GSO enables parallel distribution to multiple lo-SDX's.
4) Finally, the elected lo-SDX installs the received intent and updates the GSO.

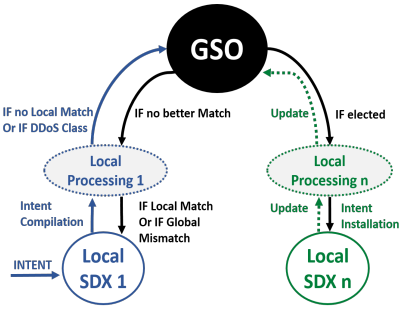



Fig. 4: General Workflow between lo-SDX and GSO

To ensure proper workflow, multiple components coordinate to enable the analysis at both the local and global stage. The components collect and process data related to lo-SDX structure, Autonomous Systems and advertised IP prefixes, along with other necessary BGP attributes; and related updates. We proceed by describing the system structure at both tiers.

### A. Local SDX

The local SDX structure is defined by an SDN fabric, SDN controller, Route Server (RS) and BGP-speaking border gateways representing each Autonomous System. The lo-SDX has several SDN applications running on top of the SDN controller as shown in Figure 5. These applications have different modules that interact according to the workflow presented in Figure 6. Their behaviour depends on the type of input, ranging from intent "request_type" (add, delete, modify, query), RS updates (resulting from BGP advertisements, withdrawals, updates) or SDX structural updates (AS, infrastructure changes). We briefly describe the applications and their internal modules:

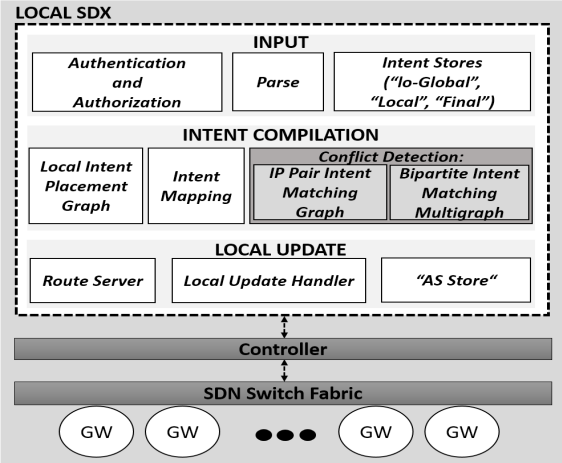



Fig. 5: Components of the lo-SDX architecture

*1) Input:*

- **Authentication and Authorization Module:** First and foremost each AS follows an authentication and authorization process at its lo-SDX. Depending on the level of authority, members can have different roles and privileges at the SDX, accompanied with different access and views of resources.
- **Parse Module:** The Parse module contains three submodules that serve to process the intent, validate the intent flow space and determine the intent class.

  **1.2.1 NLTK pre-processing:** First, we use NLTK to process the intent through tokenizing and tagging. During this process we determine if the intent is actually comprised of a set of different intents. For each we create an intent object per the structure described in Section V. For an intent to be deemed eligible it must abide by the defined format. If mandatory attributes are unspecified the intent may proceed with some default values where applicable, or further input will be required from the AS member. Based on the NLTK tag assigned, we further parse and assign the adequate input values from the provided intent to its JSON intent object, if and only if the flow space verification yields approval.

  **1.2.2 Flow Space Validation:**
  For an intent to be authorized we follow a two-step verification process. First, the AS member requesting the intent must have authorization over the intent attributes. Second, either the source or destination attributes must be a subset of the AS-advertised prefixes. If we denote sets $A$ and $B$ as intent source and destination, $E$ as the set of AS-advertised prefixes; then the intent flow space is valid if at least one of the conditions is met: $A \subseteq E$ or $B \subseteq E$. This ensures an AS can only influence traffic destined to or originating from its own networks.

  **1.2.3 Detect Intent Class:** For each intent object we determine the intent class according to attributes from the intent object. The intent class determines the mapping function.
- **Local, lo-Global and Final Intent Stores:**
  The intent objects are stored in databases. A "Local Intent Store" stores intents received locally from a participating AS at a lo-SDX. In contrast, intents sent from the GSO to a lo-SDX are stored in a "lo-Global Intent Store". This separation is necessary for other modules and intent management. We note that the intent objects that arrive from the GSO have already been processed by the Parse module in their original lo-SDX, therefore they can be directly stored in the "lo-Global Intent Store". A "Final Intent Store" database is kept to maintain records for each intent, whether it is installed locally or globally.

*2) Intent Compilation:*

- **Local Intent Placement Graph Module:**
  This module determines if the intent should be installed locally or be submitted to the GSO. It builds a Patricia IP Trie from all the advertised prefixes at the lo-SDX. Each node represents a pair of an advertised prefix and its BGP "AS Path" attribute. The trie is updated if BGP or other relevant updates occur, through the RS which maintains up to date best prefix routes for each AS member.
  Input for a graph search is the intent source IP address; if a matching prefix is found then the intent can be installed locally since the source traverses the SDX. If a matching prefix is not found, the intent is sent to the GSO.
- **Intent Mapping Module:** Described in subsection V-B. Only intents to be compiled locally proceed further.

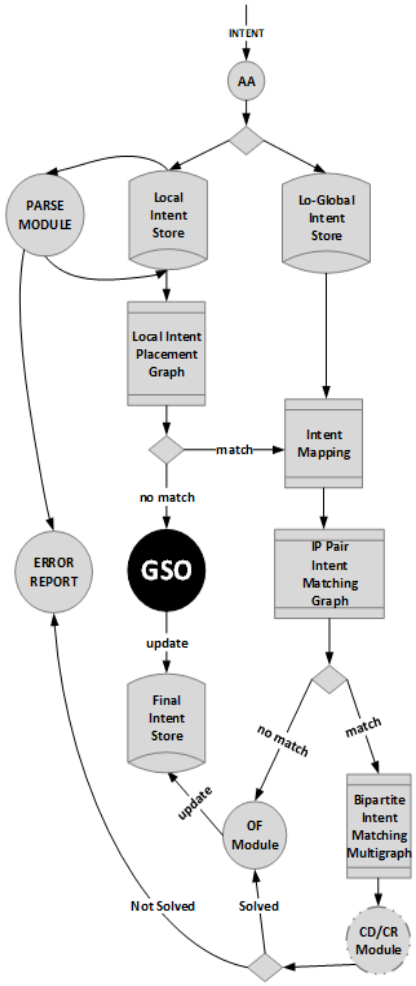



Fig. 6: lo-SDX Workflow

- **Conflict Detection Module:** Each intent prior installation is verified against the existing intents by completing a simple conflict check using the two graphs described next. This approach is a naïve way of performing a conflict check. Further processing is required to enable proper conflict detection and classification; as well as conflict resolution.

**2.3.1 IP Pair Intent Matching Graph Module:**
This module serves as an initial pre-conflict analysis. It maintains a Patricia IP Trie where each node consists of a source IP prefix and a list of destination IP prefixes. Additionally, there is a dedicated "wildcard" node (for cases where source is specified as "any"). For each installed intent at the lo-SDX, the source and destination vertices are added to the trie within a single node. The trie is updated if intents are deleted or updated.
For each intent we perform a Trie search with the intent source IP address (prefix subset, superset and exact match are considered); the "wildcard" node is also checked. If any matches are found, then we compare the list of destination IP prefixes with the new intent destination IP prefix. If there is no overlap between the destinations then the intent is submitted to an OpenFlow Generator module which invokes controller APIs to install rules in the SDN fabric. If there is an overlap, then the intent proceeds with the graph below. The complete process is outlined in Algorithm 1.

**2.3.2 Bipartite Intent Matching Multigraph Module:**
If a match was found in the previous graph, then we search for the specific pair in a "Bipartite Intent Matching Graph". This graph has two sets, sources belong to set $U$, and destinations belong to set $V$. An edge between a vertex in $U$ and a vertex in $V$ denotes an installed intent for that pair, defined by the function over the edge. We analyze the match using the vertex indegree, outdegree and the function over the edge to determine whether the new intent conflicts with the existing intent match.

**Algorithm 1:** Local Intent Graph Compilation

```
CompileLocalIntent
    inputs  : G = (V, E); #Intent digraph;
              G_lo_ip = (IP, [ASPath]) #Intent Placement Graph;
              G_ip_pair = (SrcIP, [DestIP]) #IP Pair Graph;
              G_B = (U, V, E) #Bipartite Multigraph
    outputss: OF #List of flow entries
              Global #Intent for GSO submission
              Outcome #Awaiting AS admin decision
    OF, Global, Outcome ← ∅;
    Intent, local, Bipart, Match, dest ← ∅;
    assert authorization
    foreach intent src_node s ∈ V do
        if s ∈ G_lo_ip then
            local ← getIntentAttributes(s, d, [edge_f])
            if s ∈ G_lp_pair then
                dest ← G_lp_pair([DestIP])
                foreach i ∈ netaddr.IPSet(d) do
                    if i ∈ IPNetwork(dest) then
                        Match["intent_id"] =
                          getIntentOrigin(intent_id)
                        Match["conflict_ip"] = i
                         Bipart.append(Match)
            else
                OF.append(local)
        else
            Intent ← getIntentAttributes(s, d, [edge_f])
            Global.append(Intent)
    Outcome ← Bipartite(Bipart) #Bipartite checks indegree,
      outdegree and edge function for each Match in Bipart
    return OF, Global, Outcome;
```

*3) Local Update:*

- **Route Server Module:**
This module follows the typical well-established RS model. It maintains a BGP session with each border gateway, and stores a copy for each participant. It selects and advertises best routes to all the participants, unless export or import policies are in place. The RS enables default forwarding behaviour allowing backward compatibility to a regular IXP.
- **Local Update Handler Module:**
The Handler updates the "AS Store" database whenever BGP advertisements, withdrawals or updates occur. The "AS Store" database has a table per AS, containing "SDX_id", "AS_id", advertised IP prefixes, "AS Path" and border gateway related information (IP address, Ethernet address, physical and virtual ports).
Changes in the "AS Store" database trigger updates to the "Local Intent Placement Graph" and the "Global Update Handler" (described in subsection VI-B). Additionally, changes in the "AS Store" database may require intent re-validation. For instance, if a prefix is withdrawn, we search the "IP Pair Intent Matching Graph" to determine whether there was an active intent for the withdrawn prefix. Lo-SDX RS updates generally affect local intents only, therefore we re-validate affected intents locally and re-establish them as deemed necessary. Last, if the lo-SDX is no longer applicable for the intent in question, the GSO is invoked.

### *B. Global SDX Orchestrator*

The GSO enables intent orchestration to one or multiple selected lo-SDX's. Each lo-SDX then finalizes the intent compilation and installation processes. Figure 7 shows the main GSO components alongside the general GSO workflow for intent placement discovery. The components have multiple roles; for instance, the Input component serves to receive and store intents, which upon authorization are processed by the Placement Analysis component. An algorithm determines placement and distributes the intents to the selected lo-SDX's accordingly. All modules are continuously updated through the Global Update component to reflect any topology-related or BGP-related updates.

*1) Input:*

- **Authentication and Authorization Module:** Each lo-SDX authenticates and communicates via a secure out-of-band channel with the GSO. Local SDX's send processed JSON intent objects to the GSO for placement analysis.
- **Global Intent Store:**
A "Global Intent Store" database contains separate tables per SDX to store received and forwarded intents by the GSO

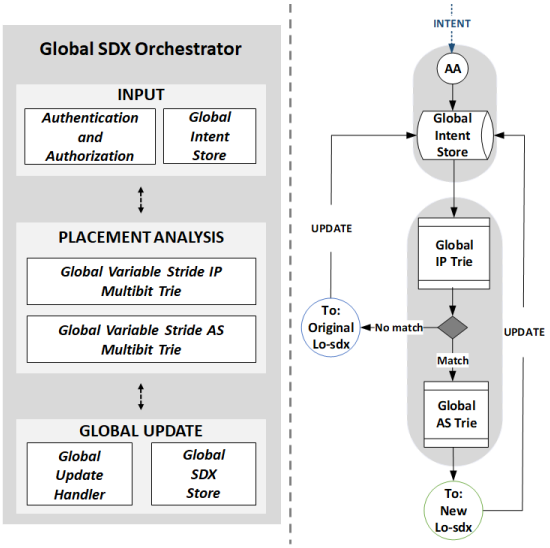



Fig. 7: GSO Main Components and General Workflow

to the lo-SDX's for compilation. Each table stores entries related to the intent attributes. In case of an intent update, query or delete request from the originating lo-SDX, the GSO can determine which lo-SDX compiled the intent.

*2) Placement Analysis:* This component contains a forest graph which consists of two separate tree graphs as shown in Figure 7. To determine which lo-SDX will install the intent, first we search the "Global Variable Stride IP Multibit Trie". If an IP prefix match is found for the supplied intent source, we examine the BGP "AS Path" attribute for the prefix match found. Each "AS_id" from the "AS Path" is then used to search the second graph, the "Global Variable Stride AS Multibit Trie". Once an exact match is found in this second trie, we have determined the SDX that will execute the intent.

- **Global Variable Stride IP Multibit Trie Module:**
  The GSO maintains a "Global variable stride IP multibit" trie which is populated with advertised prefixes and their best routes ("AS Path") from participating lo-SDX's. Each node at the trie contains an IP prefix and a list of "AS_ids" to denote the BGP "AS Path" attribute.
  A trie search is perfomed using an intent source IP address; if a prefix match is not found, the intent is sent back to the requesting lo-SDX. If a best matching prefix or multiple matching prefixes are found in the trie, Algorithm 2 comparatively examines each match (using IP prefix, "AS path" and SDX data). For each matching prefix, if there are multiple "AS Path" attributes, each "AS Path" is examined. The algorithm starts with the right-most "AS_id" in the "AS Path" attribute (since it represents the originating AS for the prefix in question). For each "AS_id" that is not found in the "AS Trie" graph described below, we proceed and examine the next in line "AS_id" to the left in the "AS Path" attribute. The outcome of the algorithm determines a final lo-SDX placement for intent compilation. For the DDoS intent class the outcome enables load distribution among multiple lo-SDX's in parallel.
- **Global Variable Stride AS Multibit Trie Module:**
  To determine if an AS participates at any lo-SDX we use an "AS Trie", where each node is a pair of (AS, SDX). Since AS numbers range from 16 to 32 bits, we consider a variable stride multibit trie to search for an exact AS match. The trie is populated with all the participating AS's at each participating lo-SDX.

*3) Global Update:*

- **Global Update Handler Module:**
  Each lo-SDX communicates RS-related updates to this module. The updates are processed and stored in a "Global SDX Store" database with separate tables per SDX, each containing data related to "SDX_id", "AS_id", "IP prefixes", "AS Path" etc. Both Global tries are updated by change triggers in the "Global SDX Store" database.

**Algorithm 2:** Global Analysis Intent Placement

```
AnalysisPlacementGraph
    inputs : Intent #Intents received;
             G_ip = (IP, [ASPath]) #Global IP Graph;
             G_as = (AS, SDX) #Global AS Graph;
    outputs: sdx_exec; #Dict of elected lo-SDX's
             nomatch #Dict of mismatch intents
    match, nomatch, sdx_exec ← ∅
    assert authorization
    foreach intent src_node n ∈ Intent do
        if n ∈ G_ip then
            #check if IP prefix match
            f ← G_ip([n])
            #f gets pair of "SrcIP, ASPath"
            match.append(f)
            foreach match["ASPath"] do
                foreach AS_id ∈ match["ASPath"][:: −1] do
                    if AS_id ∈ G_as then
                        SDX ← G_as([AS_id])
                        sdx_exec.append([ "AS": AS_id, "SDX":
                          SDX, "IP": n ])
        else
            nomatch.append(n)
    return sdx_exec, nomatch;
```

## VII. Evaluation and Results

To evaluate the proposal we conduct an experiment in the SAVI testbed as shown in Figure 9a. The topology is created using VXLAN connections between nodes, where each node is a separate VM in the SAVI testbed [27], a total of approximately 80 VMs. White nodes represent the SDX's, green nodes (firewall and victim) represent an internal network. SDN controllers and GSO are represented by blue and grey nodes respectively. Each link between a red node (DoS attacker) and an SDX represents aggregated attack traffic only per AS. In this scenario SDXA and SDXB use SDX2 as an upstream connection, SDXC, SDXD and SDXE use SDX1 as their upstream, similarly SDX3 serves to SDXF. We note that none of these links in reality are point-to-point, instead topologies are abstracted for simplicity.

Each lo-SDX is implemented using Quagga [28], Ryu (SDN Controller), OVS (Open vSwitch) and the components

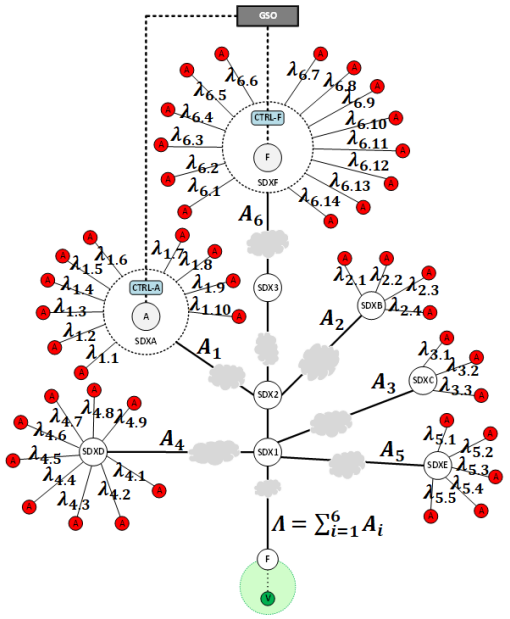
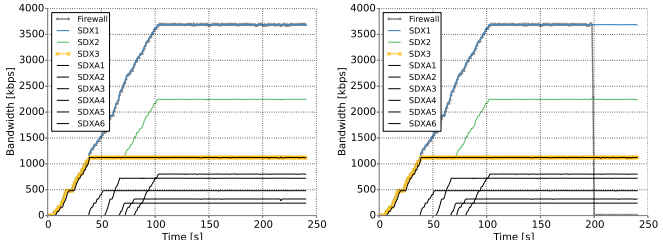
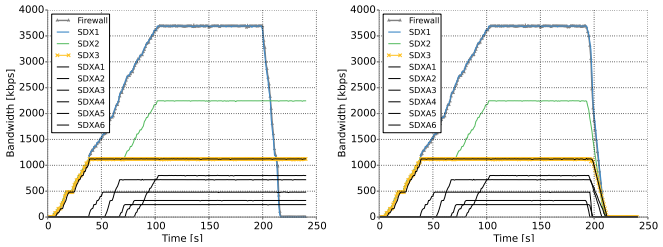



Fig. 8: Experiment topology (for brevity SDXA and SDXF are shown expanded; the other SDX's have same structure and own GSO connection)




(a) (b)

Fig. 9: DDoS Attack vs Firewall defense-based approach




(a) (b)

Fig. 10: Single IXP block vs Multiple SDX collaboration block approach

described earlier. The tries used by the GSO and lo-SDX are populated with data from the SDX topology. Additionally, we complement the tries with parsed raw data collected by the RRCs (RIS (Routing Information Service) Route Collectors) available online from RIPE [29].

During the DDoS attack each attacker node generates attack traffic ($\lambda$). Each aggregated upstream attack traffic $Ai$, ($i = 1..6$) is equal to the sum of $\sum_{j=1}^{m} \lambda_{i.j}$, where $m$ is the number of AS's per SDX that carry attack traffic. The sum $\sum_{i=1}^{6} A_i$ represents the total attack traffic. To simulate the attack we generate DDoS UDP floods from each attacker node.

We define three separate trials of the experiment, namely traditional firewall defense-based model, single lo-SDX blocking approach (closest relative to the intent user), and blocking at collaborating SDX's closest to attack sources (our approach). During each trial we measure the network transmit throughput in time at the Firewall and each SDX node. The throughput consists of the attack traffic only, and an occasional ARP request of negligible size that does not skew the results.

The results of the experiment are given in Figures 9a, 9b, 10a and 10b. In Figure 9a we show the DDoS attack without any defense method involved. As more attacker nodes start UDP floods, the attack functions grow; around t=100 all attacking nodes are participating, and the maximum attack volume is reached. The Firewall and SDX1 have same traffic level since they both receive all of the traffic from the source SDX's as denoted in the top line of Figure 9a.

In Figure 9b around $t = 200$s, rules to block the offending sources are added to the Firewall, after which the attack traffic is no longer transmitted from the Firewall to the victim node (V). This is evident by the drop in the top line in Figure 9b.

In Figure 10a, a monitoring system initiates the action by sending a list of the offending sources to SDX1 (the SDX where the victim network participates). Upon intent compilation, the blocking rules are installed at the switch, corresponding to the drop of the top line for the Firewall (since it no longer transmits attack traffic) and consequently for SDX1 as well. The other nodes still transmit attack traffic.

Finally, Figure 10b shows the collaborative approach stopping the attack traffic at each first SDX point of entry. The monitoring system sends a DDoS intent to SDX1, which then sends the intent to the GSO (becomes active at around $t = 175$s in Figure 10b). The GSO invokes the algorithm to determine the location placement for each blocking intent. The GSO then distributes the intents to the elected lo-SDX's. Each lo-SDX invokes the algorithm to compile and install the received intents, while ensuring no conflicts exist. In summary, the intents are distributed and compiled in parallel at the elected lo-SDX's. This corresponds to the fast decaying transmitting throughput functions for all the nodes in Figure 10b. As these functions get turned off, the flow arriving at the Firewall drops adequately, ultimately stopping completely.

## VIII. Conclusions and Future Work

We have presented a proposal for a future security architecture that can commence as more IXPs embrace SDN. The SDX collaboration enables an effective approach to DDoS mitigation while addressing scalability challenges. In addition multiple future opportunities related to traffic engineering, end-to-end QoS and security services can be enabled.

The implementation of the proposed architecture on the SAVI testbed [27] is part of on-going work. Additionally, the Conflict Detection and Resolution modules will be further expanded to improve and reach a full automation cycle. For the purpose of achieving the diverse set of opportunities enabled by the collaborative approach potential areas of ML application will also be reviewed.

## REFERENCES


[1] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," in *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4. ACM, 2010, pp. 75–86.

[2] K. G. Coffman and A. M. Odlyzko, "Internet growth: Is there a moores law for data traffic?" in *Handbook of massive data sets*. Springer, 2002, pp. 47–93.

[3] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: defining tomorrow's internet," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 347–356.

[4] S. Goldberg, "Why is it taking so long to secure internet routing?" *Commun. ACM*, vol. 57, no. 10, pp. 56–63, Sep. 2014. [Online]. Available: http://doi.acm.org/10.1145/2659899

[5] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin, *Firewalls and Internet security: repelling the wily hacker*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[6] B. Rowe, D. Reeves, and M. Gallaher, *The role of internet service providers in cyber security*. Institute for Homeland Security Solutions, 2009.

[7] T. Killalea, "Recommended Internet Service Provider Security Services and Procedures," RFC 3013, Nov. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc3013.txt

[8] G. M. Jones, "Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure," RFC 3871, Sep. 2004. [Online]. Available: https://rfc-editor.org/rfc/rfc3871.txt

[9] R. Anderson and T. Moore, "The economics of information security," *Science*, vol. 314, no. 5799, pp. 610–613, 2006.

[10] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 312–321.

[11] N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "There is more to ixps than meets the eye," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 19–28, Nov. 2013. [Online]. Available: http://doi.acm.org/10.1145/2541468.2541473

[12] C. Dietzel, A. Feldmann, and T. King, "Blackholing at ixps: On the effectiveness of ddos mitigation in the wild," in *International Conference on Passive and Active Network Measurement*. Springer, 2016, pp. 319–332.

[13] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a large european ixp," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 163–174, 2012.

[14] N. Chatzis, G. Smaragdakis, J. Böttger, T. Krenc, and A. Feldmann, "On the benefits of using a large ixp as an internet vantage point," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 333–346.

[15] M. Yannuzzi, X. Masip-Bruin, and O. Bonaventure, "Open issues in interdomain routing: a survey," *IEEE network*, vol. 19, no. 6, pp. 49–56, 2005.

[16] N. Feamster, J. Rexford, S. Shenker, R. Clark, R. Hutchins, D. Levin, and J. Bailey, "Sdx: A software defined internet exchange," *Open Networking Summit*, p. 1, 2013.

[17] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "Sdx: A software defined internet exchange," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 551–562, 2015.

[18] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "An industrial-scale software defined internet exchange point." in *NSDI*, vol. 16, 2016, pp. 1–14.

[19] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 99–110.

[20] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480, Feb. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6480.txt

[21] C. Janz, N. Davis, D. Hood, M. Lemay, D. Lenrow, L. Fengkai, F. Schneider, J. Strassner, and A. Veitch, "Intent nbi - definition and principles." Open Networking Foundation, 2015.

[22] L. Garber, "Denial-of-service attacks rip the internet," *Computer*, vol. 33, no. 4, pp. 12–17, Apr. 2000. [Online]. Available: http://dx.doi.org/10.1109/MC.2000.839316

[23] S. Mansfield-Devine, "Ddos goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare," *Network Security*, vol. 2016, no. 11, pp. 7–13, 2016.

[24] *Connected networks. AMS-IX.* [Online]. Available: https://ams-ix.net/connected/

[25] D. R. C. Moore, J. Strassner, E. J. Ellesson, and A. Westerinen, "Policy Core Information Model – Version 1 Specification," RFC 3060, Feb. 2001. [Online]. Available: https://rfc-editor.org/rfc/rfc3060.txt

[26] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed. O'Reilly Media, Inc., 2009.

[27] J.-M. Kang, H. Bannazadeh, and A. Leon-Garcia, "Savi testbed: Control and management of converged virtual ict resources," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 664–667.

[28] *Quagga*. [Online]. Available: https://www.nongnu.org/quagga/

[29] *RIPE Routing Information Service (RIS)*. [Online]. Available: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data