ENDEAVOUR: A Scalable SDN Architecture for Real-World IXPs

Gianni Antichi*, Ignacio Castro†, Marco Chiesa‡, Eder L. Fernandes†, Remy Lapeyrade§, Daniel Kopp¶, Jong Hun Han*, Marc Bruyere††, Christoph Dietzel¶‖, Mitchell Gusat**, Andrew W. Moore*, Philippe Owezarski§, Steve Uhlig†, Marco Canini‡‡
*University of Cambridge †Queen Mary University of London ‡Université catholique de Louvain §LAAS-CNRS ¶DE-CIX ‖TU Berlin **IBM Research ††University of Tokyo ‡‡KAUST

***Abstract*—Innovation in interdomain routing has remained stagnant for over a decade. Recently, IXPs have emerged as economically-advantageous interconnection points for reducing path latencies and exchanging ever increasing traffic volumes among, possibly, hundreds of networks. Given their far-reaching implications on interdomain routing, IXPs are the ideal place to foster network innovation and extend the benefits of SDN to the interdomain level.**

In this paper, we present, evaluate, and demonstrate ENDEAVOUR, an SDN platform for IXPs. ENDEAVOUR can be deployed on a multi-hop IXP fabric, supports a large number of use cases, and is highly-scalable while avoiding broadcast storms. Our evaluation with real data from one of the largest IXPs, demonstrates the benefits and scalability of our solution: ENDEAVOUR requires around 70% fewer rules than alternative SDN solutions thanks to our rule partitioning mechanism. In addition, by providing an open source solution, we invite everyone from the community to experiment (and improve) our implementation as well as adapt it to new use cases.

***Index Terms*—Software Defined Networking, Internet eXchange Points, Inter-domain routing, Peering.**

I. Introduction

Internet eXchange Points (IXPs) have become ubiquitous elements fueling a surge of peering interconnections and ultimately leading to a denser and flatter Internet [1]. Nowadays, the largest IXPs interconnect hundreds of Autonomous Systems (ASes) and carry huge traffic volumes (even several Tbps) [2]. Around 80% of the announced address space is reachable through the existing IXPs [3] and their presence extends to the most remote regions [4], [5]. While IXPs have facilitated greater interconnectivity and affordability, many fundamental drawbacks in the Internet are rooted in the protocol enabling interdomain connectivity, i.e., the Border Gateway Protocol (BGP). BGP only provides a very limited indirect control for ASes to set their reachability policies: it is exclusively based on IP destination and its influence is restricted to neighboring networks. To overcome these limitations, Software Defined Networking (SDN) has been recently proposed: SDN supports logically centralized network control at a fine level of granularity, well beyond the IP destination-based approach of BGP. Unfortunately, only the intradomain level has benefited from SDN deployment so far [6]. Because of their centrality and relevance, IXPs are the ideal place from where to extend the benefits of SDN to the Internet at large [7]. However, existing SDN solutions for IXPs are either non-deployable at real fabrics, because they cannot handle multi-hop topologies [8], [9], or provide only very limited SDN capabilities [10], [11].

This paper presents and evaluates ENDEAVOUR, a platform that enables an SDN-based IXP architecture fully exploiting SDN capabilities and deployable in virtually, any real-world IXP topology. Building upon the ideas of Gupta et al. [8], the scalability techniques of Gupta et al. [9], and the fabric management approach of Bruyere [10], ENDEAVOUR is highly scalable, deployable in multi-hop IXP topologies, and reduces broadcast traffic within the fabric. Enabling full SDN capabilities in a multi-hop IXP entails significant challenges: we identify such challenges and their solutions. First, we study how to distribute the forwarding rules across the IXP switches to curtail the consumption of their scarce resources. ENDEAVOUR frees resources from the core switches by circumscribing their tasks to intra-fabric forwarding: only the edge switches have routing policies installed and use them to select the path towards the fabric egress. We also identify how to reduce the amount of routing policies installed in the edge switches. By exclusively installing the outbound policies in the edge switch where the policy owner is connected to, we limit forwarding state duplication, further helping to save switch resources. Secondly, we identify multiple mechanisms for failover recovery and fast BGP route withdrawal. While single-switch IXPs can simply reroute the affected traffic according to the next best or new policy, a multi-hop IXP is challenging. In ENDEAVOUR, we identify three mechanisms to tackle this issue: duplication of outbound policies, bouncing packets back to the ingress switch, and injection of recovery information in the packets. Finally, using real data one of the largest European IXP, and a realistic multi-hop topology, we evaluate ENDEAVOUR and show its scalability and advanced use cases [7] such as internal IXP load-balancing, i.e., across the core-switches, and blackholing. Our evaluation shows that ENDEAVOUR requires around 70% rules less than alternative SDN solutions [9], [10], its internal load balancing approach achieves similar results to the more complex but typically used Equal-Cost Multi-Path Routing (ECMP) [12], and provides a fast fail over recovery mechanism.

In short, the main contributions of this paper are as follows:

- We present the ENDEAVOUR platform, an SDN enabled IXP architecture deployable in real-world (multi-hop) IXP topologies.

- We evaluate several use cases of ENDEAVOUR using real data from one of the largest IXP.
- We make publicly available the implementation and documentation of the platform to the community [13].

The remainder of the paper is organized as follows. Section II introduces the relevant background while Section III presents the ENDEAVOUR architecture. The challenges faced and how ENDEAVOUR tackles them are discussed in Section IV and Section V evaluates the ENDEAVOUR architecture. Finally, Section VI overviews the related work and Section VII concludes this paper.

II. Background

A. IXPs and Route Servers

IXPs are typically implemented as a simple layer-2 broadcast domain where member networks connect through BGP-speaking routers to exchange traffic. While small IXPs' fabrics might be organized with just one switch, larger IXPs frequently have a multi-hop topology composed of edge and core switches [2]. The edge switches connect to the IXP member routers whereas the core switches only interconnect edge switches, ensuring redundancy and scalability. IXPs operates at the Ethernet level, and are rarely involved in routing decisions. Originally, IXP members had to establish BGP peerings using TCP connections with every IXP member. As IXPs grew in size [14], this solution became impractical because it involved keeping too many BGP sessions, with the associated administrative and operational overheads. IXPs introduced Route Servers (RS) [15] to address this problem: IXP members can receive all the routing information available to the IXP through a single BGP session with the RS, which acts as sophisticated route reflector.

B. iSDX: A Scalable Software-Defined Exchange

The iSDX [9] is a scalable SDN solution for IXPs with a single switch topology, that builds upon the previous SDX [8] project. SDX proposes a programmable SDN fabric that allows members to override default BGP routing behavior with more fine-grained SDN policies. The SDX approach relies on a RS through which IXP members exchange reachability information via BGP, and the SDX controller, that provides each member fine-grained control over the incoming and outgoing traffic flows. The design of the SDX controller ensures that the forwarded traffic complies with the SDN policies and the advertised prefixes. Each member runs an SDN control application either on the central controller or on a local machine while its border router exchanges BGP information with the IXP's RS. The SDN controller collects the SDN policies from all members, reconciles them with the current BGP routing state, and computes the forwarding rules that are installed at the IXP fabric. While a naïve implementation of the above would result in the explosion of the forwarding state [8], iSDX devises innovative techniques to reduce the SDN rule compilation time, number of forwarding table entries, and forwarding table update rates. The first aspect is tackled by distributing the control plane computation load across the members. The second and third aspects are addressed by decoupling the BGP state from the forwarding state of the IXP fabric. This solution greatly compresses the forwarding state and the update rate.

ENDEAVOUR builds upon the iSDX solution to decouple the BGP and the SDN control plane, but differently from the iSDX solution it is deployable in any real IXP topology, whether single or multi-hop.

C. Umbrella: A Scalable Multi-Hop Fabric

To exchange traffic, IXP members typically learn each other physical address, i.e., MAC address, using the Address Resolution Protocol (ARP). However, ARP traffic volumes in large IXP fabrics can be high enough to overwhelm members' routers with low capabilities [16]. The amount of ARP traffic is even higher during network outages [17] which occur quite frequent [18]. When many routers attempt to resolve the IP addresses of peers that are not available anymore, generating the so-called "ARP storms". The network congestion resulting from such events can impact the BGP connectivity, thus causing severe disturbances [16], [19]. Umbrella [10] eliminates the need of location discovery mechanisms based on packet broadcasting, i.e., ARP request or IPv6 neighbor discovery, by taking advantage of the static nature of the exchange. Umbrella features on-the-fly translation of broadcast packets into unicast ones by exploiting the OpenFlow (OF) ability to rewrite the destination MAC addresses of frames. In particular, for any packet entering the fabric, the ingress OF switch encodes the path towards the egress in the destination MAC field, i.e., transforming into unicast the ARP broadcast traffic. Core switches then use the encoded path to forward the packets accordingly. Finally, the egress edge switch replaces the encoded MAC address with the original one.

ENDEAVOUR leverages the Umbrella design to obtain an efficient transport layer over a multi-hop topology, by removing all the side effects related to broadcast traffic, and by enabling effective access control over the fabric. Additionally, ENDEAVOUR enables on-the-fly recomputation of the internal IXP paths, when member's policies change or when a major network disruption happens. Finally, Umbrella does not deal with link failures, a problem that we discuss extensively in Sect. IV.

III. The ENDEAVOUR architecture

Figure 1 presents the architecture of ENDEAVOUR. At the physical level, a set of OF-enabled switches performs the forwarding functions within the IXP fabric. The switches are interconnected according to the IXP operator's most common network topology, e.g., full-mesh or spine-leaf.

In line with the SDN principle, ENDEAVOUR relies on a logically centralized `Fabric Manager` component that acts as an interface to the low-level physical network. The fabric manager exposes to the members and IXP operators a logically coherent view of the network and a set of high-level primitives, greatly simplifying the data plane management tasks. Through the fabric manager, the IXP operators can easily deploy customized applications such as Distributed Denial of Service (DDoS) mitigation or internal load balancing, while

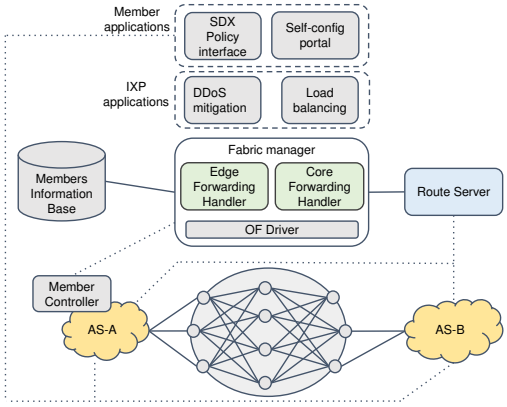

Fig. 1. The ENDEAVOUR architecture.

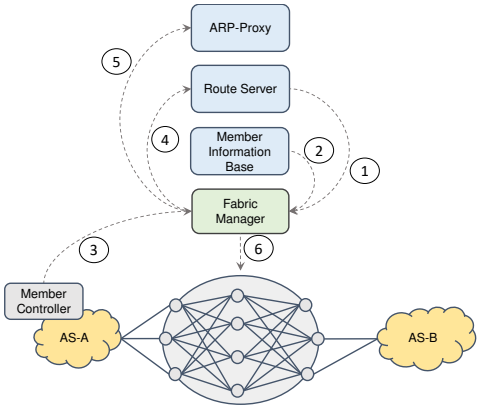

Fig. 2. The ENDEAVOUR route setup phase.

IXP members can define fine-grained routing policies such as inbound/outbound traffic engineering or Access-Control-Lists (ACLs) configuration using the `SDX policy interface`. Note that the routing policies defined using the SDX policy interface are visible to the IXP: members that regard their routing policies as strictly private information, should not reveal them through the SDX policy interface. Different techniques such can be used to enhance privacy [20], [21].

To install the appropriate forwarding states into the switches, the fabric manager gathers information from several ENDEAVOUR components. First, to ensure forwarding consistency with the BGP state, the fabric manager collects all the BGP routes advertised by the members from the RS component[1]. Second, it collects the policies from the `members' controllers`. Finally, the fabric manager also reads the current members' configurations (e.g., port bandwidth, physical MAC addresses) from the `Member Information Base` component to compute the intended forwarding state.

To forward packets throughout the IXP network, the fabric manager makes use of two crucial elements: (i) the EDGE FORWARDING HANDLER (EFH), based on iSDX and Umbrella, and (ii) the CORE FORWARDING HANDLER (CFH), based on Umbrella. The EFH is responsible for handling the packets entering the fabric: for each packet, the EFH selects an egress IXP port and an internal path. The egress IXP port selection is based on the iSDX module while the internal path selection process leverages the source-based routing encoding scheme of Umbrella which is dictated by the internal load balancing configuration of the IXP operator. The CFH, instead, is solely responsible of forwarding packets across the IXP network, leading to very light requirements for the core switches. Applying a source-routing approach within the fabric is possible thanks to the inherently static nature of the IXP environment. Indeed, connecting a new IXP member device is coordinated between the member and the IXP: first, the member communicates the MAC address of its device, then the IXP assigns a physical port in the IXP network to such device. This information is stored in the `Member Information Base`. This setup coordination between the IXP and its members breaks the premises of traditional neighbor discovery mechanisms, such as ARP or ND, where new devices can be connected to a network without any coordination with the network administrator.

[1]The RS also guarantees backward compatibility with traditional IXP designs, where members can enforce their policies through standard BGP mechanisms, e.g., BGP communities, AS-Path prepending.

A. *ENDEAVOUR IXP fabric in action*

This section describes the ENDEAVOUR control and data plane operations. In particular, the paragraph *Route Setup phase* describes the steps performed by the fabric manager to produce the forwarding state required to enable connectivity among members, while *Data exchange phase* shows the sequence of operations performed by the forwarding plane to route a packet across the fabric.

Route Setup phase: Figure 2 depicts the route setup phase in ENDEAVOUR. Initially, the fabric manager learns BGP reachability information from the RS (step 1) as well as the configuration, e.g., mapping of MACs, IPs, ports, from the member information base (step 2), and the SDN policies of the members' controllers (step 3). To reduce the forwarding state, the fabric manager then combines the BGP reachability information and SDN policies to determine which destination prefixes have the same forwarding behavior within the IXP network. Such prefixes are aggregated into Forwarding Equivalent Classes (FECs), each of which is assigned to a virtual IP (vIP) and a virtual MAC (vMAC) address. The fabric manager then sends the vIP to the RS (step 4) and the associated vMACs to the ARP-Proxy (step 5). The RS uses the vIP as the next-hop of those routes announced to its members that are destined towards a prefix contained in the FEC associated to such vIP. The vMAC contains the encoded set of members that announced a route towards a specific prefix. The ARP-Proxy sends the appropriate vMAC as a gratuitous ARP-reply for any IP address in the FEC associated to vIP, thus binding a packet to its designated vMAC. Whenever a member announces or withdraws a route, the EFH modifies the vMAC accordingly, leaving the forwarding state unaffected. Finally, the fabric manager installs the forwarding rules for support source-routing within the fabric in both the edge and core switches (step 6).

Data exchange phase: In Algorithm 1, we present the ENDEAVOUR data plane pipeline, i.e., the sequence of operations that are performed by a switch `S` when it receives a packet `pkt` from one of its incoming ports `iport`. Initially, `S` verifies that `iport` is an IXP ingress port, i.e., whether it connects to an IXP member's device (line 1). If so, the fabric egress port `eport` for the packet is selected according to (i) the sender's outbound policies and the receiver's inbound ones as stored in the `EFH` tables, (ii) the vMAC encoded in the destination MAC `dmac` of the packet, and `iport` (line 2). Based on the chosen egress port, the sequence of outgoing ports that has to be traversed by `pkt` to reach `eport` is encoded in `dmac` as a stack of 8-bit labels (line 3). Since there are only 48 bits in the destination MAC, this forwarding approach limits the maximum number of hops within the IXP to be less than 6. We note this requirement is typically met by design in current fabrics. In addition, the flexibility of OF allows us to use less bits in the encoding scheme to increase the number of supported hops, as suggested in [22]. Regardless of whether `iport` is an IXP ingress port, `S` processes the packet according to the value on the top of the stack (line 4), which is popped before forwarding the packet (line 5)[2]. If this operation leaves the stack of labels empty (line 6), then the next hop of the packet is a member's device, which requires the switch to rewrite the destination MAC with the address associated to the member's physical port (line 7). Finally, the packet is forwarded through its designated outgoing port (line 8).

```
   input : A packet pkt received from port iport
   output: The updated packet and its outgoing port
1  if iport is an ingress port then
2  |  IXP_eport ← EFH.iSDX(pkt.dmac,iport)
3  |  pkt.dmac ← EFH.Umbrella-setup(IXP_eport)
4  oport ← CFH.Umbrella-peek(pkt.dmac)
5  pkt.dmac ← CFH.Umbrella-update(pkt.dmac)
6  if pkt.dmac is all zeros then
7  |  pkt.dmac ← member_mac(oport)
8  send pkt out from oport
```

Algorithm 1: ENDEAVOUR data plane pipeline at switch `S`.

IV. From Single to Multi-Hop IXP Fabrics

Realizing SDX in practice is challenging. From the members' perspective, an IXP acts as a high-performance *virtual switch* that provides sub-millisecond interconnectivity latency among a multitude of different organizations. In practice, however, IXPs rarely consist of a single switch. In fact, to cope with the increasing volumes of traffic exchanged among their members, a number of IXPs (e.g., LINX, AMS-IX, DE-CIX) have grown from a single switch topology to large infrastructures with distributed switches located in different data centers, where packets are forwarded through intermediate *core* switches before leaving the fabric. As such, realizing SDXes in practice entails addressing a series of non-trivial challenges that stem from aligning the need for high-performance packet forwarding operations with the underlying distributed environment. Specifically, IXP operators must carefully understand how the forwarding state scales with the number of configured SDN policies, how to easily load-balance the forwarded traffic across the high-speed IXP fabric, and how to quickly reroute traffic in response to network failures — three crucial operations that we discuss below.

[2]To enable such a forwarding scheme, every switch must have two action tables: forwarding and copy-field. OF 1.5 specifications allow copying and rewriting part of a header field.

A. Scaling the forwarding state

Today's approaches lack multi-hop support. Traditional IXP fabrics provide a layer-2 interconnection network to their members, thus limiting the amount of state within the network to a bare minimum, i.e., one entry for each member's layer-2 address. However, in SDN enabled IXPs, as members' (legacy) routers are unable to forward packets according to fine-grained policies (e.g., layer-4 information), the fabric requires storing information regarding each member's policies into the IXP switches. Unfortunalty, they only have limited forwarding state resources. While previous work [9] devised techniques for reducing the amount of forwarding state installed within a single-switch IXP fabric, the problem of distributing the rules across the fabric has remained uncharted.

Our approach limits forwarding state duplication. To improve the scalability of the IXP fabric, we distribute the routing policies across the switches as follows: member's routing policies are installed at the edge switches by the Edge Forwarding Handler, whereas the core switches are left with the simpler task of forwarding traffic towards its designated egress point — a task that is managed by the Core Forwarding Handler (as explained in the Algorithm 1). A straw man solution would be to simply replicate the members' policies across each of the IXP switches, but this would result in a waste of the already limited switches' forwarding table resources.

To efficient utilize resources, we draw two key observations. First, for each IXP member, it suffices installing the member's outbound policies at the ingress edge switches where the member connects to. Indeed, packets (potentially) matching a given member's policy will enter the IXP network only from those ingress switches. Second, inbound policies must be installed in every edge switch because the computation of a packet's egress point occurs when the packet first enters the IXP network. Finally, members' blackholing [23] policies are replicated at those IXP ingress switches where the targeted malicious traffic is expected to enter the fabric. However, given the current nature of DDoS (Distributed Denial of Service) attacks where multiple sources target a single destination, blackholing policies can be aggregated and installed at the IXP egress switches only, thus trading lower forwarding state space for higher IXP fabric bandwidth overhead. We show the benefits of such partitioning of the forwarding rules in Section V.

B. IXP internal load balancing

Today's load balancing mechanisms are hash-based. Even though a portion of the overall traffic might remains local at a certain edge switch, a large fraction of the overall traffic needs to traverse the IXP fabric. This large quantity forces network operators to carefully configure their internal routing paths in order to optimize the load per link within the IXP network. To this end, large IXPs deploy highly symmetrical topologies and use ECMP Routing [12] to equally spread the load among their internal links. In hash-based ECMP, the hash of the flow identity (i.e., IP addresses and transport port numbers) is used to deterministically select the outgoing port. Thus, ECMP guarantees that each packet of the same flow is forwarded along the same path.

Simpler SDN-based load-balancing. We rely on a simpler, yet effective, mechanism for load balancing traffic across the IXP fabric. Instead of employing the "black-box" optional capabilities of commercial legacy switches, such as hash-based forwarding, we capitalize on the features available in any OF-enabled switch. In particular, we compute the outgoing port according to the least significant bits in the IP source and destination addresses as also proposed in [24]. This mechanism provides a main advantage: while a black-box solution cannot be easily upgraded if not suitable anymore, this simple load balancing technique can be adapted on-demand. Our evaluation (Sect.V) demonstrates that this approach attains a load balancing performance comparable to hash-based approaches in multi-hop topologies. This is a crucial gain in the IXP context where replacements costs are high.

C. IXP fast failover recovery

Legacy IXPs have limited fast reroute capabilities. To attain the highest reliability within the IXP fabric, IXPs currently leverage the fast rerouting capabilities of the legacy routing protocols (e.g., OSPF, MPLS). Upon any internal link failure, such mechanisms quickly reroute packets towards their designated egress ports along a pre-computed alternate path. The extent to which these approaches improve network robustness is, however, inherently limited by the lack of information about alternative IXP egress ports through which traffic can be rerouted. As a result, whenever a link connecting an IXP egress switch to a member's BGP router fails, all packets forwarded through that port are dropped, a highly undesirable behavior from an operational perspective. For example, any member with two ports connected to the IXP fabric might desire to receive traffic through any of these two ports whenever a port is down or the sending member may want to reroute its traffic towards a different member in case of failure.

Current single-switch SDN-based fast reroute does not extend to multi-hop topologies. In a SDX fabric with a single switch topology, rerouting along a link failure is a trivial task: as soon as a link between the IXP switch and a member's router fails, the former one quickly disables the forwarding rules affected by the link failure, thus rerouting the traffic according to the next best SDN policy. In a multi-hop fabric, performing fast reroute is more challenging as the failed link may not be located on the same switch that computes the IXP egress port of a packet. We discuss three different approaches to tackle this challenge, each achieving a different trade-off in terms of the size of the forwarding tables, bandwidth overhead and packet processing overhead. Finally, we show that such approaches can be re-used to quickly re-route traffic in response to BGP withdrawal messages.

Approach #1: duplicating outbound policies. The *first approach* is to replicate all the member's inbound/outbound policies across all the switches and to keep a copy of the vMAC within the packet header (e.g., in the source/destination MAC address fields). Whenever a link fails, the Edge Handler Controller at the affected egress switch recomputes using the iSDX tables a new egress port according to the member's policies and the updated vMAC of the packet, exactly as in the above single-switch setting. As discussed in Section IV-A, this approach comes at the price of extensive duplication of the forwarding state, i.e., each member must store both the inbound and outbound tables of all members.

Approach #2: bounce packets back to ingress switch. The *second approach* consists in bouncing back to the ingress switch any packet that has to be forwarded through a failed link. In this way, a new egress port can be recomputed by the Edge Handler controller using the iSDX tables. While this approach prevents unnecessary duplication of the forwarding state, a waste of bandwidth and increased packet latency will occur with respect to the first approach.

Approach #3: inject recovery information into packets. Finally, the *third approach* strikes an interesting trade-off between all of the above overheads at the cost of additional information stored into the packet header. Whenever a packet enters the IXP network, the ingress switch leverages the iSDX tables to compute a primary and a backup egress ports. This can be achieved in different ways. One possible way is to first process a packet through the iSDX tables to compute a primary egress port, which, in turn, removes the owner of that egress port from the vMAC of the packet. After performing this operation, the modified packet is processed again by a second set of iSDX tables, which, in this case, will then return the second best egress port. Since OpenFlow does not allow switches to recirculate packets through the same forwarding table, duplication of the iSDX tables is necessary to support this operation. In this case, the switch performs two operations. It first stores the Umbrella encoding of the path towards the primary egress port into the destination MAC field. Then, it stores the backup egress port identifier in the source MAC address, a field where in iSDX has 38 spare bits that are currently unused[3]. This additional information is used by the egress switches to reroutes packets upon any link failures between the fabric and the members' devices. While this approach guarantees a fast reroute, it requires the switch to store also the additional iSDX and Umbrella tables for the fallback path, at the cost of switch memory and processing delay.

Fast reroute upon BGP route withdrawals. Similar techniques can be also used to reroute packets in response to

[3] 10 bits are used to store an identifier of the sending member

BGP control plane modifications, i.e., withdrawals of BGP route towards a certain destination. Namely, edge switches can match BGP packets and modify their forwarding table accordingly, i.e., rerouting the packets affected by the BGP withdrawal.

V. Evaluation

We assess the scalability and performance of ENDEAVOUR using real data from one of the largest IXPs in the world. It has around 1000 provisioned member ports and more than 700 members which announce over 190 thousand distinct prefixes. The data set includes the RIB table dump of one of the IXP RSs (i.e., the BGP state) and sampled flow records of the traffic. We emulate ENDEAVOUR on a topology of 4 edge switches connected in a full-mesh with 4 core switches using Mininet (v2.1). This is a realistic IXP topology similar to current deplayments and reflects the current trend of adopting core-edge network design in production at large IXPs (LINX, AMS-IX, DE-CIX). To emulate the BGP process at the participant's routers we use Quagga. However, due to the large overhead of emulating hundreds of participants, the experiments with a large number of members do not rely on instantiation of emulated routers. Instead, BGP announcements are fed directly to the ENDEAVOUR's control plane.

A. Distribution of flows in the edges

First, we evaluate the number of flow entries required by ENDEAVOUR to support such a large IXP using the approach of Hermans et al. [25] but for different performance metrics and in a multi-hop topology. As input to the ENDEAVOUR controller we feed the RIB dump from the RSs of our vantage point. In turn, the controller generates flow rules according to the member's policies and distributes them on the IXPs fabric as explained in Sect. IV. While we extracted announcements for a total of 557 members from our RIB dump, to fully evaluate the scalability of ENDEAVOUR, we randomly generate additional members up to a total of 800. Each of the generated members announces 16 prefixes (the median in our data dump), and we assign them to the edge switches in a round robin fashion.

Figure 3 shows the average number of flow rules per edge switch in ENDEAVOUR for up to 800 members, where each has a variable number of outbound policies (4, 8 and 16) that alter the forwarding behavior of BGP towards 10% of the total participants. The graph shows that a balanced distribution of members' policies across the edge switches results in a linear growth of the required flow entries. We observe that, spreading the outbound rules across the (four) edges switches is a crucial operation for scaling the IXP fabric, in this case a gain of a factor of 4. Indeed, the number of outbound rules installed on each switch is bounded by the port density of the physical switches. Although [9], [25] use different datasets, we perform a rough comparison for similar numbers of participants and policies. ENDEAVOUR requires around 70% less rules than iSDX and the Hermans et al. solution. Whereas for 500 participants and a maximum of 4 policies each, iSDX [9] required 65.250 flows rules, ENDEAVOUR needs just an average of 22.094 flow rules per switch. Similarly, for 800 participants and 16 polices each, the Hermans et al. [25] needs over 500,000 flow rules while ENDEAVOUR requires only an average of 137.971 flow rules per switch (73% less). The results confirm that the distribution of members' outbound policies across multiple switches largely enhances the scalability of ENDEAVOUR.

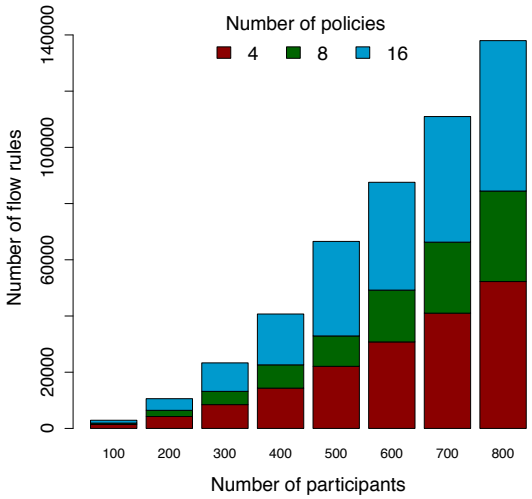

Fig. 3. Average number of flows in the edge switches of the IXP fabric as a function of the number of participants.

B. IXP internal load balancing

The ENDEAVOUR intra fabric load balancing mechanism spreads packets across core switches based on the least significant bits in the IP source and destination addresses. We evaluate this solution with a one-day data set from one of the edge switches of the large IXP. The maximum (minimum) observed throughput is 1.2Tbps at 20:57PM (0.2Tbps at 5:03AM). While IXPs frequently use the more complex hash-based ECMP to balance the traffic inside the fabric across its edge switches, we show that ENDEAVOUR succeeds with a simpler mechanism. Figure 4 depicts the traffic distribution across the four core switches over time when the ENDEAVOUR load balancing mechanism is in place, achieving significantly better performance, where each of the four links constantly receives less than 27% of traffic. An ideal ECMP scheme would obtain an almost perfect balancing at the cost of the drawbacks discussed in Section IV. While between 7AM and 9AM, our solution provides an almost perfect traffic load balancing, from 0AM to 6AM we observed a slight misalignment. To gain further insights, we analyzed the size of the flows and their total traffic share, as illustrated in Figure 5. The two measurements `t1` and `t2` spread over one hour each, with `t1` starting at 1AM and `t2` at 7AM. We define a flow as the traffic exchanged between a pair of source and destination IPs. Interestingly, a larger fraction of smaller traffic flows is forwarded during `t2`. This is consistent with previous studies on data centers, where static load balancing techniques gain in performance by increasing the fraction of smaller flows [26]. The ENDEAVOUR load balancing mechanism is

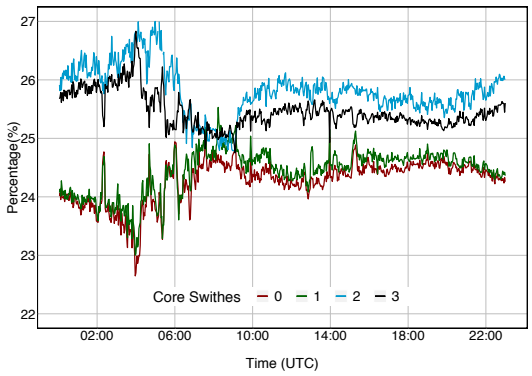

Fig. 4. Load Balancing real world performance.

hence appropriate, as IXP traffic is typically characterized by smaller flows due to the traffic requested by eyeball and ISP networks [27].

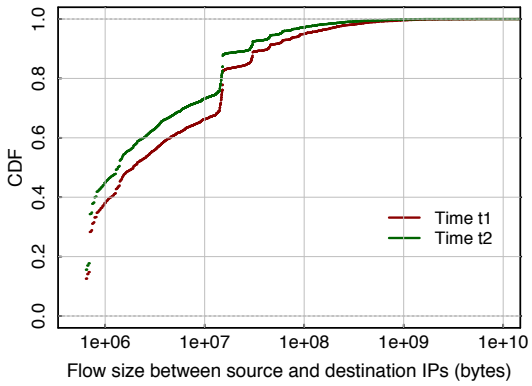

Fig. 5. Distribution of flow sizes within the performance measurement.

C. Blackholing

To evaluate the blackholing capabilities of the ENDEAVOUR platform, we record the BGP blackholing updates at a large European IXP [23] over the course of one day. These announcements and withdrawals are then replayed by using the ENDEAVOUR blackholing API. Figure 6 depicts the installation time of the new updates and reports the total number of rules installed in the fabric. Updates are performed in blocks of 50. We measured the time from calling the API to the application of the last rule on the data plane.

Figure 6 shows how the first block of updates is completed within 7 and 9 seconds. When issuing 1300 updates in consecutive blocks, times raise above 10 sec. The number of rules installed in the fabric scales to 2122 after 2000 replayed flow rule updates: the time for rules to take effect grows proportionally.

D. Forwarding State Recovery

We now evaluate the time required by a switch to retrieve its whole forwarding state from the controller, an operation that happens in case of software or hardware failures. We compare three different solutions: (i) Umbrella, (ii) a Learning

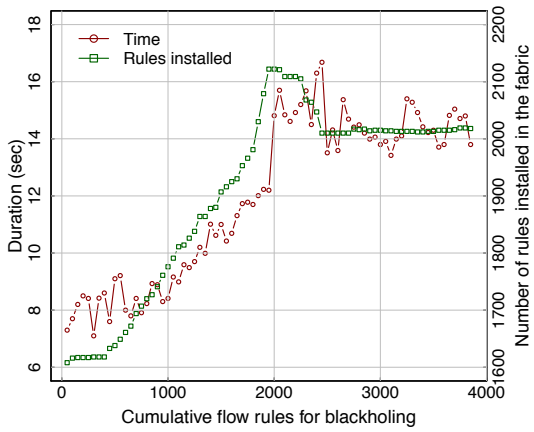

Fig. 6. Blackholing rule deployment time.

Switch, i.e., a Layer-2 forwarder which relies on the OpenFlow controller to flood broadcast ARPs, and (iii) ENDEAVOUR. In this analysis, we connect 250 members in our Core-Edge multi-hop topology. Figure 7 shows the average completion time for retrieving the forwarding state of each switch in the topology. We observe that the rule partitioning technique described in IV plays a key role in the time reduction: the lower the number of rules required per switch, the lesser the time needed to recover the forwarding state.

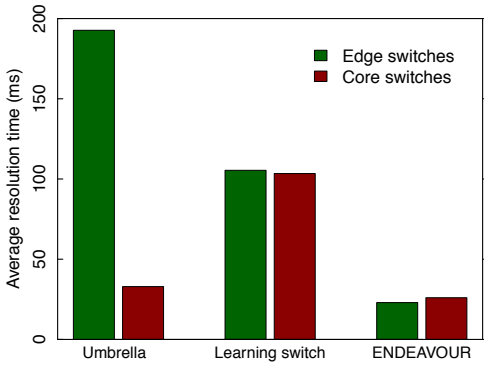

Fig. 7. Average resolve time on switch failure.

VI. Related Work

IXPs are the ideal place to extend the success of SDN from the intradomain, e.g., [6], to the interdomain level. IXPs have grown in number [1] and size [14], carrying huge traffic volumes [5], and interconnect a multitude of networks [2], [3]. With about 80% of the address space reachable through IXPs [3], [15], [28] affect a large share of the Internet. Being such a central element [2], [4], [28] makes them an ideal place to deploy the ENDEAVOUR architecture to bring SDN to the interdomain settings. Introducing OF in IXPs is a recent and promising idea [8]–[10], [29]. The Cardigan project [29] implements a hardware based, default deny policy, capable of restricting traffic based on RPKI verification of routes

advertised by devices connected to the fabric. While this approach offers the required protection for a stable IXP fabric, it is less suitable for IXPs that wish to remain neutral with regards to IP forwarding. Gupta et al. developed an SDN-based eXchange point (SDX) to enable more expressive policies than conventional hop-by-hop, destination-based forwarding [8]. iSDX shows that it is possible to implement representative policies for hundreds of participants while achieving sub-second convergence in response to configuration changes and routing updates [9]. However, iSDX only considers an IXP topology with a single switch, but in reality there might be multiple hops within the switching infrastructure. TouSIX [30] is the first European SDN-enabled IXP. TouSIX employs Umbrella [10], a multi-hop SDN fabric manager that tackles part of the control traffic, the broadcast traffic, directly within the data plane and can be implemented in most real IXP topologies. However, Umbrella lacks both on-the-fly recomputation of the internal IXP paths and protection mechanisms for link failures.

The ENDEAVOUR platform builds upon these experiences to produce an SDN-enabled architecture even more scalable than iSDX [9] and readily deployable in virtually any existing IXP topology. The set of use cases that can be enabled have been introduced in [7] and partially demonstrated in [31].

VII. Conclusions

In contrast to the rapidly evolving Internet, innovation in interdomain routing has remained stagnant. Despite the efforts to change this situation with the clean slate approach of SDN, its success has rarely gone beyond intradomain level. IXPs, with their central role in the Internet, present an ideal opportunity to break this deadlock and extend the benefits of SDN to the Internet at large. In this paper we presented, evaluated, and demonstrated ENDEAVOUR, an SDN platform for IXPs. ENDEAVOUR is a practical solution that can be deployed on multi-hop IXPs, supports a large number of use cases, is highly-scalable, and avoids broadcast storms. Our evaluation with real data from one of the largest IXPs showed the benefits, use cases, and scalability of our solution. In addition, by providing an open source solution, we invite everyone from the community to experiment, improve, and extend it.

Acknowledgments: We thank Josh Bailey, Yatish Kumar, David Meyer, and Jennifer Rexford for providing advisory guidance to the project. This research is (in part) supported by European Union's Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960).

References

[1] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: Mapped?" in *SIGCOMM*. ACM, 2009.

[2] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a Large European IXP," in *SIGCOMM*. ACM, 2012.

[3] I. Castro, J. C. Cardona, S. Gorinsky, and P. Francois, "Remote Peering: More Peering without Internet Flattening," in *CoNEXT*. ACM, 2014.

[4] D. Weller, B. Woodcock *et al.*, "Internet Traffic Exchange: Market Developments and Policy Challenges," OECD Publishing, Tech. Rep., 2013.

[5] I. Castro, R. Stanojevic, and S. Gorinsky, "Using Tuangou to Reduce IP Transit Costs," in *Transactions on Networking, Volume: 22, Issue: 5*. IEEE/ACM, 2014.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-deployed Software Defined Wan," in *SIGCOMM*. ACM, 2013.

[7] M. Chiesa, C. Dietzel, G. Antichi, M. Bruyere, I. Castro, M. Gusat, T. King, A. W. Moore, T. D. Nguyen, P. Owezarski, S. Uhlig, and M. Canini, "Inter-domain Networking Innovation on Steroids: Empowering IXPs with SDN Capabilities," in *Communication Magazine, Volume: 54, Issue: 10*. IEEE, 2016.

[8] 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," in *SIGCOMM*, 2014.

[9] 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*. USENIX, 2016.

[10] M. Bruyere, "An Outright Open Source Approach for Simple and Pragmatic Internet exchange," https://hal-univ-tlse3.archives-ouvertes.fr/tel-01393222v1, 2016, [Online; accessed 01-Apr-2017].

[11] H. Kumar, C. R. Russell, V. S. Sivaraman, and S. Banerjee, "A Software-Defined Flexible Inter-Domain Interconnect using ONOS," in *EWSDN*. IEEE, 2016.

[12] C. Hopps, "RFC 2992: Analysis of an Equal-Cost Multi-Path Algorithm," 2000.

[13] "ENDEAVOUR project," https://www.h2020-endeavour.eu/, 2017.

[14] J. C. Cardona and R. Stanojevic, "IXP Traffic: A Macroscopic View," in *LANC*. ACM, 2012.

[15] P. Richter, G. Smaragdakis, A. Feldmann, N. Chatzis, J. Boettger, and W. Willinger, "Peering at Peerings: On the Role of IXP Route Servers," in *IMC*. ACM, 2014.

[16] M. Wessel and N. Sijm, "Effects of IPv4 and IPv6 address resolution on AMS-IX and the ARP Sponge," Master's thesis, Universiteit van Amsterdam, the Netherlands, 2009.

[17] "FranceIX outage," https://www.franceix.net/en/events-and-news/news/franceix-outage-notification/, [Online; accessed 27-Mar-2017].

[18] V. Giotsas, C. Dietzel, G. Smaragdakis, A. Feldmann, A. Berger, and E. Aben, "Detecting Peering Infrastructure Outages in the Wild," 2017.

[19] V. Boteanu and H. Bagheri, "Minimizing ARP traffic in the AMS-IX switching platform using OpenFlow," Master's thesis, Universiteit van Amsterdam, the Netherlands, 2013.

[20] M. Chiesa, D. Demmler, M. Canini, M. Schapira, and T. Schneider, "Towards Securing Internet eXchange Points Against Curious onlooKers," in *ANRW*. ACM, 2016.

[21] M. Chiesa, R. di Lallo, G. Lospoto, H. Mostafaei, M. Rimondini, and G. Di Battista, "PrIXP: Preserving the Privacy of Routing Policies at Internet eXchange Points," in *IM*. IFIP/IEEE, 2017.

[22] A. Hari, U. Niesen, and G. T. Wilfong, "On the Problem of Optimal Path Encoding for Software-Defined Networks," in *Transactions on Networking, Volume: 25, Issue: 1*. ACM/IEEE, 2017.

[23] C. Dietzel, A. Feldmann, and T. King, "Blackholing at IXPs: On the Effectiveness of DDoS Mitigation in the Wild," in *PAM*, 2016.

[24] N. Kang, M. Ghobadi, J. Reumann, A. Shraer, and J. Rexford, "Efficient traffic splitting on commodity switches," in *CoNEXT*. ACM, 2015.

[25] S. Hermans and J. Schutrup, "On the feasibility of converting AMS-IX to an Industrial-Scale Software Defined Internet Exchange Point," http://ext.delaat.net/rp/2015-2016/p26/report.pdf, 2016, accessed Mar-2017.

[26] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in *NSDI*. USENIX, 2010.

[27] 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 *IMC*. ACM, 2013.

[28] N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "There Is More to IXPs than Meets the Eye," in *CCR*. ACM, 2013.

[29] J. P. Stringer, Q. Fu, C. Lorier, R. Nelson, and C. E. Rothenberg, "Cardigan: Deploying a Distributed Routing Fabric," in *HotSDN*. ACM, 2013.

[30] "TouSIX Official Website," http://www.touix.net/en/content/tousix-project, [Online; accessed 13-Feb-2017].

[31] C. Dietzel, G. Antichi, I. Castro, E. L. Fernandes, M. Chiesa, and D. Kopp, "SDN-enabled Traffic Engineering and Advanced Blackholing at IXPs," in *SOSR*. ACM, 2017.