# ENDEAVOUR: D4.5: Implementation of the Selected Use Cases for the IXP Members

Christoph Dietzel, Sebastian Abt, Marco Chiesa, Philippe Owezarski

# ENDEAVOUR: Towards a flexible software-defined network ecosystem

ENDEAVOUR

| | |
|---|---|
| **Project name** | ENDEAVOUR |
| **Project ID** | H2020-ICT-2014-1 Project No. 644960 |
| **Working Package Number** | 4 |
| **Deliverable Number** | 4.5 |
| **Document title** | Implementation of the Selected Use Cases for the IXP Members |
| **Document version** | 1.0 |
| **Editor in Chief** | Dietzel, DE-CIX |
| **Authors** | Dietzel, Kopp, Abt, Chiesa, Owezarski |
| **Date** | 31/01/2017 |
| **Reviewer** | Castro, QMUL |
| **Date of Review** | 08/12/2016 |
| **Status** | *Public* |

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 23/11/16 | 0.1 | First structure | Dietzel,Kopp,Abt |
| 26/11/16 | 0.2 | Advanced Blackholing added | Dietzel, Kopp |
| 28/11/16 | 0.3 | Inbound/Outbound TE added | Chiesa |
| 29/11/16 | 0.4 | Traffic Anomaly Detection added | Owezarski |
| 30/11/16 | 0.5 | First draft | Dietzel, Kopp |
| 02/12/16 | 0.6 | Traffic anomaly use case rewritten | Owezarski |
| 02/12/16 | 0.7 | Improved draft | Dietzel |
| 08/12/16 | 0.8 | Review | Castro |
| 12/12/16 | 0.9 | Review incorporated | Dietzel, Owezarski |
| 15/12/16 | 1.0 | Final version | Dietzel |

# 1 Executive Summary

Over the course of the ENDEAVOUR project the consortium developed a wide range of use cases as potential candidates to be implemented within the ENDEAVOUR prototype. After consolidating the most compelling use cases we implemented them into the ENDEAVOUR platform. To this end, the present deliverable showcases the implemented use cases of the ENDEAVOUR platform for Internet eXchange Point (IXP) members. Each use case is demonstrated in a video. In addition, Deliverable 4.4 discusses the relevant use cases for IXP operators. In combination, these two deliverables reflect the current state of the ENDEAVOUR platform prototype. We present technical background necessary to understand the implementation of each use case, the high level implementation itself, as well as a workflow of each demonstration.

# Contents

# 2 Introduction

In this report, we present the implemented ENDEAVOUR use cases for IXP members. This includes demo videos for all use cases and a description of the implementation of all demonstrators. We first outline the setup of the development environment in Section 3. Then we present the implemented use cases (Section 4.2) together with technical details. Next, we discuss the workflow of the demonstrator (Section 5). The final Section 6 summarizes the report.

Note, that this document is not self-contained, but accompanies the demonstrator videos[1] and the code[2]. Details on the implementation of the ENDEAVOUR framework can be found in Deliverable 2.3 and the monitoring architecture in Deliverable 3.3. For a comprehensive description of the use cases for IXP members we refer to Deliverable 4.3.

# 3 Development Environment

The ENDEAVOUR consortium maintains its code on the GitHub platform. We agreed on only having deployable code in the master branch. The development is coordinated by weekly calls and the ENDEAVOUR Waffle board[3]. Additionally, the ENDEAVOUR GitHub repository provides up-to-date setup instructions to provision a Virtual Machine (VM) with the ENDEAVOUR platform and install the necessary software for running the ENDEAVOUR platform.

Before the use case specific deployment scripts can be started, the ENDEAVOUR platform must be configured and started. The following commands set up the environment in a vagrant VM:

```
$ git clone https://github.com/h2020-endeavour/iSDX.git
$ cd ~/iSDX
$ vagrant up
$ vagrant ssh
$ git clone https://github.com/h2020-endeavour/endeavour.git
$ cd ~/iSDX/test
$ sh buildall.sh
```

[1] `https://www.youtube.com/playlist?list=PLl6z513p1ClFZqcITtxGTbgCe0wQOU0Ew`
[2] GitHub - `https://github.com/h2020-endeavour/endeavour`
[3] Waffle IO - `https://waffle.io/h2020-endeavour/endeavour`

As a result the current version of the platform is ready to run specific use cases or tests for use cases. A use case test can be started with the following command, "test-name" refers to a specific test:

```
$ sudo bash startup.sh --stats test-name
```

Further details on the platform and the test architecture can be found in Deliverable 2.3.

# 4 Implemented Member Use Cases

In this Section we present the implemented use cases, namely Inbound/Outbound Traffic Engineering, Advanced Blackholing, and Traffic Anomaly Detection. For each use case we discuss briefly the motivation, a high-level description of the implementation, and how we evaluated the correctness of the code for each use case. Before, we outline the reasons for the selection of these three use cases.

## 4.1 Selection Process

During the ENDEAVOUR project a large number of different use cases have been collected, discussed, and analyzed. An extensive collection of use cases from related work can be found in Deliverable 4.1. These use cases, in addition to those developed by the ENDEAVOUR consortium and the ones discussed at the IXP member workshop (summary in Deliverable 5.3) were joined and structured. With this comprehensive list at hand we were able to present our most promising candidates to a wider audience during the RIPE71 meeting. As a result we compiled a list with an extensive motivation, summary of the current situation, and a technical description, documented in Deliverable 4.3.

Eventually, DE-CIX's operational insights combined with even more feedback of the customers helped to select the first set of promising candidates to be implemented. The positive Extended Advisory Board feedback workshop (summary in Deliverable 5.5) encouraged us to proceed with the implementation of the selected set.

## 4.2 Implementation of Use Cases

Subsequently, we discuss the implementation of the three IXP member use cases together with a brief motivation and evaluation.

### 4.2.1 Inbound/Outbound TE

Traffic Engineering (TE), i.e., the task of tuning routing protocol parameters so as to optimize traffic flows, is a fundamental and crucial operation in today's network. Given the rich and flourishing connectivity ecosystem at IXPs, operators wish to carefully control how traffic enters/leaves their networks with the ultimate goal to enhance network performance. To this end, the ENDEAVOUR platform is designed to support fine-grained Inbound-/Outbound TE.

The implementation of fine-grained routing capabilities is the main part of the iSDX component. We refer the reader to Deliverable 2.2 for a detailed description of the ENDEAVOUR architecture and, in particular, the iSDX component. In addition, we refer the reader to Deliverable 2.3 to complement the architecture description with a detailed low-level explanation of the iSDX encoding mechanism at the forwarding-plane level.

The iSDX component, which closely relates to the Inbound/Outbound use case, has been extensively evaluated in Deliverable 2.2 along several dimensions: the number of IXP members, the number of announced IP prefixes, and the number of fine-grained Inbound/Outbound routing policies.

### 4.2.2 Advanced Blackholing

Blackholing is a reactive Distributed Denial of Service (DDoS) mitigation technique deployed by many Internet Service Providers (ISPs) and IXPs. At IXPs it allows a peer to announce a prefix via Border Gateway Protocol (BGP) to other peers, traffic destined for this prefixes is then discarded at the switching fabric. Advanced blackholing refers to an evolution of this services to overcome inherent limitations. With classical implementations, the finest level of granularity of blackholing is per Internet Protocol (IP) address. Given the patterns of many DDoS attacks, defining rules on transport protocol (TCP/UDP) or on transport ports is a well-perceived feature by the operators' community. Furthermore, with traditional blackholing, no statistics about the traffic (e.g, volume, protocols) are available since the traffic is dropped at the IXP. Advanced blackholing preserves these insights by providing statistical information of the discarded traffic.

Advanced blackholing is implemented on top of the iSDX and Umbrella as components of the ENDEAVOUR platform. To install a blackholing rule the participant controller pushes a new blackholing rule to the ENDEAVOUR central controller which installs it on the Software Defined Networking (SDN) edge switches. Thus, ingress packets with matching fields are discarded at

the IXP. In contrast to legacy blackholing, the volume of the dropped traffic can still be monitored.

To simplify the triggering process of Advanced Blackholing we add a web interface that abstracts the participant controller. This REpresentational State Transfer (REST) Application Programming Interface (API) enables each participant to install or remove blackholing rules in a simplified fashion.

The implemented Advanced Blackholing use case is verified through a setup that utilizes a synthetic network traffic generator. Thus, we instantiate IXP participants and configure them to exchange traffic amongst each other. Next we use the REST API to define blackholing rules that drop a subset of the generated traffic flows. The monitoring visualization tool, i.e., Grafana, is used to visualize the impact on traffic and the temporal correlation. We document this test scenario in our demonstration video, described in Section 5.2.

### 4.2.3 Traffic Anomaly Detection

Anomaly detection is an essential and crucial function in any computer network. Anomalies, be they legitimate as flash crowds or illegitimate as Denial of Service (DoS) or DDoS attacks strongly impact the performance and Quality of Service (QoS) of the networks and their communications. They can lead to Service-Level Agreement (SLA) violations and harmful situations for operators or members of IXPs. Illegitimate traffic will be discarded as soon as it is detected. Legitimate traffic can lead to routes or scheduling rules modification. Running such anomaly detection tool at IXPs is the best suited place because of the global integration of all networking function under the control of a single controller. In addition, the programmable aspect of SDN makes it able to easily and quickly deploy countermeasures for a maximum positive effect. This also enables to centralize the results of all monitoring tasks within the infrastructure of an IXP.

The Online and Real-time Unsupervised Network Anomaly Detection Algorithm (ORUNADA) algorithm that has been designed for this purpose is described in [1]. It uses the data provided by a non blocking monitoring system as input, namely in the ENDEAVOUR project the Field-Programmable Gate Array (FPGA) based Open Source Network Tester (OSNT) system, as described in Deliverable 3.3. As an output, anomaly detection software provides accurate anomaly signatures that are sent to the SDN controller for generating new SDN rules.

The anomaly detection software has been evaluated in detail, the results can be found in [1]. This includes detection accuracy, detection performance,

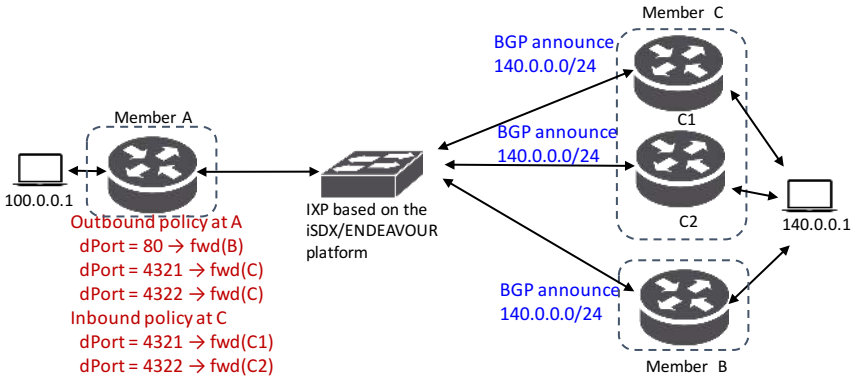



Figure 1: Simplified IXP based on the ENDEAVOUR platform for the Inbound/Outbound TE use case.

sensitivity analysis, as well as performance comparison with other anomaly detection tools and algorithms. It especially exhibits greater performances than other existing solutions on detection accuracy. The evaluation also elaborates on the limits of the anomaly detection software on the number of network features or throughput of the incoming traffic for achieving online and real-time detection.

# 5 Demonstration of Use Cases

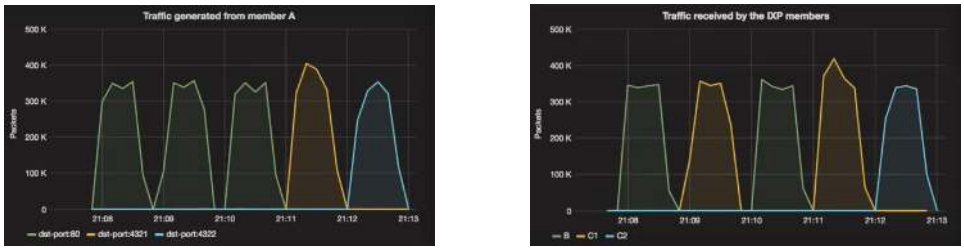
## 5.1 Inbound/Outbound TE

In this Section, we highlight how the ENDEAVOUR platform takes advantage of SDN's direct control over packet-processing rules to enable members to express flexible fine-grained policies for inter-domain traffic engineering.

**Demonstrator description.** The Inbound/Outbound use case demonstrator is built upon the simplified IXP scenario depicted in Figure 1. Member `A` wishes to send HTTP traffic towards member `B` and traffic destined to ports `4321` and `4322` towards member `C`, who, in turn, aims at steering this incoming traffic towards ports `C1` and `C2`, respectively.

The demonstration evolves as a sequence of five phases, each one spanning a one-minute time interval:

1. Host `100.0.0.1` generates a one-minute flow of HTTP traffic towards `140.0.0.1`.




(a) Traffic originated from `A1`.

(b) Traffic received at `B`, `C1`, and `C2`.

Figure 2: Demonstration of the Inbound/Outbound TE use case.

2. Member `B` withdraws its BGP announcement for the IP subnet `140.0.0.0/24`. At the same time, member `A` generates another one-minute HTTP traffic flow towards `140.0.0.1`.

3. Member `B` re-announces a BGP announcement for the IP subnet `140.0.0.0/24`. At the same time, Host `100.0.0.1` generates another one-minute HTTP traffic flow towards `140.0.0.1`.

4. Host `100.0.0.1` generates a one-minute traffic flow towards `140.0.0.1` destined to port `4321`.

5. Host `100.0.0.1` generates a one-minute traffic flow towards `140.0.0.1` destined to port `4322`.

The above five phases are depicted in Figure 2a, where we use different colored lines to draw the different type of traffic entering the IXP network from member `A`. Figure 2b shows how traffic is being received by members `B` and `C`. We can observe that HTTP traffic is correctly being received by member `B` whenever this one announces an IP prefix towards `140.0.0.0/24` via BGP (i.e., phases 1 and 3). The same HTTP traffic is re-routed through member `C` when member `B` withdraws its BGP announcement for `140.0.0.0/24` (i.e., phase 2). Finally, during the last two phases, we can observe that traffic destined to ports `4321` and `4322` is correctly being tranferred via member `C`'s border routers `C1` and `C2`, respectively.

A narrated video of this use case demonstration can be found at:

- `https://youtu.be/gP1-Wpca5p0`

**Reproducing the use case demonstration.** We leveraged *torch* for orchestrating and scheduling the network events needed to showcase the In-

bound/Outbound TE use case. The specification file containing each network event is stored in `iSDX/test/specs/test1-mh-te.spec`. The monitoring rules needed to verify the correct behaviour of the platform can be found in `iSDX/test/specs/ test1-mh-te-monitor_flows.cfg`. We leveraged Grafana [4] to visualize the flow of traffic through the IXP fabric. The web-based Grafana interface can be accessed via any browser installed on the Host VM machine by entering the following address `http://localhost:3000`. The visual dashboard for the Inbound/Outbound use case can be imported from file `iSDX/testinbound_outbound_te_dashboard.py`.

## 5.2 Advanced Blackholing

This Section presents the Advanced Blackholing use case and demonstrates how we take advantage of the fine-grained matching rules introduced by SDN.

**Demonstrator description.** The Advanced Blackholing use case demonstrator is built upon the simplified IXP scenario depicted in Figure 3. Our demo scenario consists of an IXP fabric based on the ENDEAVOUR platform. Three members namely `A`, `B` and `C` connect their border router to the ENDEAVOUR fabric. In order to simulate traffic flows across the IXP fabric, each border router connect an additional host machine. Each traffic stream has a different characteristic (e.g., dst port). We are able to see the different traffic streams monitored by the ENDEAVOUR platform in the port statistic graphs of Figure 4a and Figure 4b.

Subsequently, we describe the workflow of the demonstrator video.

1. Participant `C` activates a blackholing policy which should affect the traffic coming from `A` to dst port 53. We refer to this stream as `B1`. The blackholing policy is installed within the IXP fabric, to only drop the traffic stream `B1`. As a result, we can see the number of bytes dropped by this policy in Figure 4b. In addition, the number of bytes received by participant `C` decreases. All other streams remain unaffected.

2. Participant `C` activates a second blackholing policy for the traffic flow `B2` from participant `A`. Similar to the previous stage, we observe the number of bytes received by participant `C` further decreasing. Likewise, the blackholing policy graph rises.

[4] Grafana Visualization Tool - `http://grafana.org/`

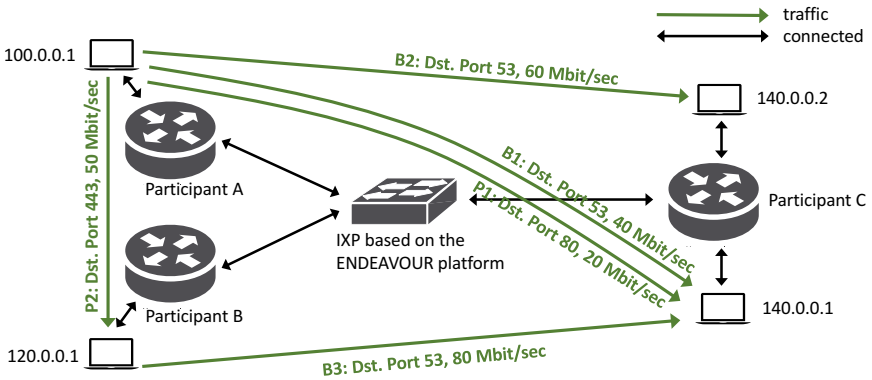



Figure 3: Simplified IXP based on the ENDEAVOUR platform for the Advanced Blackholing use case.

3. Participant `C` was able to blackhole traffic on a per port level from participant `A`. The Advanced Blackholing capabilities allow participant `C` to also specify blackholing policies affecting traffic from participant `B`. Participant `C` activates a third blackholing policy matching traffic from participant `B`, which we call `B3`.

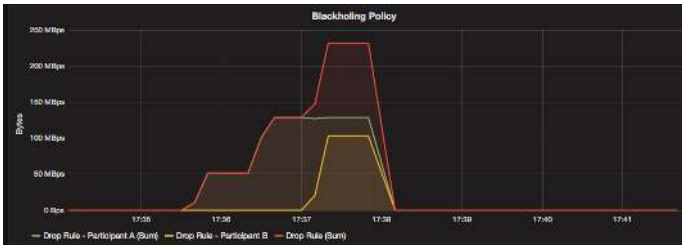
4. To simulate a real-world scenario, we terminate the three traffic streams `B1`, `B2`, `B3` at this stage, which indicates the end of the attack. Since participant `C` is able to monitor the traffic dropped by its blackholing policies shown on Figure 4b, he is able to detect the end of the attack and deactivates the blackholing policies.

5. As a consequence, the blackholing policies are removed from the ENDEAVOUR fabric. After we restart the traffic streams `B1`, `B2`, and `B3`. We can see that the traffic is received by participant `C` again.

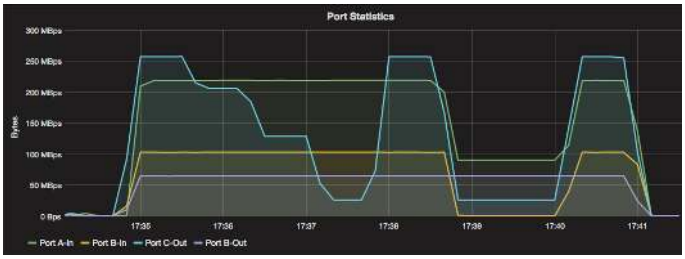
A narrated video of this use case demonstration can be found at:

- `https://youtu.be/jgaHVxq1-do`

**Reproducing the use case demonstration.** We leveraged *torch* for orchestrating and scheduling the network events needed to showcase the Advanced Blackholing use case. The configuration file containing each network participant and each event is stored in `iSDX/test/specs/test3-mh-bh.spec`. The blackholing rules need to be installed after the build process and can be found in `iSDX/examples/test3-mh-bh/policies/ participant_x_bh.cfg`.




(a) Number of bytes affected by blackholing policy installed within the ENDEAVOUR fabric.




(b) Number of bytes affected by blackholing policy installed within the ENDEAVOUR fabric.

Figure 4: Demonstration of the Advanced Blackholing use case.

We facilitate Grafana [5] to visualize the flow of traffic through the ENDEAVOUR Software Defined eXchange (SDX) fabric. The web-based Grafana interface can be accessed via any browser installed on the Host VM by entering the following address `http://localhost:3000`. The visual dashboard for the Advanced Blackholing use case is stored in the default Grafana setup.

### 5.3 Traffic Anomaly Detection

For this demonstration, we will show how the ENDEAVOUR platform can enforce filtering policies, with the help of the traffic anomaly detection tool.

**Demonstrator description.** To showcase this claim, we perform the tests on the topology presented in Figure 5. We consider two routers, named router `A` and router `B`, which are connected to an IXP. This IXP is running a SDN switch, controlled by the ENDEAVOUR platform. The owner of router `B` asked to probe its received traffic in order to block potential anomalies. As a consequence, the outbound traffic for this router will be mirrored to

[5] Grafana Visualization Tool - `http://grafana.org/`

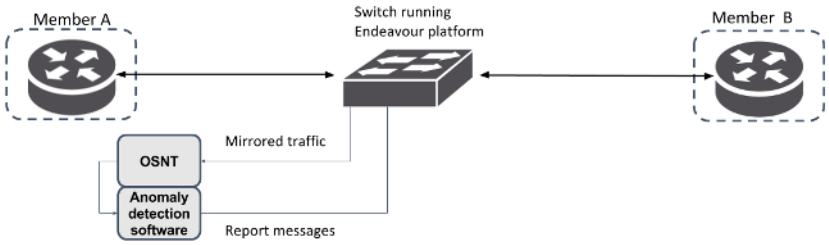



Figure 5: Topology for the Anomaly detection traffic use case

a dedicated port. Deliverable 3.3 describes the monitoring system and how different variables are made available.

To visualize this attack, we show a tcpdump filtering the upcoming attack.

The demonstration is divided into the following steps:

1. The participants are peering their IP addresses at the beginning of the video. It first exhibits some BGP traffic exchanged in router `B` traffic visualized thanks to tcpdump.

2. Router `A` sends some traffic to router `B`. The IXP forwards this traffic correctly.

3. Router `A` begins to send malicious traffic.

4. The monitoring platform detects this attack, sending a message to the ENDEAVOUR platform.

5. The ENDEAVOUR platform applies a filtering rule based on informations sent by the monitoring tool. It then stops forwarding this attack to router `B`.

A narrated video of this use case demonstration can be found at:

- `https://youtu.be/W32KgcaqHqA`

**Reproducing the use case demonstration.** This demonstration fully relies on the use of the monitoring platform developed in Deliverable 3.3, that takes advantage of a netFPGA card with a specific version of the OSNT platform. OSNT then provides significantly improved performances. Given that point, the monitoring probe could not be flexible enough to be run in other places except a prepared hardware testbed.

The test environment required to run the demonstration is available in *endeavour/examples/test-anomaly* folder (in the Git *anomaly_ detection_ demo* branch), and the controller used to accept report messages and enforce OpenFlow rules was added to the *master* branch of the ENDEAVOUR Github repository.

# 6 Summary

In this report, we describe the implemented ENDEAVOUR use cases for IXP members. Thereby, we aim to provide in addition to the demo videos an overview of the implementation. We outline the setup of the development environment, then we present the implemented use cases together with technical details. Finally, we discuss the workflow of the demonstrator.

# 7 Acronyms

**SDN** Software Defined Networking

**BGP** Border Gateway Protocol

**ISP** Internet Service Provider

**IXP** Internet eXchange Point

**QoS** Quality of Service

**SLA** Service-Level Agreement

**IP** Internet Protocol

**DDoS** Distributed Denial of Service

**DoS** Denial of Service

**TE** Traffic Engineering

**HTTP** HyperText Transfer Protocol

**VM** Virtual Machine

**API** Application Programming Interface

**REST** REpresentational State Transfer

**ORUNADA** Online and Real-time Unsupervised Network Anomaly Detection Algorithm

**DoS** Denial of Service

**DDoS** Distributed Denial of Service

**OSNT** Open Source Network Tester

**FPGA** Field-Programmable Gate Array

**SDX** Software Defined eXchange

# References

[1] J. Dromard, G. Roudière, and P. Owezarski. Orunada, an online and real-time unsupervised network anomaly detector. *IEEE Transaction on Network and System Management (TNSM)*, 2016.