A Development of Real-time Failover Inter-domain Routing Framework using Software-defined Networking

Yoshiyuki Kido[1,2], Juan Sebastian Aguirre Zarraonandia[2], Susumu Date[1,2], and Shinji Shimojo[1,2]

[1] Cybermedia Center, Osaka University,
5-1, Mihogaoka, Ibaraki-shi, Osaka 5670047, Japan
{kido, date, shimojo}@cmc.osaka-u.ac.jp
[2] Graduate School of Information Science and Technology, Osaka University,
1-5, Yamadaoka, Suita-shi, Osaka 5650871, Japan
j.sebastian@ais.cmc.osaka-u.ac.jp

Abstract. Internet Exchange Points (IXPs) play a crucial role in the interconnection between Internet Service Providers. Consequently, failures in the underlying IXP infrastructure impact the end-user experiences and can potentially translate into a financial loss for participants. Several research groups have attempted to override the Border Gateway Protocol (BGP) route selection process in IXP using a software-defined networking (SDN) approach to provide fast failover times, and to control how traffic flows between IXPs. In this research, data plane failures, such as physical disconnections between participant routers and the IXP switch, interrupt traffic flows until the configured BGP hold timer in the IXP route server expires. The objective of this research is to decrease the packets dropped during an IXP data plane failure event. We attempt to decrease the disconnection time between the IXP participants to a value lower than one second, which is the minimum configurable BGP Hold Time. In order to pursue this goal, we have developed a framework that reduces the failover process to the order of milliseconds. The results of the experiments registered an average failover time of 31 milliseconds. As a consequence of improving the failover time, the average packet loss is reduced.

Keywords: Software Defined Networking, Stream Processing, Internet Exchange Point

1 Introduction

Services that provide real-time communication such as VoIP, have stricter demands for continuous connectivity than during the origins of the Internet. Failing to do so could translate into a negative reputation and subsequently a large financial loss. At the same time, there has been an increase in the reliance of the scientific community on network infrastructures to share and store large amounts

of data, which in most cases is geographically dispersed. For example, large-scale instruments like the Large Hadron Collider produce petabytes of data that are transmitted to regional locations and are finally distributed to researchers [7].

According to current trends, the data intensity of many disciplines is projected to increase by a factor of ten or more over the next few years [8]. Scientific de-militarized zones (DMZ) are not only transferring elephant flows through national research networks, but also storing large data sets in public cloud infrastructures. In order to support these operations cloud service providers peer directly with Research & Education networks in order to avoid congesting their own networks. In some cases, cloud service providers peer with commodity IXPs that serve the traffic of a large percentage of ISP customers, which are not designed to carry long-term data set file transfers [5]. As this trend continues, IXPs will play a crucial role in the interconnection between ISPs, CDNs, and research networks. Consequently, any failure in the underlying IXP infrastructure will impact end-user experiences and research endeavors, and can potentially translate into a financial loss for its participants.

As more heterogeneous participants around the world join the network, future IXP implementations will have to support several features in their network architectures, such as:

- Quickly identify connectivity disruptions within their infrastructure.
- Resolve connectivity failures faster than current routing protocols.

Guaranteeing continuous connectivity across the Internet is a large challenge for network operators, because exercising any resilience or traffic engineering technique beyond their administrative domain is not possible. Furthermore, there are several technologies and protocols implemented in Wide Area Networks (WAN) that are difficult to change (e.g., Border Gateway Protocol (BGP)) due to their strong standardization and widespread adoption during previous decades.

In order to enable fast failover and continuous forwarding between Internet domains, researchers from both academia and service providers have proposed traffic engineering approaches and inter-domain protocol enhancements that allow manipulation of the routing decision process. Still, these techniques have not achieved sufficient standardization to be widely adopted by all Internet participants, mainly because these techniques introduce routing incompatibilities and additional deployment and management complexities [26].

Based on the previous inter-domain scenario, we attempt to approach the identification and resolution of connectivity disruptions at IXPs. The objective is to decrease the number of packets lost during a routing protocol convergence to a new route. The disconnection time between IXP participants is reduced to a value lower than one second, which is the minimum configurable BGP hold timer. We explore the enhancement of an IXP architecture with software-defined networking (SDN) and stream processing technologies to provide fast failover on the order of milliseconds.

In Section 2, the technical issues to be solved and the derived research goal, as well as the research problem, are clarified. Section 3 formally introduces the concept, design, and implementation of the proposed IXP architecture as well

as the technical issues derived in the previous section. In Section 4, evaluation experiments are conducted in order to verify the feasibility and correctness of the proposed method. Finally, Section 5 concludes this paper and suggests areas for future research.

2 Technical Issues and Related Research

This section presents the main challenges to reducing the failover time of data plane failures between Internet Exchange Point participants. An overview about how the Border Gateway Protocol works is presented, and the mechanisms that allow communication between network operators are provided.

2.1 Border Gateway Protocol (BGP)

Current Internet transport infrastructure relies on the BGP for interconnecting different administrative Internet domains, which are also technically referred to as autonomous systems (AS). As the de-facto standard for inter-domain routing for more than twenty years, the main function of the BGP is to exchange reachability information between ASes by means of BGP UPDATE or KEEPALIVE messages [9]. If a BGP speaker does not receive a KEEPALIVE or UPDATE message from a particular BGP neighbor for the duration of the Hold Time, then the speaker will close the BGP connection. By default, the Hold Time is set to 180 seconds on commercial routers. During the establishment of a BGP session, KEEPALIVE messages notify each BGP peer that the neighbor router is active. After a BGP session is established, the session is used as a mechanism to determine whether peers are reachable by being exchanged at a regular interval, which is usually one third of the configured Hold Time interval. KEEPALIVE messages must not be sent more frequently than once per second [24].

2.2 Internet Exchange Points (IXPs)

An IXP is a network infrastructure that facilitates the exchange of Internet traffic between ASes. Additional services offered at IXPs include the free use of route servers. Route servers provide participants with a scalable solution to peer with a large number of co-located ASes through a single BGP session, without having to manually configure individual adjacencies with each AS. Generally, participants in an IXP that want to exchange traffic must comply with some basic requirements:

1. Have a public Autonomous System Number (ASN).
2. Bring a router to the IXP. Connect one of its Ethernet ports to the IXP switch, and one of its WAN ports to the WAN media leading back to the network of the participant.
3. The router of the participant must be able to run the BGP.
4. Agree to the General Terms and Conditions (GTC) of the IXP.

Networks can peer publicly or privately in an IXP. Autonomous systems that publicly use an IXP network infrastructure for exchanging traffic make a one-time investment to establish a circuit from their premises to the IXP and pay a monthly charge for using an IXP port and an annual fee for membership to the operator of the IXP. Other than these costs, IXPs do not charge for exchanged traffic volume, unless there is a violation of the GTC (e.g., sending traffic to an AS without having received a route from that member). In addition, IXPs offer private peering interconnects that are separated from the public peering infrastructure and enable two ASes to directly exchange traffic and have a dedicated link that can handle stable and high-volume traffic (e.g., Internet2 Peer Exchange (I2PX) and Cloud Connect [14]).

2.3 Internet Exchange Point Data Plane Failure Detection Challenges

Data plane failures, such as physical disconnections between participants routers and the IXP switch, interrupt traffic flows until the configured BGP hold timer in the IXP route server expires. During this lapse of time, no failover mechanism is started, and, as a result, packets are dropped between the disconnected participants.

The goals of implementing a fast failover mechanism are to provide continuous service delivery after a failure has been detected and to optimize the performance of networks by controlling how traffic flows [28]. Among the motivations for providing fast failover and control to traffic that crosses multiple Internet domains are [10]:

Network failures and changes in routing policies: Neighbor domain failures and fluctuations degrade user performance and lead to inefficient use of network resources.

Violation of peering agreements: If the amount of traffic exchanged by two peer ASes is exceeded by one of the ASes, then an AS should have mechanisms to direct some traffic to a different AS without affecting its neighbor throughput.

Current limitations for providing fast failover at IXPs techniques represent a significant challenge for applications and collaborations groups between distributed domains that rely on underlying networks for services that include real-time data rendering and transfers, on demand communication tools, and processing data at high bandwidths for visual analysis. As pointed out in [15], each one of these applications has multiple network requirements, such as guaranteed bandwidth, traffic isolation, and time schedulers. Some of the limitations to providing fast failover at the IXP and performing traffic engineering in the inter-domain scenario derive from the BGP routing decision process, as presented in the following section.

The following are some of the limits that have been recognized based on the routing decision criteria of BGP that prevent fast reaction to data plane failure events and dynamic distribution of traffic between ASes at IXPs:

Single next-hop per network prefix: The BGP neighbor with the shortest AS path attribute is selected as the next-hop for forwarding all traffic to a given destination, discarding valid offers from other BGP neighbors.

Rigid architecture: Widespread adoption of new routing features is difficult because contiguous BGP neighbors need to overcome implementation incompatibilities before agreeing on their deployment.

Slow convergence: Border Gateway Protocol outages can take dozens of seconds to recover. Until the BGP Hold Time expires, a new route will not be selected for failover, and communication will not be resumed. This downtime impacts the performance of applications that rely on continuous connectivity.

Regarding the slow convergence of the BGP, the Hold Time parameter can be configured with low values. However, the problem with this is that if KEEPALIVE messages are buffered longer than the Hold Time, or if one of the BGP processes of the peering router cannot generate KEEPALIVE messages fast enough, then the BGP session is terminated. Moreover, BGP routers can protect themselves against aggressive KEEPALIVE timers from neighbors by refusing peering negotiation and never establishing a session [2].

2.4 Software-defined Networking (SDN) in Inter-domain Scenarios

In network devices (e.g., routers and switches), the packet-forwarding task is performed by its data plane according to the rules and paths calculated by standard protocols running on its control plane. In a SDN architecture, the control plane of a network device is decoupled from its data, enabling programmability through an Application Programming Interface (API) such as OpenFlow or P4 [16][4].

OpenFlow is a communication protocol that enables a centralized SDN Controller to program the data path of a network device according to the decisions of applications and protocols running on the control plane or application layer. Among the features of the SDN architecture using OpenFlow, which is frequently used, is automatic isolation of traffic flows programming the match-action tables of the underlying network devices with OpenFlow rules.

The OpenFlow version 1.5 specification [21] supports the definition of multiple flow tables with rules that match flows to multiple parameters inside the packets header, such as the source and destination IP address, VLAN ID, and TCP or UDP port number. Inside the flow tables flow rules entries are sorted by priority, where the highest-priority rules are at the top of the flow table. Inbound packets are matched to the flow entries starting from the highest-priority rules. If there is a match, then the flow matching stops, and the set of actions for the matched flow entry are performed. Packets that do not match any flow entry are dropped or sent to the controller for further analysis.

Previous studies attempted to override the BGP route selection process in IXPs using SDN, provided fast failover times, and controlled how traffic flows between ASes. In this section, we present the more relevant proposals and some of their limitations.

Software-defined IXP and iSDX Gupta *et. al.* [13] proposed a software-defined IXP (SDX) in which each participant AS interconnects to a shared data-plane and individually define forwarding policies relative to their current BGP routing table, enabling application-specific peering and fast failover. Figure 2.11 illustrates an SDX interconnection model with three ASes logically peering with an SDX Controller and pushing their routing.

In [12], how SDX can potentially fail at a large IXP as the compilation of BGP route updates scales between participant ASes is discussed. Policy compilation can take minutes as the number of participants exchanging traffic for hundreds of thousands of prefixes increases. In order to reduce the policy compilation time, Gupta *et al.* proposed a distributed model in which each participant AS runs locally an SDX controller that compiles routing policies. An IXP SDX controller collects the routing policies of all participants and updates the data plane accordingly.

Other Inter-domain SDN approaches A similar distributed approach to SDX was proposed by Wang *et al.* [27]. They introduced a framework for inter-domain path diversity based on SDN and the BGP. Each participating AS implements an enhanced SDN controller that exports and propagates to a peer AS SDN controller network functions that enable the setup of routing paths for particular applications. A richer criterion for the inter-domain route decision process is provided together with the BGP, as well as alternative routes to destinations through participant ASes.

Chen *et al.* [6] also extended SDN to an inter-domain network federation composed of several SDN autonomous domains. They proposed a network view exchange mechanism of routing criteria (e.g., Quality of Service attributes and application layer protocol numbers) across multiple SDN domains, enabling path diversity between ASes, and extending the BGP forwarding decision process. Based on the cited works in this section we recognize that the challenge of providing fast failover and application specific peering between ASes is only achievable only under the following assumptions:

- It is feasible for each participant AS to synchronously deploy the SDN technology locally and change their routing infrastructure.
- Autonomous systems with BGP routing implementation must be compatible with supplemental control-plane information and extensions, i.e., custom topology exchange algorithms.

2.5 Research Goal

The objective of this research is to decrease the packets dropped during an IXP data plane failure event. We attempt to decrease the disconnection time between the IXP participants to a value lower than one second, which is the minimum configurable BGP Hold Time. In order to pursue this goal, we propose a framework that reduces the failover process to the order of milliseconds. The proposed failover framework implements SDN technologies and stream processing concepts

in an IXP. This failover mechanism detects in real-time data plane connectivity failure events that otherwise would have to wait until the BGP convergence process ends to finish. In addition, this mechanism provides the capability of reacting to BGP topology changes in order to avoid violations of inter-domain traffic forwarding between members of an IXP. Unlike previous studies, we look for a design that does not require the adoption of SDN technology in all participants of the IXP in order to not affect their throughputs. There are two main technical challenges to achieve the research goal:

Data plane failure detection: Even though the BGP hold timer can be configured with a value of as low as one second, network device manufacturers recommend not configuring the hold timer below 30 seconds to avoid route flapping. In addition, in an IXP scenario, it is not feasible to demand that every participant configure a low BGP hold timer. Therefore, in order to achieve fast failover, it is not possible to rely on the BGP.

Correct failover path: This challenge derives from the previous challenge. If the BGP is not to be implemented in the failover process, it is difficult to guarantee that traffic redirection is to be done through an IXP participant that can reach the same routes that the compromised participant advertised. Lack of synchronization between the failover process and BGP route advertisements can lead to inter-domain policy violations.

Considering these technical challenges, the proposed failover framework should be designed to achieve a response time lower than the BGP Hold Time, and process BGP UPDATE messages to route traffic consistently between IXP participants.

3 Proposed Real-time Failover Framework

The basic idea behind the proposal is to reduce the IXP failover time upon data plane failures by enhancing its architecture with SDN to execute the traffic redirection task as a control plane application. A data plane failure event is defined as the disconnection of the physical link between the border routers of the AS and the IXP data plane switch. The proposed framework adds software modules to the IXP in order to process these connectivity failure events in real time.

One of the most important advantages of adopting this framework is that it allows control plane applications to express their peering intents according to a set of rules or an algorithm, thus enabling control of the failover path. A hypothetical implementation of SDN in an IXP could be as follows. First, after a data plane failure event is detected, and an SDN controller is notified via a control channel. Then, an SDN application processes the event and proactively updates the data plane via the SDN controller northbound API. After the flow tables are updated, the failover process is completed and traffic continues crossing the IXP through an alternate AS. Even though this approach can failover to another path within a short time it overrides the next-hops calculated by the

BGP. This can potentially violate BGP peering agreements and routing policies between IXP participants, so implementation becomes infeasible.

The main challenge in handling the failover process as an SDN application is that there is no direct channel or API with the IXP route server to learn valid alternative BGP neighbors to redirect traffic to. Network automation techniques allow applications to establish SSH sessions with a router, to collect and parse the output of commands that echo the Routing Information Base (RIB), and to store sessions in a database management system. However, collecting routing information in this fashion is prone to errors and lacks consistency in cases in which the result of the commands provide unexpected outputs or between routers operating systems. Moreover, BGP UPDATE messages that add or withdraw routes can arrive during the interval between SSH session establishment, making this approach ineffective in terms of correct packet forwarding to valid BGP neighbors. The failover process requires a mechanism that learns and considers valid BGP next hops from the IXP route server.

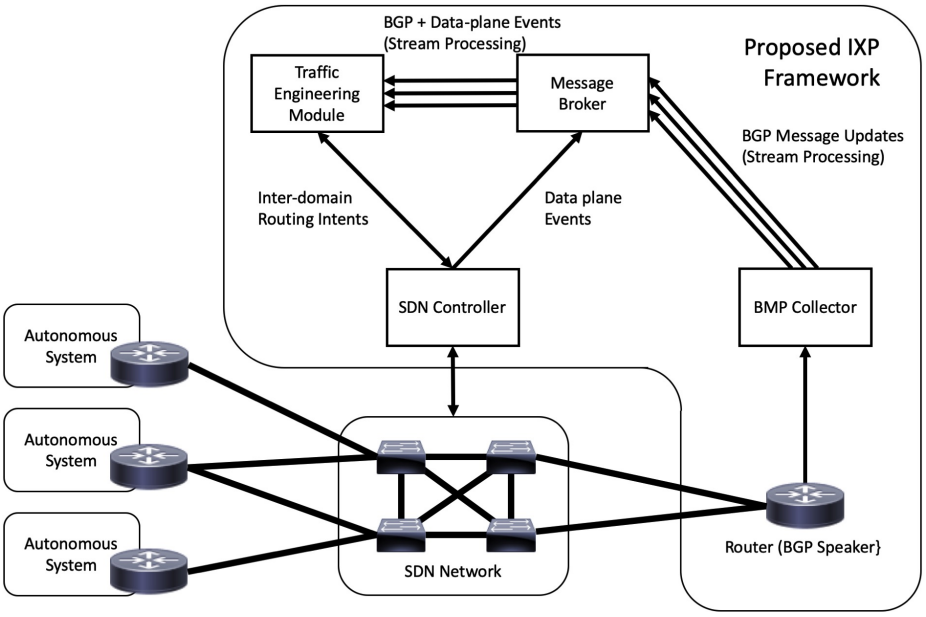

Fig. 1. Basic concept of the proposed framework.

We propose a real-time processing approach in which data plane failures and BGP updates are handled as a stream of events by a traffic engineering application. Figure 1 illustrates the concept of the proposed framework. Valid BGP alternatives for failover are learned from the IXP route server, bacause BGP UPDATE messages are handled as a stream of events and stored in a hash

map data structure by the traffic engineering application. When a connectivity failure occurs in the IXP data plane, the event is detected by an SDN controller and published to a message broker system. Events are made available for the traffic engineering application to consume and make a failover decision.

3.1 Real-time Failover Framework

Stream Processing and Messaging System The failover processing time is proportional to the latency of the pipeline between event sources and processors. The proposed framework requires data plane failures and BGP route withdrawal events to be processed on the order of milliseconds. Because of this requirement a stream processing system is chosen due to its capability to process data continuously as it arrives at a system [17]. Two types of actors compose a stream processing system:

- Producers: write event records to the system datastore.
- Consumers: poll the system datastore periodically and read event records.

The consumer-polling task introduces overhead and delays into the system whenever records are written into the data store at a faster rate than they can be read. In order to overcome this drawback, a message broker server is implemented, and both producers and consumers connect to this server as clients. The message broker is an optimized database that stores producer events in memory and allows fast reads from connected consumers. Low latency and high bandwidth between the broker and its clients are recommended for implementations that require processing events at a fast rate [1].

Software-defined Networking IXP Implementing SDN and OpenFlow at an IXP enables external applications to manage the underlying switches forwarding tables. The BGP can run as an SDN application that synchronously configures the IXP data plane as BGP updates are received from participants. Apart from this, SDN features do not introduce any additional inter-domain benefits directly, but provide two additional capabilities that can be exploited for the failover process:

- A data plane failure detection mechanism based on port status notifications.
- Flow rules match inbound packets in priority order.

OpenFlow enables data plane switches to exchange messages with an SDN controller through an OpenFlow channel [21]. The channel is instantiated using Transport layer Security or a TCP connection and is only used to exchange control information (e.g., physical port status changes). As ports are added, removed, or modified from data plane devices, notifications are sent to the SDN controller with an OFPT_PORT_STATUS message. Details describing the reason for this message are presented in Table 1. Whenever a loss of connectivity is detected, the OFPPS_LINK_DOWN flag of the status message is set to “true”

Table 1. OpenFlow Switch Port Status Messages.

Status	Description
OFPPR_ADD	A port was added.
OFPPR_DELETE	A port was removed.
OFPPR_MODIFY	An attribute of the port has changed.

in order to notify the SDN controller that there is no physical link present at the port.

Changes to the OFPT_PORT_STATUS message are useful in the proposed framework because such changes provide a relatively faster data plane failure detection mechanism than the BGP hold timer. OpenFlow switches send status messages asynchronously to the SDN controller without soliciting them.

In an IXP with OpenFlow switches, inbound packets header contents are matched to flow table entries. A flow table consists of entries defined by a match field and a priority. For the case in which there is a match for multiple flow entries, only the entry with the highest priority is considered [21]. For the proposed failover framework, once a data plane failure is detected it is necessary to install flow entries with a higher priority value than that installed by a BGP application. Following this, inbound traffic matching a BGP route that flows through a compromised BGP peer is redirected to another valid BGP next hop so that communication is not interrupted.

BGP UPDATE Message Ingestion The proposed framework follows inter-domain traffic engineering guidelines by not redirecting flows to ASes that are not expecting to receive them (e.g., do not advertise routes to a compromised destination). Candidates for routing traffic between IXP peers should be valid according to their advertised BGP update messages. This principle is respected by the BGP Monitoring Protocol (BMP) [25]. The BMP allows a server to export an unfiltered BGP routing table from a router and to gather all possible routes that can be used to reach each destination.

The BMP is implemented by two entities, i.e., a BMP collector and a BMP client, that communicate over a TCP connection. Once the BMP session is opened, the BMP client will start dumping to the BMP collector Route Monitoring (RM) messages containing the Adjacent Routing Information Base Inbound (Adj-RIB-In). In addition, incremental updates to advertising or withdrawing routes are dumped. The peer session status is also dumped by a BMP client. Because of these features, we herein consider a BMP collector in order to guarantee that failover paths are valid according to the BGP. Moreover, whenever IXP participants add routes or stop advertising them, the proposed failover framework can automatically update the set of candidates considered for the traffic redirection task.

Traffic Engineering Module As mentioned in Section 2, an IXP route server performs the best path selection process on behalf of all of its participants. In cases in which peering ASes implement route filters, these must be considered by the IXP as well. In order to facilitate both features, the proposed framework includes a Traffic Engineering module as an SDN application. Peering intents are pushed to the SDN controller through its exposed Northbound API. Route filters can be applied in the form of intents that explicitly specify the MAC address of the next hop for traffic being previously forwarded by an unreachable IXP participant. The Traffic Engineering module can also support custom traffic engineering algorithms that make decisions based on additional route attributes included in BGP update messages (e.g., LOCAL_PREF), or other criteria, maximally leveraging the capabilities of SDN technology.

3.2 Failover Mechanism Implementation

The message broker acts as a buffer between event producers and consumers and is implemented as a stream processing system that receives and makes available the following information with low latency:

- BGP update messages from the BMP collector.
- Communication failure between the IXP switch and participants.

In both operations, the consumer is the Traffic Engineering module. Upon receiving any routing update or host down events, the failover process is triggered. The message broker receives events related to connectivity failures, or BGP route updates. Depending on the case, the child node identifies the unreachable next-hop MAC address, or the IP address of an unreachable network. This information is used as input by the next child node, which selects an alternative next hop that can continue forwarding traffic. The final child node in the process installs flow rules through the Northbound API of the SDN controller using a flow priority value higher than that used by a BGP application.

Message Broker and Traffic Engineering Module The Message Broker module implemented is Apache Kafka [17], which is chosen because of its capability as a stream processing platform for storing messages written by producers, and makes the messages available to consumer software components with low latency. Event records that arrive to the system are classified into topics and written into a database table residing on the memory of the Kafka broker. The Traffic Engineering module is an application implemented using the Kafka Java consumer API (Listing 1.1), which feeds from three topics, each one running as a separate thread:

- Data plane link failures.
- BGP Updates (new routes advertised or route withdrawals).
- BGP neighbor additions or deletions.

Listing 1.1. Traffic Engineering module main thread

```
public class KafkaConsumerDemo {
  public static void main(String[] args) {
    boolean isAsync = args.length == 0 || !args[0].trim().
        equalsIgnoreCase("sync");
    ConsumerHost thread1 = new ConsumerHost("pmacct.acct");
    ConsumerMessage thread2 = new ConsumerMessage("pmacct.
        bmp_msglog");
    HostDown thread3 = new HostDown("HOST");

    thread1.start();
    thread2.start();
    thread3.start();
  }
}
```

Each of these threads reads events and triggers the process for calculating a valid next hop for compromised traffic. The thread that reacts to data plane failures is shown in Listing 1.2. This thread starts executing by subscribing to the "HOST" Kafka topic, which is where the SDN controller commits records whenever it detects a link down in the data plane. The consumer poll method is an infinite loop that continuously polls the HOST topic to retrieve new commits at a frequency of 10 milliseconds. Whenever a data plane failure event is committed to the HOST topic, a control loop is executed to perform the failover task in the following order:

- Identify the unreachable IXP participant MAC address using the ConsumerHost.get() method.
- Remove the compromised IXP participant from the Traffic Engineering module data structure using the alt.remove() method.
- Construct a POST message with a new peering intent and send it to the SDN controller through its REST API through the Postman.postman() method.

The desired method for the next-hop selection process is implemented on individual Java classes: one for pre-configured next hops, and another for a round robin algorithm that randomly selects a single next hop from among all valid options.

Listing 1.2. Data Plan Failure Reaction Thread

```
@override
public void doWork() {
  consumer.subscribe(Collections.singletonList("HOST"));
  ConsumerRecords<Integer, String> records = consumer.poll();
  for (ConsumerRecord<Integer, String> record : records) {
    if (!record.value().isEmpty()) {
      alt = Consumerhost.get();
      hostDown = (record.value().substring(record.value().length
          () - 10)).substring(0, 8);
      alt.remove(hostDown);
    }
  }
```

```
12   try {
13     Postman.postman();
14   } catch (IOException ioe) { ioe.printStackTrace(); }
15 }
```

For the failover process the Traffic Engineering module must identify the port of the unreachable IXP participant. To complete the task, the module must also select a valid participant. The reachability information of each IXP participant is stored in a hash map. The benefits of implementing this type of data structure are:

- Reachability information can be stored and retrieved in no particular order.
- Data can be stored based on a key-value pair: For this proposal the key-value pair consists of an IXP participant MAC address and the interface port number used by the IXP switch to reach the address.

Software-defined Networking Controller The Open Network Operating System (ONOS) is an SDN controller that implements OpenFlow as its primary southbound protocol [3]. Within ONOS features, there are two key functionalities required in an IXP environment that can be fulfilled:

- Physical layer failure notification to a message broker.
- Connection to external networks using BGP.

Section 3.2 introduced how the SDN controller detects data plane failure events. Next, we clarify how this type of event is handled for failover purposes. In order to share the failure events with other applications (e.g., the Traffic Engineering module), the SDN controller must expose an interface where the MAC address of an unreachable next hop can be propagated. It is desirable to transmit this information between software components with low latency. For this purpose, ONOS provides a Kafka Integration application [19] that serves as a data plane event producer. As soon as ONOS receives an OFPT_PORT_STATUS message with the OFPPS_LINK_DOWN flag set to "true", ONOS reacts by generating an event record containing the MAC and IP address of the now unreachable peer. These events are written into a Kafka topic, which is later consumed by the Traffic Engineering module.

In order to establish BGP sessions with other networks, ONOS provides the SDN-IP application [20]. When activated in an IXP, SDN-IP allows the ONOS controller to learn BGP routing information from the route server. The BGP routes are then translated by SDN-IP into OpenFlow rules that are installed on the IXP data plane.

Border Gateway Protocol Monitoring Protocol Collector The implemented BMP collector is pmacct [23], an open-source telemetry tool that dumps BMP updates to a message broker like Kafka in JSON format. The IXP route server is configured as a BMP client that establishes a BMP session with pmacct. In the proposed framework it is of special interest when the IXP route server send pmacct the following messages:

– Peer Down notifications, which indicate that a BGP peering session was terminated.
– Route Monitoring messages, which provide a snapshot of the Adj-RIB-In of each monitored peer.

After pmacct receives both messages, the messages are published to the Kafka broker for the Traffic Engineering module to consume and make the necessary changes to the IXP data plane in order to maintain continuous communication between participants.

4 Evaluation

Three experiments have been conducted to evaluate the failover time and packet loss of the proposed real-time failover framework. The number of packets dropped and the processing time overhead of the framework are measured for two scenarios:

– PP: Failover to a pre-configured path in the Traffic Engineering module.
– RR: Failover to a path that is randomly selected by a round robin algorithm.

The third experiment is designed to estimate the number of packets forwarded incorrectly to an IXP participant after advertising a BGP route withdrawal.

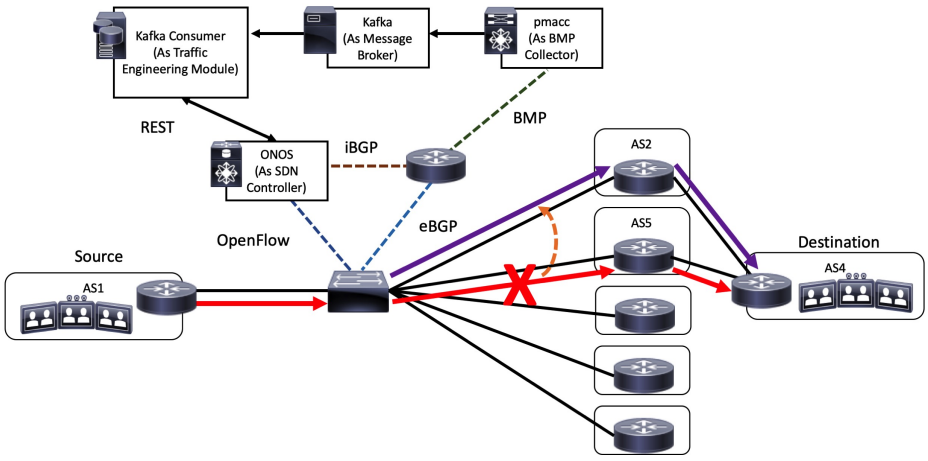

Fig. 2. Internet Exchange Points (IXP) testbed used for evaluation.

Figure 2 illustrates the proposed real-time failover framework testbed used for the experiments. Link failures are simulated between the IXP data-plane and the AS that is the next hop for traffic toward the destination host. The failover time is determined by measuring the processing time between the reception of a

link failure event by the ONOS controller and the time at which the data plane intent is sent by the TE module.

Packets are captured in pcapng format using Wireshark [22]. These traces are used to perform live monitoring of the traffic flows between hosts before and after the failover process. In addition, Wireshark generates a timestamp of the time when packets are captured, which is later used to determine the number of packets dropped.

4.1 Experiment Environment

The IXP testbed has been emulated using GNS3 [11], which is a network emulator that supports the complete functionality of network devices operating system. The emulated routers run the Cisco Internetwork Operating System version 15.5 and are configured with 256 MB of RAM. The data-link layer is implemented using Ethernet, and network devices interfaces support a bandwidth of 10,000 Kbits/s, an MTU of 1,500 bytes, and a delay of 1,000 microseconds. The SDN IXP is responsible for translating BGP route entries to OpenFlow rules for each peer AS in the following form:

packets from peer A destined for network C → forward to peer domain B.

The SDN implementation includes the ONOS SDN controller and a single Open vSwitch. The SDN-IP and Kafka-integration applications are activated in the ONOS controller. Within the IXP, SDN-IP installs and updates the forwarding rules of the data-plane according to BGP route updates. The Kafka-integration application is also activated in the ONOS controller in order to generate connectivity failure events in real-time as a stream of records.

For evaluating the packets dropped during the failover process two virtual machines that exchange traffic are deployed in the testbed. As Fig. 2 shows, the source host is configured in the same network of AS1, which peers with the IXP route server. The destination host is configured in the network of AS4, which does not peer with the IXP directly. In this configuration, AS1 has multiple valid paths to reach AS4, but due to the BGP best-path selection algorithm, only one path is selected to forward the traffic between the two hosts. User Datagram Protocol traffic is generated using netsniff-ng [18] at four different rates, and the transmitted frames have a total length of 42 bytes.

For the first scenario, AS2 is pre-configured as the failover path and is selected as the next hop during the failover process. It is possible to implement an interface with the TE module that validates the pre-configured failover path against the hash map containing the list of valid BGP next hops. We have decided not to include such functionality because it will not impact the failover time or packets dropped during a data plane failure event. Moreover, if an OpenFlow rule is installed for forwarding traffic through a pre-configured next hop, and later a BGP update message withdraws the route, then the TE module detects the event and rollback to the route determined by the BGP best-path selection algorithm.

For the second scenario a round-robin function selects the best path from available options stored in a hash map. The data structure holds key value pairs corresponding to each IXP participant MAC address and port number at which the pairs connect to the IXP switch. For both cases, the TE module sends a new peering intent through the REST API exposed by the ONOS controller. A POST method is constructed with its content type defined as a JSON object that contains the MAC address, the port number of the next hop, and an alphanumeric tag that uniquely identifies the installed flow rule.

4.2 Experimental Results and Discussion

Failover simulations were performed five times for each transfer rate. The disconnection interval was registered by measuring the time difference between timestamps of data plane failures and peering intents sent by the TE module. The round-trip times between the SDN controller and the Open vSwitch in the data plane is not considered, because this time is on the order of microseconds. The round-trip time between the Open vSwitch and each IXP participant is also within the microseconds range, and so is discarded. Figure 3 shows traffic arriving at AS4 during one of the multiple experiments.

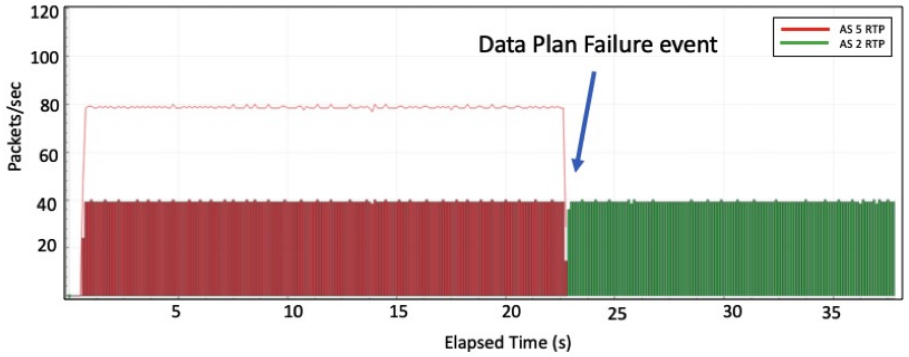

Fig. 3. Traffic forwarded through the IXP toward AS4.

Before the link failure, traffic toward AS4 is forwarded by AS5. When the link failure event is generated an interruption can be observed for a short interval of time until the proposed failover framework redirects the traffic to AS2 (PP case) or another IXP participant (RR case). Figure 4 illustrates the failover time for both pre-configured paths (PP) and round-robin (RR) selection. The results registered an average failover time of 31 milliseconds for PP, and 617 milliseconds for RR. Table 2 shows the average number of packets dropped at different transfer rates.

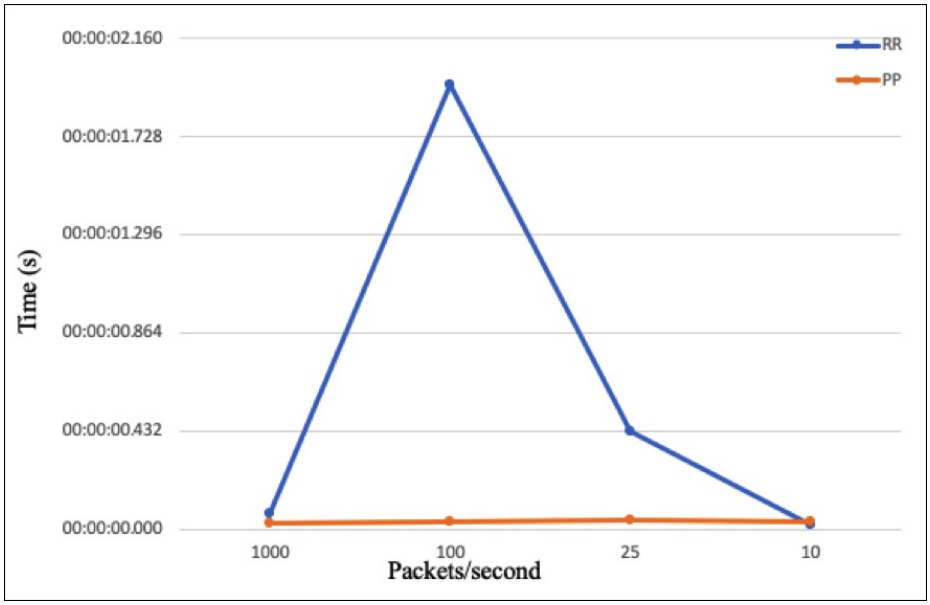

Fig. 4. Proposed framework failover time for pre-configured path and round-robin selection.

Table 2. Packets Dropped during Data Plane Failure Events.

Packets/s	Packets
10	0
25	1
100	2
1000	27

5 Conclusion

In the present study, we approach the challenge of reducing the failover time at an IXP by proposing a real-time failover framework that implements SDN and a stream processing system. Integrating SDN into the IXP architecture enables two features for improving the failover time provided by the BGP hold timer. The first feature is detection of connectivity failures between participants using OpenFlow, and the second feature is collection of the BGP routing tables of all the IXP participants through the BGP Monitoring Protocol (BMP). We evaluates the feasibility of the proposed framework by performing two experiments. These experiments investigate the failover time and packets dropped during data plane failure events. Traffic is exchanged between two hosts through an experimental IXP testbed, and a link failure event is simulated between the data plane

switch and a participant AS. Although the proposed framework makes assumptions about the IXP underlying data plane infrastructure, we believe that its implementation in production has the potential to reduce the number of packets dropped between participants during data plane failures events, and the BGP Hold timer is configured with standard values (e.g., 90 seconds).

Acknowledgements. A part of this research was supported by the JSPS KAKENHI Grant Number JP18K11355.

References

1. Akidau, T., Chernyak, S., Lax, R.: Streaming Systems. O'reilly (2018)
2. Beijnum, I.V.: BGP: Building Reliable Networks with the Border Gateway Protocol. O'reilly (2002)
3. Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., Parulkar, G.: ONOS: towards an open, distributed SDN OS. In: HotSDN '14: Proceedings of the third workshop on Hot topics in software defined networking. pp. 1–6 (August 2014). https://doi.org/10.1145/2620728.2620744
4. Bosshart, P., Daly, D., Izzard, M., McKeown, N., Rexford, J., Talayco, D., Vahdat, A., Varghese, G., Walker, D.: P4: Programming Protocol-independent Packet Processors. ACM SIGCOMM Computer Communication Review **44**(3), 87–95 (July 2014). https://doi.org/10.1145/2656877.2656891
5. Breen, J.: What Happens When the Science DMZ Meets the Commodity Internet? In: The 2015 Internet2 Technology Exchange (October 2015), https://meetings.internet2.edu/2015-technology-exchange/detail/10003935/
6. Chen, Z., Bi, J., Fu, Y., Wang, Y., Xu, A.: MLV: A Multi-dimension Routing Information Exchange Mechanism for Inter-domain SDN. In: 2015 IEEE 23rd International Conference on Network Protocols (ICNP). pp. 438–445 (November 2015). https://doi.org/10.1109/ICNP.2015.34
7. Dart, E., Antypas, K., Bell, G., Bethel, E., Carlson, R., Dattoria, V., De, K., Foster, I., Helland, B., Hester, M., Klasky, S., Klimentov, A., Lehman, T., Livny, M., Metzger, J., Milner, R., Moreland, K., Nowell, L., Pelfrey, D., Pope, A., Prabhat, P., Ross, R., Rotman, L., Tierney, B., Wells, J., Wu, K., Wu, W., Yoo, B., Yu, D., Jason, Z.: Advanced Scientific Computing Research Network Requirements Review: Final Report 2015. Lawrence Berkeley National Laboratory Report (June 2016), https://escholarship.org/uc/item/8zd204n4
8. Dart, E., Bell, G., Benton, D., Canon, S., Cowley, D., Dattoria, V., Fagnan, K., Foster, I., Giuntoli, P., Hester, M., Hettich, R., Hnilo, J., Howe, A., Jacob, R., Jacobsen, D., Love, L., Madupu, R., Metzger, J., Palinisamy, G., Rabinowicz, P., Rotman, L., Sears, R., Strand, G., Wehner, M., Williams, D., Zurawski, J.: Biological and Environmental Research Network Requirements Review 2015 - Final Report:. Lawrence Berkeley National Laboratory Report (September 2015), https://escholarship.org/uc/item/3647b0mm
9. Denning, P.J.: The ARPANET after Twenty Years. Computers under attack: intruders, worms, and viruses pp. 11–19 (February 1991). https://doi.org/10.1145/102616

10. Feamster, N., Borkenhagen, J., Rexford, J.: Guidelines for Interdomain Traffic Engineering. ACM SIGCOMM Computer Communication Review **33**(5), 19–30 (October 2003). https://doi.org/10.1145/963985.963988
11. GNS3, https://www.gns3.com/
12. Gupta, A., MacDavid, R., Birkner, R., Canini, M., Feamster, N., Rexford, J., Vanbever, L.: An Industrial-scale Software Defined Internet Exchange Point. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). pp. 1–14. USENIX Association (March 2016)
13. Gupta, A., Vanbever, L., Shahbaz, M., Donovan, P.S., Schlinker, C.B., Feamster, N., Rexford, J., Shenker, J.S., Clark, J.R., Katz-Bassett, B.E.: SDX: a Software Defined Internet Exchange. ACM SIGCOMM Computer Communication Review **44**(4), 551–562 (October 2014). https://doi.org/10.1145/2740070.2626300
14. Networking for Cloud, Internet2, https://www.internet2.edu/products-services/advanced-networking/networking-for-cloud/
15. Kiran, M., Pouyoul, E., Mercian, A., Tierney, B., Guok, C., Monga, I.: Enabling Intent to Configure Scientific Networks for High Performance Demands. Future Generation Computer Systems **79**(1), 205–214 (February 2018). https://doi.org/10.1016/j.future.2017.04.020
16. McKeown, N., Anderson, T.E., Balakrishnan, H., Parulkar, G., Peterson, L.L., Rexford, J., Shenker, S.J., Turner, J.: " OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review **38**(2), 69–74 (March 2008). https://doi.org/10.1145/1355734.1355746
17. Narkhede, N., Shapira, G., Palino, T.: Kafka: The Definitive Guide. O'reilly (2017)
18. netsniff-ng, http://netsniff-ng.org/
19. Kafka Integration, ONOS Dashboard, https://wiki.onosproject.org/display/ONOS/Kafka+Integration
20. SDN-IP Architecture, ONOS Dashboard, https://wiki.onosproject.org/display/ONOS/SDN-IP+Architecture
21. OpenFlow Switch Specification Version 1.5.1, https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf
22. PCAP Next Generation Dump File Format, https://wiki.wireshark.org/Development/PcapNg
23. pmacct project, http://www.pmacct.net/
24. Rekhter, Y., Li, T., Hares, S.: A Border Gateway Protocol 4 (BGP-4). RFC 4271 (January 2016). https://doi.org/10.17487/RFC4271
25. Scudder, J., Fernando, R., Stuart, S.: BGP Monitoring Protocol (BMP). RFC 7854 (January 2016). https://doi.org/10.17487/RFC7854
26. Silva, W.J., Sadok, D.F.: A Survey on Efforts to Evolve the Control Plane of Inter-domain Routing. Information **9**(5), 125 (May 2018). https://doi.org/10.3390/info9050125
27. Wang, Y., Bi, J., Zhang, K.: A SDN-Based Framework for Fine-Grained Inter-domain Routing Diversity. Mobile Networks and Applications **22**, 906–917 (April 2017). https://doi.org/10.1007/s11036-017-0857-2
28. Xiao, X., Hannan, A., Bailey, B., Ni, L.: Traffic Engineering with MPLS in the Internet. IEEE Network **14**(2), 28–33 (March 2000). https://doi.org/10.1109/65.826369