Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

Visualizing BGP RIB Changes into Forwarding Plane by Leveraging BMP and IPFIX

Master Thesis

Author: Livio Sgier

Tutor: Tobias Bühler, Thomas Graf (Swisscom)

Supervisor: Prof. Dr. Laurent Vanbever

April 2020 to October 2020

Acknowledgement

I would like to thank Monica, Guillermo, Almerima, Raphaël and Heng from Swisscom for their warm welcome and their valuable inputs regarding all aspects of this thesis. A special thanks goes to Matthias for his readiness to offer help quickly regarding everything related to the infrastructure at Swisscom. Another special thanks goes to my mentor at Swisscom, Thomas, whose deep knowledge in networking, his guidance and his readiness to offer competent advice inspired and motivated me throughout the thesis. I would also like to thank Paolo for developing and maintaining the wonderful network monitoring software pmacct which was heavily used throughout this thesis. He was always very helpful and quick in implementing feature requests.

At ETH, I would like to specially thank Tobias for his supervision. He was at all times very supportive and available to offer valuable advice regarding every detail of the thesis. I would also like to thank Prof. Laurent Vanbever, the head of the Networked Systems group at ETH, for offering me the possibility to write this thesis and his valuable feedback.

Abstract

At its core, the Internet and modern large-scale data centers use BGP as its main control-plane protocol to exchange routing and reachability information among its peers. BGP has been around for decades and has seen countless updates and extensions to support the growing number of use cases and ever increasing demands. The desire to gain deeper insights into the internal states of BGP prompted the introduction of the BGP Monitoring Protocol (BMP). We present a system that collects, stores and visualizes BMP messages in real-time in a lab network that supports MPLS layer 3 VPNs as well as in production. In addition, we collect forwarding-plane metrics and correlate them with BMP in time and on a VPN level to visualize cause and effect across both planes and pave the way for root-cause analysis and real-time performance monitoring.

Contents

Chapter 1

Introduction

Network monitoring is a difficult task. Service providers and other large companies maintain networks that consist of thousands of networking devices that exchange huge amounts of control- and data-plane information. Many protocols on different layers influence each other and cause actions that are fully automated. Because the protocols were not designed to be transparent in their reasoning, it becomes challenging to understand cause and effect. This lack of understanding and deeper insights into the network can cause performance degradation, network outages and delays in identifying and fixing the problem. Therefore, it is important to gather information from many sources and to monitor many different angles of the network to gain a high degree of visibility. To achieve that goal, large amounts of data have to be collected, correlated and processed in real-time to guide a human operator or a machine to identify anomalies and act accordingly.

The control-plane protocol of focus in this thesis is the Border Gateway Protocol (BGP) [41], which modern large-scale networks heavily rely on. The task of BGP is the exchange of network reachability information among its peers. It connects different Autonomous Systems (ASes), a collection of entities that form the backbone of the Internet. However, from a monitoring perspective, state information export is not natively supported by BGP, which motivated the introduction of a new protocol, the BGP Monitoring Protocol (BMP) [43]. It can be used to monitor BGP peerings and provides an interface to observe the router's internal BGP state.

The goal of this thesis is to incorporate the newest BMP drafts into a full-fledged end-to-end real-time monitoring system to gain deeper insights into the internal state of the network. This goal can be broadly split into the following distinct tasks: Collection, correlation, storage and visualization of the control- and forwarding-plane data. BMP and data-plane monitoring protocol data are exported from routers in a lab network. A collector application is then used to correlate the information and aggregate the forwarding plane with control-plane information. The different streams of information are fed into a message broker, from where the data is again ingested into a database. The data is then queried and visualized in a front-end with the option of further exploring the data.

The thesis is structured as follows. Chapter 2 introduces relevant protocols and technologies that are necessary to understand the work. It also mentions related work in the area of network monitoring protocols and systems. Chapter 3 gives a detailed explanation of the lab network, which will be a reference point throughout the thesis. Chapter 4 gives a detailed analysis and explanation of the end-to-end pipeline and its components. Chapter 5 evaluates the capabilities of the system, extends it to a production environment and compares different vendor implementations. Chapter

6 highlights the current state of BMP in more detail and proposes future research directions that build up on this thesis. Finally, Chapter 7 summarizes the work.

Chapter 2

Background and Related Work

The first section of this chapter introduces relevant protocols and technologies. The subsequent section targets related work, which compares competing standards as well as monitoring systems that build up on them.

2.1 Background

We first explain the control-plane protocol BGP, followed by the monitoring protocol BMP. Next, we discuss the data-plane monitoring protocol IP Flow Information Export (IPFIX) [12]. We then focus on MPLS VPN layer 3 networks [42], on which the lab environment is based on.

2.1.1 Border Gateway Protocol (BGP)

BGP is a standardized gateway protocol that the Internet heavily depends on. It is designed to exchange network reachability information between autonomous systems (ASes). BGP is also used as a protocol within an AS, which is referred to as Internal BGP (iBGP). This work focuses solely on peerings between ASes, also called Exterior BGP (eBGP). BGP runs on top of TCP and the connection between two peers that run the protocol is called a BGP *Peering*. While there are message types in BGP that are concerned with error handling and connection establishment and maintenance, the transfer of routing information is encoded in a BGP *Update* message. The basic structure of this type of message consists of one or many routes that are advertised or withdrawn, each associated with a set of attributes.

The routing information is stored in Routing Information Bases (RIBs). A BGP process has multiple RIBs, as displayed in Figure 2.1. The Adjacency RIB Inbound (Adj-RIB-In) contains routing information that has been advertised by its peers. The Adj-RIB-In can be further split into pre- and post-policy. The BGP process can define a set of inbound policies for each peering, which can modify or reject incoming route messages. Only routes that pass that inbound policy check end up in the post-policy Adj-RIB-In. Another filter is applied by BGP's *Decision Process*. The Local RIB (Loc-RIB) contains the routes that have been selected by that process. The Adjacency RIB Outbound (Adj-RIB-Out) exists for each peer and normally matches what is in the Loc-RIB. Analogously to the Adj-RIB-In, the Adj-RIB-Out is split into pre- and post-policy that reflects the per-peer filtering and/or modifications of route messages.

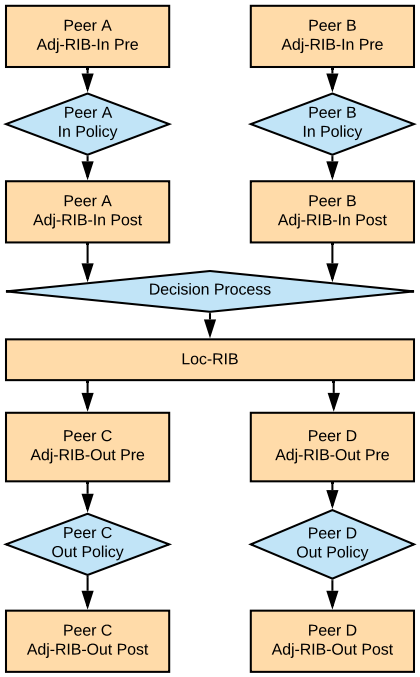


Figure 2.1: BGP RIBs and Policies involved in a BGP Process

2.1.2 BGP Monitoring Protocol (BMP)

The BMP protocol was born out of the desire of researchers to have a view of the BGP information which cannot be realized with standard BGP protocol mechanisms. The original BMP draft became a standard in June 2016 as RFC 7854 [43]. A router is configured to send BMP messages to one or more BMP monitoring stations, also referred to as BMP servers. The BMP message format consists of a header that contains the BMP version, the length of the message as well as the message type. An additional per-peer header follows for all messages that are exchanged in the context of a BGP peering. As an example, the per-peer header of an Adj-RIB-In route update message carries the peer dimensions of the router the update messages are received from. This additional peer header contains information such as the peer type (global instance peer, route distinguisher instance peer etc.), flags that indicate the IP version of the peering as well as attribute formats among others. Also, BGP information like the AS of the peer and its BGP ID are included. The per-peer header ends with timestamps of the message arrival. Messages that are not per-peer based omit the per-peer header.

The original BMP standard [43] defined the following message types:

- **Route Monitoring**: This message type carries the Adj-RIB-In information of a peer in the form of BGP update messages, pre- and post-policy. After a BMP router establishes a connection to a BMP server, all routes in the Adj-RIB-In of the BMP router are sent to the server. After the initial table dump, the BMP router sends incremental state-compressed Adj-RIB-In updates.

- **Statistics Report**: Statistics reports contain information that might be of interest to an operator. Transmission of these events can either be periodically transmitted by a predefined

interval or event-driven. Examples include number of duplicate advertisements/withdraws, number of updates invalidated due to loops in the AS path and many more.

- **Peer Down Notification**: When a peering goes down, it triggers a peer down notification that contains a reason code and associated data.
- **Peer Up Notification**: Peer up notification messages indicate that a peering session has been established as an encapsulated BGP *Open* message with additional local IP and port information.
- **Initiation Message**: This message type is used by the BMP router to provide vendor specifics, software versions and other information in the form of information TLVs (Type-Length-Value). Initiation messages can also be used to advertise capabilities to the BMP server.
- **Termination Message**: The termination message provides a way for the monitored router to indicate the reason for the termination in the form of an information TLV.
- **Route Mirroring Message**: Route mirroring messages are used to duplicate the received messages on the BGP peer *without* state compression, for example to help in the debugging process. This is in contrast to the state-compressed route monitoring messages. The message may or may not include a route monitoring message. Other uses of this message type are information about lost or faulty Protocol Data Units (PDUs).

Complete access to the Adj-RIB-In (pre- and post-policy) is useful, but it neglects all other RIB information in the BGP process. As a result, the BMP community developed an update, which became a standard in November 2019 as RFC 8671 [16], to enable access to (pre- and post-policy) Adj-RIB-Out. Yet another draft [15] proposed access to the Loc-RIB, thus enabling full RIB coverage. However, having full RIB coverage introduced new challenges and opportunities. The Loc-RIB contains the routes that are selected by the BGP *Decision Process* (cf., Figure 2.1), but it does not give any information about how the routes are used, e.g., which one is best, best-external, backup or used with equal-cost multi-path (ECMP). We also lack information about which route-policy permitted, denied or modified which prefix and attributes. To cover these use cases, more drafts were introduced [6, 49].

2.1.3 IP Flow Information Export (IPFIX)

In conjunction to control-plane monitoring protocols, the forwarding plane has its own protocols to monitor actual traffic flowing through the network. This is most often done in the unit of flows due to the increased scalability to monitoring on the packet-level. RFC 7011 [12] defines a flow as “a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties“. The specific set of properties vary, but common values are source and destination IP addresses, port numbers, transport layer protocol and other meta information. One differentiates between inspecting every packet and sampling at a defined rate (e.g., 1:1000, every 1000th packet is inspected) to increase scalability at the cost of decreased accuracy.
IPFIX messages are split into three different types: Templates, option templates and data records. To decrease the footprint of the transmitted data, the description of the exported data is encoded into a template which contains the global IDs that map to a specific field as well as the length of that field. IANA maintains a list of all possible fields, also called information elements, which can be

found in [23]. The template is periodically exported at a predefined interval. The option templates allows for additional information to be provided about the data records, which would not be possible with the data records, because the data records are just a concatenation of the information elements defined in the template. The information to which template an option template or data packet belongs is encoded in a Template ID, which is exported in all packet types.

2.1.4 BGP/MPLS IP Virtual Private Networks

The lab environment, which acts as a reference point throughout this work, is a network that allows multiple customer Virtual Private Networks (VPNs) to be run over the same physical infrastructure, a common use case for service providers [42]. This type of network is called BGP/MPLS IP VPNs. The customer edge (CE) sends its VPN routes to the provider edge (PE), which distributes the routes to other CEs of that VPN. Routers in the core that are not edge routers are simply called provider (P) routers. Routes within a VPN are assigned an MPLS label which is transmitted alongside the VPN route. As a result, the core routers do not need to know about the VPN routes and make forwarding decisions solely based on the MPLS transport labels. It is important to isolate routes from different VPNs in order to allow for overlapping address spaces as well as to guarantee a higher level of security. The isolation is accomplished via VPN routing and forwarding tables (VRFs). Each PE/CE circuit is associated with one or multiple VRFs. The PE routers use BGP to distribute the routes. However, if two VPNs advertise the same address to the PE router that refer to different systems in different VPNs, it is important that BGP does not treat it as the same address to prevent installing only one route, making the other unreachable. To deal with this issue, RFC 4760 [2] defines a BGP extension to carry routing information from multiple address families, including VPNv4/6. A VPNv4 address consists of an 8 byte route distinguisher (RD), followed by an IPv4 address[1]. The sole purpose of the RD is to make multiple equal IPv4 addresses, that denote different systems advertised by different VPNs, unique by choosing distinct RDs.

Another important concept is that of a route target (RT), which identifies VPN membership. Every VRF can have RT import lists and RT export lists. The RTs that are placed in the export list are attached to routes that are advertised to other PE routers. The PE that receives the route compares the RTs that are attached to the route with its import list. If there is a match, the route is imported into the VRF, otherwise it is rejected. If the import and export lists are identical across all VRFs, we have an any-to-any topology. However, more sophisticated topologies, e.g., hub-and-spoke, can be created with the concept of RTs by constructing the RT import and export lists accordingly.

These concepts apply in the context of a single AS. What if VPN routes can be advertised across AS boundaries? RFC 4364 [42] defines Option A, B and C that differ in their complexity, scalability and trust relationships between the ASes. Option A installs a VRF on the PE router for every VPN. Both AS border routers act as a CE and PE simultaneously. This simple option allows to advertise IPv4/6 unicast addresses without exposing internal routing tables but with the cost of maintaining a large number of VRFs which hinders scalability. Option B advertises labeled VPNv4 routes between the trusted endpoints of two ASes. This improves scalability because there is no need for separate VRFs per VPN and isolated routing tables, because VPNv4 addresses are globally unique within an AS. However, the AS border router can still become a bottleneck if it

[1] RFC 4659 defines the VPNv6 address family to accommodate IPv6 addresses.

has to maintain routes for every VPN. Scalability is further improved with Option C. With this option, the AS border routers do not maintain and distribute VPNv4 routes. Instead, there is a multi-hop BGP connection between PE routers or route reflectors (RR) in the different ASes. RRs are used to increase scalability by reducing the number of peerings and to decouple the control and forwarding plane. Routers peer with the RR, but not among themselves to achieve a linear relationships between the number of routers and peering sessions. With Option C, the AS border router is only needed to exchange the loopback address of the routers in the different ASes and the MPLS transport labels for the label switched path. Because we have an eBGP peering between two routers in different ASes, the next-hop is changed when a route is advertised over that peering. However, that causes the label switched path to be split into one segment to the router and another segment to the PE. To mitigate this issue, the next-hop can be left unchanged on that peering to have a single label switched path to the destination PE. This option results in the most scalable exchange of VPNv4 routes across different ASes.

2.2 Related Work

BGP monitoring systems can be broadly classified into two categories [48]: (1) Route collector servers that use the BGP protocol to receive messages from interesting routers and (2) separate protocols that actively or passively receive BGP information by other means. The first category involves active BGP peering sessions from another router in the network or from a software router like Quagga [38]. The accessible information with active peering is limited as only a subset of the BGP state information of a router is transmitted. Examples from the second category include the usage of SNMP that offer information about BGP monitoring activities [19] or directly issuing BGP commands (e.g., *show ip bgp*) to the router via SSH or telnet, a process that is referred to as *screen-scraping*. There are also BGP routers that make routing information publicly available for research purposes. BGP *looking glasses* allow users to directly access a router, e.g., via a web-interface, to query RIB information that is useful for interactive explorations. Another proposal is to use Netconf YANG [3] to create a BGP model that is vendor-neutral and aims to cover a wide range of BGP features [24]. The most recent protocol in the latter category is BMP. Systems that make use of BMP exist in [36] and [37], but they lack support for the newest BMP features (e.g., access to Loc-RIB and Adj-RIB-Out), they focus primarily on public networks and they do not correlate with the forwarding plane.

More established standards exist in the forwarding-plane monitoring space. Development of the first protocols started already in the early nineties because it enabled and facilitated security analysis, capacity planning, accounting, profiling and many others [20]. The lab environment uses IPFIX, which is based on Netflow [11], developed by Cisco. Before IPFIX was introduced, Netflow was the de facto standard and many other vendors implemented their own version, such as Jflow from Juniper Networks [27] or Cflowd from Nokia [35].

Chapter 3

Lab Network Environment

In this chapter we show the lab network environment and explain the topology as well as important concepts in more detail. We then trigger specific events and observe the control- and data-plane message it produces to illustrate the described concepts and the wealth of information it entails.

3.1 Lab Topology

As already mentioned in Section 2.1.4, the lab network, shown in Figure 3.1, is an MPLS/BGP IP VPN Inter-AS Option C network. It consists of two VLANs with the address space *192.0.2.0/25* and *192.0.2.128/25*, respectively. There are two CE routers *daisy-51/53* and six PE routers *daisy-*

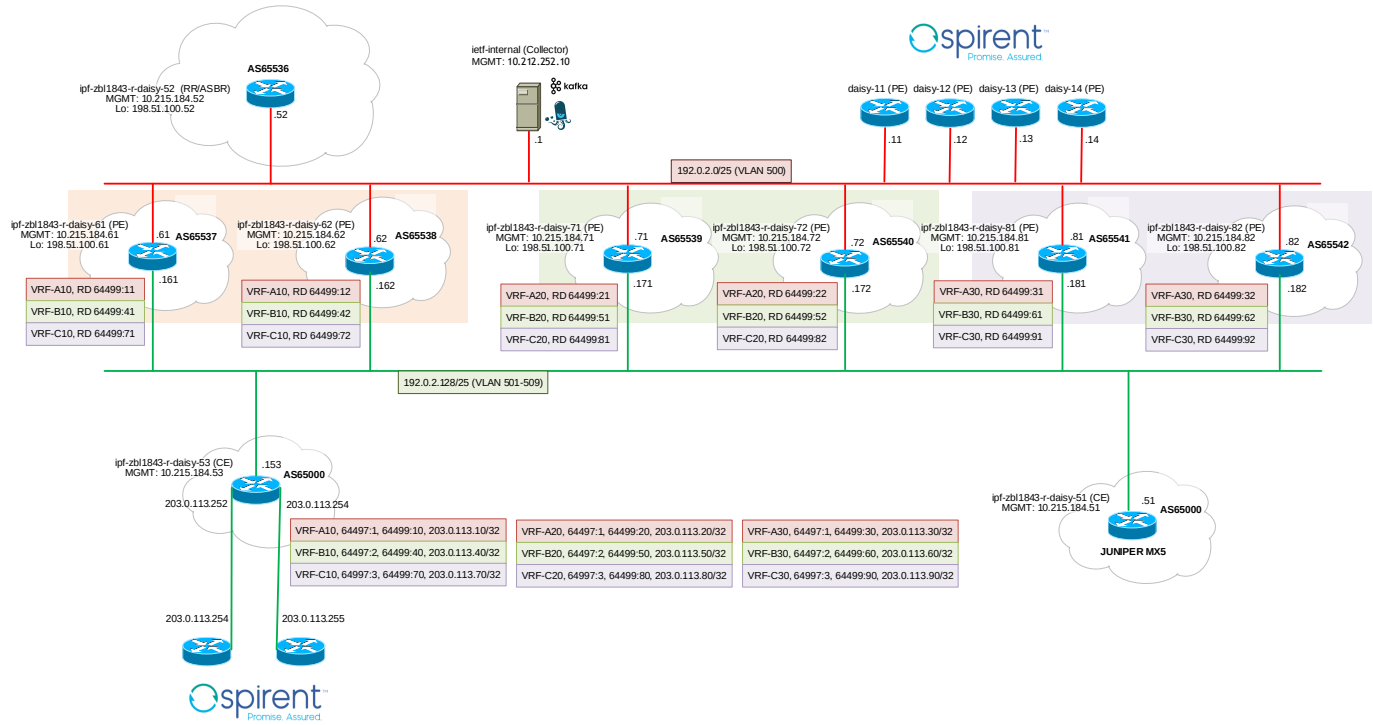


Figure 3.1: MPLS/BGP IP VPN Lab network topology

61/62/71/72/81/82 that route/forward traffic. There is also a route reflector (RR) and an AS border router (ASBR) in one device, denoted as *daisy-52*. RRs are used to increase scalability. They peer with every PE router and, thus, remove the necessity of a full-mesh BGP peering among every pair of PE routers. This reduces the number of peering sessions from quadratic to linear in the size of the PE routers, thereby increasing scalability. *daisy-52* is also an MPLS P router, thereby having more than just routing responsibilities. We employ a number of traffic generators from Spirent that can generate large amounts of control- and data-plane traffic in order to test the correctness and the scalability of the network. The routers export their control- and data-plane traffic to the collector *ietf-internal*. All network devices with the exception of CE *daisy-51* and the traffic generators are Huawei devices and they are all assigned their own AS. This design is inspired by RFC 7938 (Use of BGP for Routing in Large-Scale Data Centers) with the implication of employing eBGP as the sole routing protocol [29].

The CE router *daisy-53* maintains nine VRFs. Every router has an RD attached to each of its VRFs. There are multiple RDs for each VRF (each router has its own unique RD per VRF) to allow multiple paths to reach the destination (e.g., in the case of ECMP [21]). A route that is associated with a specific VPN is assigned a specific community value. For example, all routes that correspond to VPN *A* encode the standard community value *64497:1*. Likewise for VPN endpoint identifiers. To distinguish between the two cases, standard communities with a prefix of *64497:* are assigned to VPN identifiers and standard communities with a prefix of *64499:* are assigned to VPN endpoint identifiers. The PE routers *daisy-61/62/71/72/81/82* and *daisy-52* have their own unique loopback address in the form *198.51.100.??* where *??* stands for the associated number of the routers, e.g., *198.51.100.61* is the loopback address of *daisy-61*. On the CE router, each VRF has their own loopback address defined, e.g., *203.0.113.10* for VRF *A10*.

```
Marker: ffffffffffffffffffffffffffffffff
Length: 74
Type: UPDATE Message (2)
Withdrawn Routes Length: 0
Total Path Attribute Length: 46
▾ Path attributes
  ▸ Path Attribute - ORIGIN: IGP
  ▸ Path Attribute - AS_PATH: 65000
  ▸ Path Attribute - NEXT_HOP: 192.0.2.153
  ▸ Path Attribute - MULTI_EXIT_DISC: 0
  ▸ Path Attribute - COMMUNITIES: 64496:299 64496:1001 64497:1 64499:10
▾ Network Layer Reachability Information (NLRI)
  ▾ 203.0.113.10/32
      NLRI prefix length: 32
      NLRI prefix: 203.0.113.10
```

Figure 3.2: BGP update message according to the configuration of VRF instance *A10* on *daisy-53*

To show how we can monitor the network and what the exported data looks like, we activate an interface that causes control-plane messages to be exported and in a next step we send traffic over the VPN that is captured with IPFIX. For the control-plane event, we activate the IP address *203.0.113.10* on the CE router. This event triggers dozens of BMP messages of which we show a small subset to illustrate the wealth of insight into the BGP process it enables. *tcpdump* [46] is used on the collector to capture all incoming BMP messages. The resulting messages can then be dissected and visualized in Wireshark [13]. The IP address is bound to the VRF instance *A10* (cf., Figure 3.1). VRF instance *A10* is associated with two PE routers *daisy-61/62* with which *daisy-53* peers and exchanges information.

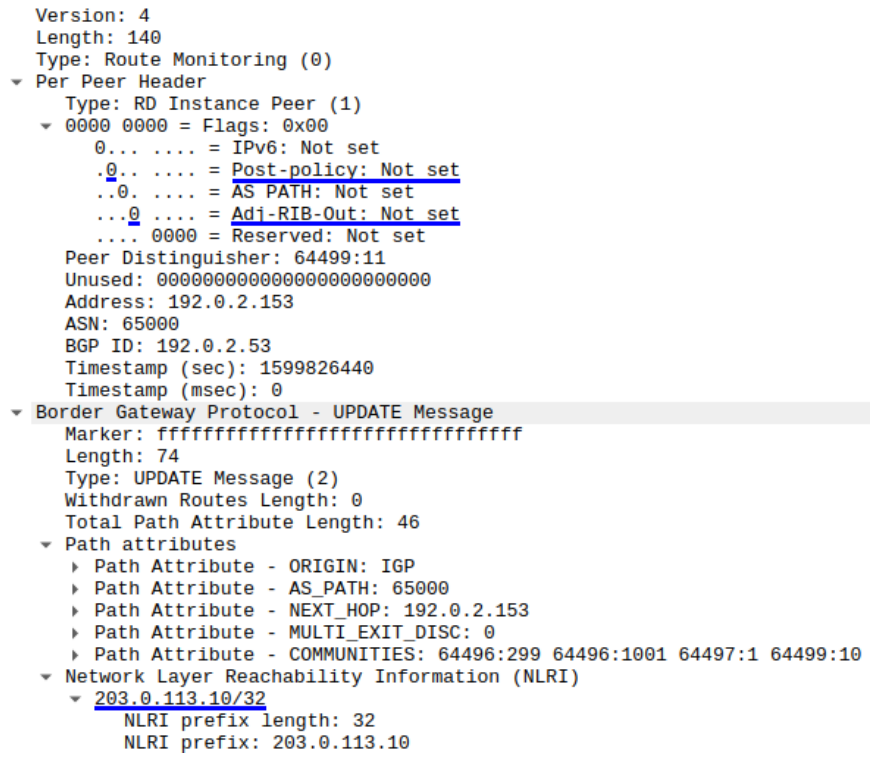


Figure 3.3: BGP Adj-RIB-In Pre-policy of *daisy-61* peering with *daisy-53*

Figure 3.2 shows the BGP message that was triggered by the interface activation. The message is an update message for the enabled IP address. The route carries multiple path attributes, most important for our purposes are the *NEXT_HOP* and *COMMUNITIES* fields. The former specifies over which IP address a message for the specified prefix is routed through and the latter is a method to add extra information to a route for which the semantics can be chosen arbitrarily. Some specific community values, such as the VPN identifier, will be explained in more details in the subsequent sections. The next section describes the BMP message this event produced in detail.

3.2 BMP in Action

The following BMP messages are exported from *daisy-61*. The BGP message from Figure 3.2 triggers multiple BGP events which are encapsulated in BMP messages and sent to the collector. Figure 3.3 depicts the Adj-RIB-In Pre-policy information of the peering with *daisy-53*. As mentioned in Section 2.1.2, every BMP message contains a per-peer header that contains additional information about the peering, such as flags that distinguish the RIBs (e.g., Adj-RIB-In Pre-policy) among other information, the peer distinguisher that refers to the VRF if the peering is of type *RD Instance peer* (in this example *64499:11* for *VRF-A10*) and the IP address of the peer and its AS number. The encapsulated BGP message that follows is equivalent to Figure 3.2.

As mentioned in Section 2.1.2, route policies that were triggered within the BGP process are exported with BMP as well. A BMP route policy message that was triggered by the previous BGP update message, can be seen in Figure 3.4. The name of the triggered route policy is *RP-A10-IP-IN*. This route policy adds the community value *64496:1033* to the route, as defined in the configuration

```
Version: 3
Length: 760
Type: Route Policy and Attribute Trace Message (100)
▸ Flags: 0x00
Route Distinguisher: 0x0000fbf30000000b (277021095624715)
Prefix Length: 32
Prefix (Reserved): 00000000000000000000000000
Prefix (IPv4): 203.0.113.10
Route origin: 3221226137
Event count: 2
Total Event Length: 721
Single event length: 373
Event count: 1
Timestamp (sec): 1599826440
Timestamp (msec): 462705
Path Identifier: 0
AFI: 1
SAFI: 1
▸ TLV: (t=0,l=7) VRF/Table
▸ TLV: (t=1,l=46) Policy TLV
▸ TLV: (t=2,l=46) Pre Policy Attribute
▸ TLV: (t=3,l=50) Post Policy Attribute
▾ TLV: (t=4,l=186) String
    Type: String (4)
    Length: 186
    Value: 786d6c6e733a7274703d2275726e3a6875617765693a79616e673a6875617765692d726f…
    String: xmlns:rtp="urn:huawei:yang:huawei-routing-policy" select="/rtp:routing-policy/rtp:policy-definitions/rtp:policy-definition[rtp:name='RP-A10-IP-IN']
Single event length: 348
Event count: 2
Timestamp (sec): 1599826440
Timestamp (msec): 463034
Path Identifier: 0
AFI: 1
SAFI: 1
```

Figure 3.4: Triggered route policy *RP-A10-IP-IN*, carried in BMP Route policy and attribute tracing message triggered by BGP update

of the router: *route-policy RP-A10-IP-IN permit node 10 apply community 64496:1033 additive.*

BGP maintains two versions of the Adj-RIB-In, before and after applying the defined policies. The Adj-RIB-In Post-policy message can be seen in Figure 3.5, including the added standard community value as defined by the route policy. The definition of the fields is equivalent for the Pre- and Post-policy RIBs.

Both, Adj-RIB-In Pre- and Post-policy do not indicate whether and how the route is installed in the forwarding table. As seen in Figure 2.1, the RIB that contains this information is called Loc-RIB and the relevant route monitoring message can be seen in Figure 3.6. The peer type in this BMP message is *Loc-RIB Instance Peer* as defined in [15]. There is additional information stored in the route monitoring message that indicates how the route is installed (e.g., best-external, ECMP etc.), called *path marking* according to the draft [6]. However, there is no dissector available for Wireshark yet. After the router received the BGP update message, potentially modified it and installed the route in its Loc-RIB, the BGP process exchanges the newly learned route with the RR *daisy-52*. Because *daisy-61* has a peering with the RR to advertise the routes, it also exports the associated Adj-RIB-Out (Pre- and Post-policy). Adj-RIB-Out messages are constructed analogously to Adj-RIB-In messages, which we omit here for brevity.

3.3 IPFIX in Action

In addition to the control-plane information from the previous section, we now show selected messages from the data plane that are exported after sending traffic over the newly activated interface to another endpoint for that specific VPN. We will cover the correlation between the two planes in the next chapter. As mentioned in Section 2.1.3, IPFIX packets are split into templates, option templates and data records. The template specifies the schema for the exported data. Figure 3.7 depicts a template schema that all routers in the lab use to export IPFIX data. The template has an ID of 1538 that is used to match all corresponding IPFIX data packets.

```
Version: 4
Length: 144
Type: Route Monitoring (0)
▾ Per Peer Header
    Type: RD Instance Peer (1)
  ▾ 0100 0000 = Flags: 0x40, Post-policy
      0... .... = IPv6: Not set
      .1.. .... = Post-policy: Set
      ..0. .... = AS PATH: Not set
      ...0 .... = Adj-RIB-Out: Not set
      .... 0000 = Reserved: Not set
    Peer Distinguisher: 64499:11
    Unused: 00000000000000000000000000
    Address: 192.0.2.153
    ASN: 65000
    BGP ID: 192.0.2.53
    Timestamp (sec): 1599826440
    Timestamp (msec): 0
▾ Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 78
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 50
  ▾ Path attributes
    ▸ Path Attribute - ORIGIN: IGP
    ▸ Path Attribute - AS_PATH: 65000
    ▸ Path Attribute - NEXT_HOP: 192.0.2.153
    ▸ Path Attribute - MULTI_EXIT_DISC: 0
    ▸ Path Attribute - COMMUNITIES: 64496:299 64496:1001 64497:1 64499:10 64496:1033
  ▾ Network Layer Reachability Information (NLRI)
    ▾ 203.0.113.10/32
        NLRI prefix length: 32
        NLRI prefix: 203.0.113.10
```

Figure 3.5: BGP Adj-RIB-In Post-policy of *daisy-61* peering with *daisy-53* with the Post-policy bit set in the Per Peer Header

```
Version: 4
Length: 144
Type: Route Monitoring (0)
▾ Per Peer Header
    Type: Loc-RIB Instance Peer (3)
  ▾ 1000 0000 = Flags: 0x80, Loc-RIB
      1... .... = Loc-RIB: Set
      .000 0000 = Reserved: Not set
    Peer Distinguisher: 64499:11
    Address: ::
    ASN: 65537
    BGP ID: 192.0.2.61
    Timestamp (sec): 1599826440
    Timestamp (msec): 0
▾ Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 78
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 50
  ▾ Path attributes
    ▸ Path Attribute - ORIGIN: IGP
    ▸ Path Attribute - AS_PATH: 65000
    ▸ Path Attribute - NEXT_HOP: 192.0.2.153
    ▸ Path Attribute - MULTI_EXIT_DISC: 0
    ▸ Path Attribute - COMMUNITIES: 64496:299 64496:1001 64497:1 64499:10 64496:1033
  ▸ Network Layer Reachability Information (NLRI)
```

Figure 3.6: BGP Loc-RIB of *daisy-61* with the Loc-RIB bit set in the Per Peer Header

The first three fields capture layer 3 information: IP addresses of the communicating parties as well as the IP next hop at the point of capture. Traffic information such as number of packets and number of bytes follow. The time window for which the traffic information is aggregated for is encoded in the *switched* fields (first and last). The BGP next hop field refers to the VPNv4/6 next

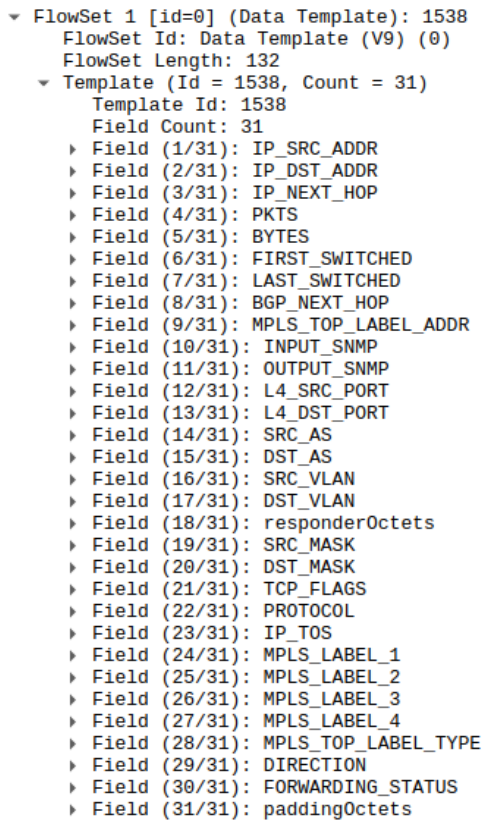


Figure 3.7: IPFIX Template Definition, exported by lab routers

hop [42]. The MPLS top label address is the loopback address of the MPLS PE router that this flow will be forwarded to. What follows are the indexes of ingress and egress interfaces, transport layer port numbers, AS numbers and VLAN IDs of the sender and the receiver. The *responderOctets* field is the total number of layer 4 payload. The source and destination masks correspond to the prefix length of the source and destination IP addresses, respectively. TCP control bits are included in case TCP is used as the transport-layer protocol. The IP TOS field refers to the type of service (TOS) field in the IPv4 header (Traffic Class field for IPv6), that prioritizes the IP packet. The MPLS label stack follows from top to bottom. The MPLS top label type identifies the control protocol that allocated the top label of the MPLS label stack which includes, among others, BGP and VPN [23]. The direction specifies whether the flow is ingress or egress at the point of capture. The last field before the padding refers to the forwarding status and any attached reason, e.g., whether the packet was forwarded or dropped [23].

To capture a flow on a PE router, we send ICMP ping messages between different endpoints of the VPN. The source and destination are *VRF-A10* with IP address *203.0.113.10* and *VRF-A20* with IP address *203.0.113.20*, respectively. As can be seen in Figure 3.1, *VRF-A10* connects to PE routers *daisy-61* and *daisy-62*. The (label switched) forwarding path between two PE routers is via the MPLS P router *daisy-52*. Figure 3.8 shows the exported IPFIX data packet from *daisy-61*. The source and destination IP addresses are listed as described. Because we use MPLS and, thus, a label switched path, the IPv4 next hop is not defined and forwarding is solely based on the MPLS labels. The top label address is the address of the MPLS next hop, i.e., the MPLS P router *daisy-52*. The BGP next hop is the VPNv4 next hop, i.e., the PE to which *VRF-A20* is

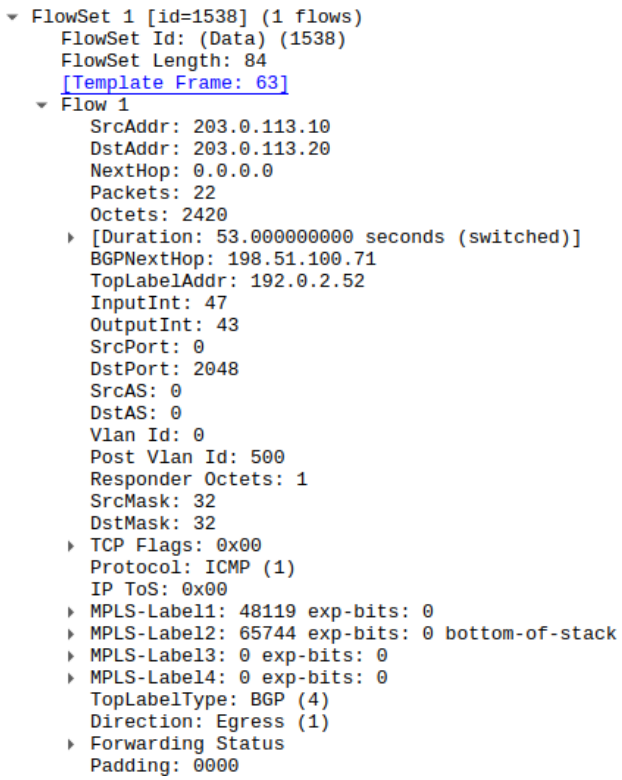


Figure 3.8: IPFIX Data Packet exported from *daisy-61*

attached to. The meaning of the ingress and egress interfaces, transport-layer port numbers, AS number, address masks and VLAN IDs are described in the template description. The TCP flags are zero because there is no TCP used in the transmission of ICMP ping packets. The IP type of service is set to zero, which corresponds to best-effort delivery. The MPLS label stack has depth two. The top label refers to the label switched path to the PE router. This label is then popped and the bottom label is required to forward the packet to the correct VRF. The control protocol that allocated the top label of the MPLS label stack is BGP. The packet was captured at the egress interface of *daisy-61* and the forwarding status is not further described.

This chapter showed what and how control- and data-plane messages in the lab environment are exported and collected. The raw format of the data is not particularly useful to infer valuable insights. To be able to correlate and visualize the data, we have to further process and structure it. The next chapter explains a design to export, collect, store and analyze this data in a more systematic way to support a diverse set of use cases in order to increase network visibility.

Chapter 4

Design

We have now seen the lab environment and what kind of information BMP and IPFIX provide in this context. Equipped with that knowledge, we now approach the problem in a more structured manner. We build an end-to-end pipeline and control every aspect from exporting the data from the network routers up to the visualization in a web front-end to guarantee scalability and extensibility requirements. To contribute towards that goal, the pipeline leverages the battle-tested big data technologies Apache Kafka [1] and Apache Druid [47] as well as the collector software pmacct [32] that is written in the low-level and fast programming language C. The following sections describe the pipeline and its components in more detail, followed by a section dedicated to the correlation of the data and an external library to directly query router information.

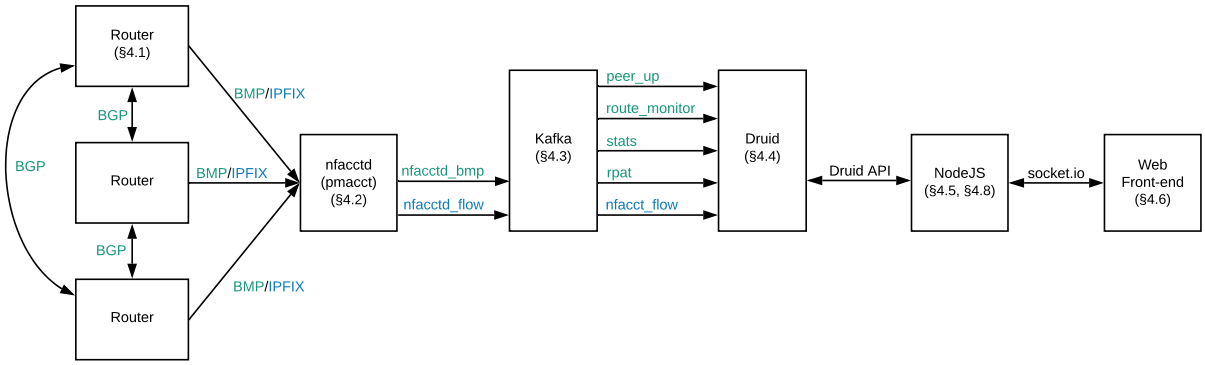


Figure 4.1: End-to-End Pipeline and data flow between the components (green = control plane, blue = forwarding plane)

4.1 Router Export

As mentioned in Chapter 3, the dominant router vendor in the lab network is Huawei, the reason for which is extensive BMP support, compared to other network vendors (cf., Section 5.3). The CE router *daisy-53*, the RR *daisy-52* and the PE routers *daisy-61/62/71/72/81/82* establish a BMP session on a predefined port with the collector *ietf-internal* (cf., Figure 3.1). The BMP configuration on the Huawei routers [22] defines which RIBs from which VPN instances to monitor. The configuration also defines other parameters, such as the time between periodical statistics dumps

and more. After a connection between the router and the collector has been established and initiation messages exchanged, the router sends the content of its RIBs according to the configuration. After that initial dump, the BMP routers encapsulate incremental BGP updates and forward them to the collector as they arrive.

4.2 Collection

The collector software is called *pmacct*. It is written in the C programming language and is a multi-purpose passive network monitoring tool. *pmacct* can account, classify, aggregate, replicate and export forwarding-plane data, e.g., IPFIX, collect and correlate control-plane data via BGP and BMP and much more [32]. To further enable aggregation and analysis of the collected data, *pmacct* supports a wide range of back ends, e.g., Apache Kafka. There are multiple daemons, that are either stand-alone, e.g., *pmbmpd* is the BMP daemon, or as a thread in a correlation process, e.g., *nfacctd*, which is used to correlate IPFIX and BMP data. To show the data format of a specific example, we revisit our data-plane traffic capture from Section 3.3. The IPFIX data packet that is exported from the router (cf., Figure 3.8) is collected with *nfacctd* and enriched with control-plane data, such as AS path and standard community values of both endpoints. Listing 4.1 displays a correlated and aggregated IPFIX capture from *daisy-53*. We explain the details on how the correlation works in Section 4.7.


```
{
  "event_type": "purge",
  "comms": "64496:1001, 64496:299, 64497:1, 64499:20",
  "ecomms": "null",
  "lcomms": "null",
  "as_path": "65000_65539_65536_65537",
  "peer_ip_src": "192.0.2.53",
  "comms_src": "64496:1001, 64496:1033, 64496:299, 64497:1, 64499:10",
  "ecomms_src": "null",
  "lcomms_src": "null",
  "as_path_src": "65000",
  "iface_in": 56,
  "iface_out": 57,
  "mpls_vpn_rd": "0:64499:23",
  "ip_src": "203.0.113.10",
  "net_src": "203.0.113.10",
  "ip_dst": "203.0.113.20",
  "net_dst": "203.0.113.20",
  "mask_src": 32,
  "mask_dst": 32,
  "port_src": 0,
  "port_dst": 0,
  "tcp_flags": "0",
  "ip_proto": "icmp",
  "tos": 0,
  "timestamp_start": "2020-10-15T09:13:06.000000+02:00",
  "timestamp_end": "2020-10-15T09:13:06.000000+02:00",
  "timestamp_arrival": "2020-10-15T09:13:42.807504+02:00",
  "forwarding_status": "0",
  "packets": 9,
  "bytes": 918,
```

```
  "writer_id": "ietfinternal/3575"
}
```

Listing 4.1: Example of collected data-plane information from *daisy-53*, enriched with control-plane data such as AS path and standard communities

4.3 Message Broker

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications [1]. Clients that interact with Kafka are split into producers, which can publish (write) events (i.e., *pmacct*) and consumers that subscribe to (read) events. Events are organized and stored in *topics*. Figure 4.1 shows control plane *topics* in green color and forwarding plane *topics* in blue color. *nfacctd_bmp* and *nfacctd_flow* are the *topics* that *pmacct* ingests all collected BMP and IPFIX data into. Because *topics* are inherently schema less, the *nfacctd_bmp* topic contains all BMP messages. These are further split into individual *topics*, each one corresponds to a specific BMP message type. The advantages of this splitting are increased performance for the storage layer. Less data has to be sorted through if we query a specific message type. And the schema is smaller and tailored to the specific message type and, thus, more efficient. The reason for choosing Kafka is its high performance, its interoperability with other software/systems and its ability to store and buffer messages, which results in the decoupling from connected producers and consumers.

4.4 Storage

Apache Druid is a widely used, column-oriented, open-source, distributed data store written in Java. Its strengths lie in quick ingestion of massive amounts of event data and to provide low-latency queries on top of it [47]. Druid has native support for consuming Kafka data. We define a schema for IPFIX and all BMP message types, as depicted in Figure 4.1. The schema for both BMP and IPFIX data is a subset of the collected and correlated data from the collector. We only choose a subset to increase scalability and because not all data is necessary for most visualization use cases. As an example, we discard TCP flags or port numbers in Listing 4.1 before we ingest them into the database. In order to support timely precise queries, the query granularity has been set to seconds. Druid offers an API that can be used to query the database externally. Additionally to Druid SQL, it supports a *native query language* with JSON over HTTP. The queries are built on the *select - from - where* pattern, commonly used with relational databases. The *select* clause defines the dimensions/columns that are to be extracted, the *from* clause defines a data source for which there is a 1:1 mapping to a Kafka *topic* and the *where* clause defines the time range as well as additional filters that can be defined in a wide variety of formats. The reason for choosing Druid is its optimization of event data. Correlation of control and forwarding plane in time form the largest share of visualization use cases. Moreover, column-oriented databases outperform row-oriented databases in case of ranged-based queries if the schema defines many columns but only a few are of interest for an average query.

4.5 NodeJS

NodeJS is a JavaScript runtime built on Google's V8 JavaScript engine which allows the code to be run outside of a browser context [14]. The main task of this component is to handle queries from

the front-end. Communication between NodeJS and the front-end is facilitated by the *socket.io* library [40] which enables bidirectional communication in real-time. The most common query from the front-end is to get control or forwarding-plane data. The NodeJS application receives a request with the desired parameters, forwards it to Druid and sends the data back in a structured form upon reception from Druid. Another type of query are route policy requests. According to [49], the *rpat* message type includes the name of the route policy as a parameter. The reason for using NodeJS is its rich ecosystem of libraries and its interoperability with front-end frameworks that are predominantly written in JavaScript. Using an additional layer can also enable potential future applications that are not necessarily web-based.

4.6 Front End

All the previous steps and components facilitate the visualization of correlated control- and data-plane information in real-time. Another use case is to allow for exploratory queries by specifying a time-range to iteratively step through the events. We implement a real-time and exploratory mode for three different control-plane visualizations: 1) Peerings, 2) route monitoring messages and 3) route policy and attribute tracing messages. Each control-plane visulaization can be correlated with forwarding-plane information. The UI is split to include filter options for both planes, which differ among the different use cases. Examples include which router and RIB to inspect, which VPN to display data from and the time range in case of an exploratory visualization. The filter options for the data plane include the standard community (which refers to VPN identifiers or VPN endpoint identifiers), the router from which the IPFIX metrics originate and the time range. The API with which the visualizations are created is built to facilitate fast prototyping of new use cases.

The visualizations are based on d3.js [4], a JavaScript library that can produce dynamic and interactive visualizations in web-browsers. The control-plane view consists of a network graph of nodes and links with the semantics of the entities tailored to each use case (cf., Section 5.1). The data-plane view is a coordinate system with the x-axis specifying the timestamp of the event and the y-axis the amount of traffic for the configuration specified by the filters. Because the forwarding-plane data is state-less, e.g., only shows aggregated data in periodic time intervals, there is an additional option for the real-time visualization to define the time range (in minutes) after which the displayed data is to be discarded. This is not desirable for the control-plane data because historical data about nodes and links can still reflect the actual topology and, therefore, should not be discarded after a certain amount of time.

4.7 Correlation

The collected data on the control and data plane have an informational value on its own. But the full potential can only be realized if that information is correlated. We focus on two different correlations: 1) VPN level and 2) time. VPN level correlation refers to the ability to map the data-plane traffic to a VPN and all of its associated attributes. There are multiple ways to achieve this correlation. The simplest way is to leverage the information element *mplsVpnRouteDistinguisher*, defined by IANA [23]. This value encodes the VPN RD. However, this field does not have widespread vendor support at this point and is also not present in the lab network. As a workaround, an explicit mapping file can be created, which maps an (ingress or egress) input interface of a router to its corresponding RD. The creation of this mapping file can be automated by querying the relevant management information base (MIB) with the SNMP protocol [7]. This correlation enables *pmacct*

to enrich the forwarding plane with control-plane data, e.g., source/destination communities that encode VPN information. This in turn enables forwarding-plane queries that are targeted at a specific VPN.

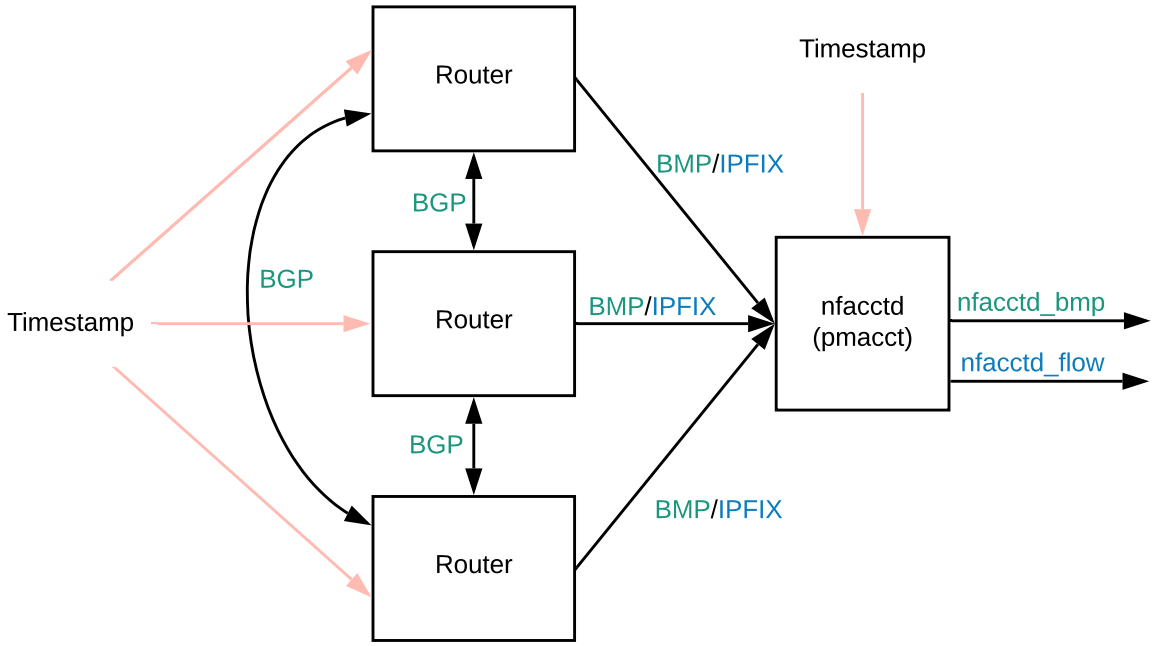


Figure 4.2: The routers and the collector pmacct insert timestamps that are used for correlation

Another correlation is with regards to time. If we want to make cause and effect visible across both planes, we need additional information on when the events occurred. Figure 4.2 shows the first two components of the pipeline (routers and the collector *nfacctd*). These are the only two components in which timestamps are created. The BMP standard [43] defines two granularities for the timestamp fields: seconds and microseconds. It is part of the per-peer header and refers to the time when an update message was received. However, the recording of a timestamp is not mandatory by the standard, but it must be set to zero if the time is not available. If we assume normal operation without resource constraints of the BMP process, BMP messages are directly forwarded to the BMP server upon the reception of BGP messages.

For the data plane, specifically the IPFIX protocol, this is different. Because of the large amount of flow data, the incoming and outgoing messages are sampled and aggregated over a specified time period, during which it is cached. At the end of the time period, the data is forwarded to the collector. The duration of the time window is a variable that can be configured. IPFIX also offers information elements that record a timestamp of the first and last inspected packet during the time window. At the collector, another timestamp is inserted for all incoming packets. Depending on the router capabilities and configurations, the additional timestamp inserted at a collector guarantees at least one timestamp for the data. In order to use the timestamps that are inserted by hundreds or thousands of network devices, it is important that all clocks from which timestamps are derived are synchronized. If this cannot be guaranteed, the collector timestamp provides a remedy, albeit less accurate.

4.8 Route Policy Query

At the current stage of BMP, it is possible to export route policy and attribute tracing messages [49] that are exported in the connection of a triggered route policy. The message contains information on whether the policy was matched, permitted or denied, and indicates a potential route modification as a result of applying this policy. It also includes the policy class and other information. However, the message does not contain the actual route policy, only its name. To bridge that gap and make the content of the route policy available, we use the Netconf YANG modeling language [3] to query the router directly. The NodeJS application makes use of a JavaScript library that implements the Netconf protocol. To query the route policy, we tunnel a *get-config* RPC call through SSH to the router, which in turn responds with the requested route policy.

Chapter 5

Evaluation

This chapter presents different use cases and visualizations of the system whose design was explained in the previous chapter. In order to test the scalability in a real-world setting, we moved away from the lab and tested the system with production data. Furthermore, we evaluate and compare different vendors and the extent to which they support BMP and IPFIX standards and drafts.

5.1 Visualization

In the following subsections we show different visualizations and explain their use cases. The following BMP messages are supported: peer up, route monitoring and route policy and attribute tracing. This list is non-exhaustive and as mentioned in Section 4.6, new visualization can easily be created by using the framework API. For every BMP message, it is possible to combine the control-plane visualization with the data-plane graph. Every visualization has two modes: exploration and real-time.

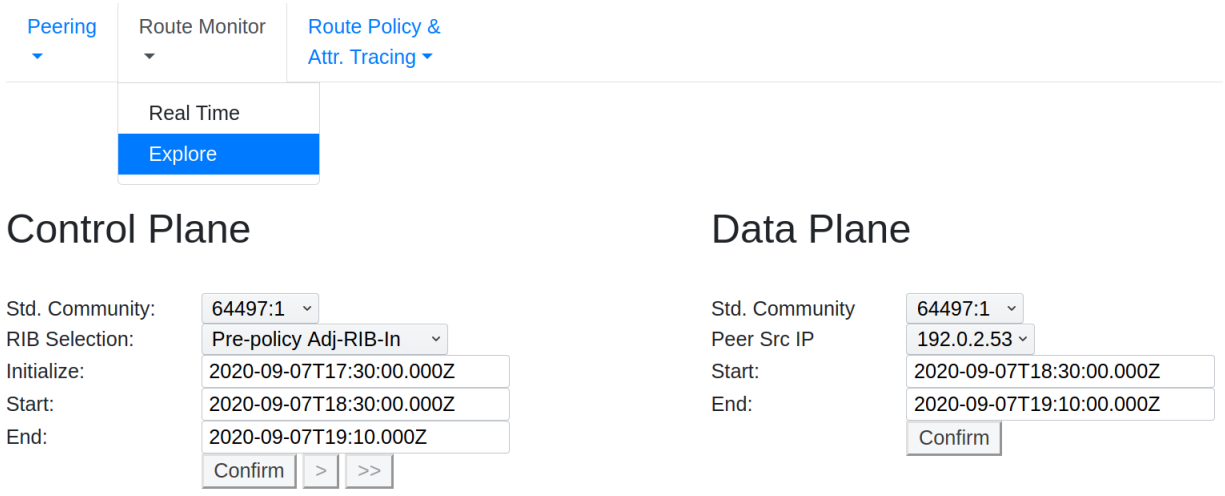


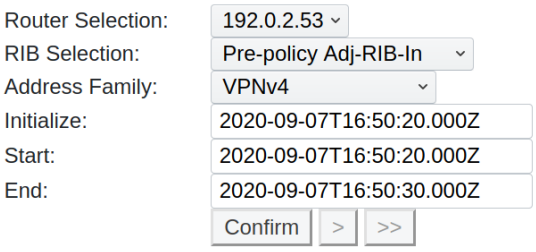
Figure 5.1: Web front end: Visualization selection, control-plane and data-plane filters

Figure 5.1 shows the user interface of the system with the route monitoring visualization selected in exploratory mode on the left side and the data plane on the right side. The exploration visualizations are defined with three timestamps for the control plane, an initialization timestamp as well as a start and end timestamp. The time period between the first two is used to build up the

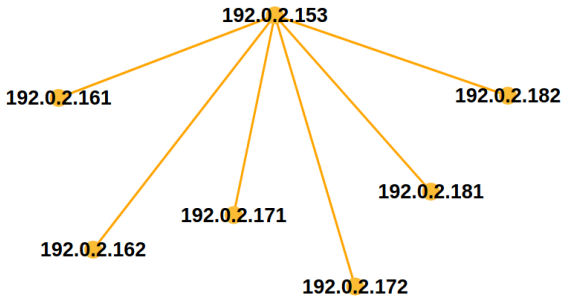
graph state because in most use cases it is desirable to see changes of an existing graph. If that is not desired, the initialization and the start timestamp can be set to the same value. The user can then iteratively click through single or multiple messages that span the time window between the start and end timestamp and the state of the graph is updated with the new information. There only exist two timestamps for the data plane: start and end. No initialization is necessary, as the graph is stateless. For the real-time mode, the last timestamp is omitted for both planes.

5.1.1 Peering

BMP contains a message type for new peering sessions, which are wrapped BGP open messages [41]. Figure 5.2a shows the different filtering options for the visualization of peering sessions. A RIB for a specific router can be selected which results in a star topology as seen in Figure 5.2b. An additional filter selects a specific address family, e.g., IPv4/6 or VPNv4/6. RFC 7854 [43] defines different peer types that are used to infer the address family. The center of the star topology is the local IP of the selected router and the leaf nodes represent the peer IP address of the router that the selected router peers with. This visualization gives a first insight into the topology of the network across different address families.



(a) Peer up filter option to select router, RIB, address family and time window

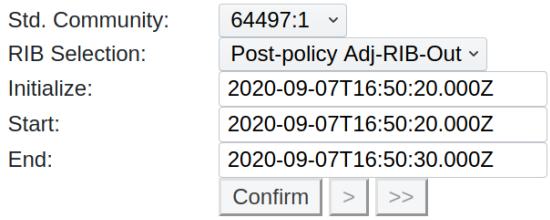


(b) Peer up graph for the selected properties

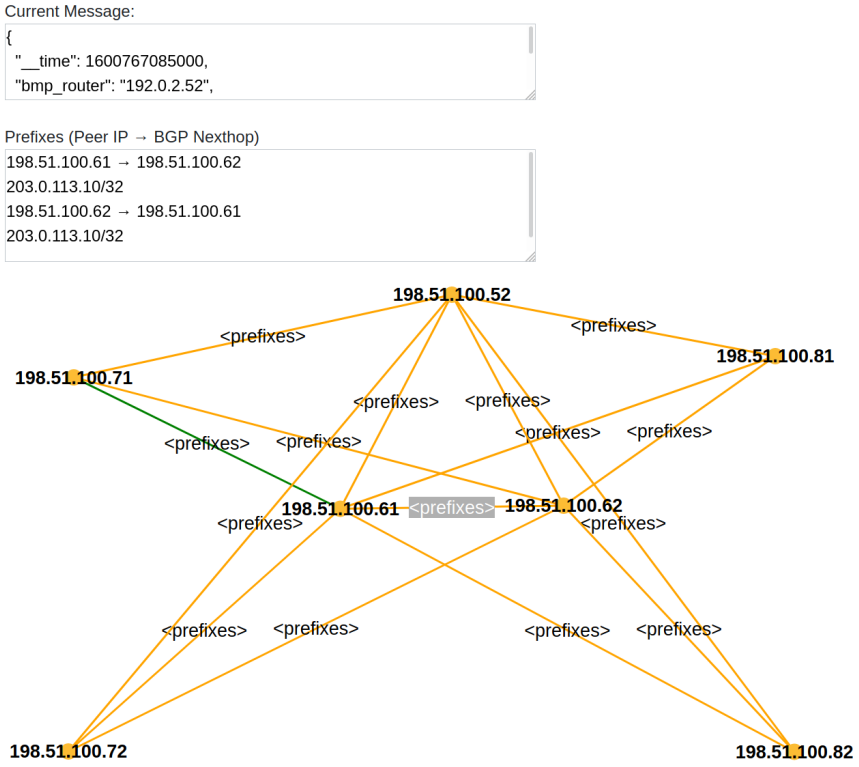
Figure 5.2: BMP Peer up message type to visualize BGP peerings

5.1.2 Route Monitor

BMP route monitoring messages carry BGP update messages, which transfer routing information between BGP peers [41]. BGP update message can either advertise (i.e., update) or withdraw a route. These messages contain several attributes for a specific prefix, such as AS path, BGP nexthop, community values and other information that is exported with BMP. Figure 5.3a shows the available filter options. Additionally to the timestamps and the RIB, there is a filter for standard community values. This filter enables a user to see traffic for a specific VPN identifier or VPN endpoint identifier. As mentioned in Chapter 3, there exists a predefined mapping between standard community values and VPN identifier and VPN endpoint identifiers. After pressing the *Confirm* button, the graph is built with all messages that occurred between the first two timestamps. After that, the buttons > (forward) and >> (fast-forward) can be pressed to incrementally see changes in the graph. The links are color-coded: green refers to update messages, red refers to withdraw messages and orange indicates that there was no link state change in the last message. The graph is directed and points from the peer IP address towards the BGP nexthop that was carried in the BGP update message for a given IP prefix. The link label is marked with $<prefixes>$. By clicking



(a) Route monitor filter option to select VPN (std. community), RIB and time window



(b) Route monitor graph that displays BGP nexthop for prefixes

Figure 5.3: BMP Route monitoring message type to visualize BGP update messages

on it, we can see a list of prefixes, potentially from both directions. A route monitor carries more information than what is displayed in the graph. Therefore, the raw data of the current BMP message is displayed in a separate text field. This information includes, additionally to the BGP nexthop, the address family, the AS path, the standard and extended communities and much more. An exhaustive list can be found in the documentation of pmacct [32].

We return to our example from Section 3.2 where we activated the IP address *203.0.113.10/32* that belongs to VPN *A* which is mapped to the community value *64497:1*. Figure 5.3b shows the resulting graph of the Post-policy Adj-RIB-Out after enabling the IP address. We can see that the PE routers *daisy61/62* for that VPN (cf., Figure 3.1) have a link to all other routers, because they are the BGP nexthop for the activated IP address. The same is true for the RR *daisy-52* to propagate the route through the network. The link between the PE routers is highlighted in grey and the associated BGP nexthops for that prefix point in both directions due to the configured BGP mul-

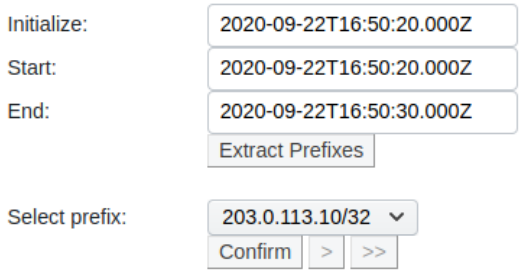
tipath, which can be seen in the textfield *Prefixes*. This textfield contains a list of all prefixes that are associated with that particular pair of peer IP address and BGP nexthop. Because BGP update messages are the proximate cause by which packets are routed and rerouted through networks, the monitoring of these messages can significantly help in troubleshooting, anomaly detection and root cause analysis [5, 17].

5.1.3 Route Policy

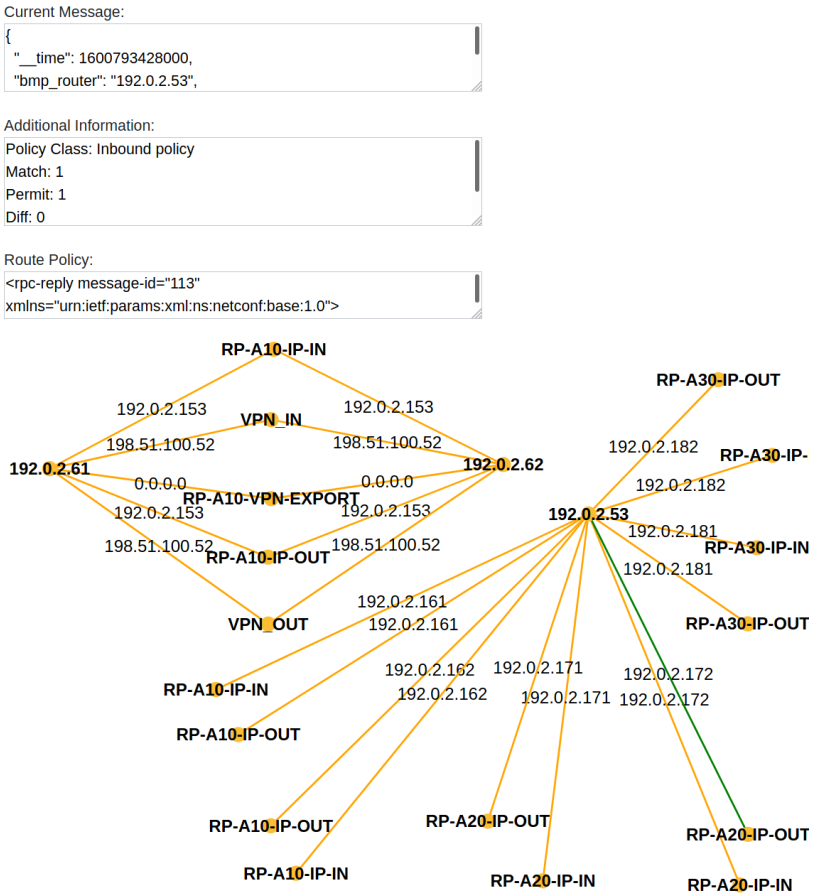
As shown in Figure 2.1, the multiple RIBs in a BGP process differ because route policies can be defined to allow, deny or modify routes. While the RIBs allow the visibility of the full network state, it does not explain how route policies affect route propagation and BGP attribute changes. As of October 2020 still in draft status, the document from [49] describes a method to use BMP to record how BGP routes are (not) changed under the process of route policies. A BMP route policy message is associated with a prefix. Prefixes are dynamic and new prefixes can be advertised as shown in the route monitoring visualization. Thus, it is not desirable to hardcode a list of prefixes into the software. Instead, a first phase consists of extracting all prefixes of all route policy messages for the predefined time window, as shown in Figure 5.4a. A specific prefix can then be selected for which the graph is built. As with the previous visualizations, the graph is built for the time window between the first and the second timestamp. Further state changes are incrementally applied by clicking the fast and fast-forward buttons. The resulting graph consists of multiple bipartite sub-graphs with BMP routers as one set of nodes and the route policy name as the other set of nodes. The links are labeled with the peer IP address of the router the BMP router peers with (the IP address *0.0.0.0* is used for route policies that are not related to a peering). There are multiple text fields that display further information. As in the other visualizations, the link in the graph that corresponds to the current message is colored in green and the message is displayed in a text field in its entirety, as defined in the pmacct documentation [32]. By clicking on the route policy, additional information is shown: The policy class, e.g., Inbound policy, refers to the policy category, of which a full list can be seen in [49] as well as three bits: The *Match* bit, that informs whether the policy was matched at least once, the *Permit* bit (0 = Deny, 1 = Permit) that states whether the route that is carried in the message is permitted or denied, and the *Diff* bit which is set if the route was changed by the policy. Additionally, there is a text field that shows the actual content of the route policy. This is accomplished by tunneling a Netconf YANG *get-config* RPC call through SSH directly to the router (cf., Section 4.8).

Figure 5.4 shows the resulting route policy messages that were triggered after enabling the IP address *203.0.113.10* on the CE router (cf., Figure 5.4a). For clarity, the visualization in Figure 5.4b only includes the CE router and both PE routers that are associated with the same VPN the IP address refers to. The PE routers trigger multiple route policies. Let us for example consider the policy *RP-A10-IP-IN* that both PE routers trigger over a peering with the CE router. As mentioned in Section 3.2, this policy for this route is matched, permitted and the route differs before and after applying the policy, which is due to the addition of the community value *64496:1033*.

Access to the route policies as well as additional information such as whether the policy was matched and permitted and how a policy modified a route greatly assists in discovering incorrect route policies. Examples include wrongly set community values and incorrect relative order of policies. These may result in incorrect route attribute modification, best route selection mistake, or route distribution mistake. The combined information and correlation of route monitoring and route policies allows to identify unexpected route changes and the responsible route policies. Apart



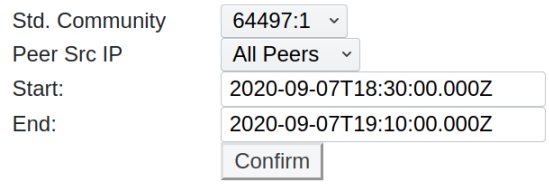
(a) Route policy and attr. tracing filter to extract and select prefix for a specified time window



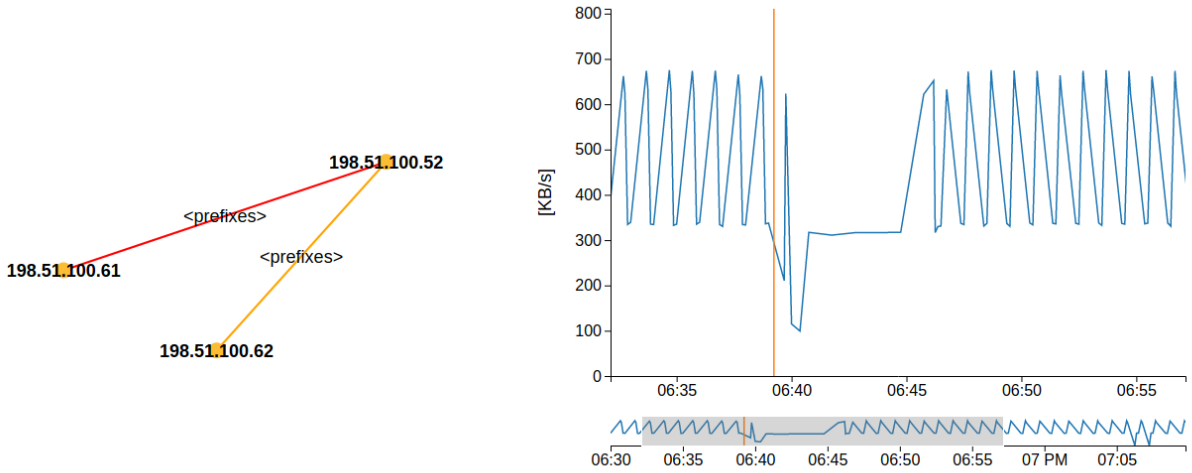
(b) Route policy and attr. tracing graph that displays the BMP routers and the triggered route policies

Figure 5.4: BMP Route policy and attribute tracing message to visualize the route policies and their and the context in which they are triggered

from the policy validation application, the route policies data can also be analyzed to discover the route propagation path within the network. With the route's inbound and outbound event collected from each router, an operator is able to find the propagation path hop by hop. The identified path is helpful for operators to better understand its network, and thus benefiting both network troubleshooting and network planning [49].



(a) Data-plane filter to select VPN (std. community) and peer(s) that exported the IPFIX messages for a specified time window



(b) Control- (left) and data-plane (right) graph to display correlation between events, indicated by the vertical bar on the right side: Effect of data-plane traffic after removing an interface.

Figure 5.5: Correlation visualization between control- (BMP) and data-plane (IPFIX)

5.1.4 Control-/Data-Plane Correlation

To add more context to the BMP messages, we add the data plane as an additional source of information. As described in Section 4.2, the collector in the pipeline runs two daemons to collect both control (BMP) and data-plane (IPFIX) information. It maintains internal datastructures to correlate the information and enriches the data plane with control-plane data. For example, the community values that correspond to a specific VPN are added to the IPFIX data, e.g., the VPN identifier *64497:1* for the VPN *A*. This enables the filtering for data-plane traffic per VPN. The visualization that combines the control and the data plane further correlates the messages across time which allows an operator to see what causes in the control plane has which impact on the data plane. Figure 5.5a displays the filter options for the data plane. The option for standard communities filters by VPN identifiers or VPN endpoint identifiers. We can either display data-plane information from all routers or select a specific router to further filter the amount of data to be processed. In contrast to the control-plane information, there is no initial timestamp due to the statelessness of the forwarding-plane data.

We activated a traffic generator in the lab that produces uniform traffic between a predefined pair of endpoints. We then deactivated the IP address of an interface over which traffic flows in VPN *A* (std. community: *64997:1*) and observed the control plane and the data plane simultaneously, as depicted in Figure 5.5b. The left side of the figure are route monitoring messages for

Adj-RIB-In. The color of the links are red which means a route withdrawal was communicated over this peering. To correlate in time, the right side of the picture shows a vertical orange line that corresponds to the timestamp of the last received BMP message. The timing of the withdraw message coincides with a change in the traffic pattern, because there is a causation between the control-plane event (route withdrawal) and the data plane (change of traffic pattern) that can be made visible with this visualization.

After six minutes we reactivated the same IP address, the result of which can be seen in Figure 5.6. The green color of the links on the left side of the figure indicate BGP update messages that triggered BMP Adj-RIB-In route monitoring messages. The right side of the figure show the reactivation of the traffic pattern from before the IP address was deactivated, caused by enabling the same IP address again.

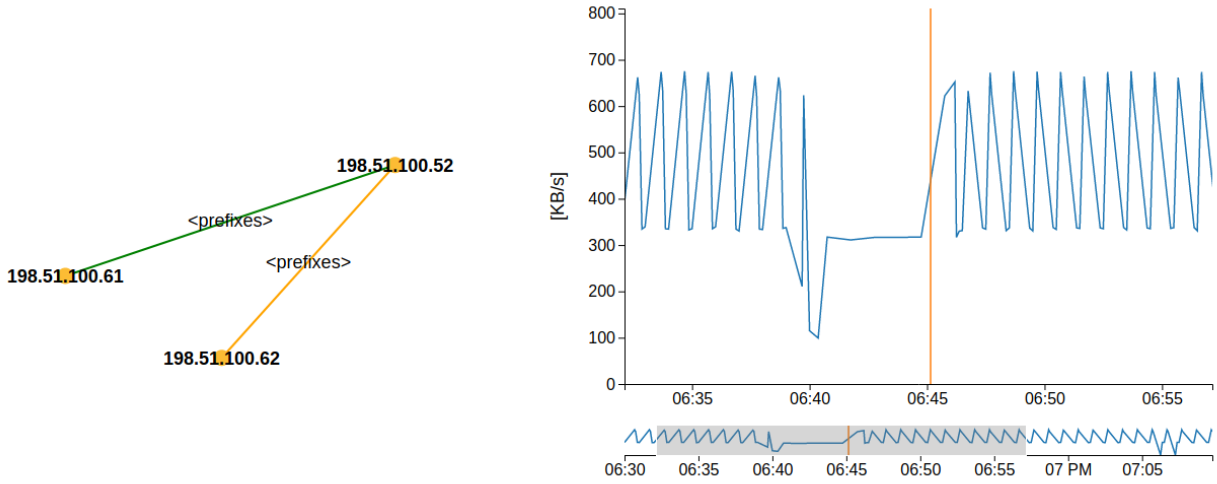


Figure 5.6: Control- (left) and data-plane (right) graph to display correlation between events, indicated by the vertical bar on the right side: Effect of data-plane traffic after enabling an interface

5.2 Production Environment

The lab network consists of a limited number of devices that generate a limited amount of control- and data-plane data. In order to increase the load and test the scalability of the system, we performed a large-scale test with production data. We ingested one hour worth of control- and data-plane traffic into the time-series database Druid. The total amount of data is close to 100GB in size with 100'000 messages per minute for control-plane data and 500'000 messages per minute for IPFIX exports. Indexing and ingestion of the data took approximately 15 minutes. Because the newer standard RFC 8671 [16] that enables Adj-RIB-Out access and the subsequent draft that enabled Loc-RIB access [15] are not yet widely deployed, the control plane included only Adj-RIB-In data in accordance to RFC 7854 [43]. Also, only BMP route monitoring messages are recorded. The control-plane data was exported by four BMP routers, each peering with 12 routers, carrying data from over 2000 different VPN identifiers and almost 3000 VPN endpoint identifiers. The data-plane traffic carried exports from more than 2500 routers. IPFIX data was exported from more than 700 VPN identifiers and an equal amount of VPN endpoint identifiers that were active

during that time window. The filter options are analogous to Section 5.1.2 for the route monitoring message and Section 5.1.4 for the IPFIX data. Despite the large amount of data, the fact that both control- and data-plane information were split across hundreds or thousands of VPNs, the data could be aggressively filtered and all possible queries resulted in a data transfer that corresponded to a small subset of the overall data. The response times did not exceed a few seconds for any query combination. To reach subsecond response times, the time window can be adjusted to a smaller range.

5.3 Vendor Comparison

Large-scale networks typically consist of multiple vendors. To support use cases on a network-wide scale, it is important that networking devices support equal capabilities. Support claims by the vendors must be carefully validated to verify whether a standard is fully or partially implemented. We updated our lab network environment and added multiple vendors in order to test their BMP capabilities. Figure 5.7 shows the topology of the updated lab. Changes relative to the original lab topology (cf., Figure 3.1) are the addition of two Cisco devices *daisy-54* and *daisy-55*: CSR1000V that runs the IOS XE operating system and XRv9000 with the IOS XR operating system. The Juniper MX5, *daisy-51*, was already part of the original lab topology. The device is now operational and exports BMP messages. As a last addition, we included the software routing suite FRRouting [30] as *daisy-56*.

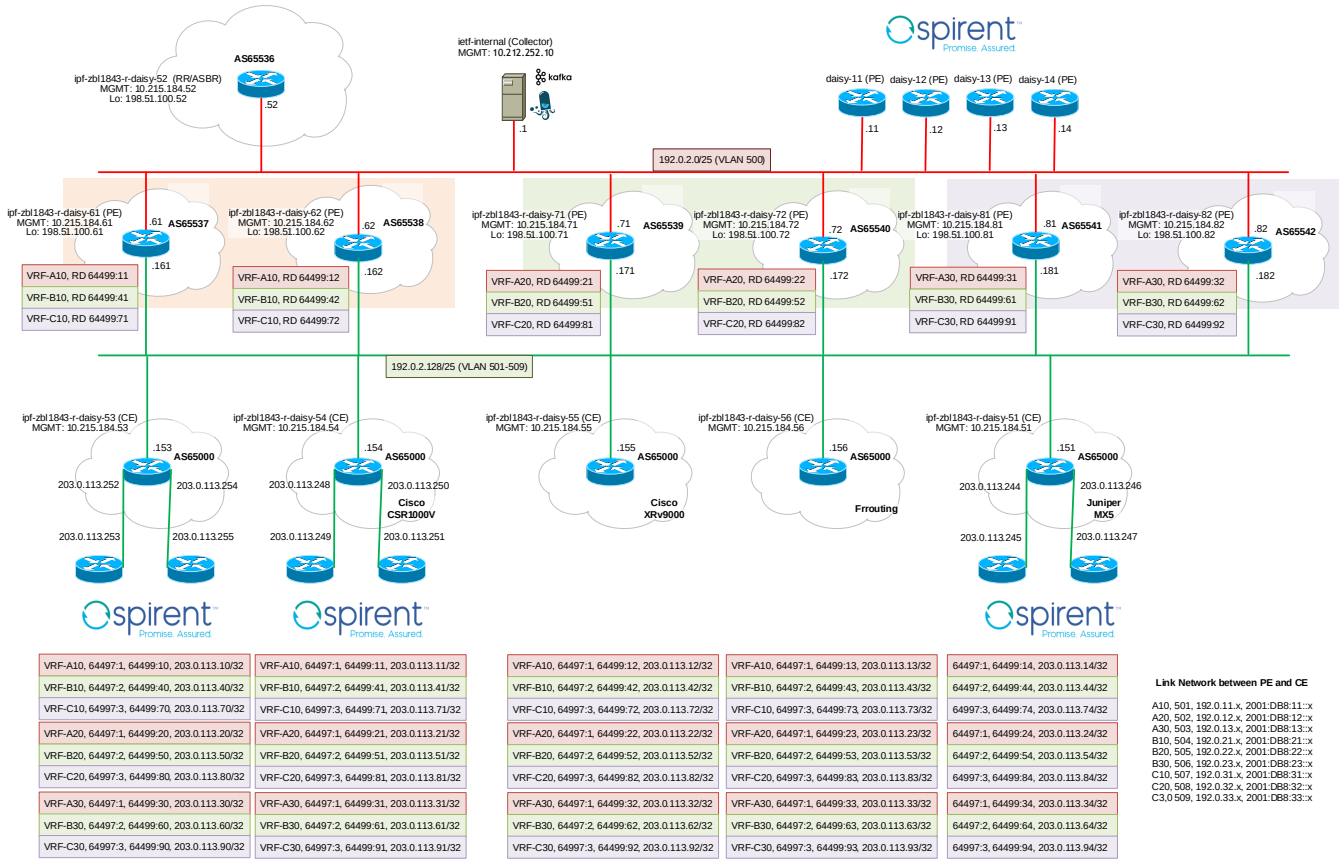


Figure 5.7: Updated MPLS/BGP IP VPN Lab network topology with additional vendor support

In order to test all devices in the context of an MPLS/BGP IP Virtual Private Networks Inter-AS Option C network, we incorporated them as CE routers with the respective VRF definitions, according to the already defined PE configurations. We chose a subset of BMP standards and drafts that were discussed in the context of this thesis and because the use cases they facilitate are dependent on one another. For example, the path marking draft [6] provides additional path status information for BMP route monitoring messages. But route monitoring messages are always in the context of a RIB, which we further differentiate in this comparison. Before we look at the vendor support for the standards and drafts, we explore the enabled use cases with an increasing level of BMP support.

The original standard RFC 7854 [43] enables support for pre- and post-policy Adj-RIB-In. Compared to traditional techniques to increase visibility into the internal state of the BGP process (cf., Section 2.2 Related Work), this standard enables a view of protocol updates that a router is *receiving*. What is not available with only access to the Adj-RIB-In, are protocol updates that the router is *sending*. For example, if a service provider desires to see what routes are advertised to customers in a scalable way, access to the Adj-RIB-Out is necessary. This additional capability is added in the updated standard RFC 8671 [16].

With both Adj-RIB-In and Adj-RIB-Out, we see incoming and outgoing BGP updates. However, Adj-RIB-In Post-policy may still contain hundreds of thousands of routes. Only a tiny subset may be installed in the Loc-RIB as a result of the BGP *Decision Process*. There is also an efficiency argument to be made for Loc-RIB access [15]: Frequent BGP update messages among peers without resulting Loc-RIB changes are common. Efficient troubleshooting, e.g., to resolve performance issues, can be achieved by only inspecting the Loc-RIB, which has a direct impact on the forwarding state of a router. Another reason Loc-RIB access is beneficial is the ability to correlate received VPNv4/6 (via Adj-RIB-In) routes with the forwarding plane. Access to Adj-RIB-In is not sufficient, because the VPNv4/6 route is learned with an RD that is not local to the router and, thus, no correlation is possible.

	Adj-RIB-In [43]	Adj-RIB-Out [16]	Loc-RIB [15]	Route Policy and Attr. Tracing [49]	Path Marking [6]
Huawei	✓	✓	✓	✓	✓
Cisco XRv9000	✓	✗	✗	✗	✗
Cisco CSR1000V	✓	✗	✗	✗	✗
Juniper	✓[1]	✓[1]	✓[1]	✗	✗
FRRouting	✓	✗	✗	✗	✗

Table 5.1: Feature Matrix that cross-references vendors to BMP standards/drafts

Additional access to the Loc-RIB allows an operator to see *which* routes are installed, but not *how* the routes are used (e.g., ECMP, best-external etc.). This is achieved with the support of the path marking draft [6]. Another valuable source that increases visibility into the internal state of the BGP process is information about triggered route policies [49]. This enables access to information on route propagation, attribute changes and it improves testing of new route policies.

[1] Only non-VRF RIB entries are supported.

The differences among the vendors in terms of the described BMP capabilities can be seen in Table 5.1. Huawei is the most advanced in terms of BMP support, which is documented [22] and unofficially tested with early access to firmware updates. Both RFC standards [43] and [16] are implemented as well as all drafts: Loc-RIB, route policy and attribute tracing and path marking [15, 49, 6]. Both Cisco devices as well as FRRouting only support Adj-RIB-In [9, 10, 31]. Juniper supports all RIBs [26, 25]. However, this is not true for the type of MPLS VPN networks we looked at in this work, because they cannot distinguish different VRFs and do not support the VPNv4/6 address families.

Chapter 6

Outlook

We have seen the potential of BMP and its integration into a system that is able to correlate the monitored control-plane information with the forwarding plane in real-time. In this chapter we take a more general view and look at other current progress in the area of BMP and identify potential future work.

6.1 Current State and Future of BMP

We have covered several standards and drafts of BMP in this thesis. This included the original standard that supported Adj-RIB-In. Followed by the updated standard that included Adj-RIB-Out support. The last RIB that is missing for full coverage is the Loc-RIB, which is still in draft status. Other documents that we covered are support for path marking, which indicates how a route is installed in the RIB, e.g., primary, ECMP etc. Another draft is about route policies and attribute tracing. This draft offers insights into how route policies affect route propagation and modify BGP attributes.

One of the big differences between BMP and other control-plane monitoring mechanisms is its focus on BGP. BMP messages are essentially encapsulated BGP messages with additional peering information. For example, BMP route monitoring messages are BGP update messages with an additional per-peer header. This fact makes BMP very efficient in exporting control-plane information compared to more gerneralized approaches such as a BGP YANG model [24].

In addition to the standards and drafts we covered in the thesis, there is more current work that aims to bring visibility to all aspects of the BGP process in an efficient manner. The document in [34] adds general TLV support. TLV stands for *Type-Length-Value* and is the data format that BGP uses to encode optional data. It consists of a type field, which is a numerical encoding defined by a standard, followed by its length and the actual data itself. Most BGP message types allow for optional trailing data in the TLV format. With this document, a way to support additional TLV data, regardless of the message type, is introduced.

Another draft [33] adds support for enterprise-specific TLVs. As noted, the type field in the TLV is a numerical representation, governed by the standard body IANA. This draft enables vendors to define vendor-specific TLVs, marked with an enterprise bit (E-bit).

BMP speaking routers in large-scale networks that enable pre- and post-policy monitoring on Adj-RIB-In and Adj-RIB-Out and additionally support the drafts we covered, produce large amounts of data that can cause a significant load on the TCP connection between the BMP speaking

devices. With the trend of monitoring and transmitting more data, BMP heads towards an I/O bound protocol. One mechanism, defined in this document [45] describes a compression mechanism to reduce the size of the transmitted data. Inevitably, this causes more load on the CPU to compress/decompress the data, which might or might not be worth it, depending on the available resources. To advertise the capability for compression, a compression information TLV is sent as an additional TLV field in the BMP initiation message (cf., Section 2.1.2).

From a security perspective, a draft [18] was introduced to enable real-time route leak detection with BMP. Route leaks are defined as "propagation of routing announcements beyond their intended scope", which means that an announcement of a BGP route from one AS to another is in violation of the intended policies of the receiver, the sender, and/or one of the ASes along the preceding AS path [44]. The draft provides a security scheme that is suitable for ISPs and can detect route leaks on the prefix level.

Because our system is designed to quickly enable new visualization use cases, we can make use of the additional information that results from the mentioned drafts in this section without difficulties. Depending on the implementation in the collector software and the structural change of BMP, we might need to modify the database schemas and the message broker configuration.

6.2 Control-Plane Delay & Loss

With the wider adoption and reliance on BMP as a control-plane monitoring protocol, questions regarding the reliability of the protocol and its implementations become more important. Delayed, lost and duplicated messages can make certain use cases impossible. The problem is exacerbated if a pipeline is employed with multiple components that each can cause disruptions in the system, whether they are caused by overflown buffers, lost messages in transit or by other means. We will first consider the first step of the pipeline: The router and the transmission to the collector.

To illustrate the problem, we consider the following use cases that are made significantly harder or impossible without reliability guarantees.

- **Peer Migration**: We want to migrate all routes from one peer to another, one router can send a BGP *Route Refresh* before and after the peer migration, which causes a re-advertisement of the prefixes in the form of BGP update messages, which in turn causes BMP to send route monitoring messages. If we rely on BMP, the current protocol is not capable to detect delay or loss, which leads to inconsistent routing tables.

- **Route Policy Testing**: We want to apply and test a new route policy on a specific peering. To do that, we install the route policy on that peering and send a BGP *Route Refresh* which causes the route policy to be applied on all re-advertised prefixes.. By comparing the routes before and after the policy was installed, we can test whether the route policy produced the desired changes. However, without a consistent view, potentially caused by lost messages, no statement can be made about the effect of the route policy.

- **Closed Loop & Zero Touch Operations**: While inconsistencies can be detected manually in some cases, closed loop & zero touch operations require a higher degree of reliability to successfully reason without human intervention.

The first two use cases assume the ability to re-advertise all routes without restarting the peering session. One such possibility was introduced with the BGP *Route Refresh* capability [8]. To further

improve the reliability, another BGP capability, *Enhanced Route Refresh*, was introduced in [39]. This standard adds markers to the beginning and ending of a route refresh. Combined with TCP as the transport protocol, the resulting guarantees are loss prevention and delay detection. BMP lacks both of those mechanisms.

It is logically impossible to guarantee loss prevention for every BMP message with finite resources. But state compression and buffer back pressure mechanisms can be employed to guarantee eventual (but possibly delayed) route updates. Once the messages are transmitted, TCP as the transport protocol for BMP, prevents loss in transit. It remains future work to investigate if and how protocol refinements are to be undertaken to incorporate these considerations.

If we assume a pipeline similar to Figure 4.1, we not only need to guarantee delivery from the router, but also successful reception on the collector, reliable forwarding to the message broker etc. Simply put: we need end-to-end reliability. One possibility is to use sequence numbers and timestamps for each component in the end-to-end pipeline. The question of if, where and how they can be used to increase reliability remains future work.

Chapter 7

Summary

As modern large-scale networks grow more and more complex to satisfy the growing need for diverse use cases, it becomes more challenging and important to achieve network visibility. As we have seen in this work, the protocols BMP to monitor control-plane and IPFIX to monitor data-plane activity prove to be very helpful to gain a degree of network insight that is essential to assist in root cause analysis and performance monitoring.

We first provided a context with the requisite technologies and knowledge that is assumed for the understanding of the thesis. This included specific aspects of BGP that are important to understand the full range of the capabilities of BMP. The BMP protocol is introduced in detail, starting with the first standard and then outlining the evolution and extensions up to this point. In addition to the monitoring of the control plane, we included the IPFIX protocol to track and monitor flows. To highlight the practical aspects of the monitoring protocols, we used a lab environment as a reference point throughout the thesis. The lab is a BGP/MPLS IP Virtual Private Networks Inter-AS Option C, which presents a challenging networking environment common in the service provider industry. To illustrate the introduced concepts, we showcased specific actions that triggered BMP and IPFIX messages and explained them in detail. The main part of the thesis consisted of building a system that handles the export, collection, storage and real-time visualization of monitoring data for this type of network environment. We argued for pipeline component choices that fulfill scalability requirements of hundreds of thousands of messages per minute as well as the requirement for control- and data-plane correlation across multiple dimensions.

We evaluated the system and showed different existing use cases that correspond to different BMP message types and visualized correlated information after filtering along multiple dimensions. We then tested the system in production and established its applicability in a large-scale environment with hundreds of thousands of messages per minute. Because of the differences in vendor support for BMP, we evaluated different vendors in terms of which standards and drafts they support and compared the different use cases that can be accomplished with the varying degree of support.

To conclude, we assessed the current state of BMP, the ongoing drafts and the longer term view of its vision. We also identified potential future work by investigating the lacking reliability guarantees of the BMP protocol. These guarantees are necessary to support critical use cases in the areas of troubleshooting and automation.

Bibliography

[1] Apache Software Foundation. Apache Kafka. https://kafka.apache.org/, January 2011. Accessed: October, 2020.

[2] Bates, T., Chandra, R., Katz, D., and Rekhter, Y. Multiprotocol Extensions for BGP-4. RFC 4760, January 2007.

[3] Bjorklund, M. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, October 2010.

[4] Bostock, M. D3.js - data-driven documents. http://d3js.org/, February 2011. Accessed: October, 2020.

[5] Campisano, A., Cittadini, L., Di Battista, G., Refice, T., and Sasso, C. Tracking back the root cause of a path change in interdomain routing. In *IEEE Network Operations and Management Symposium* (April 2008).

[6] Cardona, C., Lucente, P., Francois, P., Gu, Y., and Graf, T. BMP Extension for Path Status TLV. Internet Draft, https://tools.ietf.org/html/draft-cppy-grow-bmp-path-marking-tlv-06, September 2020. Accessed: October, 2020.

[7] Case, J., Fedor, M., Schoffstall, M., and Davin, J. A Simple Network Management Protocol (SNMP). RFC 1157, May 1990.

[8] Chen, E. Route Refresh Capability for BGP-4. RFC 2918, September 2000.

[9] Cisco. IP Routing: BGP Configuration Guide, Cisco IOS XE Release 3S. Chapter: BGP Monitoring Protocol. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/xe-3s/irg-xe-3s-book/bgp-monitoring-protocol.html, January 2018. Accessed: October, 2020.

[10] Cisco. Routing Configuration Guide for Cisco ASR 9000 Series Routers, IOS XR Release 6.2.x. Overview of BGP Monitoring Protocol. https://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k-r6-2/routing/configuration/guide/b-routing-cg-asr9000-62x/b-routing-cg-asr9000-62x_chapter_010.html, September 2020. Accessed: October, 2020.

[11] Claise, B. Cisco Systems NetFlow Services Export Version 9. RFC 3954, October 2004.

[12] Claise, B., Trammell, B., and Aitken, P. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, September 2013.

[13] Combs, G. Wireshark. https://www.wireshark.org/, 1998. Accessed: October, 2020.

[14] Dahl, R. NodeJS. `https://nodejs.org/`, May 2009. Accessed: October, 2020.

[15] Evens, T., Bayraktar, S., Bhardwaj, M., and Lucente, P. Support for Local RIB in BGP Monitoring Protocol (BMP). Internet Draft, `https://tools.ietf.org/html/draft-ietf-grow-bmp-local-rib-07`, May 2020. Accessed: October, 2020.

[16] Evens, T., Bayraktar, S., Lucente, P., Mi, P., and Zhuang, S. Support for Adj-RIB-Out in the BGP Monitoring Protocol (BMP). RFC 8671, November 2019.

[17] Feldmann, A., Maennel, O., Mao, Z. M., Berger, A., and Maggs, B. Locating internet routing instabilities. *SIGCOMM Computer Communication Revieew* (August 2004).

[18] Gu, Y., Chen, H., Ma, D., and Zhuang, S. BMP for BGP Route Leak Detection. Internet Draft, `https://tools.ietf.org/html/draft-gu-grow-bmp-route-leak-detection-04`, September 2020. Accessed: October, 2020.

[19] Haas, J., and Hares, S. Definitions of Managed Objects for BGP-4. RFC 4273, January 2006.

[20] Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., and Pras, A. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials* (November 2014).

[21] Hopps, C. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992, November 2000.

[22] Huawei. Configuring BMP. `https://support.huawei.com/enterprise/en/doc/EDOC1100039536/3a8c0622/configuring-bmp`. Accessed: October, 2020.

[23] Internet Assigned Numbers Authority (IANA). IP Flow Information Export (IPFIX) Entities. `https://www.iana.org/assignments/ipfix/ipfix.xhtml`, May 2007. Accessed: October, 2020.

[24] Jethanandani, M., Patel, K., Hares, S., and Haas, J. BGP YANG Model for Service Provider Networks. Internet Draft, `https://tools.ietf.org/html/draft-ietf-idr-bgp-model-09`, June 2020. Accessed: October, 2020.

[25] Juniper. Change in implementation of BMP protocol in Junos OS release 19.1 and 18.2X75-D30. `https://kb.juniper.net/InfoCenter/index?page=content&id=KB34082`, May 2019. Accessed: October, 2020.

[26] Juniper. BGP Monitoring Protocol. `https://www.juniper.net/documentation/en_US/junos/topics/topic-map/bgp-monitoring-protocol.html`, September 2020. Accessed: October, 2020.

[27] Juniper Networks, Inc. Juniper Flow Monitoring. `https://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf`, 2011. Accessed: October, 2020.

[28] Klyne, G., and Newman, C. Date and Time on the Internet: Timestamps. RFC 3339, July 2002.

[29] Lapukhov, P., Premji, A., and Mitchell, J. Use of BGP for Routing in Large-Scale Data Centers. RFC 7938, August 2016.

[30] Linux Foundation. FRRouting. https://frrouting.org/, April 2017. Accessed: October, 2020.

[31] Linux Foundation. BMP. http://docs.frrouting.org/en/latest/bmp.html, August 2019. Accessed: October, 2020.

[32] Lucente, P. pmacct. http://www.pmacct.net/. Accessed: October, 2020.

[33] Lucente, P., and Gu, Y. Support for Enterprise-specific TLVs in the BGP Monitoring Protocol. Internet Draft, https://tools.ietf.org/html/draft-lucente-grow-bmp-tlv-ebit-01, May 2020. Accessed: October, 2020.

[34] Lucente, P., Gu, Y., and Smit, H. TLV support for BMP Route Monitoring and Peer Down Messages. Internet Draft, https://tools.ietf.org/html/draft-ietf-grow-bmp-tlv-03, September 2020. Accessed: October, 2020.

[35] Nokia. Cflowd. https://documentation.nokia.com/html/0_add-h-f/93-0073-HTML/7750_SR_OS_Router_Configuration_Guide/Cflowd-Intro.html. Accessed: October, 2020.

[36] Obstfeld, J., Chen, X., Frebourg, O., and Sudheendra, P. Towards near real-time bgp deep analysis: A big-data approach. https://arxiv.org/abs/1705.08666, May 2017. Accessed: October, 2020.

[37] Orsini, C., King, A., Giordano, D., Giotsas, V., and Dainotti, A. Bgpstream: A software framework for live and historical bgp data analysis. In *Proceedings of the 2016 Internet Measurement Conference* (November 2016).

[38] P Jakma. Quagga Routing Suite. https://www.quagga.net. Accessed: October, 2020.

[39] Patel, K., Chen, E., and Venkatachalapathy, B. Enhanced Route Refresh Capability for BGP-4. RFC 7313, July 2014.

[40] Rauch, G. Socket.io. https://socket.io/. Accessed: October, 2020.

[41] Rescorla, E., and Dierks, T. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[42] Rosen, E., and Rekhter, Y. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, February 2006.

[43] Scudder, J., Fernando, R., and Stuart, S. BGP Monitoring Protocol (BMP). RFC 7854, June 2016.

[44] Sriram, K., Montgomery, D., McPherson, D., Osterweil, E., and Dickson, B. Problem Definition and Classification of BGP Route Leaks. RFC 7908, June 2016.

[45] Srivastava, M., and Lucente, P. BMP Compression. Internet Draft, https://tools.ietf.org/html/draft-msri-grow-bmp-compression-00, October 2019. Accessed: October, 2020.

[46] The Tcpdump Group. tcpdump. https://www.tcpdump.org/, 1988. Accessed: October, 2020.

[47] Tschetter, E., and Yang, F. Apache Druid. `https://druid.apache.org/`. Accessed: October, 2020.

[48] Vissicchio, S., Cittadini, L., Pizzonia, M., Vergantini, L., Mezzapesa, V., and Papagni, M. L. Beyond the best: Real-time non-invasive collection of bgp messages. In *In USENIX Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN 2010)* (2010).

[49] Xu, F., Graf, T., Gu, Y., Zhuang, S., and Li, Z. BGP Route Policy and Attribute Trace Using BMP. Internet Draft, `https://tools.ietf.org/html/draft-xu-grow-bmp-route-policy-attr-trace-05`, July 2020. Accessed: October, 2020.

Appendix A

Pipeline Documentation

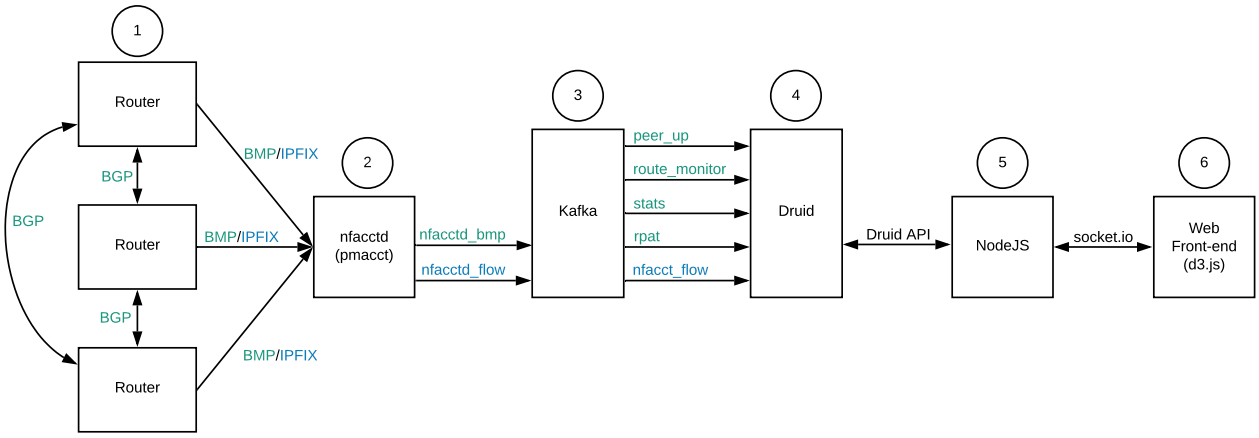


Figure A.1: End-to-End Pipeline to export, collect, store and visualize control- and data-plane information

This section serves as a documentation with practical steps to set up every component of the pipeline from Figure A.1.

1. **Router**

   Every vendor defines a different syntax to export control- and forwarding-plane information. The general information to be present for BMP is who to export to (IP address and port), on which interface and what data to export, e.g., which RIBs of which peerings. Additional parameters, e.g., timer for statistics dump, is set with predefined commands. An example of the BMP configuration on the CE router *daisy-53* can be seen in Listing A.1.

```
bmp
 statistics-timer 60

 bmp-session 192.0.2.1
  tcp connect port 1790
  connect-interface GigabitEthernet0/3/6.500
  trace-route ipv4 all
```

```
monitor all-vpn-instance
 route-mode ipv4-family unicast adj-rib-in pre-policy
 route-mode ipv4-family unicast adj-rib-in post-policy
 route-mode ipv4-family unicast adj-rib-out pre-policy
 route-mode ipv4-family unicast adj-rib-out post-policy
 route-mode ipv6-family unicast adj-rib-in pre-policy
 route-mode ipv6-family unicast adj-rib-in post-policy
 route-mode ipv6-family unicast adj-rib-out pre-policy
 route-mode ipv6-family unicast adj-rib-out post-policy

monitor vpn-instance A10
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance A20
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance A30
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance B10
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance B20
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance B30
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance C10
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance C20
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib

monitor vpn-instance C30
 route-mode ipv4-family unicast local-rib
 route-mode ipv6-family unicast local-rib
```

Listing A.1: Example BMP configuration on a Huawei *NE40E-M2K-B* router

The different configuration settings and syntax among different vendors also applies to the forwarding plane.

2. **nfacctd(pmacct)**

   To receive the exported information from the network devices, the collector pmacct listens on the ports which the routers export their data to. *nfacctd* is a daemon of the *pmacct* collection

that is capable of collecting control- and data-plane information. It correlates the data on a VPN level and enriches the forwarding plane with additional information such as community attributes. In case there is no RD export in the forwarding plane, this necessary information for this correlation is supplied with a file called *flow_to_rd_map*, which maps the ingress and egress interface IDs and IP addresses to the RD. An excerpt of an example file can be seen in Listing A.2.

```
!192.0.2.72

id=0:64499:22 ip=192.0.2.72 in=47
id=0:64499:22 ip=192.0.2.72 out=47
id=0:64499:52 ip=192.0.2.72 in=49
id=0:64499:52 ip=192.0.2.72 out=49
id=0:64499:82 ip=192.0.2.72 in=50
id=0:64499:82 ip=192.0.2.72 out=50
id=0:0:0 ip=192.0.2.72 in=4
id=0:0:0 ip=192.0.2.72 out=4

!192.0.2.52

id=0:0:0 ip=192.0.2.52 in=42
id=0:0:0 ip=192.0.2.52 out=42
id=0:0:0 ip=192.0.2.52 in=4
id=0:0:0 ip=192.0.2.52 out=4

...
```

Listing A.2: Mapping file to match interface ID and IP address to a RD

pmacct offers plugin support for multiple back ends, which have to be configured accordingly. If we want to use Kafka as a message broker, we add the IP address and the port number for the broker and define the Kafka topics the control- and forwarding-plane data is produced into as well as some configurational parameters such as buffer sizes and TLS support if desired. Another important parameter is the format of the encoded timestamps. The option *timestamps_rfc3339* [28] has to be set to *true* to be compatible with Druid's time format.

3. **Kafka**

   There are two options to organize the Kafka topics for BMP messages. We can define one large schema that entails all BMP messages. Fields that are not present in some BMP message types are exported as *null*. This simplifies the setup, but comes at an efficiency disadvantage. A more efficient strategy (in terms of response time) is to create a Kafka topic for every individual BMP message type. This is achieved by consuming from the general BMP topic and produce to the individual message topics. An example on how to do that for the route monitor message type is shown in Listing A.3.

```
bin/kafka-console-consumer.sh --bootstrap-server $KAFKA_IP:$KAFKA_PORT
    --topic $GENERAL_BMP | grep '\"bmp_msg_type\": \"route_monitor\"'
   | bin/kafka-console-producer.sh --bootstrap-server $KAFKA_IP:
   $KAFKA_PORT --topic $ROUTE_MONITOR_TOPIC
```

Listing A.3: Bash command to split up and ingest BMP messages into Kafka topics for specific message types

In case only a subset of all BMP messages are used for consumption, this potentially leads to much less data to be searched before responding to a query. As an example, the number of route monitoring messages in our lab environment were just a small subset of the route policy and attribute tracing messages. Regarding the forwarding-plane, this differentiation does not apply to IPFIX messages because all exported flow information adhere to the same schema.

4. **Druid**

   Druid has to be run with a minimum amount of resources to handle multiple Kafka topics and to efficiently respond to queries. The *small single server* configuration is the lowest pre-installed deployment configuration that was successfully tested. This requires at least 64 GB of RAM and 8 CPU cores. Depending on the previous splitting strategy for Kafka, different BMP schemas have to be defined up front. If we decide for the strategy to produce to individual topics for each message type, we can create a schema with the minimal number of dimensions that are needed for future use cases. An example for the BMP route monitoring message is shown in Listing A.4.

```
{
  "type": "kafka",
  "spec": {
    "ioConfig": {
      "type": "kafka",
      "consumerProperties": {
        "bootstrap.servers": "$KAFKA_IP:$KAFKA_PORT"
      },
      "topic": "$ROUTE_MONITOR_TOPIC",
      "inputFormat": {
        "type": "json"
      },
      "useEarliestOffset": true
    },
    "tuningConfig": {
      "type": "kafka"
    },
    "dataSchema": {
      "dataSource": "$ROUTE_MONITOR_TOPIC",
      "granularitySpec": {
        "type": "uniform",
        "queryGranularity": "SECOND",
        "segmentGranularity": "HOUR",
        "rollup": true
      },
      "timestampSpec": {
        "column": "timestamp",
        "format": "iso"
      },
      "dimensionsSpec": {
        "dimensions": [
          "bgp_nexthop",
          "bmp_msg_type",
          "bmp_router",
          "comms",
          "ecomms",
```

```
            "ip_prefix",
            "log_type",
            "peer_ip",
            "rd",
            "is_in",
            "is_loc",
            "is_post"
            }
        ]
      }
    }
  }
}
```

Listing A.4: Example Druid schema definition for BMP Route monitoring message

Depending on the requirements of the system, different parameters for query and segment granularity can be chosen.

5. **NodeJS**

   The NodeJS application has dependencies on the packages *express*, *http*, *socket.io*, *request*, *util* and *fs*. If the additional option to query route policies directly from the router is desired, the package *node-netconf* has to be included as well.

6. **Web Front End**

   The control-plane graph for route monitoring, peering and route policy and attribute tracing message types have been implemented. The library to create control-plane graphs has a well-defined API to develop new use cases as follows:

```
setNode(nodeObj)
setLink(linkObj)
getNode(nodeObj)
getLink(linkObj)
deleteNode(nodeObj)
deleteLink(linkObj)
```

   A *node* has a unique identifier *id*, a *label* that is displayed in the graph, an *info* object that stores additional information about the node and *labelCallback*, a callback funcition that is called when the node is clicked on by the user.

   A *link* consists of two endpoints *source* and *target* that refer to the node *id* of the endpoints, a link *label* that is displayed on the graph, an *info* field that stores additional information about the link and a *labelCallback* that is called when the user clicks on the link label. Because links are often used to aggregate information and maintain state, there is a an additional object *append*, that is a set which maintains data across function calls. As an example, if we use the peer IP address and the BGP next-hop as nodes, the link with these two nodes as endpoints might represent all prefixes with that specific (peer IP address, BGP next-hop) tuple.