Monitoring an EVPN-VxLAN fabric with BGP Monitoring Protocol

July, 2020

Davide Pucci
M.Sc. Security and Network Engineering
University of Amsterdam
Amsterdam, The Netherlands
davide.pucci@os3.nl

Giacomo Casoni
M.Sc. Security and Network Engineering
University of Amsterdam
Amsterdam, The Netherlands
giacomo.casoni@os3.nl

***Abstract*—The BGP Monitoring Protocol (BMP) has been defined in June 2016 and is meant to provide a complete Border Gateway Protocol (BGP) monitoring solution, while eliminating the need for clumsy workarounds and home-cooked solutions, such as setting up extra, ad-hoc, BGP speakers to work as route collectors, or even screen-scraping. Given the widely spreading adoption of Ethernet VPN (EVPN) in Data Centers nowadays, a mean to monitor such an overlay would greatly improve visibility of the internal workings of a EVPN fabric, allowing for greater effectiveness with regard to the troubleshooting flow and incident handling and prevention. In our research, we identified use cases for BMP applications in a EVPN overlay network and we proved its validity by implementing a BMP suite — based on the Free Range Routing (FRR) suite — capable of exchanging and analyzing, specifically, EVPN traffic.**

***Index Terms*—BGP, BMP, EVPN, VxLAN, routing, route distinguisher, route target, MAC mobility**

I. Introduction

The Border Gateway Protocol (BGP) is a widely known routing protocol, whose first definition is dated back in the late eighties [17]. The protocol is supposed to work on top of the Transmission Control Protocol (TCP), allowing BGP to skip the implementation of several strategies, meant to guarantee reliability and security capabilities, as all features partly offered by the transport protocol, already. When configured and running on a node, called a *speaker*, a connection is established with neighboring speakers and portions of information within the routing scope are exchanged. Such information unit is called Network Level Reachability Information (NLRI): relying on data obtained with the cooperation between a network of BGP peers, a virtual graph of reachable networks is built, with each peer contributing to the availability of parts of them, and altogether basing it off the physical architecture. Alongside the NLRI, further information is exchanged, to ensure that every speaker is able to detect whether certain inferred configurations lead to loop or to apply desired policies. All the data is being kept, by each node, on a database structure called Routing Information Base (RIB). Each speaker has three different kind of RIBs: 1) the Adj-RIB-In stores all the routes information received a specific peer (each Adj-RIB-In is linked to a specific peer), 2) the Local-RIB stores the effective routing information that is applied on the local speaker, after having processed and enforced local policies and 3) the Adj-RIB-Out stores all the routes information that the local speaker selected to be advertised (similarly to the Adj-RIB-In, there is a Adj-RIB-Out per neighboring peer).

The introduced behavior applies to what is commonly known as External BGP (eBGP), which represents the part of BGP strictly related to the inter-Autonomous System (AS) routing. Originally, eBGP and its counterpart for intra-AS routing, Internal BGP (iBGP), were logically entirely separated, as it initially appeared to be obvious to make a separation between their roles and scopes. Nowadays, while eBGP is still behind the routing of the highest hierarchy level of the Internet, it also expanded to be suitable in further scopes. Since when BGP was first defined, more and more additions have been done to extend the capabilities of the two layers, like the introduction of private AS numbers [13], the definition of AS Confederations [30], or BGP Route Reflection [2]. Moreover, the belief that the original distinction between iBGP and eBGP was no more indispensable started spreading: both scopes could be covered entirely by eBGP. In fact, in August 2016, the Request for Comments (RFC) 7938 [16] appeared, to claim BGP to be a valid solution to handle routing within a single private Data Center. Such environments are always more relying on virtualization technologies and protocols to build flexible infrastructures, which have to be able to be dynamically stretched and shrunk to hold complex and different-scales systems, to isolate or intersect networks segments, to possibly enforce granular policies. These needs seem to be flawlessly supported by the BGP capabilities, which has been kept on being upgraded and extended, over and over.

In such infrastructure, among the other requisites, stability represents a key component: in order to properly guarantee it, a conscious and consistent view of how the various components of the network are inter-working with each others is needed. This brings the attention to monitoring: being aware of the (in)correct functioning of such complex environment represents an important challenge, even more so when the

environment itself, the network, is representing the baseline onto which further application components rely to offer their services. With this in mind, the BGP Monitoring Protocol (BMP) [27][11] represents a new solution to cover such need, specifically for BGP: it allows to collect meaningful information involved in the inter-speaker communication and setup, *e.g.*, various neighboring speakers vendor and implementations specifications, BGP sessions enumerations, exchanged prefixes, constructed `AS_PATH`s, as well as statistics reports containing information not inherently visible on the BGP protocol level. This new protocol relies on a well-known distinction of roles, being the BMP server assigned of collecting (more precisely, passively receiving) the aforementioned information from its neighboring speakers, the BMP clients. Given the young nature of BMP, however, it does not represent a *de-facto* standard for monitoring BGP, yet.

Thanks to its high flexibility, BGP has been successfully applied to very different use cases, such as the aforementioned Data Centers, as well as university campuses. In such contexts, it is also often required for different parties, possibly in different locations, to share a private communication channel, *e.g.*, separate machines belonging to the same department spread across a campus, or servers from a single client placed in different Data Centers. In scenarios like these, a Virtual Private Network (VPN) strategy is usually adopted, to allow the two areas aiming for privacy to send and receive to/from each other with a public network in between, guaranteeing privacy to the parties. Ethernet VPN (EVPN), a specific case of VPN [24], is defined as a BGP-based control plane for layer-2 and layer-3 connectivity. It operates on a Provider Edge (PE) and Customer Edge (CE) infrastructure, where reachability information is exchanged among PEs, which also exchange encapsulated traffic. As far as reachability goes, PEs make use of MultiProtocol BGP (MP-BGP) [3][4][5][8] to exchange information relying on a IP/MPLS backbone. On the other hand, encapsulation can be achieved by many means, and while several options are available for EVPN, Virtual Extensible LAN (VxLAN) [18] is usually the preferred one in Data Centers. VxLAN is a tunneling technology used to encapsulate Ethernet frames in User Datagram Protocol (UDP) packets. Moreover, VxLAN expands the ID space which is quite limited in conventional Virtual Local Area Network (VLAN). EVPN uses the concept of Route Distinguisher (RD), also exchanged via MP-BGP, to make locally significant identifiers globally unique and achieve network separation on shared links. To give further policy control over what foreign RD-dependent routes are to be imported locally in a specific (possibly different) RD, the extended community Route Target (RT) is used.

II. Problem and Previous Research

While BMP offers a native solution to BGP route advertisement data collection, its adoption is not widespread yet. This could be due to the protocol being relatively new, although all major networking equipment manufacturers already provide support for it, or to the fact that alternative solutions had already been used before BMP came to be. Traditionally, BGP monitoring has been done by having a full session with a so called *route collector* [31], which is nothing more than a BGP speaker used to collect routes from other routers for monitoring purposes. BGPmon [32], hypothesized in 2009, improves on the collectors idea and relies on a overlay network on top of the original fabric, which is used to exchange BGP advertisements between routers and collectors. The presence of existing BGP monitoring systems could be hampering the adoption of BMP.

Furthermore, it is still unclear what kind of information can be gathered with BMP: by design, the protocol is meant to collect BGP messages exchanged between the monitored station and its peers, as well as introducing specific type of counters and data, in the form of the aforementioned statistics reports. It is still not entirely clear how such information can be interpreted and to what extent it can be applied to real-life common use cases, beyond simple statistics collection. For example, is it capable of tracking Virtual Machine (VM) movements within a network? Can it detect a security incident while happening?

While BMP by design seems to not impose limits in terms of network architectures — as long as BGP is taking care of the routing component, BMP can be introduced — with no consideration of Address Family Identifier (AFI) / Subsequent AFI (SAFI), on the other hand, it remains unclear whether this is the case in practice. Even more so, given that open BGP/BMP implementations such as Free Range Routing (FRR) seem to support it only for tracking a Protocol Data Unit (PDU) whose scope is strictly linked to IPv4 and IPv6 AFIs and unicast and multicast SAFIs only. Its behavior when attempting to monitor an overlay network, namely built relying on EVPN- and VxLAN-based strategies, is still not clear.

Finally, it is worth mentioning that open source products that revolve around BMP are also being released. OpenBMP, for example, is a BMP collector used to aggregate information from several BMP enabled routers. Moreover, OpenDayLight Software Defined Network (SDN) controller also implements a BMP plugin. However, even for the few BMP implementations that do exist, the depth is relatively low, to the best of our knowledge. For example, OpenBMP only provides functionality such as AS number lookup and geo-location, which only make sense in a traditional AS-to-AS BGP context.

Given the widespread use of BGP as, initially, the only Exterior Gateway Protocol (EGP) solution, and nowadays as Interior Gateway Protocol (IGP) in Data Centers [16], it is only natural that much research went into attempting to reliably monitor the state of BGP sessions, in order to detect and troubleshoot faults or attacks.

Already in 2004, Nordström et al., in *Beware of BGP attacks* [21], warned about the possibilities of attacks against the BGP infrastructure. Threats like traffic blackholing, redirection and sniffing were already very much a concern then. Moreover, as mentioned in *BGP in the Data Center* [9], cloud environments in Data Centers change quickly, and new virtual networks and machines are spinned up in seconds, calling for a way to detect

the location of virtual components at a given time.

Monitoring BGP advertised routes could provide a solution to the aforementioned problems. The task, however, proved to be all but trivial in recent researches. Mainly two problems need to be addressed when attempting to perform BGP monitoring:

- How can BGP route advertisements be collected?
- How can the collected data be analyzed?

Several solutions were proposed in the literature to collect BGP route advertisements. Notably, BGPmon [32] first advanced in 2009 a variation of the traditional BGP data collection method. Originally, BGP routes were monitored by creating a full BGP session with an extra router, which was only used for monitoring. The approach described in the proposed software, while based on the same principles, implements only the components necessary to monitor. This method, however, while allowing much greater scalability, is still flawed. For example, the only routes that can be monitored are the ones exported by the observed BGP speaker. This means that no information can be inferred about the raw, unprocessed, advertisements the BGP speaker receives. One year later, in 2010, the research *Beyond the Best: Real-Time Non-Invasive Collection of BGP Messages* [31] proposed an entirely different method for BGP data collection, which relied on selective IP packets duplication and forwarding. At the time of writing, BMP was already proposed, but the authors deemed it to be too experimental and not mature enough. While allowing wider visibility on the the state of the BGP environment, this method introduces complexity on the side of the collector, which has to take care of demultiplexing streams and reassembling TCP segments. Finally, in 2016, BMP was defined [27], promising a built-in, native approach to tap and observe the internals of the BGP pipeline. In the subsequent RFC 8671 [11], support was added to BMP for monitoring not only the content of Adj-RIB-In, but also Adj-RIB-Out. For both databases, pre-policy and post-policy monitoring is available nowadays.

As far as analyzing collected BGP information, much research has been focusing on the issue of BGP routes hijacking. Papers such as *Visual analytics for BGP monitoring and prefix hijacking identification* [6], *HEAP: reliable assessment of BGP hijacking attacks* [26], *IP prefix hijack detection using BGP connectivity monitoring* [1] and *ARTEMIS: Neutralizing BGP hijacking within a minute* [28] all propose different solutions to prevent or mitigate this phenomenon. Much in the same direction, the research *Unsupervised real-time detection of BGP anomalies leveraging high-rate and fine-grained telemetry data* [23] focuses on detecting general BGP anomalies, which could imply attacks on a network infrastructure or malfunctions of network equipment. On a different note, *Netcohort: Detecting and managing vm ensembles in virtualized data centers* [14] proposed monitoring BGP routes advertisements to detect performance bottlenecks caused by ensembles of virtual machines. Finally, projects such as *BGPStream: a software framework for live and historical BGP data analysis* [22] aim at an overall view on a environment given by BGP data.

No research so far has focused explicitly on the integration of BMP with EVPN-VxLAN environments. This is a gap to be filled, however, given the increasing reliance Data Centers have on EVPN-VxLAN [10], which operates based on MP-BGP [24] and could therefore be monitored with BMP. More specifically, no further research seems to have been done about possible approaches on monitoring cross-networks VM movements in a EVPN-based environment.

III. Research Questions

How can BMP effectively used to monitor an EVPN-based overlay networks?

This question can be divided into these sub-questions:

1) Can BMP, by design, be used to monitor an EVPN overlay network?
2) Can a proof of concept be realized to cover such a scenario and with which open source technologies (*e.g.*, FRR or OpenBMP)?
3) What information can be collected and in what way can this information be used to monitor the overlay network?
4) In which way did the adoption of BMP impact the network in its design?

IV. Research Considerations

According to the BMP's RFC [27], the way BGP is monitored is not AFI/SAFI-dependent. In fact, regardless of the approach to encapsulate a BGP message within a BMP packet, the monitoring protocol provides a restricted range of messages types and none of them is explicitly relying on a given AFI/SAFI assumption. In the predominant and most important use case, there is the routing information: the current status of announced and exchanged route is provided to the BMP servers. This information might come in two different shapes: 1) Route Monitoring, in which case the BMP client, *i.e.* the monitored BGP speaker, sends all the routes processed and stored in the Adj-RIB-In (or in the Adj-RIB-Out, as defined in a further BMP reiteration RFC [11]) as soon as the BMP session is up, while keeping the monitoring server up-to-date by sending further routes as received by the speaker in the shape of `BGP UPDATE` PDU subsequently; 2) Route Mirroring, in which case the speaker proxies and forwards every BGP message it is encountering, giving the BMP server a full-fidelity view of all the network. Considering that this approach is much noisier within the network, it is advised to be used for troubleshooting purposes only.

Another message type is the stats report: this kind of message is not created by directly encapsulating BGP messages, but rather by deriving BGP statistics to observe interesting events that could occur on the speakers. Event- or timer-driven, such reports basically contain counters of aggregated information, *e.g.*, number of (known) duplicate prefix advertisements or withdrawals, number of invalidated updates due to `CLUSTER_LIST`, `AS_PATH` or `AS_CONFED` loops, number of routes in a given RIB and so forth.

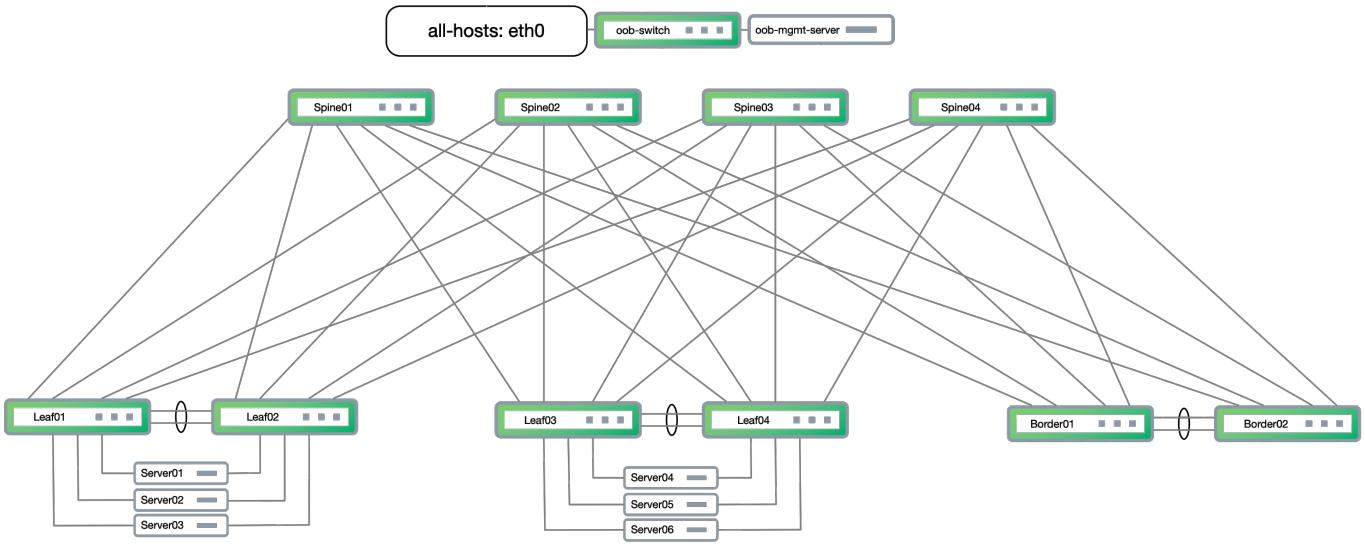

Fig. 1. Experiment network topology. At the bottom, the main spine-leaf structure, consisting of four spines, four leaves and two border nodes: each of the spine and leaf nodes are BMP clients. At the top, the nodes covering management-related purposes: the management server is the designed BMP server, connected via dedicated links to each BMP client.

Finally, Peer status message type is used to give an insight about BGP speakers overall status, as when they come up or down, bringing BGP sessions up and down as well.

As per the aforementioned reasons, protocol-wise, BMP does not impose any constraint in terms of AFI/SAFI, enabling for the proposed use case of monitoring EVPN fabrics the same way it can be done for more traditional environments based on IPv4/6 unicast and multicast routing only.

Given that the mean opens up to such a monitoring scenario, further work has been done about identifying what kind of information is worth to be traced to gather what kind of behavior observation. On this extent, the following events have been defined:

1) VM movements history, to give an insight about when and where movements have been done within the network for a given Media Access Control (MAC) address.
2) Infrastructure convergence time, to give an accurate estimation of how long the network took to converge and stabilize.
3) MAC flapping, to detect whether a MAC address is being advertised by a single PE or if conflicting advertisements are being exchanged.
4) Inconsistencies in MAC Mobility counters [25], to show whether in the network MAC movements unexpectedly happened.
5) BGP sessions status, to track down when and to which extent a speaker has stopped peering.
6) Prefixes authority history, to show an insight about where prefixes are originated and, if applicable, when such prefixes authority has been moved from one node to another.

V. Experiments and Discussion

Network topology

As shown in fig. 1, the network topology adopted for the experiment is represented by a typical spine-leaf infrastructure. Four spines are full-mesh and point-to-point link connected to a set of four leaves and two borders: each of them is running BGP, with each link translating into a BGP session. The four leaves are grouped in couples, each representing a Multichassis Link Aggregation (MLAG) pair, to which racks of three servers are directly connected. These servers are hosting VMs whose MACs are being advertised in the network. Every node in the network has a dedicated `eth0` interface connected to the virtual `oob-switch`, to guarantee connectivity to the `oob-mgmt-server`, which represents the BMP server: the BMP communication is entirely isolated from the BGP logical layer. Every BGP speaker is running Cumulus Linux 4.1.1 and shipping with FRR 7.0.

BMP server

Being that BMP is a relatively new protocol, not many open server implementations exist at the time of writing, being OpenBMP, from Streaming Network Analytics System [29], arguably the most popular. The tool, however, has questionable EVPN support: while EVPN routes can be recognized, the analysis features are mainly focused on classical BGP applications, such as AS-to-AS BGP sessions. For example, OpenBMP provides the functionalities for IP geo-location and AS number lookups. Clearly, these features are of no use when applied to an EVPN environment. As far as parsing goes, as of version 3.3.0, Wireshark can fully parse BMP messages. However, Wireshark provides no analysis and unflexible vi-

sualization capabilities. A custom solution has been therefore built, in order to satisfy all the following requirements:

- full parsing capabilities for both BMP message headers and BGP EVPN messages, including relevant Path Attributes and Extended Communities;
- analysis of the received data, with regard to the defined use cases;
- visualization of the analytics drawn from the BMP collector.

The parser has been implemented to parse BGP messages containing the `AS_PATH`, `EXTENDED COMMUNITIES`, `MP_REACH_NLRI` and `MP_UNREACH_NLRI` path attributes. For the `EXTENDED COMMUNITIES` path attribute, the focus went towards parsing the RT and the MAC Mobility counter extended communities. While all extended communities have the same size, *i.e.* eight bytes, and all use one byte for the type and subtype, the remaining six bytes can be used differently. Parsing only works correctly for extended communities that use a two bytes field followed by a four bytes one. For `MP_(UN)REACH_NLRI`, parsing is only implemented if the AFI and SAFI are, respectively, 25 and 70 (Layer-2 Virtual Private Network (L2VPN) and EVPN). As far as the BMP message goes, only message type 0 (Route Monitoring), 2 (Peer Down Notification), 3 (Peer Up Notification), 4 (Initiation Message), and 6 (Route Mirroring Message) are processed.

The server has been implemented in Python [7], relying on an ElasticSearch instance for data storing and retrieval. While Kibana was initially used, the too specific nature of the use case in analysis made impossible to leverage its visualization functions: this led to therefore resort back to Python plotting libraries, namely Matplotlib [19] and NetworkX [20].

BMP client

The BMP client is the BGP speaker to be monitored. As an extension of BGP the protocol is implemented along in the routing protocol stack. As mentioned in Network topology section, each node is running FRR 7.0, which is not yet supporting BMP. Support for BMP has been added from version 7.2 onward: hence, a preventive FRR upgrade has been done to enable it, as well as explicitly enabling the `bmp` module as a startup option to the `bgpd` daemon:

```
spine01$ grep bgpd_options /etc/frr/daemons
bgpd_options = " -M snmp -A 127.0.0.1 -M bmp"
```

Listing 1. Add `bmp` module to `bgpd` options

Furthermore, according to the suite documentation, at the time of writing, the current implementation of the monitoring protocol is only supporting tracking IPv4/v6 AFI and unicast/multicast SAFI when enabled in monitoring mode [12]:

> *Only IPv4 and IPv6 are currently valid for AFI, and only unicast and multicast are valid for SAFI. Other AFI/SAFI combinations may be added in the future.*

Although, as described in Research Considerations section, BMP protocol is (S)AFI-independent, its FRR implementation presented some use cases constraints to be overcome. Mainly, such constraints were imposed as per the assumption about the layout of a (S)AFI RIB: being the EVPN RIBs made of two different layers, this is conflicting with the simple single-layer structure of a IP-based RIB. This latter RIB represents a single set of routes (and possible further information attached to them), while for EVPN this is not the case: a route is not self-described uniquely, as several Virtual Routing and Forwarding (VRF) could be using same networks referenced in routes to enforce different behaviors. EVPN relies on the concept of RD (and RT) to identify to which VRF a (otherwise ambiguous) route is to be installed. The RD is associated to a VRF and prepended to a specific route to confer uniqueness to the route itself. This structure gives an insight about the reason why FRR implemented EVPN RIBs to be organized in two-layers tables. Steps to add support for this use case involved the steps below.

AFI/SAFI enablement

To configure Route Monitoring for a supported (S)AFI, such as IPv4 unicast, the following commands are to be used:

```
spine01# configure terminal
spine01# router bgp
spine01# bmp targets default
spine01# bmp monitor ipv4 unicast pre-policy
```

Listing 2. VTY shell commands to add BMP pre-policy monitoring for IPv4 unicast

The function called by the `bmp monitor` command corresponds to the `bmp_monitor_cmd` function installed using the macro `DEFPY` in `bgpd/bgp_bmp.c`. Allowing the command to support L2VPN AFI and EVPN SAFI is a matter of extending a constant:

```
DEFPY(bmp_monitor_cfg,
  bmp_monitor_cmd,
- "[no] bmp monitor <ipv4|ipv6>
    <unicast|multicast>
    <pre-policy|post-policy>$policy",
+ "[no] bmp monitor <ipv4|ipv6|l2vpn>
    <unicast|multicast|evpn>
    <pre-policy|post-policy>\$policy",
  NO_STR
  BMP_STR
  "Send BMP route monitoring messages\n"
- "Addr Family\nAddr Family\n"
- "Addr Family\nAddr Family\n"
+ "Addr Family\nAddr Family\nAddr Family\n"
+ "Addr Family\nAddr Family\nAddr Family\n"
  "Send state before policy and filter ...\n"
  "Send state with policy and filters ...\n")
{ ... }
```

Listing 3. Patch to allow enabling EVPN monitoring

Correct queue routes identification

When a BMP session is established in a Route Monitoring scenario, the `bgpd` daemon is appending items pointing to registered routes to a special BMP-related queue. Each of these

items is represented by the following `struct` definition:

```
struct bmp_queue_entry {
  struct bmp_qlist_item bli;
  struct bmp_qhash_item bhi;
  struct prefix p;
  uint64_t peerid;
  afi_t afi;
  safi_t safi;
  size_t refcount;
};
```

Listing 4. BMP routes queue entry structure

As clearly visible, the only field referring to the specific route is the `struct prefix p`: the problem here is that, as mentioned, for EVPN this is just not enough, as a prefix could very much likely be not unique on its own. This is why a further field is needed to be taken into account: `struct prefix_rd rd` field adds the missing discriminator to properly identify EVPN routes. When handling queue operations, two auxiliary functions are called to perform comparison of objects and compute their hash: `bmp_qhash_cmp` and `bmp_qhash_hkey`. The `prefix_rd` field can be exploited to correctly compute such operations in a coherent and unambiguous way.

The first function performs the comparison between two objects and was originally only taking into account the `prefix p` field as well as performing a memory comparison. It has been extended to check the RD field as well, when applicable:

```
int bmp_qhash_cmp(struct bmp_queue_entry *a,
                  struct bmp_queue_entry *b)
{
  int ret;

+ if (a->safi == SAFI_EVPN
+     && b->safi == SAFI_EVPN) {
+   ret = prefix_cmp(&a->rd, &b->rd);
+   if (ret)
+     return ret;
+ } else if (a->safi == SAFI_EVPN)
+  return 1;
+ else if (b->safi == SAFI_EVPN)
+  return -1;

  ret = prefix_cmp(&a->p, &b->p);
  if (ret)
   return ret;
  ret = memcmp(&a->peerid, &b->peerid,
    offsetof(struct bmp_queue_entry, refcount) -
    offsetof(struct bmp_queue_entry, peerid));
  return ret;
}
```

Listing 5. Patch to fix queue EVPN-related entries comparison

On the other hand, the function to calculate the hash for a specific entry needed to take the RD into account as well to extend the hash key material, when applicable:

```
uint32_t bmp_qhash_hkey(struct bmp_queue_entry *e)
{
  uint32_t key;

  key = prefix_hash_key((void *)&e->p);
  key = jhash(&e->peerid,
   offsetof(struct bmp_queue_entry, refcount) -
   offsetof(struct bmp_queue_entry, peerid), key);

+ if (e->afi == AFI_L2VPN && e->safi == SAFI_EVPN)
+   key = jhash(&e->rd,
+    offsetof(struct bmp_queue_entry, rd) -
+    offsetof(struct bmp_queue_entry, refcount) +
+    PSIZE(e->rd.prefixlen), key);

  return key;
}
```

Listing 6. Patch to fix queue EVPN-related entries hash generation

Extended these two functions, it is prevented to see IP routes overwriting EVPN routes in the queue, as well as being able to discriminate between EVPN routes with the same network structure but different RD.

Final step was to inflate the `bmp_queue_entry` object with the RD information when the route is originally handled by BMP and put in the queue, *i.e.* in the `bmp_process_one` function.

```
void bmp_process_one(struct bmp_targets *bt,
     struct bgp *bgp, afi_t afi, safi_t safi,
     struct bgp_node *bn, struct peer *peer)
{
  struct bmp *bmp;
  struct bmp_queue_entry *bqe, bqeref;
  size_t refcount;

+ if (safi == SAFI_EVPN && bn->prn)
+   prefix_copy(&bqeref.rd,
+    (struct prefix_rd *)
+     bgp_node_get_prefix(bn->prn));

  ...
}
```

Listing 7. Patch add RD inflation in case of EVPN routes

Two-layer tables iteration

When sending out routes, BMP is iterating on each RIB, encapsulating each route into a `BGP UPDATE` packet and then surrounding it with a BMP header. To iterate, the function `bmp_wrsync` is called repeatedly with the object `struct bmp *bmp` which maintains information about the current state of sync with the BMP server: the field `bmp->syncpos` points to the last element in the table which has been synced, *i.e.* a prefix in the table.

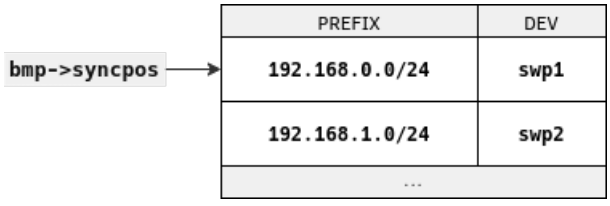

Fig. 2. Traditional single-layer IP RIB structure and synchronization mechanism

If the `bmp_wrsync` function were to be called in a context like shown in fig. 2, the second entry of the IP RIB would be synced.

FRR's BMP implementation takes for granted that the table to iterate is just one level deep: to extend this behavior to

cover deeper tables too, the `bmp_wrsync` function has been patched to keep working transparently for all the already covered cases, while performing few more operations in case of EVPN: a new `bmp->syncrdpos` field is introduced, to point to the last element handled in the middle-layer table. Combining both `bmp->syncpos` and `bmp->syncrdpos`, the current middle-layer table position, *i.e.* the current RD, can be associated with the corresponding current final-layer table position, *i.e.* the effective route prefix.

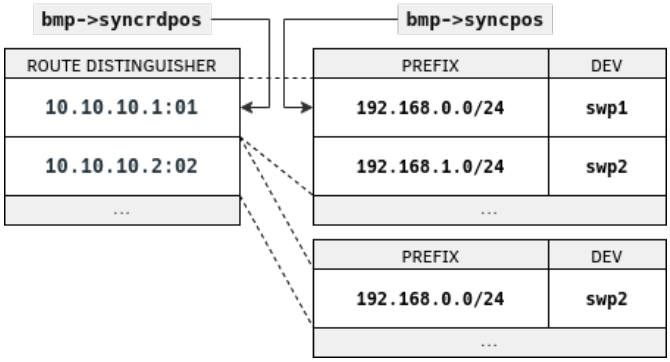

Fig. 3. Dual-layer EVPN RIB structure and synchronization mechanism

This is clearly visible in fig. 3, which shows how the `bmp->syncpos` pointer is still used to point to the (inner) prefix table, while the newly introduced `bmp->syncrdpos` one keeps track of the (outer) route distinguisher table.

To conclude, the last part is about ensuring that, in case of deeper tables, at the end of the `bmp->syncpos`-based iteration, instead of concluding the synchronization, `bmp->syncpos` was reset and `bmp->syncrdpos` increased to the next outer table entry.

For a matter of briefness and due to the complex nature of the patch to cover the mentioned case, it is not shown: nevertheless, the patches are entirely available on the FRR upstream repository[1].

Final configuration

Once the BMP support for EVPN routes is correctly introduced, effective EVPN monitoring configuration is straightforward:

```
spine01# configure terminal
spine01# router bgp
spine01# bmp targets default
spine01# bmp connect $HOST port $PORT max-retry $X
spine01# bmp monitor l2vpn evpn pre-policy
```

Listing 8. VTY shell commands to add BMP monitoring for EVPN

[1]The patches have been submitted to the official maintainers of FRR and have been successfully merged on the upstream repository:

1) *"lib: prefix: add prefix_rd type"*, https://github.com/FRRouting/frr/pull/6582
2) *"bgpd: bmp: add support for L2VPN/EVPN routes"*, https://github.com/FRRouting/frr/pull/6590

For a further look and their full version, the references above can be used.

Or, via the configuration file, within the BGP tag block:

```
router bgp $ASN
 bgp router-id $BGP_ID
 ...
 !
 bmp targets default
  bmp monitor l2vpn evpn pre-policy
  bmp monitor l2vpn evpn post-policy
  bmp connect $HOST port $PORT max-retry $X
!
```

Listing 9. Configuration file instructions to add BMP monitoring for EVPN

Use cases and analysis

VM movements and convergence time estimation

Tracking reachability information (`MP_(UN)REACH_NLRI`) for a given MAC address can give insights into when, where from and where to a VM was moved.

Messages regarding a given address must be grouped coherently and on a time basis so that each group reflects a single event (such as a VM movement). In order to do so, after retrieving all reachability messages regarding a specific MAC address, the mean time interval between message reception and the related standard deviation are computed. Given a user-provided $tolerance$ and the current time interval t, a message is evaluated as belonging to the current event, or being the first of a new event by the means of formula 1:

$$t > \mu_{time} \times \left(\frac{tolerance \times \#messages}{\sigma_{time}} \right) \qquad (1)$$

Once events are identified, it is possible to observe which PE was sending reachability advertisements (`MP_REACH_NLRI`) and which one was advertising unreachability (`MP_UNREACH_NLRI`). This gives an indication as to which rack a VM was moved from and to. Fig. 4 is a visualization of what an event should look like.

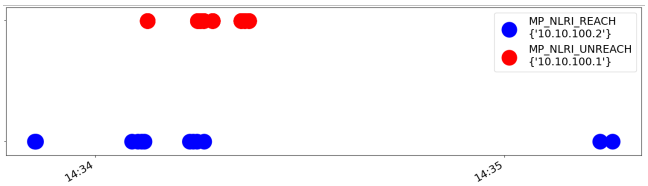

Fig. 4. Plot of received reachability information upon VM movement. The y-axis represents the type of advertisement, while the x-axis the time, with format `HH:MM`.

The blue nodes represent reachability advertisements, while the red ones represent unreachability ones. In a correct environment, all reachability advertisements should come from a single `Next Hop Network Address` and similarly for all the unreachability ones. Given all the advertisements in an event, the difference between the last and first timestamp gives an indication of how long it took to the infrastructure to converge to a stable state, where all the nodes had correct information about where a MAC could be reached from. Given several identified events, it is then possible to

compute a mean convergence time. For the proposed experimental setup, the mean convergence time for MAC address `44:38:39:ff:00:19` was 1.46 seconds, with a standard deviation of 0.25 seconds.

Fig. 5 represents a plot of a possibly incorrect EVPN behavior. In this case, reachability are exchanged without older information being revoked. This could be due to formula 1 failing to detect events correctly. In this case, an operator could fine tune the output by increasing the *tolerance* value. This could also, however, be an indication of MAC flapping occurring.

Fig. 5. Plot of incorrect received reachability information. The y-axis represents the type of advertisement, while the x-axis the time, with format `HH:MM`.

MAC flapping detection

While in a traditional layer-2 network MAC flapping is defined as a switch receiving frames with the same source MAC address on different ports, in a EVPN context it refers to an address being advertised by several PEs. According to the Juniper knowledge base [15], MAC flapping "*makes the network more vulnerable and wastes network resources*".

At the heart of MAC flapping detection is the process of traversing the events identified with the same method described in Section VM movements and convergence time estimation and ensuring that at no point in time a specific MAC address is advertised by more than a single PE. MAC advertisement changes can be represented by a graph, as shown in fig. 6.

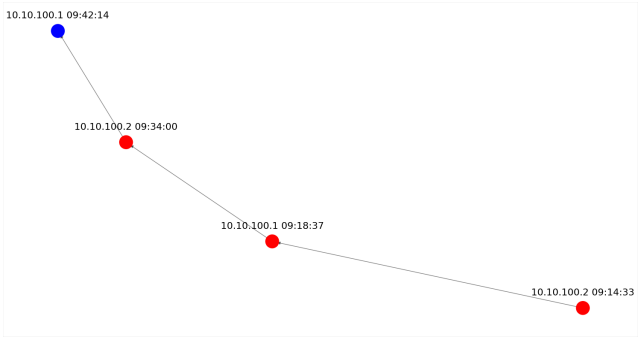

Fig. 6. Directed graph of correct MAC advertisement flow.

Fig. 6 represents a correct flow of MAC advertisements across the infrastructure, where each node represents which PE advertised a given MAC and at which time. Mainly, it is important for the graph to only have a single blue node. Blue nodes are those that are still valid at the moment of analysis, meaning that no unreachability information has invalidated a previous reachability advertisement. Moreover, it is important that in the directed graph that represents the advertisement flow all non-terminal node are of degree two and terminal nodes of degree one. This shows that at no point in time any reachability information was received without the previous one being invalidated. This is not the case, for example, in fig. 7, an example of a graph that shows the EVPN infrastructure running into a faulty state at timestamp 09:34, but eventually fixing itself, since it can be noted that only one node is valid at the time of analysis (only one blue node).

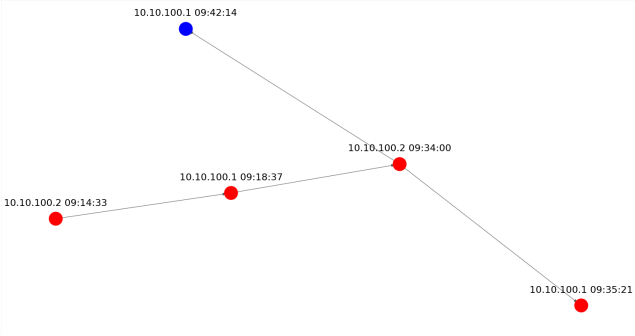

Fig. 7. Directed graph of incorrect MAC advertisement flow.

MAC Mobility counter inconsistencies detection

The MAC Mobility counter, associated to each MAC address, is increased whenever a MAC address is learned by a PE [25]. The counter had been introduced, among other reasons, to prevent MAC duplication issues and it is therefore intended to stop any BGP action for a given MAC until corrective measure is taken by the operator. BMP offers a different way to track the value of the counter and detect more faults than just MAC duplication.

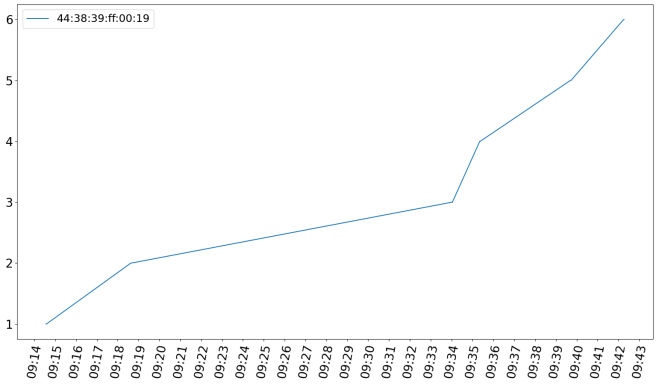

Fig. 8. Plot of correct MAC Mobility counter. The y-axis represents the MAC Mobility value, while the x-axis the time, with format `HH:MM`.

Fig. 8 shows a correct plot of MAC Mobility counter against time, while fig. 9 represents a faulty situation where the counter is seen decreasing at timestamp 09:43. This could, for example, be due to a node advertising false information or due to the reception of old information, caused by large

latency in the network. This could help uncover problems such as, for example, bugs in the implementation of a routing protocol, possible loss of connections of some nodes or network congestion.

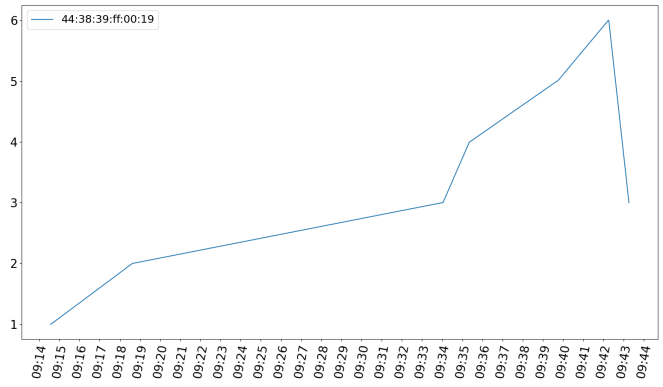

Fig. 9. Plot of incorrect MAC Mobility counter. The y-axis represents the MAC Mobility value, while the x-axis the time, with format `HH:MM`.

BGP sessions

A BGP session can be defined as a couple of BGP peers IDs (id_1, id_2), with no ordering applied. In order to track the lifetime of a specific session, BMP offers the Peer Status information, which is basically the encapsulation of two `BGP OPEN` messages for Peer Up, and `BGP CEASE NOTIFICATION`, *i.e.* type-6 `BGP NOTIFICATION`, for Peer Down. On one hand, the Peer Up message keeps track of both IDs involved in the session bringup: each of the two `BGP OPEN` messages encapsulated contains the BGP Identifier field. For Peer Down the case is slightly different, as BMP only encapsulates a `BGP NOTIFICATION` with type-6, which does not contain any reference to a session, but it does for the specific peer which has gone down, thanks to the BGP Identifier field in the BMP header. While it is straightforward to infer that, if a peer has entirely gone down, such event impacts on all the sessions which that peer was involved in, this reasoning does not apply in cases in which either 1) the peer has specifically invalidated a session, meaning that the `BGP CEASE NOTIFICATION` only applies to that very session, or 2) the peer has unexpectedly gone down, not allowing the `bgpd` daemon to track the event and sending out the notification accordingly.

In the case shown in fig. 10, though, a single node has been entirely shut down, allowing for each session tracked to be invalidated.

As clearly visible, at first, all the sessions are up, as it was possible to reconstruct thanks to the `BGP OPEN` read by the BMP server. At 14:04:30, the node `leaf02`, with BGP Identifier 10.10.10.2, is manually shut down, triggering `BGP CEASE NOTIFICATION`s to be sent out to the server. Just five seconds later, the node is back up again, resulting on new `BGP OPEN` messages, thanks to which the BMP server is able to infer the new status of the sessions.

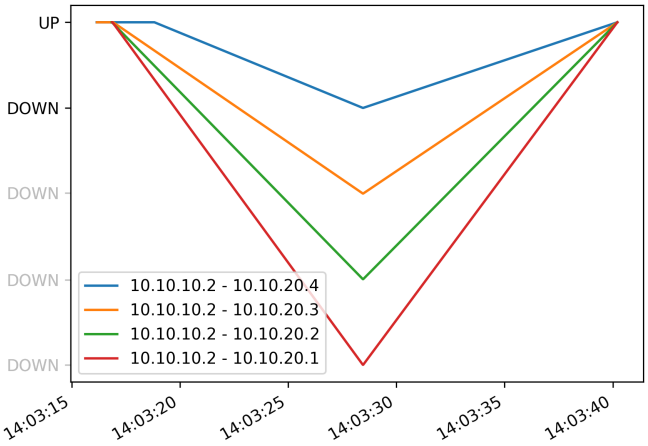

Fig. 10. Plot of peer sessions status, inferred via BMP Peer Up and Down events, correlated with time. The y-axis represents the BGP session status, while the x-axis the time at which a certain event occurred, with format `HH:MM:SS`.

Prefix authority

The prefix authority tracking allows to check whether a certain prefix has started being advertised by new peers. This information can be gathered reading the content of a `BGP UPDATE` message, which is encapsulated in both `BMP Route Monitoring` and `Mirroring` messages. The prefix authority can be inferred relying on two different fields:

1) in the `NLRI` of a EVPN type-5 route, *i.e.* IP route Prefix, both the RD and the inner prefix are exchanged;
2) in the `AS_PATH` path attribute, *i.e.* the list of AS numbers which are to be traversed to reach the announced prefix, the last element represents the original peer that advertised the aforementioned prefix.

Correlating these two information allows to build a plot as shown in fig. 11: the y-axis enumerates the AS numbers advertising certain prefixes, while the x-axis shows when such prefixes have been advertised. In the case proposed, there is no prefix authority overlapping, nor transfer of ownership.

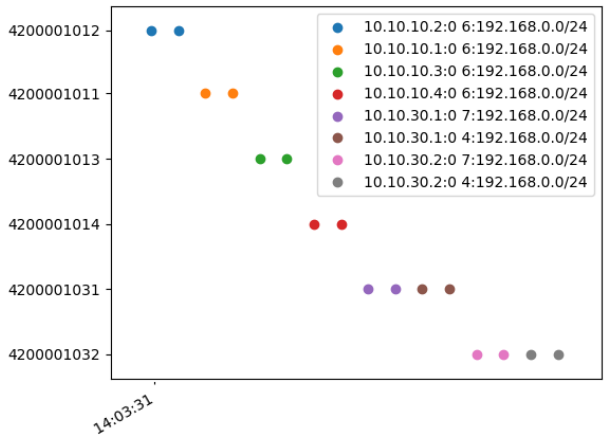

Fig. 11. Plot of EVPN prefixes advertised by BGP peers. The y-axis represents the AS number, while the x-axis the time at which the prefix announcement is being done, with format `HH:MM:SS`.

VI. Conclusions

In order to answer the research question, three main points had to be addressed.

Firstly, a working BMP client which would support BGP/EVPN was needed: FRR provided a good starting point, already implementing BGP/EVPN and BMP for IPv4 and IPv6. The FRR implementation has been successfully extended in order to support EVPN.

Secondly, a working BMP server/collector which could perform analysis specifically on EVPN BGP messages was needed, as well. Given the lack of open implementations, a custom solution capable of receiving, parsing, analyzing and visualizing BMP messages exchanged by a BGP/EVPN speaker has been built and published on Github [7].

Lastly, a set of use cases that could possibly be adopted in the context of monitoring a EVPN overlay network were identified and the server/client solution was successfully applied to meet the requirements of the defined use cases. Overall, BMP has been proven to be a viable solution as far as EVPN overlay monitoring is concerned. Furthermore, the adoption of such protocol does not impact anyhow directly the way the infrastructure is to be designed.

Future Work

In the future, it could be worth improving both the client and the server implementations. The BMP implementation on FRR could be more stable and less prone to sudden crashes, as it was implemented mainly for the specific use case, rather than for production purposes. More testing on it would be beneficial, if not necessary to ensure stability. BMP on FRR, moreover, is currently not VRF aware: this could also be implemented in the future.

As far as stability goes, the same applies to the BMP server. In fact, many BGP components are ignored by the server and only a small set of all Path Attributes or Extended Communities have parsing implemented for, namely only the ones needed for the given use cases. Adding support for more of these and testing, would not only make the implementation more complete, but also more stable.

The identified use cases, although sufficient to demonstrate the viability of BMP as a tool for monitoring EVPN, are not, by any means, a comprehensive set of all its possible applications: more could be identified and investigated. As an example, the proposed experiment was mainly based on `pre-policy` messages and use cases related to this kind of information. Studying `post-policy` messages as well would grant insights into how a specific router applies policies, enabling for investigating new applications of BMP. Moreover, when studying VM movements and convergence times, it could be interesting to observe the behavior of the tool whenever several events happen at the same time.

VII. Acknowledgement

First and foremost, we would like to thank Cumulus Networks and, especially, Attilla de Groot, Donald Sharp and Vivek Venkatraman, for the original idea behind the project and the complete support throughout the duration of the research, in all its aspects.

Moreover, we are grateful to the OS3 staff and the University of Amsterdam for allowing us the use of their facilities, in such difficult time in which COVID-19 impacted so hard on the way structures and institutions can offer their services.

REFERENCES

[1] Alshamrani, H. and Ghita, B. "IP prefix hijack detection using BGP connectivity monitoring". In: *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*. IEEE. 2016, pp. 35–41.

[2] Bates, T. and Chandra, R. *BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*. RFC 1966. Internet Engineering Task Force (IETF), June 1996, pp. 1–12. URL: https://www.rfc-editor.org/rfc/rfc1966.txt.

[3] Bates, T. et al. *Multiprotocol Extensions for BGP-4*. RFC 2283. Internet Engineering Task Force (IETF), Feb. 1998, pp. 1–9. URL: https://www.rfc-editor.org/rfc/rfc2283.txt.

[4] Bates, T. et al. *Multiprotocol Extensions for BGP-4*. RFC 2858. Internet Engineering Task Force (IETF), June 2000, pp. 1–11. URL: https://www.rfc-editor.org/rfc/rfc2858.txt.

[5] Bates, T. et al. *Multiprotocol Extensions for BGP-4*. RFC 4760. Internet Engineering Task Force (IETF), Jan. 2007, pp. 1–12. URL: https://www.rfc-editor.org/rfc/rfc4760.txt.

[6] Biersack, E. et al. "Visual analytics for BGP monitoring and prefix hijacking identification". In: *IEEE Network* 26.6 (2012), pp. 33–39.

[7] Casoni, G. *EVPN-BMP-Listener*. 2020. URL: https://github.com/giacomo270197/EVPN-BMP-Listener (visited on July 1, 2020).

[8] Chen, E. et al. *Revised Error Handling for BGP UPDATE Messages*. RFC 7606. Internet Engineering Task Force (IETF), Aug. 2015, pp. 1–19. URL: https://www.rfc-editor.org/rfc/rfc7606.txt.

[9] Dutt, D. G. *BGP in the Data Center*. Aug. 2017.

[10] Dutt, D. G. *EVPN in the Data Center*. July 2018.

[11] Evens, T. et al. *Support for Adj-RIB-Out in the BGP Monitoring Protocol (BMP)*. RFC 8671. Internet Engineering Task Force (IETF), Nov. 2019, pp. 1–9. URL: https://www.rfc-editor.org/rfc/rfc8671.txt.

[12] FRR. *BMP — FRR 7.2 Documentation*. 2020. URL: http://docs.frrouting.org/en/stable-7.2/bmp.html (visited on July 1, 2020).

[13] Hawkinson, J. and Bates, T. *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. RFC 1930. Internet Engineering Task Force (IETF), Mar. 1996, pp. 1–10. URL: https://www.rfc-editor.org/rfc/rfc1930.txt.

[14] Hu, L. et al. "Net-cohort: Detecting and managing vm ensembles in virtualized data centers". In: *Proceedings of the 9th international conference on Autonomic computing*. 2012, pp. 3–12.

[15] Juniper. *Changing Duplicate MAC Address Detection Settings*. 2020. URL: https://juniper.net/documentation/en_US/junos/topics/task/configuration/configuring-mac-mobility-settings.html (visited on July 1, 2020).

[16] Lapukhov, P., Premji, A., and Mitchell, J. *Use of BGP for Routing in Large-Scale Data Centers*. RFC 7938. Internet Engineering Task Force (IETF), Aug. 2016, pp. 1–35. URL: https://www.rfc-editor.org/rfc/rfc7938.txt.

[17] Lougheed, K. and Rekhter, Y. *A Border Gateway Protocol (BGP)*. RFC 1105. Internet Engineering Task Force (IETF), June 1989, pp. 1–17. URL: https://www.rfc-editor.org/rfc/rfc1105.txt.

[18] Mahalingam, M. et al. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348. Internet Engineering Task Force (IETF), Aug. 2014, pp. 1–22. URL: https://www.rfc-editor.org/rfc/rfc7348.txt.

[19] Matplotlib. *Visualization with Python*. 2020. URL: https://matplotlib.org (visited on July 1, 2020).

[20] NetworkX. *NetworkX Documentation*. 2020. URL: https://networkx.github.io (visited on July 1, 2020).

[21] Nordström, O. and Dovrolis, C. "Beware of BGP attacks". In: *ACM SIGCOMM Computer Communication Review* 34.2 (2004), pp. 1–8. DOI: 10.1145/997150.997152.

[22] Orsini, C. et al. "BGPStream: a software framework for live and historical BGP data analysis". In: *Proceedings of the 2016 Internet Measurement Conference*. 2016, pp. 429–444.

[23] Putina, A. et al. "Unsupervised real-time detection of BGP anomalies leveraging high-rate and fine-grained telemetry data". In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2018, pp. 1–2.

[24] Sajassi, A. et al. *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*. RFC 8365. Internet Engineering Task Force (IETF), Mar. 2018, pp. 1–33. URL: https://www.rfc-editor.org/rfc/rfc8365.txt.

[25] Sajassi, A. et al. *BGP MPLS-Based Ethernet VPN*. RFC 7432. Internet Engineering Task Force (IETF), Feb. 2015, pp. 1–56. URL: https://www.rfc-editor.org/rfc/rfc7432.txt.

[26] Schlamp, J. et al. "HEAP: reliable assessment of BGP hijacking attacks". In: *IEEE Journal on Selected Areas in Communications* 34.6 (2016), pp. 1849–1861.

[27] Scudder, J., Fernando, R., and Stuart, S. *BGP Monitoring Protocol (BMP)*. RFC 7854. Internet Engineering Task Force (IETF), June 2016, pp. 1–27. URL: https://www.rfc-editor.org/rfc/rfc7854.txt.

[28] Sermpezis, P. et al. "ARTEMIS: Neutralizing BGP hijacking within a minute". In: *IEEE/ACM Transactions on Networking* 26.6 (2018), pp. 2471–2486.

[29] SNAS. *OpenBMP*. 2020. URL: https://github.com/SNAS/openbmp (visited on July 1, 2020).

[30] Traina, P. *Autonomous System Confederations for BGP*. RFC 1965. Internet Engineering Task Force (IETF), June 1996, pp. 1–7. URL: https://www.rfc-editor.org/rfc/rfc1965.txt.

[31] Vissicchio, S. et al. "Beyond the Best: Real-Time Non-Invasive Collection of BGP Messages". In: *INM/WREN*. 2010.

[32] Yan, H. et al. "BGPmon: A real-time, scalable, extensible monitoring system". In: *2009 Cybersecurity Applications & Technology Conference for Homeland Security*. IEEE. 2009, pp. 212–223.