Beyond the Best: Real-Time Non-Invasive Collection of BGP Messages

Stefano Vissicchio Luca Cittadini Maurizio Pizzonia
Luca Vergantini Valerio Mezzapesa Maria Luisa Papagni
Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, Rome, Italy
{*vissicch,ratm,pizzonia,verganti,mezzapes,papagni*}*@dia.uniroma3.it*

Abstract

Interdomain routing in the Internet has a large impact on network traffic and related economic issues. For this reason, BGP monitoring attracts both academic and industrial research interest. The most common solution for collecting BGP routing data is to establish BGP peerings between border routers and a *route collector*.

The downside of this approach is that it only allows us to trace changes of routes selected as *best* by routers: this drawback hinders a wide range of analyses that need access to *all* BGP messages received by border routers.

In this paper, we present an effective technique enabling fast, non-invasive and scalable collection of *all* BGP messages received by border routers. By selectively cloning BGP traffic and sending it to a remote monitor, we are able to collect BGP messages without establishing additional BGP peerings. Our technique does not require any new feature to be implemented by routers and we experimentally show that our approach incurs a negligible processing overhead at the border routers. Our prototype implementation is able to process and archive all BGP messages in near real-time on commodity hardware.

1 Introduction

The Internet provides connectivity among a great number of *Internet Service Providers* (ISPs) that exchange routing information via the Border Gateway Protocol (BGP) [21]. For inter-domain traffic, BGP has the final say on routing decisions and BGP messages received from neighboring ISPs can have a dramatic impact on the service actually provided by an ISP.

BGP monitoring enables ISPs to perform business-critical activities like troubleshooting and anomaly detection [18, 22]. Recently, it has been shown that BGP data can be exploited for business intelligence [14], traffic engineering [5], root cause analysis [8, 12], oscillation detection [10, 13], routing table analysis [16] and agreement compliance verification [11].

Despite such a rich set of potential applications, current BGP monitoring practices are quite limited: very often, they employ open source BGP daemon implementations to establish extra BGP peerings with border routers. The daemon acts as a *route collector*, in the sense that it collects information received via those extra peerings, dumps it in some format, and stores it for future analyses. For example, this is the approach adopted by Route-Views [20] to collect BGP data for the Internet community. Such a practice has two major drawbacks: *(i)* it is only able to collect those routes that have been selected as best by the routers that peer with the collector; and *(ii)* it is only able to collect BGP messages after ingress policy application, which can modify the messages.

Unfortunately, these drawbacks prevent exploiting the monitoring system for interesting applications like fine tuning of ingress policies, verification of Service Level Agreements and analysis of what-if scenarios [9]. Recently, the BGP Monitoring Protocol [23] has been proposed to overcome those limitations, but it is still experimental and requires software support on the routers.

In this paper we present a practical approach enabling real-time, non-invasive and scalable collection of all BGP messages received by BGP border routers. For this purpose, we exploit a usually overlooked feature that allows a router to selectively clone IP packets and send them to a remote collector. We make use of such a feature to copy every incoming TCP segment belonging to BGP sessions. After possibly reordering out-of-order segments, our collector parses the BGP messages and stores them in the standard MRT format [6].

By means of experimental evaluation on one of the cheapest commercial routers targeted to ISPs, we show that deploying our solution negligibly affects the performance of border routers with respect to traffic forwarding throughput, packet latency and router CPU usage. We show that our prototype implementation can monitor hundreds of BGP routers on commodity hardware. We also check the accuracy of the collected data. Finally, by comparing our approach to existing solutions, we show that our solution better fulfills the requirements we identify for an ideal monitoring system.

We also believe that the same approach can be adopted to monitor other signaling protocols. In this light, we consider this paper as a first step towards a centralized monitoring solution for the whole control plane.

The rest of this paper is organized as follows. In Section 2, we define the requirements we mandate for an ideal BGP monitoring system. In Section 3, we describe our proposal for a BGP monitoring system, outlining its architecture and discussing the most relevant components. Then, based on the requirements defined in Section 2, we evaluate our technique (Section 4) and we compare it with existing solutions (Section 5). We conclude in Section 6.

2 Requirements for a BGP Monitor

In this section, we describe a set of requirements that a BGP monitoring system should ideally fulfill.

Collection of non-best routes updates. BGP routers select a single *best* route among a set of candidates. Although non-best routes have no impact on where packets are forwarded, keeping track of them allows an ISP to better engineer its traffic and analyze what-if scenarios.

Policy independent data collection. An ideal collection system should reconstruct the original BGP messages as sent by neighboring ISPs, without being affected by the locally configured policies. This allows ISPs to decouple BGP data from BGP policies, so that policy changes cannot affect the consistency of historical data.

Real-time data collection. A BGP monitoring system should be able to collect data in real-time, or at least in near real-time. That is, a BGP update should be available for applicative analysis within few seconds.

Low impact on router resources. A typical constraint on management systems is to have a small impact in terms of extra resource demand (e.g., CPU usage, throughput and bandwidth) on the network infrastructure. This is especially true for BGP monitoring, given that BGP border routers typically have to forward huge amounts of traffic.

Cost-efficient deployment. To be realistically deployable in large networks, the monitoring system should be able to handle hundreds of border routers employing few machines equipped with commodity hardware.

3 Proposed Architecture

In this section we propose an architecture for a BGP monitoring system that aims at satisfying all the requirements listed in Section 2. The main idea is to mandate *border routers* to capture all the incoming TCP segments belonging to BGP sessions with eBGP peers and forward them to a remote *route collector*. The route collector is responsible for reassembling the TCP segments, decoding BGP messages and storing them in MRT format [6]. We show that this technique can be implemented using a feature commonly available on routers together with ad-hoc software employed on the collector side.

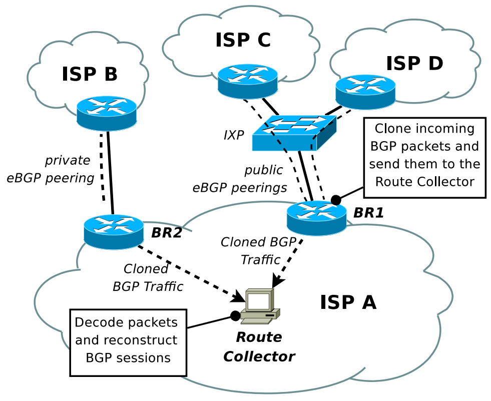

Figure 1: A deployment scenario of the proposed monitoring system.

Fig. 1 depicts the architecture of our solution in a typical deployment scenario. In this example, ISP A configures its border routers $BR1$ and $BR2$ to clone BGP packets and send copies to a remote Route Collector. Since packet cloning is performed before applying local policies, the route collector will receive BGP messages exactly as they are sent by eBGP peers. This feature allows ISP A to monitor what routes are announced by its peers B, C, and D. Of course, this approach supports private peerings between ISPs as well as peerings at public Internet exchange points (IXPs).

Fig. 1 highlights the role of the two main architectural components: the border router (BR) and the route collector (RC). We now provide details on each component.

3.1 Border Routers: Cloning BGP Traffic

The majority of ISP-targeted commercial routers provides the feature to clone IP packets and send copies to a remote machine. This is mostly used for copying traffic to Intrusion Detection Systems [3]. Leading vendors also provide filtering capabilities that allow operators to specify which packets must be cloned. To maintain a vendor-independent terminology, we will refer to this feature as *Selective Packet Cloning* (SPC). An SPC-enabled BR copies the packets received from user-specified *source interfaces* and matching an optional *filter* to another interface, which we call *destination interface*.

Depending on the capabilities of the device, a destina-

RITE Configuration Steps

Step 1 - Define a *filter* to select BGP traffic

```
7201(config)#access-list 100 permit
                            tcp any any eq bgp
```

Step 2 - Define a *destination* interface

```
7201(config)#ip traffic-export profile myPr
7201(config-rite)#interface vlan1
7201(config-rite)#incoming access-list 100
                                     mac-address <addr>
```

Step 3 - Select one or more *source* interfaces

```
7201(config)#interface ge0/0
7201(config-if)#ip traffic-export apply myPr
```

Figure 2: Steps for configuring SPC on Cisco routers.

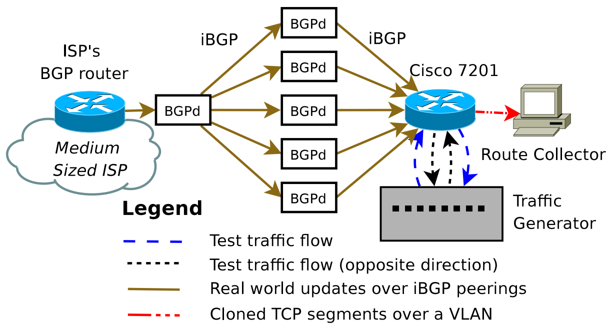

Figure 3: Testbed topology.

tion interface can be either a physical interface (e.g., an Ethernet interface), a VLAN interface (via 802.1q encapsulation), or a tunnel interface (e.g., IP-in-IP encapsulation or Generic Routing Encapsulation).

We now briefly describe the SPC feature as implemented in Cisco and Juniper devices. The cheapest Cisco devices targeted to ISPs (e.g., Cisco 7200 and 7300 routers) provide the *Router IP Traffic Export (RITE)* feature [3]. A RITE-enabled router can select packets received on certain interfaces applying IP- and TCP-based filters, and forward cloned packets over a VLAN interface. More expensive Cisco routers (i.e., 7600 series or greater) support the *Encapsulated Remote SPAN (ERSPAN)* feature [1], which provides a superset of the functionalities offered by RITE, e.g., the possibility to forward cloned traffic over a tunnel. Both RITE and ERSPAN can be used to implement the SPC feature on Cisco devices. Juniper's SPC support is called *Port Mirroring* [2]. Traffic received via user-specified ingress interfaces can be cloned and forwarded over a VLAN or a tunnel (IP-in-IP or GRE) interface.

Enabling SPC feature on BRs requires a very small amount of extra configuration. For example, Fig. 2 shows how to configure RITE on Cisco routers. Steps 1 and 2 only need to be performed once, while Step 3 has to be repeated for each of the BR's interfaces used for eBGP peerings.

3.2 Route Collector: Receiving, Reconstructing, and Storing BGP messages

Cloned TCP segments are sent from BRs to the RC which decodes and stores BGP messages. The RC performs the following activities.

Packet reception. The RC receives cloned packets and buffers them for further elaboration.

TCP stream reconstruction. Since the RC does not establish a TCP session with the BR, cloned TCP segments might arrive out of sequence. Therefore, for each eBGP peering the RC needs to reorder packets to extract the TCP stream. Duplicated segments are discarded. To keep resource consumption at the BR as low as possible, the RC silently ignores lost cloned TCP segments, if any.

BGP message decoding. The reconstructed TCP stream is analyzed to decode BGP messages and infer BGP session state changes.

BGP message storing. BGP messages and inferred state changes are stored (e.g., in MRT format [6]).

We developed a prototype RC that is based on the standard `tcpdump` utility for receiving cloned packets. We used `nice` to schedule the receiving process with high priority, and then send the received packets to a Perl script that is able to perform TCP stream reconstruction in pipeline. Finally, another Perl script takes the reconstructed stream in input and writes BGP messages in MRT format on a file.

4 Evaluation

In this section, we evaluate performance, accuracy and scalability of the proposed monitoring system.

In all the experiments we ran, we found that **no cloned packet was dropped** and **BGP messages were always correctly reconstructed and stored on disk**. Hence, we focus on the performance degradation at the BRs and on the scalability of the RC component.

We evaluate the router load in terms of frame loss (throughput), average CPU usage, and average packet latency. In our experiments, we used a Cisco 7201 router, referred to as *device-under-test* (*DUT*) in the following. The router is equipped with four Gigabit Ethernet ports, 1 Gigabyte of RAM, and a 1.67 GHz Motorola Freescale 7448 processor. We chose the Cisco 7201 because it is considered one of the cheapest router targeted to ISPs.

4.1 Performance of Border Routers

The network topology for our tests on BRs is depicted in Fig. 3. The DUT was connected to a traffic generator (a SmartBits 600B) using two interfaces. We use the traffic generator for stressing the router with high amounts of traffic (200 bidirectional IP flows) that the router handles using routes learned via BGP (using third-party next-hop).

On the third interface of the router, we set up five iBGP peerings, each providing a real-world update stream from a medium sized ISP announcing the full routing table (about $310,000$ prefixes). Namely, we interposed BGP daemons between DUT and the ISP, in order to amplify the original stream five times, as shown in Fig. 3.

We configured SPC such that incoming traffic belonging to the BGP peering was cloned on the fourth interface of the router over a VLAN. A packet sniffer was attached to the same VLAN and acted as a RC.

In this setting, the router was able to route packets with a negligible frame loss (less than 0.01%) for traffic **up to** 60% of the maximum packet rate obtainable on a full-duplex Gigabit Ethernet. Higher packet rates resulted in many more dropped frames. For this reason, we do not report results of tests made for higher packet rates.

We measured packet loss, average CPU usage and average latency at increasing packet rates. We ran tests both with SPC enabled and disabled and compared the results.

Fig. 4 reports the results of our tests. The y-axis shows the difference (in percentage) between the performance of the router when SPC is enabled with respect to when it is disabled. The x-axis represents packet rate. It is easy to see that activating the SPC feature essentially has no impact on the frame loss and on the average latency. The worst latency we recorded was 375 μseconds with SPC enabled and 301 μseconds with SPC disabled.

Differences for CPU load are small and highly dependant on the presence of BGP bursts. Anyway, activating SPC never affected CPU load for more than 2%.

We also ran a 5 minutes experiment sending traffic at a constant packet rate (45% of the maximum packet rate) while tearing down and bringing up the five iBGP peerings every minute, in order to generate huge BGP update bursts. Even under this extremely heavy load, the router dropped less than 0.005% of packets. More details can be found in [9].

4.2 Performance of the Collector Software

To assess the amount of resources required on the RC side, we captured five BGP sessions during the initial full table transfer (nearly 1.5 million prefix updates, 37,157 TCP segments, most of them of the maximum length). We separately measured the processing time needed for receiving the packets, reconstructing the TCP stream, decoding BGP messages and storing them in MRT format on commodity hardware (a laptop equipped with a dual-core 2.6 GHz CPU and 4G of RAM). We stress that

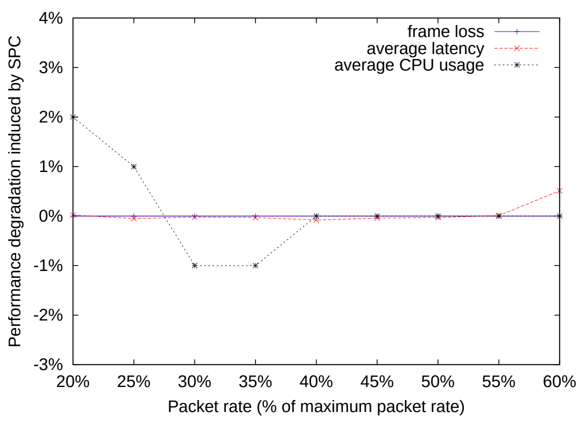

Figure 4: Performance degradation induced by SPC.

summing the measures we obtained in this experiment provides an upper bound on the performance that can be achieved by a RC, since processing times can be greatly enhanced by enabling pipelining and parallel processing.

We re-played the capture file with `tcpreplay` using the topspeed option on a 100Mbit ethernet link connected to our prototypical RC. Actual throughput was about 80Mbit/sec, much higher than the throughput of regular BGP sessions. Re-playing the capture file with `tcpreplay` took 3.38 seconds, while originally the BGP sessions lasted more than 2 minutes. A regular BGP session can reach such a high speed just sporadically. Even in this extreme experiment, we were able to capture all the packets with `tcpdump` and store them to an output file. TCP stream reconstruction from the output file took 2.6 seconds, while BGP session decoding and storage in MRT format took 1.7 seconds. Overall, a single prefix update was processed in less than 5.23 μseconds on average. Given that real world BGP sessions exhibit an average of less than 100 prefix updates per second, our prototype implementation can handle hundreds of BRs on commodity hardware. Of course, multiple RCs can be installed in the same network. However, given the performance of our prototype, we expect that even tier-1 ISPs need one or few collectors.

5 Related Work

Two naive approaches for BGP monitoring can be enabling debug option on router devices and installing optical taps and ad-hoc filtering boxes near each BR. However, debugging output stream "might render the system unusable" [4]. On the other hand, installing an optical tap and a dedicated filtering box for each optical fiber of each BR would be too expensive.

Existing approaches can be broadly classified in two categories: those employing some kind of route collectors to which BGP messages are pushed by border routers, and those adopting separate protocols to pull BGP information from the routers.

The typical architecture of a BGP monitoring system belonging to the first category essentially consists in a route collector, deployed inside the network, that is configured to maintain iBGP peerings with every BR. Quagga [17], OpenBGPd [7] and PyRT [19] are probably the most famous and widespread tools to set up a route collector this way.

BGP monitoring systems based on separate management protocols are designed to pull information from routers. In particular, SNMP has a number of MIB objects that are dedicated to BGP monitoring activities [15]. Often, operators pull information by *screen scraping*, i.e., using software that connects to the device, e.g., via Telnet or SSH, issues a specific command, e.g., `show ip bgp`, and collects the output.

Recently, a new ad-hoc protocol has been proposed in the IETF (the BGP Monitoring Protocol, or BMP) [23]: it is based on the idea of sending received BGP messages via a TCP connection with a monitoring station.

5.1 Comparison with Related Work

Table 1 summarizes the main differences between our approach and existing solutions. In the following, we discuss them in more detail.

Collection of Non-Best Routes Since Quagga, OpenBGPd, and PyRT rely on an iBGP peering, updates for routes that the BR does not select as best routes will never be collected at the RC. Non-best routes can be collected by screen scraping (e.g., via `show ip bgp` queries), and there exist SNMP managed objects for every route received. BMP and the solution we present in this paper are currently the only way to continuously monitor non-best routes.

Policy Independent Data Collection Quagga, Pyrt, and OpenBGPd can only monitor routes selected as best, and they are forced to collect BGP messages after ingress policy application. On the contrary, polling-based mechanisms typically provide a way to gather BGP messages as they are before BGP filters are applied (see [15] for SNMP based mechanisms). Both BMP and our approach also allow an ISP to collect policy independent data.

Real-Time Collection Solutions that employ additional iBGP peerings, such as Quagga, OpenBGPd and PyRT, are, in principle, capable of collecting BGP messages in real time. However, if messages are dumped periodically, additional delay is introduced before data are available for an application to analyze. For example, Quagga can dump BGP data not faster than one file per minute. Real-time is of course unfeasible with SNMP and other polling-based mechanisms: their usage is restricted to periodic snapshots of BGP routes received by BRs. The current BMP specification asserts that BMP messages "are not real time replicated messages received from a peer" [23]. Section 4 shows that our approach can collect data in near real-time.

Low Impact on Router Resources Handling an iBGP peering is a lightweight task for a BR, hence solutions based on Quagga, OpenBGPd, or PyRT do not put stress on routers. On the other hand, polling-based solutions employing SNMP or screen scraping heavily affect the performance at the BR, since it must process the whole BGP table and send a snapshot to the monitor. Our experimental tests show that our approach affects the performance of the BR only minimally, see Section 4. We expect that also BMP has a low impact on router resources in most of the cases. See Section 5.2 for a more detailed comparison between BMP and our solution.

Cost efficient deployment Since Quagga and OpenBGPd emulate a real router, CPU cycles and memory are wasted at the route collector for activities that are useless to a BGP monitoring system, e.g., performing the best route selection process. This makes them unable to handle a large number of peers providing a full Internet routing table. PyRT is not affected by this problem since it only implements a minimal set of features, disregarding activities that are not relevant to the monitoring system. Since SNMP and screen scraping have no real-time constraint, a single monitor could be able to handle hundreds of BRs. The performance study in Section 4.2 ensures that our approach and, reasonably, also BMP can handle hundreds of BRs on a single RC.

5.2 Comparison with BMP

Section 5.1 highlights that only our approach and BMP can reasonably be used in a monitoring system which aims at satisfying all the requirements listed in Section 2. However, BMP is not yet standardized and, currently, only JunOS versions later than 9.5 support BMP.

The main technical difference between BMP and our approach is that BMP relies on TCP while our solution forwards packets from BRs to the RC over IP tunnels or VLAN. Our solution is based on enabling SPC at the BRs and does not need any additional daemon to be run. SPC involves only switching capabilities (either software or hardware) which are usually highly optimized. On the contrary, BMP is not implementable using switching mechanisms, must rely on conventional TCP implementation, and usually requires an additional daemon.

Adopting TCP, BMP guarantees reliable delivery of copied BGP messages to the collector. However, it is not clear what the router resource consumption would be under extreme circumstances, e.g., when the RC tries to slow down the BR by shrinking the TCP congestion

	Quagga, OpenBGPd	**PyRT**	**SNMP**	**BMP**	**SPC**
collection of non-best routes	no	no	yes	yes	yes
policy independent data collection	no	no	yes	yes	yes
real-time data collection	no	no	no	almost	yes
impact on router resources	very low	very low	heavy	very low	very low
cost efficient deployment	no	yes	yes	yes	yes

Table 1: Comparison between our solution and related work with respect to the requirements defined in Section 2.

window. Our proposal does not mandate the router to maintain any state. Observe that RCs can easily check whether some TCP segments are missing by analyzing sequence numbers of cloned traffic.

Essentially, our approach pushes as much complexity as possible to the collector. The benefits are twofold: on one hand, a simpler router-side component results in precious resource savings; on the other hand, our solution is easy to extend to monitor other control-plane protocols than just BGP without requiring changes on routers.

6 Conclusions

We envision that enhanced BGP monitoring techniques can allow ISPs to make better high-level economic decisions about peerings and commercial agreements. Within this context, ISPs could offer and buy new connectivity services with Service Level Agreements that involve BGP updates. To support the above scenario, as well as better troubleshooting and other business intelligence analysis, we propose and evaluate an innovative technique for real-time collection of all BGP messages sent by BGP peers. Through experiments, we show that our approach accurately records the BGP updates received, it is easy to configure on current routers, it is scalable, and it has a negligible impact on the performance of the monitored border routers. As future work, we plan to extend this approach to monitor other signaling protocols and to deploy our solution in real networks.

Acknowledgements

We are especially grateful to the anonymous reviewers for constructive and insightful comments. This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, FIRB project "Advanced tracking system in intermodal freight transportation" and under Project "ALGODEEP: Sfide algoritmiche per elaborazioni data-intensive su piattaforme di calcolo emergenti", MIUR PRIN.

References

[1] Configuring local span, remote span (rspan), and encapsulated rspan (erspan). Cisco Systems, Inc. Official Cisco ERSPAN documentation.

[2] Configuring port mirroring. Juniper Networks, Inc. Official Juniper Port Mirroring Documentation.

[3] Router ip traffic export packet capture enhancements. Cisco Systems, Inc. Official Cisco RITE documentation.

[4] Using debug commands on cisco ios xr software. Cisco Systems, Inc. Official Cisco IOS XR documentation.

[5] BALON, S., AND LEDUC, G. Combined intra- and inter-domain traffic engineering using hot-potato aware link weights optimization. In *Proc. SIGMETRICS* (2008).

[6] BLUNK, L., KARIR, M., AND LABOVITZ, C. MRT routing information export format. Internet-Draft, 2009.

[7] BRAUER, H., AND JEKER, C. OpenBGPd. www.openbgpd.org.

[8] 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 *Proc. NOMS* (2008).

[9] CITTADINI, L., MEZZAPESA, V., PAPAGNI, M., PIZZONIA, M., VERGANTINI, L., AND VISSICCHIO, S. Beyond the best: Real-time non-invasive collection of bgp messages. Tech. Rep. RTDIA165-2010, Roma Tre Univ., 2010.

[10] CITTADINI, L., RIMONDINI, M., COREA, M., AND DI BATTISTA, G. On the feasibility of static analysis for BGP convergence. In *Proc. IM* (2009).

[11] FEAMSTER, N., MAO, Z. M., AND REXFORD, J. BorderGuard: detecting cold potatoes from peers. In *Proc. IMC* (2004).

[12] FELDMANN, A., MAENNEL, O., MAO, Z. M., BERGER, A., AND MAGGS, B. Locating Internet Routing Instabilities. In *Proc. SIGCOMM* (2004).

[13] FLAVEL, A., ROUGHAN, M., BEAN, N., AND SHAIKH, A. Where's Waldo? Practical Searches for Stability in iBGP. In *Proc. ICNP* (2008).

[14] GAO, L. On inferring autonomous system relationships in the internet. *IEEE/ACM Trans. Netw. 9*, 6 (2001).

[15] HAAS, J., AND HARES, S. Definitions of managed objects for BGP-4. RFC 4273, 2006.

[16] HUSTON, G. Analyzing the internet's BGP routing table. *The Internet Protocol Journal 4*, 1 (2001).

[17] K. ISHIGURO AND ET AL. Quagga. `www.quagga.net`.

[18] MAI, J., YUAN, L., AND CHUAH, C.-N. Detecting BGP anomalies with wavelet. In *Proc. NOMS* (2008).

[19] MORTIER, R. PyRT. `research.sprintlabs.com/pyrt`.

[20] OREGON ROUTEVIEWS PROJECT. `www.routeviews.org`.

[21] REKHTER, Y., LI, T., AND HARES, S. A Border Gateway Protocol 4 (BGP-4). RFC 4271, 2006.

[22] ROUGHAN, M., GRIFFIN, T., MAO, Z. M., GREENBERG, A., AND FREEMAN, B. IP forwarding anomalies and improving their detection using multiple data sources. In *Proc. SIGCOMM workshop on Network troubleshooting* (2004).

[23] SCUDDER, J., FERNANDO, R., AND STUART, S. BGP monitoring protocol. Internet-Draft, 2009.