

BGP Monitoring System

Yan Chen, Dave Matthews, Dan Massey

Computer Science Department

Colorado State University

Fort Collins, Colorado, USA

Email: {cheny, dvmthws, massey}@cs.colostate.edu

I. INTRODUCTION

Real BGP routing information is an essential resource for both researchers and operation communities in Internet routing. In order to collect large number of data in real time, BGP Monitoring System (BGPMon) is designed to monitor BGP updates and routing tables from BGP routers. It uses modular architecture to scalably monitor many BGP routers by distributed deployment while allowing a consolidated and neat interface to end users. BGPMon uses the Extensible Markup Language (XML) [1] as BGP data log format. This format can accurately record BGP data without any information loss and it is extendable for possible new features in BGP updates.

II. SYSTEM ARCHITECTURE

BGPMon aims for a scalable and robust system to accurately log all BGP messages from monitored BGP routers and provide the ability of real time data access. This system should be able to monitor a large number of BGP routers and support various client applications. Additionally, it should be resistant to session failures and update bursts etc.

BGPMon adopts a modular architecture in its design. Each BGPMon can monitor BGP data and provide data access service independently. Or a number of BGPMons can monitor different routers and interconnect with each other in a distributed mode. More specifically, a BGPMon is designed to support the following functional requirements.

- Reliably connect to multiple routers to silently get BGP data.
- Maintain a RIB table for each router that it monitors and update it with received updates.
- Provide BGP data to other BGPMons and end users upon requests, or request BGP data from other BGPMons.

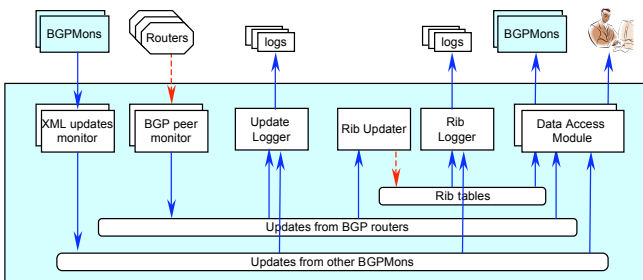


Fig. 1. BGPMon system components

As shown in Fig. 1, the implementation of a BGPMon consists of several event-driven modules connected by two queues of updates. The system input can be BGP updates from BGP routers or BGP data in XML from other BGPMon. And the output is real time BGP data to a client application or another BGPMon. More specifically, the BGP Peer Monitor module peers with BGP routers and enters received updates to the router update queue. While XML Update Monitor gets BGP data from other BGPMons and inputs them into the BGPMon update queue. The RIB Updater module applies each router update to the RIB tables of the routers it monitors. The Update Logger module copies all updates to a log file. The RIB Logger module copies the RIB tables (either the updated tables for routers or the XML updates from other BGPMons) to a log file on a periodic basis. Finally, the Data Access module provides a copy of a peer RIB table or the update stream as the request from the client application or other BGPMon.

The main concern in BGPMon design is scalability, which gives it the ability to monitor large number of BGP routers. Each router requires one TCP connections for a BGP peering session and a table for its *RIB-Out*. Currently, there are about 200,000 entries in the *RIB-Out* of a BGP router. To maintain such a huge number of TCP connections and tables is a big challenge. This scalability requirement is the reason why we design the BGPMon chain-able, as shown in Fig. 2.

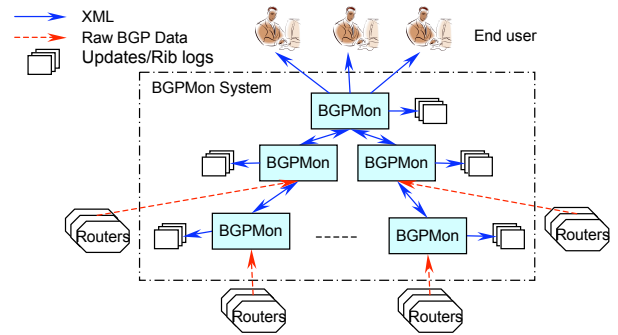


Fig. 2. BGPMon system architecture

In this Figure, each BGPMon only monitors some of the local BGP routers and maintains their rib tables. A BGPMon is configured as a reception to accept data access requests from various applications. It does not monitor any BGP router. It simply passes the request to other connected BGPMons.

The BGPMon with the requested data will reply and send the result back. In this way, all end users use the same interface from one BGPMon to access all monitored routing information without knowledge of the internal structure. This flexibility and simplicity allow regional deployment of BGPmons to support local routers and logs, while also allowing a consolidated global view.

III. LOG DATA FORMAT

The logged data of BGP updates and RIB table messages should be complete and accurate. More precisely, it is the full record of everything happened and can be used to replay history. In our design, BGPMon takes advantage of Extensible Markup Language (XML) format to provide a simple yet very flexible text-based log. Fig. 3 shows an example of a BGP update logged in a XML format. This format has the following advantages.

- All BGP information can be completely recorded and ready for replay. Also XML format files can easily be decorated with additional information.
- XML logs are easily extensible. An *unknown* field will be used to store the new BGP update feature in ASCII format in XML log. Thus the data can be saved without any loss or misdata.
- XML logs are human and machine-readable. By using CSS or XSL, BGP data in XML can be easily displayed on the web.
- XML is a common interface to other applications. XML files can be processed by a variety of existing tools.

```
<?xml version="1.0"?>
<bgp>
<message>
  <time>2007-03-22T19:00:07Z</time>
  <source_as>65001</source_as>
  <source_ip>129.82.138.4</source_ip>
  <destination_as>65009</destination_as>
  <destination_ip>129.82.47.109</destination_ip>
  <address_family>1</address_family>
  <interface_index>0</interface_index>
  <update>
    <path_attributes>
      <origin>
        <transitive/>
        <igp value=0/>
      </origin>
      <as_path>
        <transitive/>
        <as_sequence>65001 14041 3356 22351 </as_sequence>
      </as_path>
      <next_hop>
        <transitive/>
        <value>129.82.138.4</value>
      </next_hop>
    </path_attributes>
    <nleri>
      <prefix>82.206.163/24</prefix>
    </nleri>
  </update>
</message>
```

Fig. 3. BGP update in XML format

Another popular BGP data log format is Multi-threaded Routing Toolkit (MRT) [2]. It uses a binary format to record

BGP updates and RIB table messages and attach additional information such as timestamps, source and destination etc. However, current MRT also has some disadvantages as follows.

- MRT is represented in a binary format. To be human-readable, MRT data has to be converted to ASCII by some tools such as bgpdump [3]. Yet some information may be lost during the conversion. For example, current bgpdump ignores SAFI information in the BGP updates.
- MRT RIB log format is inefficient. In MRT, RIB tables are logged by prefixes. In other words, each record in the log contains one prefix and its path attribute. Given about 2/3 duplicated path attributes in a RIB table, MRT wastes lots of space and may increase processing costs.

Format	Raw File Size(Bytes)	Ratio	Compressed File Size(Bytes)	Ratio
XML	37,986,143	7	713,405	0.83
bgpdump	14,617,146	2.7	812,071	0.95
MRT	5,398,008	1	859,280	1

Fig. 4. The sizes of BGP update log in various formats

From storage perspective, XML seems to require more space than MRT since it contains lots of redundant information. However, our analysis shows that the compressed BGP updates in XML format require less space comparing with MRT format, although the uncompressed ones require more. Fig. 4 shows the sizes of the logs containing 15-minute BGP updates in various formats. Before compression, the size of XML log is 7 times bigger than that of MRT log. After using the default compression parameters for bzip2, the size of XML log is dramatically decreased to 83% of MRT's. Also note that BGPMon can write a single attribute with all of its associated prefixes in XML format rather than write individual attribute/prefix pairs in the MRT format. This will save 2/3 space of the rib table entries which have duplicating attributes. Since attributes consume the majority of space in a rib table, quite a bit of space is saved over the MRT format even before compression.

IV. SUMMARY AND FUTURE WORK

We have presented a scalable scheme BGPMon and used XML format to provide complete, accurate and reliable BGP data in real time. Our next step will focus on the following:

- The optimization of the module design to increase the processing speed and ability to handle the update burst.
- Adding metadata in the data log file to facilitate efficient data access.

REFERENCES

- [1] <http://www.w3.org/TR/2006/REC-xml-20060816>
- [2] L. Blunk, M. Karir, and C. Labovitz. *Mrt routing information export format*. Internet draft, Internet Engineering Task Force, 2005. <http://www.ietf.org/internetdrafts/draft-ietf-grow-mrt-00.txt>.
- [3] <http://www.ris.ripe.net/source/>