BGPRV: A Library for Fast and Efficient Routing Data Manipulation

Kevin Butler, Sophie Y. Qiu*, and Patrick D. McDaniel
Dept. of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
{butler,mcdaniel}@cse.psu.edu

*Dept. of Computer Science
The Johns Hopkins University
Baltimore, MD 21218
yuqiu@cs.jhu.edu

***Abstract*— BGPRV is a tool aimed to aid the analysis of BGP updates or routing table snapshots. It provides a set of library functions that make it possible to retrieve and process archived BGP data with efficiency and convenience. It encapsulates the functions of scanning the Route Views route repository, downloading data for specified time frame, processing the binary MRT format, and filtering incomplete or undesired data etc., and returns BGP data as single stream. With the abstraction of operations and simplified usage, it provides users with clean and organized BGP data that is ready for further processing and analysis.**

I. Introduction

BGP is *the* protocol that drives contemporary inter-domain routing [7]. Its performance is one of the key elements contributing to the success of the Internet. To date, there have been numerous efforts devoted to BGP related measurements and dynamic behavior analysis. Obtaining real-time information about the global routing system from the perspective of tens of different viewpoints around the world, Route Views data [12] have become one of the most important resources to Internet operators, networking community, and academic researchers.

Route Views acts as a route reflector for over 40 different ASes, from around the world. The routers collect both RIBs and UPDATEs and provide data in Cisco and MRT formats. RIBs are routing table snapshots and UPDATEs are BGP update messages. Route Views collect RIBs every two hours and UPDATEs every 15 minutes. The archived data are extensively used for the study and analysis of the Internet, for example, Internet topology and hierarchy, address usage and prefix advertisement, routing table growth, AS relationship inference, BGP instability, and convergence etc.

Collection and manipulation of Route Views data has been an interesting problem in itself. Tools have been built to periodically download RIBs and dampened routes [3], to support automatically queries [9], to convert the MRT format RIBs and UPDATEs to ASCII format [1], [5], or to parse and organize the collected data [11].

With the BGPRV toolset, we have created an easy to use set of functions that encapsulate many disparate tasks and hide low-level details from users, making it possible to collect and manipulate Route Views data with ease. Of chief importance, BGPRV is written in Perl for platform independence and its usage is simple. With a single command that specifies the period of interest (start time and end time), the tool takes care of tasks such as scanning the Route Views website, replicating the desired routing data, translating the binary format, and generating streams of BGP updates from files. BGP data can be collected and processed efficiently. It can run on either online or offline modes, based on the requirements. Moreover, the tool is not limited to manipulate Route Views data. It can be applied for collecting and parsing other MRT formatted data such as RIS data [10] and the logs from any Zebra or Quagga router that output data in MRT format.

BGPRV has been used in research on the address use structure and advertisement stability characterization in the Internet [8], origin authentication [2], and routing validation system [6]. We expect its usage will be further broadened in the Internet routing community.

II. Overview of BGPRV

Route Views data are stored as many individual files and the data are organized based on the year, month, and smaller time units when they were retrieved from the peering routers. BGPRV (see Figure 1) scans the Route Views website periodically and keeps an updated record for all the available data. It generates an entry for the URL that points to the stored Route Views data and puts them in a file named "rv_state.dat". Each entry is associated with a time stamp, representing the last time the file was examined. Because the full list of files that comprise the repository is kept, BGPRV can detect if a file is missing or has been deleted, and reacquire the file. Given the specified start time and end time, BGPRV pulls the individual files stored at Route Views and returns a sequence of hashes which point to tokenized records. The downloaded data are stored in

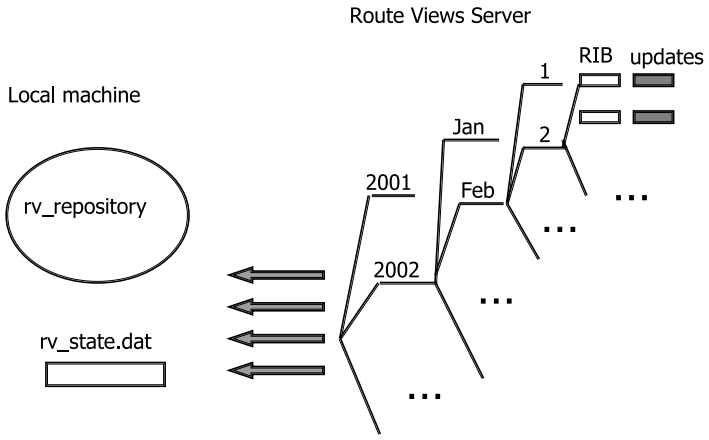

Fig. 1. BGPRV: a tool to collect and manipulate Route Views data

a repository named "rv_repository". BGPRV provides abstraction from individual directories and individual files and allows the user to read the BGP data as a single file. It decompresses the downloaded file, converts MRT format to ASCII format, and returns BGP data stream for the specified period, while hiding all the implementation and operation details from the user.

III. TOOLS

The major functions provided by BGPRV include the following:

A. GETRV

Using LWP modules, *getrv* first scans the Route View website, then download all files for the specified period. The files are mirrored in a directory in the current directory "rv_repository", which must be created before running the script. To run the utility, use the following parameters:

getrv <start time> <end time>

where the time is in the following format: "MM/DD/YY HH:MM:SS". For example, $getrv "01/01/03 00:00:00" "12/31/03 23:59:59" retrieves all the UPDATE data for 2003.

B. BGPPDUMP

bgppdump provides similar function as Tim Griffin's bgpdump tool [4], but does so in a platform-independent manner through the use of Perl, while extracting only the pertinent information from the BGP data. To run the utility, use the following parameters:

bgppdump [-f] <start time> <end time>

where the time is in the format "MM/DD/YY HH:MM:SS". The "-f" is optional, and indicates that the utility should operate in offline mode (e.g., no scanning or obtaining files over the web). The output of the tool is similar to that found in the normal bgpdump. However, only UPDATE and WITHDRAW data is reported, and all community strings and most attributes are stripped. The output format for Updates is:

<time>|A|<serc IP>|<prot> |<prefix>|<path>

and the output format for WITHDRAWS is:

<time>|W|<src IP>|<prot>|<prefix>.

Note that any UPDATE containing an AS set or IPv6 address is ignored.

C. BGPSTAB

bgpstab examines the stability of BGP by analyzing BGP updates. To run the utility, use the following parameters:

bgpstab [-f] [-r <src>] <start time> <end time>

where the time is in the format "MM/DD/YY HH:MM:SS". The "-f" is optional, and indicates that the utility should operate in offline mode. The "-r" parameter indicates the viewpoint of the test. Where specified, all announcements not from that viewpoint are ignored. A second mode allows to restart a test in progress. The mode is invoked as

bgpstab -s <statefile>

and restarts a test. A statefile named with the test date range (so multiple tests can run at the same time) is created periodically (typically after processing every

250th UPDATE file). The file name is encoded in the format of YYYYMMDD.HHMM-YYYYMMDD.HHMM-bgpstad.dat. There is no need to give any other parameters, as the statefile contains all the original times, sources etc.

The output files are named by the ranges of times they cover as for the statefile, except the ending of the file indicates the types of data that is covered. *-ases.out is the AS centric output file. It gives the AS number and the events observed from this AS, including the number of prefixes, the number of origin change events, the number of direct AS to AS change events, the number of announcements, and the number of withdraws. Similarly, *-prefixes.out is the prefix centric output file. It gives the prefix and the events related to this prefix, for example, the number of ASes the have originated the prefix, the number of related origin change events, the number of related direct AS to AS change events, the number of announcements, and the number of withdraws.

The uptimes observed during the test is given by *-upfile.out, which contains a single observed uptime (continuous period when a prefix was available and originated by the same AS) per line. Similarly, *-downfile.out outputs downtimes (continuous period when a prefix was not available) observed during the test.

bgpstab also outputs the overall statistics results. *.out contains the information such as completion date, test coverage, the specific viewpoint for this test (All announcements for non-viewpoints are ignored), and the number of UPDATE messages that the stability program observed[1]. Moreover, it also gives the number of messages whose origin is IBGP (PROT_IBGP), the number of messages whose origin is EBGP (PROT_EBGP), the number of messages whose origin is reported as being incomplete (PROT_ICOM), the number of messages whose origin is unknown (PROT_UNKNOWN), e.g., no origin attribute included in the message, and unfiltered sources which are the message counts for all sources in the test data (if source is selected, many of these are ignored).

IV. Performance

As an example to illustrate the performance of BGPRV, we run bgppdump in offline mode on an Apple XServe with dual 1.8 GHz G5 processors and 4 GB of RAM, connected in a RAID-5 configuration by Fibre Channel to an Apple X-Raid disk array. We processed the archived Route Views BGP updates for a one-month period (Jan 2005). The program took 1390.76 CPU seconds to process 211,970,676 announcements and 22,183,935 withdrawals. Much of the computation time was spent inflating the archives, which are compressed 95% compared to their full-sized equivalents. The size of the generated update stream is 16GB, with all community and other irrelevant attributes stripped and incomplete information filtered.

[1]Note that the number of UPDATEs will be less than the number of announcements, as many withdraws or announcements could be included in the same message.

V. Summary

In this paper, we introduce BGPRV, a tool which makes it possible to retrieve and process MRT-formatted BGP data with ease. Hiding implementation and operation details, BGPRV encapsulates the functions of scanning Route Views website, data downloading and decompression, converting MRT format to ASCII format, and allows the user to read BGP data as single stream instead of many individual files. Moreover, BGPRV is platform independent, efficient, and simple to use.

References

[1] http://www.cs.purdue.edu/homes/fahmy/software/as/documentation.html.

[2] W. Aiello, J. Ioannidis, and P. McDaniel, "Origin Authentication in Interdomain Routing," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*. ACM, October 2003, pp. 165–178, washington, DC.

[3] BGP data collector, http://www.routeviews.org/scripts/collector.

[4] BGP tools, http://nms.lcs.mit.edu/software/bgp/bgptools/.

[5] M. d'Itri, "Perl dump parser scripts," http://www.linux.it/md/software/zebra-dump-parser.tgz.

[6] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin, "Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing," in *Proceedings of Network and Distributed Systems Security 2003 (NDSS)*. Internet Society, February 2003, pp. 75–85, san Diego, CA.

[7] K. Lougheed and Y. Rekhter, "A border gateway protocol (BGP)," RFC 1163, June 1990.

[8] S. Qiu, P. McDaniel, F. Monrose, and A. Rubin, "Characterizing address use structure and stabillity of origin advertizement in interdomain routing," in *11th IEEE Symposium on Computers and Communications*, Pula-Cagliari, Sardinia, Italy, June 2006.

[9] Rancid Clogin, http://www.shrubbery.net/rancid/.

[10] RIS raw data, http://www.ripe.net/ris/rawdata.html.

[11] C. A. tools, http://www.caida.org/funding/routing/atoms/.

[12] University of Oregon Route Views Project, http://www.routeviews.org/.