BGPmon: Using Real-Time Data in Research and Operations

1 Introduction

Monitoring BGP[1] routing is important for both operations and research. To provide access to BGP data, a number of public and private BGP monitors are deployed and widely used [2, 3]. BGPmon is part of the Oregon RouteViews project[2] and uses a publish/subscribe overlay network to provide real-time access to vast numbers of peers and clients. For over decade, RouteViews has collected BGP routing updates and BGP routing tables from routers around the globe. This data, in MRT format[4], is publicly available from *http://archive.routeviews.org*. The archives include historical data dating back a decade as well as relatively recent data (within the last few minutes/hours). BGPmon builds on this successful system and extends this infrastructure in three ways.

First, BGPmon provides a real-time feed of both BGP updates and routing tables. Instead of downloading data from a site (which could incur delays of hours), a user can simply open a TCP connection and receive that data in real-time (which incurs a delay on the order of network propagation times). This change to easily accessible real-time data opens up a range of new opportunities for tools and live analysis.

Second, BGPmon provides the data in an XML format. The format includes both binary "bits off the wire" attributes and more human/parser friendly ASCII text in the same message format. Since the format is XML, users can easily add new tags while still maintaining a consistent format that can be shared between sites. A site might tag particular messages with a local attribute (such as my prefix or some other label meaningful only to that site). Other sites can still use the data, either stripping the tag before exchanging data if that tag has private information or simply sending the data with the new tag as other XML parsers can simply ignore unrecognized tags. Similarly, a site can easily drop some tags from the format.

For example, BGP messages must include a timestamp. However different applications may desire different types of time representations. In some cases, a unix timestamp is the preferred representation and the MRT format uses a unix timestamp. But while timestamps are convenient for coding, other applications may want a finer granularity of milliseconds. In yet another example, if the data is to be viewed by humans rather than processed by code, a human friendly representation listing year, month, day, and time is preferred. The XML format supports all three ways of displaying time. In fact, the XML stream includes all three time representations (using different XML attributes in the <TIME> section of the message). Applications desiring one time format can simply use that attribute and ignore the others. Applications can also strip out the other time representations to save space and later other users can reconstruct the human friendly time from the timestamp (or vice versa).

Third, for users gathering their own data or aggregating streams of data, BGPmon provides a new approach to data collection. Existing monitors typically collect data using a full implementation of a BGP router. In contrast, BGPmon eliminates the unnecessary functions of route selection and data forwarding to focus only on the monitoring function. In its place, BGPmon adds chaining functions so that feeds can better scale and can be combined to a large-scale mesh monitoring infrastructure. For example, one can take the real-time feed from RouteViews, chain it with feeds from local BGPmon, and direct the results into a few servers that provide the organization with real-time BGP access.

2 Background

Before introducing our new BGP Monitoring system, we will review already existing Routing Collectors (RC) in the Internet. Oregon RouteViews [2] and RIPE RIS[3] projects provide a routing data to interested researchers and operators by establishing a BGP peering agreements with different ISP's around the world. Typically, RC is simply another BGP peer router, it does not advertise any routes to peers, it receives and logs the BGP messages from neighbor ISP's.

Currently, RouteViews project provides update files that are roughly 15 minutes in duration and provides routing table snapshots every 2 hours. This is sufficient for analysis of past events, but real-time monitoring of BGP activity requires update files be available in seconds. For example, BGP prefix hijack alert systems would like to detect a potential route hijack within a few seconds. At best, today's RouteViews system only allows hijack alert systems to report hijacks that occurred many minutes ago.

In addition to providing data in real-time, an ideal BGP monitoring system would scale to dramatically increase the number of peers providing data. Given data from more locations, BGP analysis systems and tools could potentially provide better answers. For example, a BGP prefix hijack may only be visible in a small portion of the network and ideally one would like to have a monitor present in that same portion of the network. Thus our goal is not only to make the data available in real-time, but also to dramatically increase the volume of data available.

In summary, already existing Routing Collector systems are useful, but it would be useful to make the routing data available in real-time, provide the data in an extensible XML format, and simultaneously increase the amount of data collected and dramatically increase the number of locations obtaining the data. All this should occur without lose of data fidelity.

3 BGPmon: Using the Data

3.1 Receiving Routing Data

BGPmon can provide a real-time routing event stream to a large number of clients. All routing events are integrated into two XML streams: update and RIB-IN streams. Both streams send ongoing messages in XML format. XML was chosen as the message format for the streams because it is extendable, for both clients and servers, and also readable by both applications and humans.

BGPmon peers with a number of routers (peers), either directly or via chains to other BGPmon instances. Each BGP update message[5] from a peer is converted to XML format and forwarded to the update stream queue. The resulting data can be filtered with simple parser libraries like LibXML2[6] or Expat[7]. People, interested in receiving XML stream of BGP update events, should establish the TCP connection to BGPmon server *live-bgp.netsec.colostate.edu* port *50001* or run simple telnet command to become familiar with XML message format.

BGPmon also stores RIB-IN tables on a per-peer basis. For each incoming BGP update message from a peer, BGPmon stores it in the peer's RIB-IN table. BGPmon periodically injects peer's route table into the XML RIB-IN stream. XML RIB-IN event stream is available at *live-bgp.netsec.colostate.edu* port *50002*.

Also, BGPmon periodically announce XML *status* messages to both the update and RIB-IN streams. *Status* messages provide additional data about the load of BGPmon internal functions and summary information of each peer, for example, number of received BGP messages, prefixes, attributes and so on.

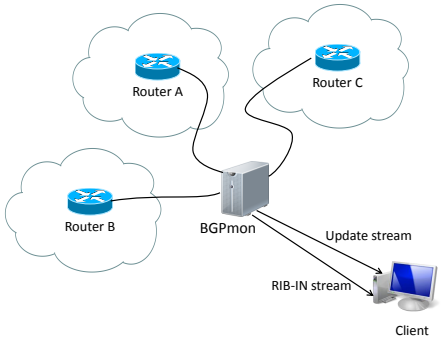

Figure 1. Receiving data from BGPmon

3.2 An Example Use

BGPmon clients are able to detect the possible BGP prefix hijack attacks. For example, suppose ISP A owns *98.158.88/23*, *98.158.90/23* and was assigned AS1000. ISP A would like to monitor its prefixes and moreover, to receive an alert message when part or all network prefixes are announced by some misconfigured router with AS2000 number of ISP B (not shown in figure). To do it, ISP A should be a client of BGPmon and open two TCP connections to receive RIB-IN and update messages.

Figure1 shows a topology where BGPmon has BGP peering with three routers: Router A, B and C. Also, BGPmon has a client (ISP A). To use the data, ISP A should establish connection to RIB-IN stream and receive a RIB-IN table snapshot. By filtering *98.158.88/23* and *98.158.90/23* prefixes and AS paths from snapshot, ISP A may see the *current* routing picture how other routers (in our example, Routers A, B and C) in the Internet are able to reach ISP A subnets. In particular example, if Routers A,B and C can reach ISP A, their AS path should end with AS1000 number.

Next, ISP A should monitor the changes in AS path for its prefixes in real time. In order to do it, ISP A should establish a connection to update stream and constantly receive and filter XML messages. For instance, if some misconfigured router of ISP B will create a BGP announce message with *98.158.88/23* prefix to Router B (see Figure1), ISP A will be able to detect the problem easily by looking in AS path of Router B announce: the last AS number in AS path will be AS2000. Thus, by filtering real time XML data, ISP A is able to see *precisely* when prefix *98.158.88/23* was announced by malicious ISP B.

This example shows the combination of the freely available BGPmon data streams can be combined with simple XML parsing to detect prefix hijacking as well as monitor

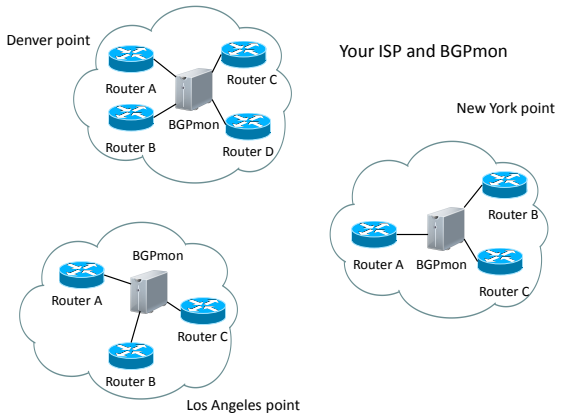

Figure 2. BGPmon mesh 1.

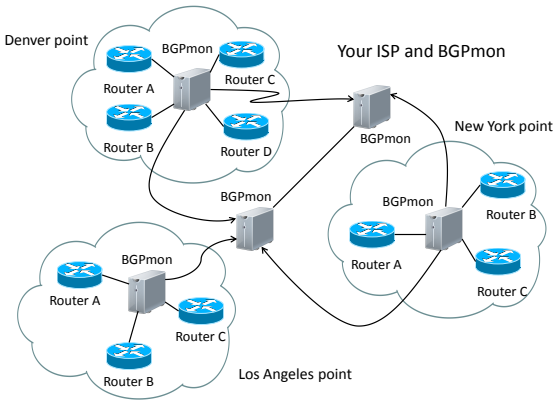

Figure 3. BGPmon mesh 2.

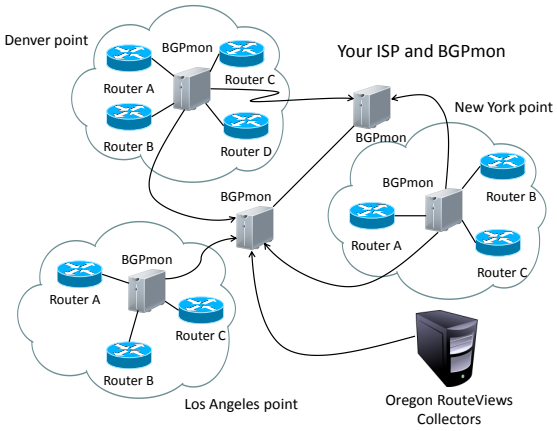

Figure 4. BGPmon mesh 3.

the status of ISP routes. The data can also be used a number of other ways and is limited only by the users ability to parse XML.

4 BGPmon: Deploying Your Own Monitors

While we have endeavored to design a BGPmon that scales to a large number of peers and clients, we allow BGPmon to scale out through the interconnection of multiple BGPmons. Figure 2 shows example of ISP with different router locations around the country. Los Angeles and New York locations has 3 core routers, while Denver location has 4 core routers. All of them runs BGP peering with neighbor peers (not shown in Figure). In this example, we have 3 BGPmon monitors installed in each location. Clients, who's interested in live routing data available at New York location, should subscribe (open a TCP connection) to BGPmon in NY area.

Moreover, BGPmon is able to to peer with each other monitors and form an overlay network (mesh). Figure 3 shows a improved network topology. In our example, we have added two BGPmon core monitors. All BGPmon instances in three locations monitor a unique set of peers and forward their events to BGPmon core monitors. Each BGPmon core monitors will log the event stream and forward their events to any clients attached. Clients can subscribe to BGPmon core monitors and receive a single stream of routing events happening around the county.

Although BGPmon is able to peer directly with routers, there are exist other Routing Collectors in the Internet (see Section 2). BGPmon is able to work with these external RCs with a few modifications to the RC software. Currently, Oregon Routeviews[2] collectors provide a routing data to BGPmon. Figure 4 shows an example, how, early described ISP network topology, can be modified to receive routing events from RouteViews. This example describes how BGPmon enables scalable real-time monitoring data distribution and create an overlay network (mesh), which provides a single stream without modifying the monitors.

5 Conclusion

Overall, we believe BGPmon represents an important change in how BGP route monitoring is accomplished in the Internet. We hope that the addition of BGPmon will make it much simpler for researchers and operators to obtain BGP data and the addition of widely available real-time BGP data will lead to the development of new tools for better understanding Internet routing. If you would like to know more about the BGPmon project, please visit our web page *http://bgpmon.netsec.colostate.edu* or contact the public maillist *bgpmon@netsec.colostate.edu*

References

[1] "A border gateway protocol 4 (bgp-4)," http://www.ietf.org/rfc/rfc4271.

[2] "University of oregon route views project," http://www.routeviews.org/.

[3] "Ripe (rseaux ip europens) routing information service," http://www.ripe.net/projects/ris/.

[4] "Mrt routing information export format," http://www.ietf.org/internet-drafts/draft-ietf-grow-mrt-11.txt.

[5] "A border gateway protocol 4 (bgp-4)," http://www.ietf.org/rfc/rfc4271.txt.

[6] "The xml c parser and toolkit of gnome," http://www.xmlsoft.org/.

[7] "The expat xml parser," http://expat.sourceforge.net/.