

# bgpmon

## Real-time Collection and Distribution of BGP Updates

Dave Matthews, Yan Chen  
Department of Computer Science  
Colorado State University  
Fort Collins, Colorado, USA  
{dvmthws, chen}@cs.colostate.edu

**Abstract** — The Border Gateway Protocol (BGP) facilitates the exchange of routing information on the Internet. Routers send their peers updates to the routes for destinations as they change. The routers also maintain the current state of routes to all internet addresses in a local table called the Routing Information Base (RIB). Analysis of the BGP update and RIB information can help identify problems with the routing topology of the Internet. The current approach to collection and monitoring of update information is file-based - tools collect updates from various routers distributed throughout the internet and write them to files. The tools also periodically capture the RIB tables for each router and write them to a file. Applications obtain the latest RIB table and update files to recreate an initial state and update stream for analysis purposes. Real-time monitoring is not possible with the current implementation. This paper describes *bgpmon*, a new tool that allows real-time monitoring of BGP updates in addition to capturing the updates and table dumps in files for later processing. The *bgpmon* architecture and implementation attempt to address scale, robustness, and other issues present in the existing implementation. *Bgpmon* is a first step in the creation of a new monitoring infrastructure for BGP.

**Keywords**-*Border Gateway Protocol; BGP; Multi-threaded Routing Toolkit; MRT; Prefix Hijack Alert System; PHAS; RouteViews; NetViews*

### I. INTRODUCTION

*Bgpmon* was a course project for CS557 Advanced Networking at Colorado State University. The goal of the project [1] was to create a tool that peers with a CISCO router, receives Border Gateway Protocol 4 (BGP-4) [2, 3, 4] updates and sends the updates in MRT [5] format to the Prefix Hijack Alert System (PHAS) [6, 7] for processing. Key aspects of the course project included:

- Connect to a CISCO router, receive updates, and forward these updates to an application in MRT format.
- Handles session failures, update bursts, etc. in the communications with the router so the monitor does not fail.
- Filter the updates sent to the client to minimize client processing and network traffic.
- Periodically refresh the routes from the router to minimize errors if the router provides this capability.

- Maintain connections to several peers and feed information to several applications.

While the focus of the course project was support for PHAS, the goal was also to design and build a robust, scalable system to support future research projects. Additional aspects of the course project included:

- Maintain RIB tables based on updates received.
- Monitor multiple routers.
- Feed multiple client applications.
- Log updates for later playback.

These additional capabilities coupled with specific improvements to address shortcomings in the existing RouteViews [8] implementation may allow *bgpmon* to play a role in the NetViews project [9].

The remainder of the paper focuses on the course project and future extensions. Section II provides background information on the key elements. Section III discusses considerations that affect the design. Section IV highlights noteworthy aspects of the implementation. Section V outlines future work to improve the tool and add capabilities to address other needs. Section VI concludes the paper.

### II. BACKGROUND

The project created a new monitoring tool based on BGP and the MRT format from the Internet Engineering Task Force (IETF) [10] that will integrate with PHAS to provide true real-time alerts. This integration will replace an existing file-based RouteViews integration and provide a component for NetViews, the proposed next generation of the RouteViews system.

#### A. Border Gateway Protocol (BGP)

BGP defines a protocol to exchange network reachability information between different Autonomous Systems on the Internet. The Internet is comprised of thousands of Autonomous Systems (AS) where each AS is a network of routers in a single administrative domain. The reachability information describes a set of network address prefixes reachable from particular point in the Internet and the route a packet might follow through the Autonomous Systems to reach that address. Each Autonomous System selects an appropriate

path to a specific address prefix from the alternatives offered from its peers using a set of local policies.

The network reachability information exchanged between peer AS are sent over a TCP connection in the form of BGP update messages. Each message includes a set of address prefixes withdrawn (no longer reachable), a set of attributes that includes a path, and set of address prefixes reachable via that path. Other messages in the BGP protocol handle maintenance of the session between BGP peers.

Each peer stores the attributes for each address prefix in Routing Information Bases (RIB). BGP peers typically maintain three RIB tables:

- RIB-In contains the routing information learned from peers.
- RIB-Local contains the routing information determined by the application of local policies to the RIB-In information. This is the basis for the routing decisions made by this peer.
- RIB-Out contains the routing information advertised to peers.

Bgpmon only requires a RIB-In table for each peer since it does not perform routing or advertise routes to its peers.

BGP also defines a set of events, a set of timers, and a finite state machine that governs the operation of each peer in the exchange of this information. In order to obtain the BGP update information from peer routers, bgpmon must also implement these aspects of the protocol.

#### B. MRT format

The MRT format defines a way to export and exchange routing information. The Multi-threaded Routing Toolkit [11] that provided protocol libraries and services for researchers and application developers originally defined this format. The IETF draft “MRT routing information export format” defines the MRT formats used in this project. The MRT format includes additional information not included in the BGP protocol:

- Timestamps, source, destination, and address type information to aid the analysis of routing information.
- Control messages identify the start and end of sessions.
- State change messages to identify significant events in the operation of the peer’s finite state machine.
- RIB table information to act as starting points in an analysis to which updates may be applied.

#### C. Prefix Hijack Alert System (PHAS)

PHAS is a web-based service that identifies possible prefix hijacks based on BGP update information collected by RouteViews. Prefix hijacks pose a serious threat to the Internet, preventing the delivery of network traffic to the intended destination. Prefix hijacks may be the result of a malicious attack or an inadvertent administration error.

PHAS uses a variety of techniques to analyze the BGP information provided by RouteViews to determine potential

prefix hijack attempts. The current implementation operates on a 3-hour delay, alerting prefix owners well after the event occurred. It is desirable to provide this information in real time so the prefix owner may take prompt action to address the problem.

#### D. RouteViews and Netviews

RouteViews is a project at the University of Oregon to provide Internet operators with a global view of routing information at different locations in the Internet. This information allows the operators to analyze the dynamics of routing changes and monitor routing table growth.

The information has proved useful beyond the original goals. Other projects use this information for research to guide the future development of routing protocols. The information also aids efforts to visualize paths, study address space utilization, and generate geographic locations for hosts.

NetViews is a proposed development effort intended to build the next generation of the RouteViews system. Key aspects of the system include a global route monitoring system that provides high quality, real-time data to the research, education, and network operations communities. NetViews will address issues identified during the operation of the current RouteViews implementation over the past several year.

- Improve robustness and recovery of BGP peering sessions.
- Include information to identify missing BGP updates.
- Improve the data integrity of logs so the routing table dumps reflect the stream of updates in the log.
- Improve support for IPv6.
- Provide better methods to access desired portions of the data.
- Provide real-time data access to support automated detection tools.

Bgpmon should benefit from the previous RouteViews experience and address these issues. The resulting tool may be a candidate for use the NetViews project.

#### E. bgpdump

Bgpdump [12] is a tool that parses files containing MRT format information. Researchers use bgpdump to produce a text version for analysis by other tools. Bgpdump is also useful for testing and debugging the format of files that bgpmon produces. It is readily available from a variety of sources on the Internet.

### III. SCALABILITY

The key consideration in the design of bgpmon is scalability. The initial project need only handle updates obtained from a single router and destined for a single application. However, to be useful bgpmon must adapt to a situation in which it monitors multiple routers to feed multiple applications. Deployment on more powerful servers may produce some degree of scaling, but it may be necessary to

support distributed deployment options such as those depicted in Figure 1.

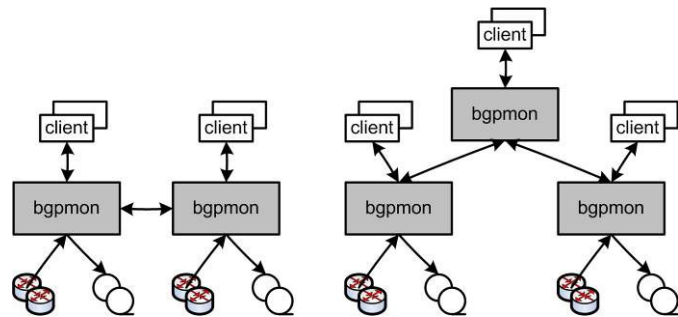


Figure 1. Bgpmon Deployment Options for Scalability

These simple examples suggest updates flowing between monitors to allow client applications to process updates originating from other monitors in peer or hierarchical fashion. This flexibility may allow regional deployments of monitors to support local routers, logs, and client applications while also allowing a consolidated global view. In a peer approach, the monitors could distribute the router monitoring function while providing applications with a combined feed of updates from all of the routers. In the hierarchical approach, a monitor could consolidate feeds from the regions for clients that monitor a global view of the updates.

To support this degree of scalability in a real-time environment, the implementation must concurrently process router updates, log the updates to a file, and distribute the updates to multiple clients in an asynchronous fashion. The design for bgpmon borrows its approach from the Staged Event-Driven Architecture (SEDA) [13] approach to support the massive concurrency demands of internet services.

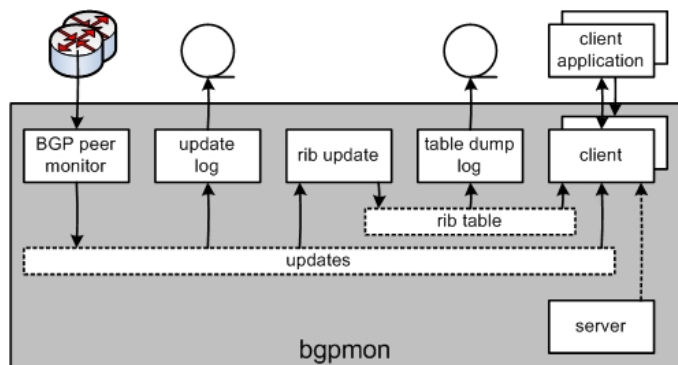


Figure 2. Initial Bgpmon Concurrency

The initial implementation of the bgpmon tool shown in Figure 2 consists of event-driven stages connected by a queue of updates. The BGP peer monitor stage enters updates into the queue for processing by each of the other stages. The update log stage copies all updates in MRT to a log file, periodically changing the log file to keep the size manageable. The rib update stage applies each update to the rib table maintained for the source of the update. The table dump log stage copies the rib table for all peers to a file on a periodic basis. Finally, the client stages may send a copy of a peer rib table or the update stream to the client application. The key concurrency issues in

this design revolve around the access to the updates in the queue and the peer rib tables.

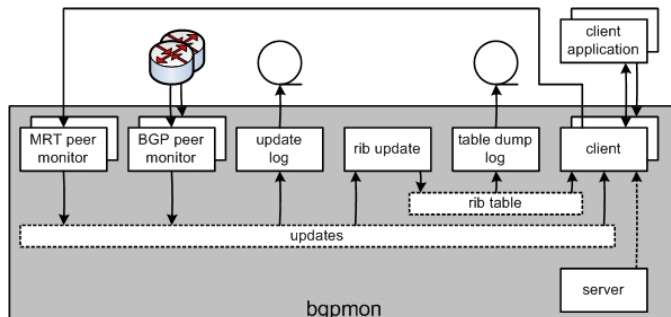


Figure 3. Future bgpmon Currency

This approach to concurrency allows a simple extension through the addition of an MRT peer monitor stage as shown in Figure 3. This new stage would be a client application that obtains updates from another bgpmon and places them in the queue for processing by the stages associated with this bgpmon. A flexible configuration module would support a variety of possible configurations of monitors, file logs, and client applications in different deployments.

The use of multiple BGP peer monitors may further minimize the impact of any one router on overall system function. Bgpmon could employ a separate table monitor for each peer.

#### IV. IMPLEMENTATION

Bgpmon compiles with the GNU C compiler and runs in both the Windows and UNIX environments. The SEDA approach used in the design suggests a threaded implementation to allow efficient sharing of the data in the update queue and RIB table. This threaded approach also performs well in typical network client/server applications [14, 15].

##### A. Bgpmon Server and Client Threads

The main program obtains the configuration information, creates the queue and RIB table structures, and initiates the threads for processing. The main thread becomes a TCP server that listens for client connections and creates a client thread to process each client request.

The server listens on a specified socket for a TCP connection from a client application and creates a client thread to process the client request. Four requests are possible.

- List of peer routers available for table dump.
- Table dump for a specific peer router.
- Update stream for a specific peer router.
- Update stream for all peer routers.

IP address and Autonomous System number identify peer routers in these requests. The client thread terminates the connection and exits after table dump related requests are complete. The client threads for update streams remain open until the client closes them. If the connection for an update

stream is lost, the client application must reinstate the connection.

### B. Thread Synchronization

A key challenge of threaded implementations is the locking mechanism used to synchronize access to the shared data in the form of the update queue and the RIB table. Bgpmon implements a different form of thread synchronization for each due to their different requirements.

The implementation of the update queue has a single writer that places new items in the queue and several readers that remove updates from the queue. In this case, every thread must process every update in the queue. When a thread exhausts its items in the queue, it must wait for new items while other threads continue to process their remaining items in the queue. The update queue uses a mutex and condition for locking to allow the lock to be dependent on whether there is any information in the queue for the thread. If the thread is successful obtaining the initial lock, it checks to see if any information is available for it. If no information is available, the thread waits on the condition to obtain the lock again. After the monitor adds new data to the update queue, the update queue broadcasts this condition to all threads so they may attempt another lock to process the new information.

The RIB table implements a read/write lock mechanism. This allows the RIB update thread to use a write lock on the table to prevent all reads from other threads when it updates the data in the RIB table. The client threads use read locks to allow simultaneous read access, but prevent the RIB update thread from obtaining a write lock to change the data. The RIB update thread may only obtain a write lock when no other threads have a read lock.

### C. Queue and MRT format

Bgpmon implements a single queue for updates rather than multiple queues to minimize the duplication of information in the implementation. The queue implements a publish/subscribe design pattern [16] where a thread may publish a new update to the queue to be read by all subscriber threads.

The publication must maintain the current location for each subscriber and a reference count for the number of subscribers remaining to read each update in the publication. When a subscriber reads an update from the publication, the queue returns a copy unless the subscriber is the last one to read the update in which case it receives the original item placed in the publication. This allows all threads to free the returned items when they have finished processing the information and insulates each thread from possible changes by other threads. Storage in the approach is proportional to the length of the queue for the slowest reader ( $n$ ), and a copy for each subscriber ( $m$ ), or  $O(n+m)$ .

Client thread failures may require cancellation of a subscription. These failures may result from the loss of connection to the client application or slow processing that result in the queue reaching its maximum size. In either case, the queue decrements all reference counts for the items

remaining in the subscription and frees all items no longer remaining in any subscriptions.

Bgpmon communicates all information received from the peer routers in MRT format. The BGP peer monitor prepends an appropriate MRT header to the BGP message before placing it in the queue. This information includes BGP open, update, keepalive, and notification messages. Bgpmon also places MRT control and BGP state change information in the queue to aid applications in their analysis of the update stream.

We chose to use MRT format for the queue because it provides the necessary information for RIB table updates (timestamp, source IP address and source AS) and can be written directly to the log file or client applications without any additional processing.

### D. BGP Peer Monitor

The BGP peer monitor thread maintains a TCP connection with each configured router. The routers send update messages containing address prefixes and their associated attributes as well as periodic keepalive messages. The monitor places all BGP messages received in the queue, including opens, updates, keepalives, and notifications. The monitor also places state change MRTs in the queue to reflect changes in the finite state machine for peer relationships.

Bgpmon sends periodic route refresh and keepalive messages to the peer routers. The route refresh message causes the peer router to send updates for all of its current address prefixes, allowing bgpmon to ensure the RIB table is up to date. The keepalive messages are required to maintain the session during periods of inactivity.

The peer router or bgpmon can send or receive a notification message that directs the other peer to cease sending messages and drop the connection. In addition, the TCP connections will occasionally fail. In both cases, bgpmon will reestablish the connection to the peer router to minimize the loss of data.

Bgpmon maintains timers associated with each router to determine when to send the keepalive messages, when to send the route refresh messages, and when to assume the connection is dead and reestablish the connection to the peer. The BGP protocol defines the keepalive and connection timers. The route refresh timer is an extension that uses the same approach as the keepalive timer with a much longer interval.

The implementation of this monitor is a single thread that uses the select system call to identify expired timeouts or peers with data available.

### E. RIB Table and RIB Update

Bgpmon maintains separate RIB tables for each peer router, modifying the table as updates arrive from that router. The RIB table reflects the current routing configuration of the corresponding router based on all prior updates received from the router.

BGP updates may associate many address prefixes with the single set of attributes and multiple BGP updates may contain

identical attributes. The RIB table maintains a mapping from the address prefixes to their associated attributes.

To facilitate efficient update processing and minimize storage requirements for the RIB table, the implementation uses a pair of hash tables – one for the prefixes and one for the attributes. This allows multiple address prefixes from multiple messages to share a single set of attributes. The hash tables allow collisions, but the hash tables sizes attempt to minimize the number of collisions. Each hash table entry is a linked list of the items that hashed to the same entry.

The rib update thread processes each BGP update, maintaining the RIB Table information with the attributes for an address prefix or withdrawing an address prefix. Extensions to the BGP update message such as MPREACH and MPUNREACH require analysis of the attributes themselves to identify additional address prefixes that must be updated or withdrawn.

#### F. Update and Tabledump Log

The update log thread writes each item in the update queue to a log file, switching log files at specified intervals. The switch is dependent on the timestamp in the update, not on the current system time. This way items remain in the log file associated with the appropriate time.

The table dump log thread writes MRT table dump messages to a log file for each of the peer routers at specified intervals. This thread locks individual hash entries in the address prefix table to synchronize with the rib update thread. The table dump log thread copies the hashed entries to a buffer and removes the lock before the entries are dumped to the log file.

Note that the current table dump format used to export RIB table information contains only a single address prefix and attribute pair. As a result, the table dump log explodes the routing information contained in the RIB table and BGP update messages. This results in very large files for table dumps.

Filenames for both log files are hour aligned for future periods. For example, log files written at 20-minute intervals will start on the hour and at 20 and 40 minutes past the hour after the initial file. This provides filenames that are easier to process by applications.

### V. RESULTS

The initial implementation exceeds the original project scope and addresses some of the issues identified with RouteViews.

- Bgpmon is a dedicated monitoring tool rather than a customization of a toolkit designed to support a wide variety of research.
- Bgpmon provides a stream of all BGP messages received and state change information for each peer to facilitate identification of gaps in the update stream.
- Bgpmon provides real-time access to the update stream, eliminating delays in the identification of

significant events on the Internet such as attempts to hijack an address prefix.

Bgpmon feature testing occurred in isolation and in concert.

- The BGP peer monitor supports multiple peer routers. Testing with seven peers placed only a light load on the server.
- The update log thread successfully writes MRT packets to a file for later processing. The files generated no errors when verified by bgpdump.
- The rib update thread maintains a rib table for each peer.
- The table dump log thread successfully writes MRT table dump packets to a file for later processing. The files generated no errors when verified by bgpdump.
- A simple client application is able to connect to the server and dump the stream of updates or table dumps to a file. Testing included multiple simultaneous client threads and dropped/restarted client connections.

The next step in the project is to integrate PHAS with bgpmon to verify operation in a production environment.

### VI. FUTURE WORK

The future work falls into two categories. The first area is deficiencies in the current implementation and testing. Some of these may require remedy before PHAS deployment.

- The initial test environment only includes a handful of CISCO 1700, 2600, 3600 routers. Access to a broader range and greater number of routers from multiple vendors is necessary to ensure the robustness of the monitoring function.
- The testing environment included only a single host with client applications residing on the same system as the server. Additional testing of other deployment options is necessary to characterize the behavior in a distributed environment.
- The configuration parameters for rib table and queue size require some study to determine suitable values for a wide variety of uses. These shared elements may automatically adjust their size as conditions dictate in future implementations.
- The address prefixes and attribute keys in the hash tables use a general-purpose hash function. Further study of hash distribution and collisions for these specialized keys may be necessary.
- The RIB table for a peer becomes invalid when the connection to the peer is lost. Bgpmon must destroy and recreate the table, or the information in the table must be marked invalid until new information arrives. The implementation does not currently address this issue.
- Slow client threads may cause the queue to reach its limit. Simply increasing the queue length does not solve the problem; it only delays the inevitable. Bgpmon must terminate slow client threads and clean up the queue to be a robust monitoring solution that

never drops the peer monitor, rib update, update log or table dump log threads.

- IPv6 support was not included in the initial implementation because the software on the test routers did not support IPv6.
- The implementation does not automatically convert the log files to a compressed format.
- While the system appears to operate with no concurrency problems, a code review is appropriate to ensure there are no hidden problems with system calls that are not thread safe.
- The system users a large amount of dynamic memory, but does not exhibit any obvious memory leak problems. Existing tools can verify this observation.

The second area of future work involves capabilities that expand the usefulness of bgpmon for large-scale deployments with PHAS, deployments involving other client applications, and potential use in the NetViews project.

- Additional testing to verify the number of supported peers and client applications in representative system configurations. The addition of an MRT peer monitor capability greatly increases the range of deployment configurations available and testing required to very scalability
- An XML client stream can simplify the analysis of routing information. Today, special purpose tools and scripts analyze MRT logs converted to text using bgpdump. The use of XML format for the routing information will make the data available to wider variety of analysis tools that are readily available. This approach will also make access to the information easier for a wider variety of programming languages since the compact MRT format entails a fair amount of bit twiddling to extract the information.
- The BGP peer monitor relies on the select system call to determine timeouts or availability of new information. This may lead to a situation where the monitor thread may choose a peer that shows data is available, but the monitor is unable to obtain the complete BGP message. In the implementation the read eventually times out, the peer connection is closed and processing occurs. While the system recovers from this situation, the timeout is undesirable. Multiple monitor threads will eliminate this situation, depending on system limitations.
- A new MRT format for RIB table dumps is necessary to overcome the duplication of information in the current format that only supports individual address prefix / attribute pairs. An approach to allow multiple address prefixes to share a common attribute specification similar to BGP update messages is

preferable. This approach provides a much more compact table dump log file. This new format will have significant impact on the current RIB table implementation and require a new data structure to allow navigation from an attribute to its associated address prefixes in addition to the current mapping from an address prefix to its attributes.

## VII. CONCLUSIONS

The project met the original goals to provide a real-time update stream for PHAS. The primary work remaining is integration and testing with PHAS for deployment.

The project also addressed some additional goals to extend its utility to other projects, such as NetViews. The current implementation does not address all of the issues identified with RouteViews. Some additional effort is required.

Bgpmon may prove a useful tool for future monitoring of BGP by allowing real-time analysis of updates.

## REFERENCES

- [1] D. Massey, "ATT PHAS implementation", personal communications, September 1, 2006.
- [2] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006 .
- [3] E. Chen. Route Refresh Capability for BGP-4. RFC 2918, September 2000.
- [4] R. Chandra, and J. Scudder. Capabilities Advertisement with BGP-4. RFC 2842, November 2002.
- [5] L. Blunk, M. Karir, and C. Labovitz. MRT Routing information export format. draft-ietf-grow-mrt3.txt, June 2006.
- [6] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A Prefix Hijack Alert System", In Proceedings of 15<sup>th</sup> USENIX Security Symposium, pages 153-166, August 2006.
- [7] PHAS: Prefix Hijack Alert System, <http://netsec.cs.colostate.edu/phas/>
- [8] RouteViews, <http://www.routeviews.org/>
- [9] NetViews, <https://netlab.cs.memphis.edu/protected/netviews/>
- [10] The Internet Engineering Task Force, <http://www.ietf.org/>
- [11] MRT Programmer's Guide. November 1999
- [12] bgpdump, <http://nms.lcs.mit.edu/software/bgp/bgptools/>
- [13] M. Welsh, D. Culler, and E. Brewer. SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. In the Proceedings of the 18<sup>th</sup> Symposium on Operation Systems Principles, October 2001.
- [14] W.R.Stephens and S.A. Rago. Advanced Programming in the Unix Environment, Second Edition. Addison-Wesley, 2005
- [15] W.R. Stephens, B. Fenner, and A.M. Rudoff. UNIX Network Programming, The Sockets Networking API, Volume 1. Addison-Wesley, 2004.
- [16] P.T. Eugster, P.A. Fleber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. ACM Computing Surveys 35, 2 (June 2003), 114-131.