Managing BGP Routes with a BGP Session Multiplexer

Vytautas Valancius and Nick Feamster
School of Computer Science, Georgia Tech

ABSTRACT

This paper presents the design, implementation, and evaluation of *BGP-Mux*, a system for providing multiple clients access to a common set of BGP update streams from multiple BGP peers. By providing multiple clients access to the same set of BGP feeds, BGP-Mux facilitates many applications, including: (1) scalable, real-time monitoring of BGP update feeds; (3) new routing architectures that require access to *all* BGP routing updates from neighboring ASes (as opposed to just the best BGP route for each destination); and (2) virtual networks running on shared infrastructure that share common underlying network connectivity. We have implemented BGP-Mux through by configuring existing features in the Quagga software router; we have deployed BGP-Mux on VINI and evaluated its scalability and performance in a controlled environment on the Emulab testbed.

1. Introduction

The Border Gateway Protocol version 4 (BGP) [12] is the Internet's interdomain routing protocol; it is used to exchange reachability information about global destinations between all the routers on the boundaries of independently operated networks, or autonomous systems (ASes). ASes, such as Internet Service Providers (ISPs) exchange reachability information about global destinations over BGP sessions. Managing these BGP sessions and routes generated by them is a burdensome task. Many applications and services could benefit from receiving up-to-date BGP routing information. For example, route monitoring applications for troubleshooting, security, and research may need to receive complete, up-to-date information about global reachability [8, 9, 15]. Virtual networks that are hosted on a shared platform also may each need to receive distinct BGP update streams [1]. Finally, new routing architectures that enhance management and control, such as the Routing Control Platform [2], may require additional visibility into the complete set of routes that are advertised from neighboring ASes.

Despite BGP's widespread use, arbitrary applications, networks, and monitoring services cannot typically obtain real-time BGP routing updates directly from routers, because these "update feeds" require direct sessions with production routers. Furthermore, because routers propagate only a single best route over their BGP sessions, even direct BGP sessions to the routers do not provide complete visibility into all of the BGP routes received by a router.

To enable on-demand, real-time BGP monitoring for these new classes of applications, this paper presents *BGP-Mux*, a Border Gateway Protocol Multiplexer. BGP-Mux provides clients with real-time, on-demand access to BGP sessions. Figure 1 shows the high-level design of BGP-Mux. BGP-Mux connects to upstream (or neighboring) ISPs with stable, persistent *upstream sessions*. Clients (or applications) connect to BGP-Mux over *client sessions*, which resemble the sessions a client would see if it had connected directly to the upstream ISPs. In this fashion, whereas typically a client domain connecting through a router would only see a single route to each destination, BGP-Mux allows each downstream client to receive *multiple* routes to each destination as if it had direct BGP sessions with upstream ISPs.

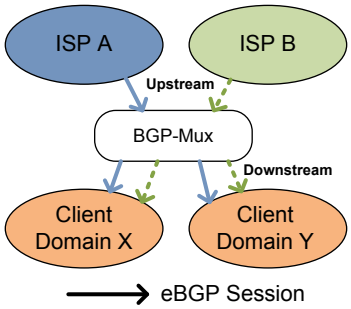


Figure 1: High-level design for BGP-Mux.

BGP-Mux "clients" may be real-time BGP update monitors; they may also be routers in physical or virtual networks. A monitoring system could connect to a BGP-Mux to obtain real-time BGP update feeds from all of the upstream ASes that provide feeds to that BGP-Mux. This service would essentially be a **real-time BGP route update server**. The route servers at Oregon RouteViews [10] provide snapshots of BGP updates; a similarly situated BGP-Mux could provide downstream clients with *real-time* feeds of BGP update streams. As another example, suppose that a virtual network wants to receive BGP updates as if it were physically connected to some set of upstream networks. For example, nodes in the VINI testbed [1] have external BGP connectivity to upstream ISPs; individual virtual networks (*i.e.*, experiments) on this testbed may wish to obtain external connectivity without having to independently negotiate and establish BGP sessions with upstream ISPs. Thus, BGP-Mux can act on behalf of multiple client networks as a **single point of negotiation for upstream BGP connectivity**.

BGP-Mux has several salient features. First, because client sessions receive upstream feeds through BGP-Mux, the clients can be transient and unreliable. Second, because it readvertises the same complete set of BGP update streams to each downstream client, BGP-Mux allows each new client to receive feeds from many upstream ASes by establishing a BGP session with a single entity (in contrast to the status quo, where each client must establish a separate BGP session with each upstream ISP). ISPs are typically reluctant to provide access to their feeds because of the provisioning overhead; BGP-Mux allows many clients to see

these routes without requiring per-client provisioning from upstream ASes.

This paper presents three main contributions. First, we present a new model for BGP interconnection that facilitates new applications that require real-time BGP update streams, such as real-time monitoring, centralized route control and virtual networks. Second, we describe the design and implementation of BGP-Mux, a system that enables this new interconnection model; we have deployed BGP-Mux on VINI and connected it to receive routes from two upstream ISPs. Third, we evaluate the scalability and performance of BGP-Mux as we increase number of clients. We have deployed BGP-Mux on the VINI testbed to provide external connectivity to virtual networks; our evaluation of BGP-Mux in a controlled setting on Emulab shows that BGP-Mux can propagate 99% BGP announcements and withdrawals to as many as 20 clients in less than 1.3 seconds with less than 5% CPU load and using 200 MB of RAM.

The rest of the paper is organized as follows. Section 2 describes three possible scenarios for BGP-Mux usage. Section 3 describes the design goals for BGP-Mux, and 4 describes the and implementation of BGP-Mux in the Quagga software router. Section 5 evaluates the performance and scalability of BGP-Mux, Section 6 discusses related work, and Section 7 concludes.

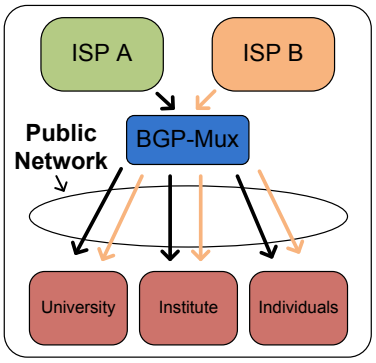


Figure 2: Remote, real-time monitoring of BGP updates.

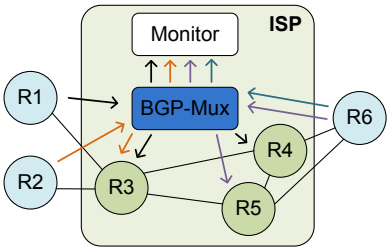


Figure 3: BGP update monitoring inside an ISP network.

2. Background and Motivation

We present a brief overview of the Border Gateway Protocol (BGP) [12] and explain its shortcomings for certain applications. We then present three motivating applications for multiplexing BGP sessions: (1) remote BGP monitoring; (2) local BGP route monitoring for intelligent route selection; and (3) access for virtual networks.

2.1 BGP: Background and Limitations

BGP is the Internet's interdomain routing protocol: it exchanges reachability information (*i.e.*, routes) about global destinations between ISPs. BGP is an *incremental* protocol: when a route to a destination changes at some BGP-speaking router, that router propagates BGP *updates* over all of its BGP sessions to neighboring routers. These updates are exchanged over BGP *sessions*; each router advertises a single "best" route to each destination over each BGP session. Although this model allows BGP to scale well, it also imposes several limitations.

Limited visibility. A BGP router selects and readvertises only a single best route per destination, which enables BGP to compute routing trees but hinders visibility for certain applications. For example, today, it is very difficult for ISPs to collect all external BGP routing updates real-time; essentially, it would require deploying (expensive) packet monitors on all peering sessions [4]. Instead, BGP-Mux could establish sessions to each eBGP session and "reflect" the routes learned on each of those sessions to a monitor.

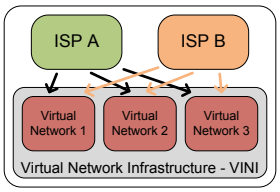
Provisioning overhead and issues with trust. The BGP session provisioning involves exchange of multiple parameters (such as IP addresses and AS numbers) between the participating parties and manual configuration on the bordering routers. Network operators are reluctant to provision new BGP sessions unless these sessions are necessary for normal network operation; further, they must typically trust the neighboring network with which it is exchanging routes; in the case of an experimental network (*e.g.*, VINI) that requires external connectivity, an operator may not trust that the experimenter will not introduce false routing information through misconfiguration (or malice).

Exposure to instability. Unstable sessions utilize CPU on border routers, especially when a full BGP feed must be resent each time the session comes back up. These issues are especially apparent when the neighboring network is a research network or a monitoring station that is not optimized for stability. BGP-Mux could "mask" the instability from virtual networks by maintaining a single, stable upstream BGP session to the upstream ISP.

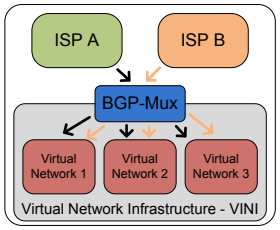
2.2 Applications for BGP Multiplexing

The limitations of BGP outlined in the previous section make several applications difficult to deploy. This section describes three such applications that BGP multiplexing could enable.

Remote, real-time BGP monitoring (Figure 2). Monitoring, security, and reactive routing applications may benefit from having access to real-time BGP update feeds from many ISPs to perform measurements (and perhaps also react or raise alarms in real-time). For example, systems that monitor routing updates for suspicious advertisements or route hijacks (*e.g.*, [8,9,15]) could benefit from having direct real-time feed of all of the ISPs that send routes to RouteViews



(a) Solution without BGP-Mux. Each virtual network must establish independent BGP sessions to neighboring ISPs. Instability in virtual networks is exposed to external ISPs.



(b) Solution with BGP-Mux. Multiple virtual networks can receive BGP routes from upstream providers without separately provisioned sessions.

Figure 4: BGP-Mux allows multiple virtual networks on the same shared physical infrastructure to exchange routes with upstream ISPs, *without independent, direct BGP sessions for each virtual network*.

(as opposed to relying on periodically archived route update files).

Unfortunately, as previously explained, establishing a separate BGP session from each upstream ISP to each new monitoring service incurs high overhead. Instead, BGP-Mux could provide transparent access to each new monitoring service or application that relies on real-time feeds, without requiring a separate negotiation with each upstream ISP every time. Both ISPs and services benefit: ISPs only need to establish only one stable BGP session to BGP-Mux; clients don't need to negotiate update feeds from multiple parties.

BGP Monitoring inside ISP networks (Figure 3). Service providers want real-time visibility into the routes that are advertised over *every external BGP session*. Monitoring every external BGP session has several applications, including monitoring these sessions to enforce peering agreements [4], modeling BGP route selection for performing what-if scenarios [5], and running network routing systems that exercise greater control over route selection (*e.g.*, RCP [2]).

Unfortunately, achieving this type of "global visibility" is difficult today: operators can log into the routers to "dump" routing tables, but this approach is complicated and only yields a snapshot; it does not give real-time stream of BGP updates. Alternatively, operators can deploy a monitor that speaks internal BGP to each router, but that monitor will learn only the best route to each destination (as opposed to all BGP routes). Instead, BGP-Mux can relay BGP update streams from every external BGP session to a central monitoring point for analysis, and reflect those BGP routes to the routers themselves.

External connectivity for virtual networks (Figure 4). Network virtualization facilities such as VINI [1] support multiple concurrent networks running on the same infrastructure. To exchange real user traffic with external, global destinations, these virtual networks must exchange routes with routers in neighboring networks, which implies that routers in the virtual network need to establish external BGP feeds with routers in neighboring networks. Ideally, each virtual network would see BGP routes from each upstream ISP as if it were that upstream's router, but the limitations discussed in Section 2.1 make this difficult (*i.e.*, these facilities are used for research and thus may be inherently unstable). BGP-Mux reduces provisioning overhead by providing on-demand access to the BGP routes from each upstream that it connects to. BGP-Mux also masks upstream ISPs from the route instability that would otherwise result from fluctuating virtual networks.

3. Design Goals

The applications described in Section 2 give rise to the following four design requirements:

Session transparency. Both the monitoring applications and the virtual networks need to receive BGP routing updates exactly as they would have received them over a direct BGP session with the upstream AS. Specifically, session initiation messages must carry the appropriate AS numbers, and update messages from upstream ASes must be passed to clients with no changes to the route attributes; for example, no additional AS hops must be added to the AS path, the next-hop IP address must appear as though the route was received from the router in the upstream AS, etc.

Scalability and fast propagation. In practice, BGP-Mux may have a large number of downstream clients. In the case of support for virtual networks, a single shared physical infrastructure may need to support tens to hundreds of co-existing virtual networks, each of which needs its own set of BGP sessions to BGP-Mux. BGP-Mux would also need to support hundreds to thousands parties (*i.e.*, researchers and operators) who would want a real-time BGP update feed (say, from RouteViews). In these cases, not only is supporting a large number of client sessions important, but all of these sessions must also receive the BGP routing updates from BGP-Mux in a reasonably short time after the route was sent to BGP-Mux.

Isolation. Because client networks may be experimental and transient, BGP-Mux must "protect" upstream ISPs against misbehaving or misconfigured experiments that mistakenly advertise erroneous BGP routes. Accordingly, BGP-Mux must have a default policy in place that filters most updates originating from client networks. BGP-Mux must also filter updates from one client network to another unless such advertisements are explicitly allowed by policy.

Upstream session stability. Provisioning BGP sessions with neighboring ASes requires configuring a BGP-speaking router on each end of the session; additionally, when sessions are destroyed or are otherwise decommissioned, BGP-speaking routers in neighboring ASes see the effects of such instability. In the case of virtual network testbed such as VINI, virtual networks may be provisioned and decommissioned frequently, as experimenters create and destroy experiments. The routers in upstream ISPs should be isolated from the instability of an experiment's associated BGP sessions (*i.e.*, as experiments come and go, the BGP session between the testbed and neighboring ASes should stay up). To reduce provisioning overhead and present a stable BGP session to upstream ASes, BGP-Mux must thus provide a stable

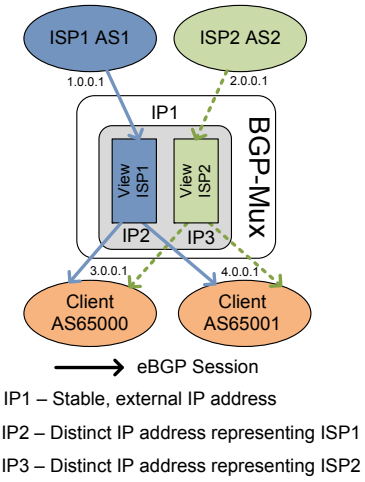


Figure 5: BGP-Mux implementation for a single BGP-Mux instance on a physical node.

```
bgp multiple-instance
!
router bgp 64512 view ISP1
 ! upstream session to ISP1
 neighbor 1.0.0.1 remote-as 1
 neighbor 1.0.0.1 route-map BLOCK out
 ...
 ! sessions to clients in the view
 neighbor 3.0.0.1 remote-as 65000
 neighbor 3.0.0.1 local-as 1
 neighbor 3.0.0.1 ebgp-multihop 255
 neighbor 3.0.0.1 attribute-unchanged
 neighbor 3.0.0.1 advertisement-interval 1
 neighbor 3.0.0.1 route-map BLOCK in
!
router bgp 64512 view ISP2
 ...
!
route-map BLOCK deny 10
```

Figure 6: Example Quagga configuration for BGP-Mux.

IP address and a session to each upstream ISP.

The next section describes our implementation of BGP-Mux and how we achieve these design goals.

4. Implementation

This section describes the implementation of BGP-Mux. We describe how we used advanced features in the Quagga software router [11] to create a BGP-Mux. We then present a single-node BGP-Mux. Finally, we describe how BGP-Mux can be distributed across multiple physical machines to improve scalability.

4.1 Implementation in Quagga

We implemented BGP-Mux with the Quagga BGP routing daemon [11]. Quagga is an open-source routing protocol suite consisting of RIP, OSPF, BGP and ISIS protocol implementations. Freely available source code makes this implementation convenient for modifications to BGP. BGP-Mux requires several advanced features that are not present in many BGP implementations: (1) multiple BGP views, (2) per-session configurable AS numbers, (3) transparent propagation of updates, and (4) a configurable minimum route advertisement interval (MRAI) timer. We explain each of these features in more detail below.

BGP views separate BGP updates from different upstreams. Updates in separate views are not subject to BGP's best path selection algorithm, so updates from each upstream ISP (*i.e.*, one per view) are forwarded downstream to clients. Another crucial BGP-Mux feature is ability to *configure its AS number separately for each session*, which allows BGP-Mux to convey the AS number of the appropriate upstream ISP and thus allows clients to see BGP routes exactly as they would have been received from an upstream ISP.

BGP-Mux must also support *update transparency*: Every update BGP-Mux forwards must be forwarded unchanged to the clients. None of the update attributes should be changed, especially AS-path, origin and next-hop attributes. Updates must be transparent not only in their form, but also in their arrival speed and rates. Such rates usually are affected by Minimum Router Advertisement Interval (MRAI). BGP-Mux thus requires *per-session modification of MRAI timer value* to decrease this timer's effect on update propagation delay.

4.2 Single-Instance BGP-Mux

Combining the above-mentioned features yields a standalone, or single-instance, BGP-Mux, as shown in Figure 5. The ISP sessions are terminated into separate BGP views. Clients that connect to those views perceive the connection as a direct connection to the ISP because each session on BGP-Mux is configured with the AS number of the appropriate ISP. When updates travel from ISPs to clients, they are propagated without any changes as quickly as possible. In addition, standard BGP features, such as AS-path access lists, are used to filter out updates coming from the clients.

Figure 6 shows the Cisco-style BGP configuration for the corresponding BGP-Mux setup shown in Figure 5. BGP-Mux peers with two upstreams, AS1 and AS2, with the addresses 1.0.0.1 and 2.0.0.1, respectively; and with two clients, with private AS numbers 65000 and 65001 (the second client configuration is not shown). Sessions to ISPs are terminated on different views. Sessions to clients are configured with *local-as* property (ensure that client sessions propagate routes using using provider's AS number) and with decreased *advertisement-interval* (to increase update propagation speed). The *ebgp-multihop* option enables BGP sessions with clients that are more than one hop away, which is needed when clients are remote monitors. Although it is not shown in the configuration, the session to the same client from view ISP2 should be configured to use a different interface. If both views used the same interface, a standard BGP client could not distinguish between the two TCP sessions.

4.3 Multi-Instance BGP-Mux

To scale large numbers of clients, BGP-Mux can be distributed across multiple machines using standard BGP configuration tools. Figure 7 shows a distributed, or multiple instance, BGP-Mux configuration. The main BGP-Mux

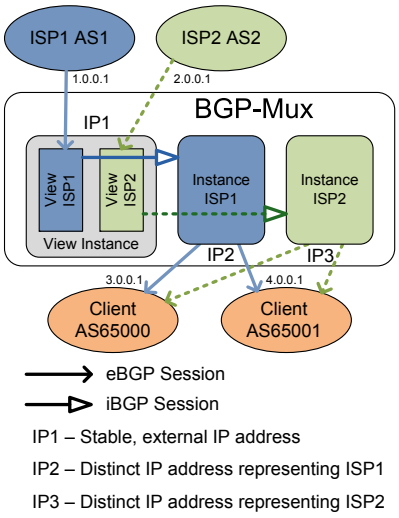


Figure 7: Scalable, Multi-instance BGP-Mux. To support a large number of clients, BGP-Mux can be distributed across multiple machines.

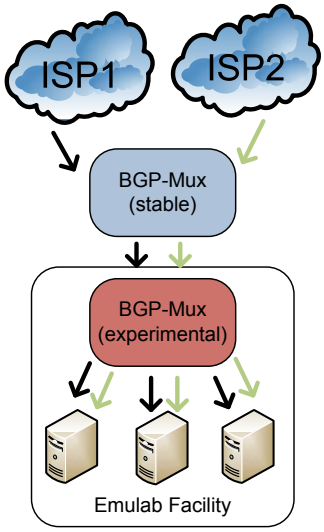


Figure 8: BGP-Mux evaluation setup

terminates sessions from ISP1 and ISP2 to views and distributes routes to separate BGP instances through iBGP sessions. These BGP instances, in turn, serve as a client session pools. As we describe in the next section, the number of BGP clients is limited by memory constraints. Memory is mainly consumed for the outgoing adjacency Routing Information Base (RIB).

5. Evaluation

Although BGP-Mux aims to maximize session transparency, it imposes processing and transmission delays on update propagation, and it might also have scalability limitations. In this section, we study the effects of BGP-Mux on update propagation time, as well as its ability to support many downstream clients using real BGP sessions terminated on a number of clients. We show that BGP-Mux impact to route propagation dynamics is very limited.

	Announcements		Withdrawals	
Clients	90% (sec)	99% (sec)	90% (sec)	99% (sec)
1	0.9528	0.9956	0.0583	0.1186
2	0.9858	1.0272	0.0595	0.8200
5	1.0332	1.0804	0.1955	0.3799
10	0.9810	1.3422	0.0599	0.3263
20	1.0299	1.0955	0.0594	0.0601

Table 1: BGP-Mux update forwarding delay. The two columns correspond to the time it took to deliver 90% and 99% of updates during the test runs. Each row corresponds to a number of clients configured to use BGP-Mux.

5.1 Experiment Setup

Figure 8 shows the evaluation setup. A BGP-Mux on VINI receives routes from two upstream ISPs: Verio and AT&T. The BGP sessions are terminated on a BGP-Mux installed on a production server with a stable IP address on the VINI testbed [1]. The BGP-Mux "under test" is at Emulab [3] and connects to a stable BGP-Mux to receive route feeds. The BGP-Mux under test also used non-default BGP ports to avoid firewalls deployed at Emulab. The BGP-Mux on Emulab belongs to a small LAN with a number of machines that are tested as BGP-Mux clients. Due to limited resources, we limit the number of clients in our tests to 20. Such a setup would be similar to the BGP-Mux deployment scenario described in Section 2.2.

The BGP-Mux under test and its BGP clients in Emulab run Linux Redhat 9 with 2.6.22 kernel and modified Quagga 0.98.6 route daemon. The Emulab nodes were Dell Poweredge 2850 servers with 3.0 GHz 64-bit Intel Xeon processor with 1MB L2 cache, 800 MHz FSB, 2GB 400MHz DDR2 RAM and Gigabit ethernet interfaces. Nodes are interconnected with a Gigabit ethernet switched network.

5.2 Propagation Delay

To streamline update propagation, we minimize Minimum Route Advertisement Interval (MRAI). MRAI is used in the internal Quagga session scheduler; and setting it to very low values can affect daemon performance. The Quagga BGP daemon is stable when MRAI is set to 1 second; smaller values cause the process to consume a lot of CPU.

To measure propagation delay, our system records the timestamp of the update as it arrives at BGP-Mux and the timestamp as the same update is sent to each BGP client. We perform this experiment for 1, 2, 5, 10 and 20 clients. The updates were measured using real feeds over 10-minute intervals, so each test scenario received slightly different update loads. Because updates and withdrawals are processed differently, we studied their propagation times separately.

Table 1 shows the update propagation times: *99% of BGP announcements are delivered within 1.3 seconds*. The propagation delay is affected by MRAI timer. When an update arrives, the BGP-Mux does not readvertise it to clients until the MRAI timer expires. Withdrawal propagation is not subject to MRAI timer; furthermore, withdrawals are subject to less checks and processing (for example, they are not subject to inbound and outbound filtering). Therefore, with-

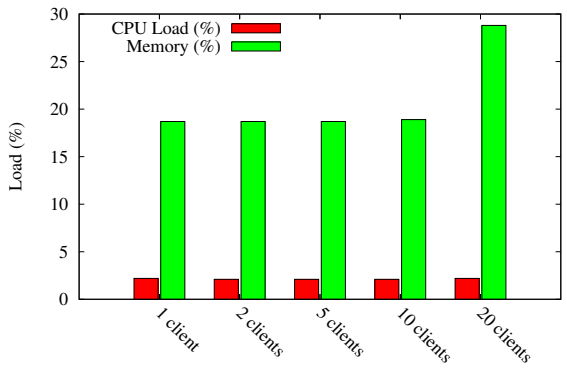


Figure 9: CPU and memory load on BGP-Mux as number of client sessions are increasing. There was 1Gb of RAM on the system. The spike in memory usage is due to Quagga's aggressive buffer pre-allocation polices.

drawals are delivered even faster than announcements. *99% of BGP withdrawals were delivered within 0.82 seconds*. We observed no noticeable increase in update propagation time as the number of clients increases.

5.3 Memory and CPU Usage

Figure 9 shows how CPU load and memory vary with the number of BGP-Mux clients. The CPU usage was primarily affected by the churn caused by upstreams but was not significantly affected by the number of clients. On the other hand, the BGP daemon consumes more memory as more clients are added because it must maintain more outbound RIB adjacency structures. Because memory in Quagga is pre-allocated to buffers in large blocks, memory usage increases abruptly between 10 and 20 clients.

6. Related Work

Oregon RouteViews [10] and the RIPE routing information service (RIS) [13] currently maintain the most widely used archives of BGP routing data. These services were initially deployed to help operators debug routing by providing routing table views from many different vantage points. Over time, however, these facilities have evolved to support a wide variety of analysis and monitoring projects, some of which could benefit from access to the real-time updates as seen by the route server itself [7–9, 15]. Unfortunately, BGP updates are logged to files only once every 15 minutes, which is too slow for many monitoring applications for both hijacking and reactive routing. The above mentioned improvements offered by BGP-Mux could enable new applications that could benefit from real-time monitoring of global BGP update data.

Many tools and software routers monitor BGP routing updates by directly connecting to a router and logging the received update stream, but these systems typically require direct BGP connections to the routers they are monitoring [6, 11]; BGP-Mux facilitates the same type of monitoring, but allows allowing *many* clients indirect access to BGP update feeds from all upstream ISPs through an indirect connection. In an expired Internet draft, Scudder describes a BGP Monitoring Protocol (BMP), which would have enabled online monitoring of incoming BGP route advertisements, albeit via a separate protocol [14].

7. Conclusion

We have presented BGP-Mux, a system that enables real-time, on-demand BGP session multiplexing for multiple downstream clients. BGP-Mux provides route monitoring services and virtual networks with the appearance of a direct connection to upstream ISPs without requiring direct connections to each upstream ISP for each downstream service or virtual network (which incur provisioning overhead and induce instability in upstream ISPs). We have deployed BGP-Mux on the VINI testbed and tested its scalability and performance by adding up to 20 downstream client networks. We have deployed BGP-Mux on VINI testbed to provide external connectivity to virtual networks; our evaluation of BGP-Mux in a controlled setting on Emulab shows that BGP-Mux propagates 99% BGP announcements and withdrawals to as many as 20 clients in less than 1.3 seconds while incurring less than 5% CPU load and using less than 200 MB of RAM. We envision that BGP-Mux will be useful not only for building virtual networks but also for projects and systems that rely on real-time BGP route monitoring.

REFERENCES

[1] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. In *Proc. ACM SIGCOMM*, Pisa, Italy, Aug. 2006.

[2] M. Caesar, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *Proc. 2nd USENIX NSDI*, Boston, MA, May 2005.

[3] Emulab. `http://www.emulab.net/`, 2006.

[4] N. Feamster, Z. M. Mao, and J. Rexford. BorderGuard: Detecting cold potatoes from peers. In *Proc. Internet Measurement Conference*, Taormina, Italy, Oct. 2004.

[5] N. Feamster, J. Winick, and J. Rexford. A model of BGP routing for network engineering. *In submission*, Nov. 2003.

[6] M. Handley, O. Hudson, and E. Kohler. XORP: An open platform for network research. In *Proc. 1st ACM Workshop on Hot Topics in Networks (Hotnets-I)*, Princeton, NJ, Oct. 2002.

[7] J. M. Hellerstein, T. Condie, M. Garofalakis, B. T. Loo, P. Maniatis, T. Roscoe, and N. Taft. Public Health for the Internet (PHI): Towards a New Grand Challenge for Information Management. Jan. 2007.

[8] J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Protecting BGP by cautiously selecting routes. Technical report, University of New Mexico, Oct. 2005. TR-CS-2005-37.

[9] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A prefix hijack alert system. In *Proc. 15th USENIX Security Symposium*, Vancouver, BC, Canada, Aug. 2006.

[10] U. of Oregon. RouteViews. `http://www.routeviews.org/`.

[11] Quagga software routing suite. `http://www.quagga.net/`.

[12] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, Jan. 2006. RFC 4271.

[13] Réseaux IP Européens Next Section Routing Information Service (RIS). `http://www.ripe.net/ris/`.

[14] J. Scudder. *BGP Monitoring Protocol*. Internet Engineering Task Force, Aug. 2005. `http://tools.ietf.org/html/draft-scudder-bmp-00` Work in Progress, expired Feb 2006.

[15] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis. A Light-Weight Distributed Scheme for Detecting IP Prefix Hijacks in Realtime. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.