

Understanding Slow BGP Routing Table Transfers

Zied Ben Houidi
France Telecom R&D Orange
Labs and
UPMC Paris Universit as
zied.benhoudi@orange-
ftgroup.com

Mickael Meulle
France Telecom R&D Orange
Labs
michael.meulle@orange-
ftgroup.com

Renata Teixeira
UPMC Paris Universit as and
CNRS
renata.teixeira@lip6.fr

ABSTRACT

Researchers and network operators often say that BGP table transfers are slow. Despite this common knowledge, the reasons for slow BGP transfers are not well understood. This paper explains BGP table transfer delays by combining BGP messages collected at a large VPN provider backbone and controlled experiments with routers of three different vendors as well as a software BGP speaker. Our results show that table transfers both in the provider network and in the controlled experiments contain *gaps*, i.e., periods in which both the sending and receiving routers are idle, but no BGP routes are exchanged. Gaps can represent more than 90% of the table transfer time. Our analysis of a software router and discussions with router vendors indicate that gaps happen because of the timer-driven implementation of sending of BGP updates. Hence, gaps represent an undocumented design choice that gives preference to more controlled router load over faster table transfers.

Categories and Subject Descriptors

C.2.3 [Computer-Communication Networks]: Network operations; C.4 [Performance of Systems]: Measurement techniques

General Terms

Experimentation, Measurement, Performance

Keywords

BGP, route propagation, routing convergence

1. INTRODUCTION

BGP route propagation is one step in BGP routing convergence. This step is particularly significant when an event triggers routing changes for many destination prefixes at the same time. Examples of such events are resets or failures of BGP sessions and intra-domain routing changes that trigger BGP changes [1]. Studies of Internet provider's networks have shown that these events can cause a router to update a significant fraction of its BGP table [1, 2, 3]. After updating its own BGP table, a router propagates this information to its BGP neighbors. We call this step a *BGP table transfer*

when the routing change affects a significant portion of a router's BGP table. Previous work has shown that BGP table transfers can take minutes [1, 3, 4].

This paper is the first to investigate the reasons of slow BGP table transfers. We combine BGP data collected at the backbone of a large VPN provider with experiments with different router models in a controlled environment. We concentrate on BGP routes from a VPN backbone because BGP tables are much larger in this environment [5]; however our findings apply to any BGP network. We summarize these findings as follows.

1. **Table transfers are slow because of gaps.** Sec. 2 examines BGP table transfers between pairs of routers in the VPN provider backbone. Our results show that table transfers can be 20 times slower than the optimal (i.e., the time to transfer the same amount of data given the link capacity). A detailed analysis of BGP message exchanges reveals that the rate to transfer the BGP table is slow because of *gaps*—periods in which no data is exchanged even though both the sender and the receiver are idle.
2. **Gaps arise in all tested routers.** We emulate the VPN provider network in a controlled environment in which we test carrier-class routers from three main vendors. As shown in Sec. 4.1, all BGP table transfers in our experiments contain gaps irrespective of the router model or configuration.
3. **Gaps are caused by a timer-driven implementation that controls the sending rate.** Gaps are not documented; hence Sec. 4.2 investigates the causes of gaps using an open source BGP speaker (SBGP). This analysis shows that SBGP has a timer-driven implementation to send BGP messages. Effectively, this implementation choice acts as a rate-limiting mechanism for BGP table transfers. Our discussions with two router vendors confirm that gaps are caused by a timer-driven implementation.

One router vendor declared that this rate-limiting is a conscious design choice to control router load, whereas the other said that gaps were unintentional. Because of our results, the latter vendor already modified its implementation of the interaction between BGP and TCP to be event-driven.

Intentional or not, this design choice has non-negligible impact on BGP table transfer time and deserves more careful consideration. Gaps represent a trade-off between fast BGP table transfers and more control over router load. Sec. 5 studies mechanisms to reduce BGP table transfer times and their consequences. The effectiveness of these mechanisms depends on the number of routes in a router's BGP table, the number of BGP neighbors of a router and the router's capacities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

Copyright 2009 ACM 978-1-60558-770-7/09/11 ...\$10.00.

Transfer number	Link type	RTD	Transfer time	Baseline
1	500Mbps	5ms	210 sec	8.91 sec
2	POS155	10ms	190 sec	17.82 sec
3	100Mbps	5ms	270 sec	8.91 sec
4	GE	1ms	210 sec	1.78 sec
5	GE	1ms	90 sec	1.78 sec
6	GE	5ms	110 sec	8.91 sec
7	GE	1ms	60 sec	1.78 sec

Table 1: Table transfers in a VPN provider backbone

2. TABLE TRANSFERS ARE SLOW

This section analyzes BGP table transfers between pairs of routers in a large VPN provider backbone. Our results are distinct from previous work, which analyzed BGP messages collected at a BGP monitor. In these datasets, it is hard to observe table transfers on BGP sessions that are more than one hop away from the monitor.

We study a large VPN provider backbone that has hundreds of border routers (provider edge routers or PEs) and thousands of VPN customers. PEs use internal BGP (iBGP) to exchange routing information between each other. Instead of having a full mesh of iBGP sessions between border routers, the provider uses a hierarchy of route reflectors (RRs) [6]. RRs keep all the routes from all the VPNs (around 680K BGP routes).

Network operators trigger table transfers between different pairs of RR-PE routers in the network by forcing a router to send a BGP route refresh message [7]. Operators monitor the routers using router-specific commands in order to determine the transfer time. We also tap the messages exchanged between one pair of RR-PE routers during one of the table transfers.

Table 1 shows statistics on some of these table transfers. Each line in the table contains information about a table transfer between a different pair of RR-PE routers. Given that BGP runs over TCP, we compute the baseline values which correspond to the time a TCP connection should take to transfer the same amount of data (30MB of routes). RWIN, the TCP receive window size advertised by the receiver, is 16KB for all transfers in Table 1. We use a TCP throughput prediction formula [8] with a loss rate of 0.01%. For all table transfers, RWIN and the round-trip delay (RTD) are the only factors that determine TCP throughput (and therefore the baseline values) because of high link capacities. Table 1 shows that the table transfer time can take up to 4 minutes 30 seconds and can be one to two orders of magnitude longer than baseline values.

To understand the reasons behind these slow times, we study the transfer between one RR-PE pair in detail. We use the packet traces and analyze the messages exchanged between the two routers. We find that the sender (RR) regularly stops sending routes to the receiver (PE) even though the receiver acknowledged all the messages it received. This creates *gaps* in the table transfer in which no messages are sent. The gaps, which can last up to two seconds each, account for around 90% of the total table transfer time. Therefore, these gaps, which are caused by the sender, are clearly the reason behind the slow table transfers. Unfortunately, we have no control on the operational routers and hence we could not get CPU measurements to understand whether routers are overloaded or not during the gaps. Therefore, we study these gaps further using a router testbed.

3. TESTBED DESCRIPTION

We perform controlled experiments to understand the gaps observed in Sec. 2. Controlled experiments allow us to monitor routers' behavior under different conditions during table transfers as well as

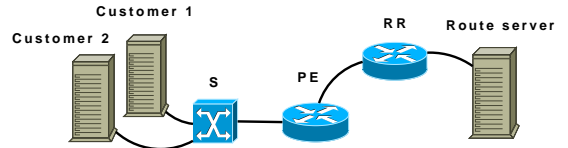


Figure 1: Testbed for controlled experiments

to study the prevalence of gaps among routers from different router vendors.

Fig. 1 depicts the testbed used in this paper. Each experiment tests two routers: a sender, one route reflector (RR) and a receiver, a PE. We test carrier-class routers from three different vendors. In this work, we mainly vary the sender (RR), we test three routers from Vendor 1, two routers from Vendor 2 and one router from Vendor 3. We test two different PEs from Vendor 1. We establish a BGP session between PE and RR and emulate the provider backbone and the customer routers using three Linux machines. The route server sends BGP routes to RR to emulate an entire VPN provider backbone. The VPN backbone has a set of route reflectors that connects hundreds of PE routers. We fix the number of routes that the route server sends to RR depending on the experiment. We use a set of 680K routes collected from one RR in the provider backbone as a basis and sub-sample this set to vary the number of routes when needed. The two other machines, Customer 1 and Customer 2, emulate customer routers and connect to PE through a switch. Customer 1 and Customer 2 can emulate each up to 250 customer routers. Customer routers communicate with PE using regular external BGP. We need to emulate customer routers so that PE is able to install routes.

In all our experiments, we follow classical BGP operational guidelines to obtain the best tuning of BGP convergence [9]. We are interested in observing the routing table transfer between RR and PE. We provoke it by resetting the BGP session between the two routers. Each section in this paper gives more details about its experiments.

4. GAPS IN TABLE TRANSFERS

This section studies the prevalence and causes of gaps observed in Sec. 2

4.1 Prevalence

We first study the prevalence of gaps. This section focuses on the sender since it is responsible for the gaps observed in Sec. 2. We perform experiments emulating table transfers in which we test each time a different sender (RR). In all experiments, RR sends a full table of 680K routes to PE. To study the prevalence of gaps, we test seven different RRs from three different vendors as well as SBGP, a simple open source BGP speaker and listener from Merit's Multi-Threading Routing Toolkit [10]. In the experiments with SBGP, we run SBGP on the route server and connect the route server directly to PE. In all experiments, we focus on the sender, so we make sure the receiver does not install the routes it receives in its forwarding tables thereby avoiding receiver overload. We tap the messages exchanged between RR and PE and study the evolution of BGP messages sent by RR as a function of time during a table transfer.

We find that the gaps are prevalent on table transfers with all the routers that we test. Fig. 2 plots the evolution of the total number of bytes sent by different RRs as a function of time. For clarity, we present a zoom on 10 seconds and only show results for one

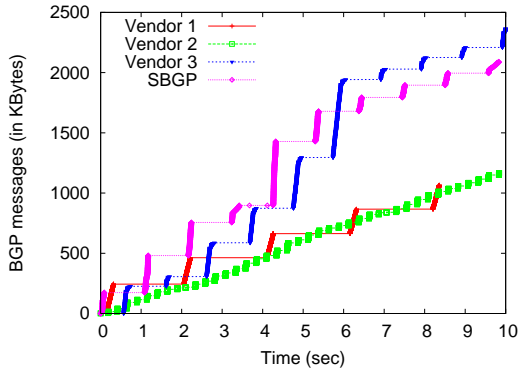


Figure 2: Gaps in table transfers

RR router from each vendor. The other routers presented similar behavior. This figure shows that table transfers contain gaps that vary from 200 ms to 2 seconds each. The gaps’ duration depends on the router and is nearly constant for each router. Vendor 1 for instance has gaps that last 2 seconds while Vendor 2’s gaps last around 200 ms. Although Vendor 1’s router has longer gaps than Vendor 2’s router, it sends more updates between two consecutive gaps, and consequently achieve the same transfer time as Vendor 2’s router. In fact, two factors contribute to the table transfer time: the duration of gaps and the number of routes sent between two consecutive gaps.

4.2 Causes

We now investigate the causes of gaps. First, we monitor the CPU load on both RR and PE using command-line interface probes. Both routers have low CPU (less than 10%) during the transfer. Our analysis of the packet traces shows that PE quickly acknowledges all packets received and is ready to receive new packets. In all experiments, when a gap starts, PE sends TCP window update messages implicitly telling RR that it can send more data but RR does not answer. The gaps are therefore caused by the sender which regularly stops sending routes. Besides, the sender’s low CPU indicate that it is just blocking during the gaps and is not performing another task.

Second, we analyze the source code of SBGP. During a table transfer, SBGP reads routes from an input file with BGP messages. We find that BGP I/O operations are timer-driven: the sending of routes is triggered by a timer that expires every second. BGP messages that have the same timestamp are sent together in the same I/O operation, while next messages wait for one second till the next I/O operation. The gaps are therefore the consequence of a timer-driven implementation of BGP I/O operations. We compile an event-driven version of SBGP in which we avoid the use of the timer. This modified version can transfer BGP tables almost as fast as baseline values (as defined in Sec. 2). We call this version *fast SBGP*.

The feature that causes gaps for router vendors is not documented. We therefore report the issue to two vendors. Our discussions with both of them confirms our analysis of SBGP source code. Vendor 2 reports that it intentionally limits the number of messages sent per interrupt. This limit prevents routers from DoS attacks and router overload and explains the gaps observed in Fig. 2. Each time the BGP process is scheduled to send routes, it only sends a certain number of routes and then waits for the next interrupt. Vendor 1 says, however, that gaps are unintentional, they are the consequence of the timer-driven implementation of one BGP pro-

cess. In fact, two BGP processes participate in route sending: the *router process* prepares routes and puts them in an update queue, the *I/O process* reads them from the queue and sends them to TCP. Once the TCP ack is received, the I/O process empties the queue. If the router process is scheduled to run while the queue is still full (due to a late ack, for instance), then the router process sleeps until a two-second timer expires.

Gaps represent an undocumented design choice that gives more priority to router protection over fast table transfers. Intentional or not, this design choice impacts BGP table transfer time and deserves further consideration.

5. REDUCING TABLE TRANSFER TIME

This section discusses approaches for speeding up table transfers and explores the trade-off between fast table transfers and controlled router load. There are two possible approaches to speed up table transfers: Increase the table sending rate or decrease the table size.

5.1 Increasing the sending rate

Two factors contribute in the table sending rate, the interval between two interrupts and the volume of data sent per interrupt.

The interval between two interrupts can be reduced by using an event-driven implementation. Because of our results, Vendor 1 already modified its implementation of the interaction between BGP and TCP to be event-driven. This change ensures that the BGP router process (see Sec. 4.2) is scheduled as soon as the I/O process gets the TCP ack. The vendor reports that this modification reduced by a factor of 15 BGP table transfer time in its controlled settings.

With a timer-driven implementation, the only way to increase the sending rate is to send more routes between two interrupts. Vendor 2 reports that it has a special configuration mode that allows tuning the maximum number of messages sent at each interrupt. Unfortunately, this configuration change is reset after a router reboot and this software add-on is not available on routers from our labs so we could not test it. For Vendor 1, we find that a possible way to increase the sending between interrupts is a careful use of TCP window scaling. Window scaling is a TCP option that allows increasing the TCP window above its maximum. This option does not remove the gaps but reduces their number, it allows a router to send more routes between two consecutive gaps.

Our experiments focused so far on the table sending time in the case of one single BGP session to a single receiver that install no routes. If we speed up the sending rate, we may hit limits on the receiver side because it has to install routes or at the sender side when the route reflector has multiple clients.

Limit 1: Installing routes

We study the possible limits induced by the receiver when it has to install routes. We perform several experiments in which we always keep the size of the table sent constant and vary each time the number of installed routes. These experiments allow us to study the impact of installing routes, given that the sending is constant.

We present results for a PE from Vendor 1 that is widely used in the provider backbone. We fix the table size in all experiments to 680K routes which corresponds to a ‘full table’ in the provider backbone. Each experiment starts by loading a new configuration in PE. PE’s configuration controls the number of routes that PE ultimately installs. Then, the route server sends the set of 680K routes to RR. After RR and PE have finished installing all routes, we reset the BGP session between PE and RR. In each experiment,

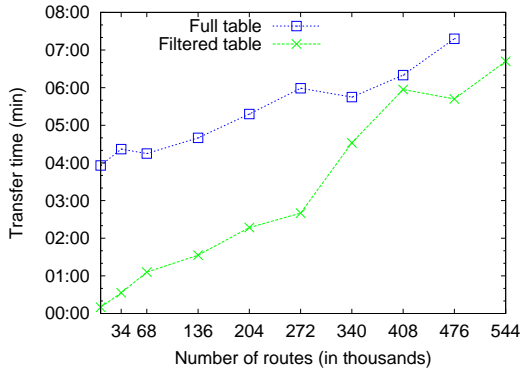


Figure 3: Varying the number of routes

we measure the table transfer time. We also run a script that uses command-line interface to probe PE every 5 seconds. The script gets the number of received and installed routes as well as PE’s BGP CPU load.

The ‘full table’ curve in Fig. 3 presents the results of these experiments. Fig. 3 presents the transfer time when varying the number of routes that PE installs. We discuss the ‘filtered table’ curve in Sec. 5.2. Each point in the figure corresponds to one experiment. We run each experiment three times and present the mean of the three runs. We verify that the results are consistent across all runs.

The ‘full table’ curve shows that even though the table size is constant across all experiments, the table transfer time increases linearly with the number of installed routes. This increase implies that the process of installing routes delays the table transfer. In fact, the analysis of messages exchanged between RR and PE shows that when PE installs routes, it delays the route reception by putting some time to acknowledge the routes it receives from RR.

Limit 2: Handling multiple BGP sessions

Our experiments only study one BGP session, but BGP routers usually connect to more than one BGP peer. Route reflectors in the provider backbone, for instance, have hundreds of clients. Depending on the network configuration and on the event that causes the table transfer, a sender might need to send its table to more than one single receiver which might overload the sender.

We observe table transfers in which RR transfers its table to two neighbors and find that the time to send a table to two BGP peers is the same to send it to one. This observation indicates that the interval between two interrupts is fixed on a per-session basis. During one gap, the BGP process stops updating one session but is free to update the other session. We study this phenomenon in detail using SBGP because we have it with and without gaps (fast SBGP).

We study the effect of increasing the number of sessions on both normal and fast SBGP. We run SBGP and fast SBGP to emulate each time an RR that sends the full table of 680K routes to its peers. We carry several experiments in which we vary each time the number of sessions from 1 to 20. We stop at 20 because the Linux machines that run SBGP and fast SBGP cannot handle more sessions.

Fig. 4 plots the table transfer time as a function of the number of sessions. This figure shows that, until 10 sessions, the transfer time using normal SBGP remains constant because, during the gaps, normal SBGP has more sessions to handle and therefore it is not stalled. Fast SBGP, however, increases with the number of

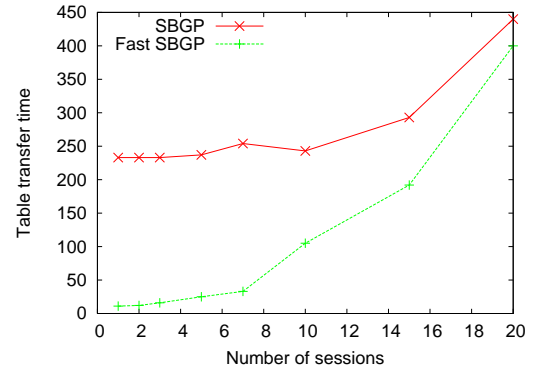


Figure 4: Varying the number of sessions

sessions. This figure shows that, the larger the number of sessions, the closer the transfer time of SBGP and fast SBGP. We expect that after a certain number of sessions, a router will take the same time to send its table irrespective of gaps. We conjecture that the benefits from removing gaps are reduced when there are many BGP sessions.

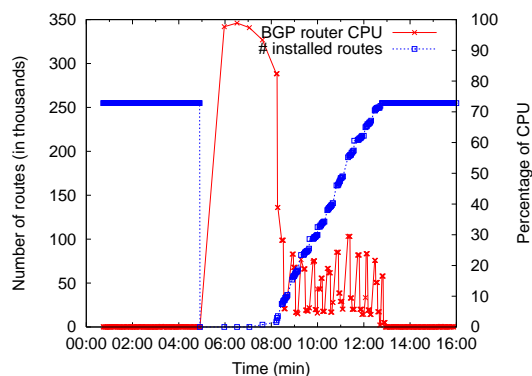
5.2 Reducing the table size

Another approach to decrease table transfer time is to reduce the table size. We study the effects of reducing the table size on table transfer times.

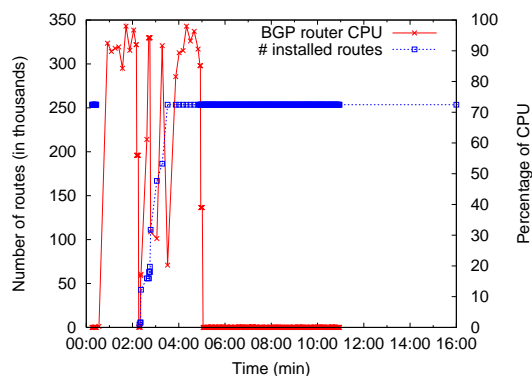
In current BGP implementation, the sender transfers a full routing table to the receiver. Sometimes, however, the receiver only installs a small fraction of these routes. In this case, sending a smaller table (fewer routes) is a possible solution to make table transfers faster. The distinction between the number of routes sent and the number of routes the receiver needs (installs) is particularly striking in VPN backbones, because a PE only needs to install routes of VPNs connected to it. For example, in the VPN provider backbone, 50% of PE routers need only between 3% and 6% of the total number of routes in the network. Sending fewer routes can be achieved by using RT-constraints [11], which is an IETF proposal to reduce the number of routes that a PE router receives during a table transfer, thereby allowing to reduce table transfer times in VPNs.

We first compare table transfer times when we send the full table versus a filtered table that contains the set of routes PE needs to install. We repeat the ‘full table’ experiments presented in Fig. 3 with the filtered table. We use the same PE router and setup as in ‘full table experiments’. Similarly, we run each experiment three times and present the mean. We also monitor PE using command-line interface probes in order to get the CPU and the number of routes PE installs. The ‘filtered table’ curve in Fig. 3 presents transfer times in these experiments. The variation among runs becomes higher when PE is overloaded (more than 340,000 routes). This high variation explains the fluctuations in the curve.

The comparison of the two curves in Fig. 3 shows that table transfers are faster when we send the ‘filtered table’. This is expected because PE receives a smaller table. The figure shows that pre-filtering routes before sending is particularly effective if PE installs only few routes. For instance, sending a filtered table reduces the table transfer from around 4 minutes to 35 seconds when PE installs only 34,000 routes. Although pre-filtering represents a clear improvement, the removal of gaps (with an event-driven implementation) achieves faster table transfers without the overhead of RT-constraints-like mechanisms.



(a) Sending the full table



(b) Reducing the table size: sending a filtered table

Figure 5: Reducing the table size

Limit 3: The receiver’s capacities

We now study possible limits at the receiver when we reduce the table size. Sending a filtered table increases the rate at which PE installs routes, which might overload it. We study PE’s behavior in both the ‘full table’ and the ‘filtered table’ experiments.

Fig. 5(a) and Fig. 5(b) show PE’s behavior when it installs around 136,000 routes in the full and filtered table experiments, respectively. The x-axis presents the experiment time, right y-axis the BGP CPU utilization, and left y-axis the number of routes. In the ‘full table’ experiments, we reset the session between PE and RR at time 04:54. The number of installed routes goes down to zero and PE starts a period of high CPU activity in which it mostly removes previously installed routes and prepares BGP withdrawal messages to customer routers. At time 07:05, the session with RR is re-established and PE starts to receive the routes. The transfer time is the time between the re-establishment of the session and when PE installs all routes. During this time, PE selects the routes needed for its customers. Then, at regular time intervals, PE installs selected routes in the corresponding VPN forwarding tables. The execution of the install process generates small spikes of BGP CPU (20%) at regular intervals.

In the ‘filtered table’ experiment presented in Fig. 5(b), the total table transfer time is around one minute and 30 seconds instead of four minutes and 40 seconds in the ‘full table’ experiment. However, the BGP CPU activity is much higher (up to 90%) because each time the router has to install routes, more new routes are available. The CPU activity remains high after PE installs all the routes.

These CPU spikes correspond to PE sending updates to the customer routers.

The comparison of the two experiments shows that even though PE installs the same number of routes, it experiences a much higher load when it receives the filtered table. Therefore, reducing the table size increases the route installing rate, which can overload the receiver.

By analogy, increasing the route sending rate can have the same effect on the receiver. There is clearly a trade-off between fast table transfers and controlled router load. Since routers have different capacities, we argue that table sending rates should not be hard coded. Operators should be able to tune the speed of table transfers according to their needs. Table transfers between powerful routers may not need rate limiting for instance.

6. RELATED WORK

Studies of events that trigger BGP table transfers. The events responsible for the largest disruptions in the Internet are of two kinds: BGP session resets or failures and routing changes caused by internal events [4]. Such events are often followed by a full or a partial BGP table transfer [2, 3]. Wang et al. [12] studied the causes and impact of BGP session failures using routing messages and router logs from an Internet backbone; whereas Teixeira et al. [1] studied BGP changes triggered by internal routing changes and reported that just the transfer of 80,000 prefixes took 80 seconds. These studies recognized that BGP table transfers are slow with no explanation.

Techniques to detect BGP table transfers. Zhang et al. [3] proposed an algorithm to detect BGP table transfers triggered by session resets between BGP monitors and their direct peers. They also observed that BGP can take several minutes to transfer a full routing table, but do not explain the reasons behind this delay. Their goal was to identify BGP table transfers on data collected at BGP monitors to remove them from later analysis.

Techniques to improve BGP table transfers. Xiao et al. [13] studied the impact of BGP timers and TCP retransmissions on BGP session failures with the goal of improving the robustness of iBGP sessions. Their analytical study cannot observe the implementation issues that we found here. Wang et al. [14] proposed a bloom-filter based mechanism that accelerates recovery after a BGP session reset. Their solution avoids full BGP table transfers by exchanging only small digests of BGP tables. Despite its promises, this solution is not yet deployed, because it requires fundamental changes to BGP.

7. CONCLUSION

This paper is the first to study the causes of slow BGP table transfers using testbed experiments and data collected from a large VPN provider. We found that BGP table transfers are slow because of the timer-driven implementation of the sending of BGP messages. This implementation decision is prevalent on all routers we tested, but it is not documented. Unfortunately, this decision considerably slows down BGP table transfers. We studied different solutions for reducing table transfer times, but these solutions come at the price of higher router load.

Our modified version of SBGP and Vendor 1’s updated software show that event-driven implementations significantly speed up table transfers. However, there is a trade-off between fast BGP table transfers and controlled router load. The need for faster table transfers or for more controlled router load depends on factors like the nature of the network, the number of BGP neighbors, the number of routes, or the routers capacities. For instance, routers at VPN

providers are often in a private address space and are consequently less vulnerable to attacks; at the same time VPN customers have stricter availability requirements. Hence, operators of VPN backbones should tune their routers to speed up BGP table transfers. On the other hand, large Internet providers may want to rate-limit BGP table transfers to ensure controlled load on their routers. When a router has many BGP sessions, the introduction of gaps at each session is an easy way to multiplex the router among different sessions. These two practical scenarios illustrate that picking one point in the solution space between fast table transfer and router control and hard-coding it in BGP's implementation is too limiting. Router vendors should expose this design decision to network operators and let them customize their networks.

8. ACKNOWLEDGMENTS

We would like to thank Stephane Litkowski for his valuable comments on this work. We are grateful to Sarah Nataf for her help in setting up and debugging our testbed. We are also grateful to Guillaume Gaulon, Bruno Decraene, Ítalo Cunha and the anonymous reviewers for their comments and helpful feedback on earlier versions of this work. We also thank the router vendors for the discussions and feedback.

9. REFERENCES

- [1] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of Hot-Potato Routing in IP Networks," in *Proc. ACM SIGMETRICS*, 2004.
- [2] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Observation and analysis of BGP behavior under stress," in *Proc. Internet Measurement Workshop*, 2002.
- [3] B. Zhang, V. Kambhampati, M. Lad, D. Massey, and L. Zhang, "Identifying BGP routing table transfers," in *Proc. ACM SIGCOMM Workshop on mining network data (MineNet)*, 2005.
- [4] J. Wu, Z. M. Mao, J. Rexford, and J. Wang, "Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, May 2005.
- [5] Z. B. Houidi, R. Teixeira, and M. Capelle, "Origin of route explosion in Virtual Private Networks," in *Proc. ACM CoNEXT student workshop*, 2007.
- [6] T. Bates, R. Chandra, and E. Chen, "BGP Route Reflection-An Alternative to Full Mesh IBGP." RFC 2796, Apr 2000.
- [7] E. Chen, "Route Refresh Capability for BGP-4." RFC 2918, Sep 2000.
- [8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation.," in *IEEE/ACM Trans. Networking*, 2005.
- [9] R. Zhang and M. Bartell, *BGP design and implementation*, pp. 62–80. Cisco Press, 2004.
- [10] C. Labovitz and M. Hirabaru, "MRT: Merit's Multi-Threaded Routing Toolkit." <http://mrt.sourceforge.net/>.
- [11] P. Marques *et al.*, "Constrained Route Distribution for BGP/MPLS IP VPNs." RFC 4684, Nov 2006.
- [12] L. Wang, M. Saranu, J. M. Gottlieb, and D. Pei, "Understanding BGP Session Failures in a Large ISP," in *Proc. IEEE INFOCOM*, 2007.
- [13] L. Xiao and K. Nahrstedt, "Reliability models and evaluation of internal BGP networks," in *Proc. IEEE INFOCOM*, 2004.
- [14] L. Wang, D. Massey, K. Patel, and L. Zhang, "FRTR: A scalable mechanism for global routing table consistency," in *Proc. International Conference on Dependable Systems and Networks*, 2004.