An Empirical Study of Router Response to Large BGP Routing Table Load

Di-Fa Chang Ramesh Govindan John Heidemann
USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA

Abstract— **Anecdotal evidence suggests that misconfiguration of backbone routers occasionally leads to an injection of large routing tables into the BGP routing system. In this paper, we investigate the detailed mechanics of router response to large BGP routing tables. We examine three commercial grade routers, and find that their responses vary significantly. Some routers exhibit *table-size oscillations* that have the potential to cause *cascading failure*. Others need operator intervention to recover from large routing tables. We also find that deployed resource control mechanisms, such as prefix limits and route flap damping, are only partially successful in mitigating the impact of large routing tables.**[1]

Keywords— **Inter-domain Routing, BGP, Router Failures, Cascading Failures, Route Flaps.**

I. Introduction

The behavior of modern-day routers under stress has not received much attention in the research literature. A particular aspect of router behavior under stress is the response of routers to *large routing table loads*. The network operator community closely monitors global routing table sizes [1], and most backbone routers are, nowadays, usually configured with several times more memory than that require to support the prevailing global backbone routing table size. However, in the past, *router misconfigurations* at a single router have been known to inject, for brief periods of time, very large routing tables into the routing system [2]. To the research community, little is known about the mechanics of router behavior under these circumstances due to rare occurrance of this event and low availability of commercial routers for testing. Industry benchmarking efforts disclosed in [3] have cataloged one aspect of this problem: How many routes can a given router hold? To study the fault-tolerance capability of current router implementation, we want to ask different questions: How do routers behave when confronted with routing table loads that exceed their capacity? Do they fail? How do they recover? How do different routers from different vendors handle this problem? The answers to these questions provide a hint to the fault-tolerance capability of current Internet. Because previously reported instances of large routing tables have been confined to the inter-domain routing system, we focus on the behavior of the BGP component of these routers. To our knowledge, ours is the first work in research community to systematically study this problem.

In this paper, we catalog the results of two kinds of route loading experiments on *three different commercial routers* (Section III):

- Set up BGP connections with a commercial grade router and microscopically observe the effects of loading the router with a routing table near or over capacity. This experiment provides the understanding of the detailed mechanics on the routers under stress.
- Examine the behavior of simple BGP topologies under large routing table loads to understand how overloading a single router can impact other routers in the topology. Since the purpose of this study is to understand the detailed mechanics of failure propagation, not to estimate the likelihood that the border routers in Internet will fail under stress, simulations with larger and more complex topologies are not necessary.

Here we catalog several interesting results that are not described in the research literature. (Deatils are found in Section IV and Section V):

- Some commercial routers exhibit a “malloc failure” when confronted with a large routing table from one or more peers. Such a failure essentially causes a soft reset of all router state, after which the router proceeds to re-establish the BGP peering sessions, and then fails again. This sequence can, in the absence of manual configuration, repeat itself indefinitely. During this sequence, the routing table size of the router repeatedly *oscillates* between zero and the maximum permitted by the router’s capacity.
- When a chain of these routers is configured such that each router in the chain peers with the neighboring router in the chain, this routing table-size oscillation can propagate down the chain. Thus, routers in the topology are subjected to routing table *flaps* (changes to route entries). In some cases, a “malloc failure” induced in one of the routers can actually *cascade* to the others. This phenomenon is dependent on the relative memory configurations of the routers in the chain and on the speed of route processing at each of these routers. Such a cascade *persists*, and may be an explanation for the meltdown described some previous large network failures [4].
- Other routers disable interfaces, or automatically disable BGP peering with routers from whom they received large routing tables. Operator intervention is required before peering can be established.
- For all the routers we studied we found that large-routing table loads do not cause packet traffic forwarding delays or drops except when the router flushes the entire forwarding table during a reset.

There exist at least two mechanisms in modern day routers that give the operator some measure of control over router resource usage. We also investigated whether these mechanisms help alleviate some of the adverse effects of large routing table loads (Section VI). First, in some newer router operating systems, a configurable *prefix limit* parameter can constrain the number of prefixes the router will accept from a neighbor. This allows the router to selectively tear down a BGP peering session with a “misbehaving” peer, while continuing to maintain

[1] This technical report (ISI-TR-2001-552) is based upon work supported by DARPA via the Space and Naval Warfare Systems Center San Diego under Contract No. N66001-00-C-8066 (”SAMAN”), and by the CONSER project supported by NSF.

routing state, and to forward traffic to and from other peers. We find, however, that in some routers this mechanism still exhibits routing table-size oscillations, repeatedly propagating routing updates to neighbors. Second, a mechanism called *route-flap damping* prevents the propagation of unstable routes across the infrastructure. In theory, this mechanism can limit the cascading router failure we observe. In practice, however, we find that the efficacy of this mechanism depends on the router failure mode which depends on router's implementation.

Our observations, we argue, indicate that modern day routers do not satisfactorily deal (and, we argue, they need to, even if routers are over-provisioned with memory to reduce the likelihood of memory overrun) with a sudden infusion of large routing table load. Clearly, the table-size oscillation and router cascading failures can be harmful to the infrastructure. Equally, the behavior in which routers require operator intervention to recover from large routing tables can cause loss of connectivity for extended periods of time. To some extent, the stateful nature of BGP contributes to the behaviors we report in this paper; a BGP speaker cannot unilaterally "shed" routes at high load because BGP speakers are required to store routes learned from neighbors [5]. Moreover, the BGP specification itself does not specify how routers should recover from resource overload. As we show, most router vendors choose to simply restart the routing processes or permanently cease the BGP connection. These approaches trade off route convergence and reachability. We conclude with some suggestions for more graceful router responses to large tables (Section VII).

II. Background and Related Work

BGP [5] is the current interdomain routing protocol employed on the Internet. Relevant to this work are two aspects of BGP: the route update mechanism and the error notification procedure. Each BGP router keeps the routing information advertised by its peering routers. Instead of periodically refreshing the peers with its local routing table, BGP speakers send *incremental* update information upon changes in its local routing database. This largely reduces the bandwidth overhead for routing messages and the processing overhead for the route decision. The error-handling mechanism defined in BGP deals with the errors in routing protocol messages and finite state machine of the routing process. Upon detecting an error, a BGP speaker sends a notification message to a peer, closes the BGP connection, releases all resources for that BGP connection, marks the routing table entries associated with that peer as invalid, and withdraws these routes from other BGP peers. However, error-handling mechanisms for resource exhaustion are not defined in the specification and are left as an implementation decision.

Recently, there has been significant work on understanding various aspects of BGP. As well, the BGP protocol continues to evolve. In the following sections, we review these advances in the areas most relevant to our work: stability and convergence.

A. Stability

One way to improve the stability of the routing system is to reduce the number of undesired routing updates that cause routing flaps. There are two mechanisms implemented in currently deployed BGP routers. One is the rate-limiting mechanism for route advertisement which is defined in BGP4. It specifies a configurable timer to control the minimum amount of time between consequent route announcements. This mechanism has the effect of packing multiple routing announcements into a single routing update, reducing routing flaps and messaging overhead. In most BGP implementation including our test routers, this timer is configurable on per-peer basis. A second mechanism is the route flap damping mechanism defined in [6]. It is used to detect unstable routes and suppress the propagation of their advertisements. It uses an exponential decay algorithm to predict the future stability of a route based on the recent history of stability. In this paper, we examine how, if at all, these mechanisms regulate the impact of large routing loads.

A recent proposed revision to the BGP protocol [7] suggests keeping forwarding state across TCP resets. It also proposes to use an empty UPDATE message as an End-of-RIB marker that indicate to its peer the completion of the initial routing update after the session is established. The restarting router defers route selection until it receives the End-of-RIB markers from all peers. These mechanisms allow the router to continue forwarding traffic while it re-establishes BGP peering sessions, a functionality called "graceful restart". This mechanism is somewhat complementary to the impact of large routing loads that we consider. It can possibly preserve routing forwarding capability across some of the kinds of failures we consider.

Other work has measured various aspects of the stability of the BGP routing system. To our knowledge, however, there is no existing literature that studies the impact of large routing loads on commercial routers. A measurement on the instability of the deployed BGP routing system, i.e., the Internet, was conducted by Labovitz *et al.* [8]. They classified various types of pathological routing updates and measured their frequencies. In a later paper [9], they reported some Internet routing instabilities caused by software bugs and artifacts of router vendor implementation decision. Shaikh *et al.* [10] studied the problem of BGP interactions with traffic. BGP adopts a keep-alive mechanism to detect the failure of the BGP session. Since current IP configurations do not typically prioritize routing traffic, routing messages share the same bandwidth with data traffic and there is a possibility that keep-alive message losses can trigger connection resets. Shaikh *et al.* provided a Markov-chain based model to analyze the times for such "false positives" to occur and recover.

B. Convergence

Unlike most interior gateway protocols using only distance metrics to compute the best route to a destination, BGP allows local policies to influence route selection. This leads to a possibility that the policy configurations at different Autonomous Systems conflict and result in the persistent route oscillations. Varadhan *et al.* [11] discovered the existence of this phenomenon. Griffin [12] provides a complexity analysis of this convergence problem, and proposes a modification [13] to BGP that would avoid such oscillations. These persistent oscillations are quite different from the table-size oscillations we observe in this paper.

Finally, Labovitz [14] examines the upper and lower bounds on BGP convergence latency for a route failure, failover and

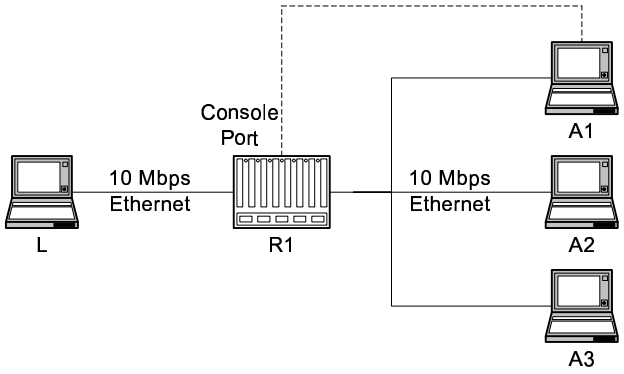

Fig. 1. Network topology used in single router experiments

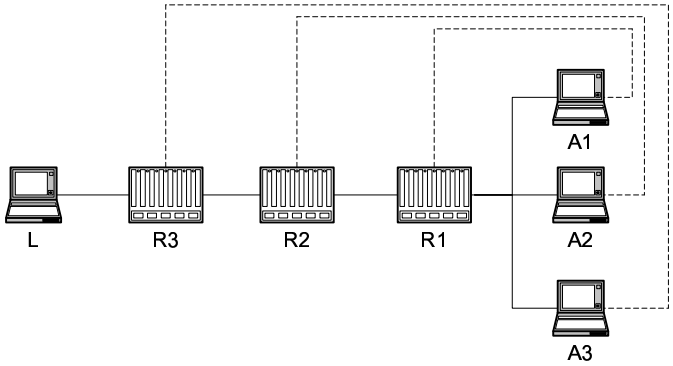

Fig. 2. Network topology used in simple topology experiments

repair. However, that work does not focus on router failure due to large table loads.

III. Experimental Setup

We conduct our experiments on two canonical topologies: a *single* router topology (Figure 1), and a *multiple router* topology (Figure 2). The first topology helps us understand the behavior of a single router under large BGP routing load, and the second helps us understand how large routing load injected at a single router affects other routers in the topology.

In the figures, we denote the advertising BGP speakers by A_i, the routers under test by R_i, and the observing BGP speaker by L. The advertisers are responsible for injecting BGP routing tables to the test router, while the observer simply peers with the test router without advertising any route. Both advertisers and observer log their interaction with the test router for later analysis.

To examine the router's response under different patterns of routing load, multiple BGP speakers (three advertisers and one observer) are used to peer with the test router such that each peer can generate different amount of routing loads. Our advertisers and observer are FreeBSD machines running GateD 3.6 [15]. This choice is partly dictated by our lack of access to many commercial routers, but it helps by providing more flexible logging facilities. As indicated in many router testing reports, turning on a commercial router's logging facilities (typically syslog to a remote UNIX machine) can dramatically degrade the router's performance. Using host-based BGP peers provides a light-weight alternative to record the router's external behavior. We do not believe that replacing host-based BGP speakers with commercial routers would qualitatively change our results.

To monitor the internal state of the routers under stress, a separate line is connected to the router's console (a serial port). We develop a program that can continuously execute commands on the router's console and record the output. All tested routers provide some internal state displaying commands. For example, the command "show ip route summary" on Cisco's command line interface displays the statistics of the current routing table. The execution of these commands places less burden on the router performance than the debug and logging facilities provided by the router OS. However, heavy processor load can delay execution of these commands. We observe command delays of up to 10 seconds for routers near failure.

We test three commercial routers shown in Table I. While we identify the vendors by name, we do not intend by this to compare vendor performance, or to malign the vendors' implementation choices. Rather, our goal is to understand how different routers behave under overload.

Each advertiser generates some large number of routes, depending on the experiment. All routes in our experiments are prefixes of length 24 bits. Different advertisers generate distinct prefixes. For all experiments described in this paper, the routers have no policy configuration for route import and export. In other words, all distinct sets of routes from different advertisers will be installed into all routing databases in the router. These choices differ somewhat from typical Internet routers, where, as of July 2001, the average prefix length advertised in inter-domain routing system is 22.6, and the average number of prefixes is 108k [1], and many routers employ policy configuration. However, these differences do not qualitatively affect our analysis of router behavior, as confirmed by experiments (not reported here) that vary prefix length and policy configurations.

To better understand how various routing events relate to each other, we use NTP (Network Time Protocol) to synchronize all machines' clocks. All our logging facilities (GateD logging and router console logging) record the time associated with the routing events including the sending and receiving of routing messages, router failures, routing table changes, etc. In this way, we can generate the timing figures for each routing event. Overlapping these figures by aligning the timestamps provides us a hint on the relation of routing events.

To understand router behavior we infer two simplified models of each router's internal databases, shown in Figure 3. The Cisco appears to have two internal tables: BGP protocol database, and a second table that combines current routes (RIB) and the line-card forwarding table (FIB). Although the BGP database and the RIB/FIB are logically separate, internal data may be shared. The Juniper, by contrast, appears to have three separate tables, one each for the BGP database, combined routes from all protocols (RIB), and line-card forwarding (FIB). We do not expect that these models capture all details of internal router structure, but they are sufficient to explain the results of our experiments.

IV. Router Response to Large Routing Tables

In this section, we present the results of experiments (Figure 1) that load a single router with a BGP table exceeding the

Router	CPU	Memory	OS (released date)
Cisco 7000	25-MHz Motorola MC68040	64M bytes	IOS 11.1 (Jun. 1999)
Cisco 12008 GSR	200-MHz IDT R5000	64M bytes	IOS 12.0 (May. 2001)
Juniper M20	330-MHz Pentium Pro	768M bytes	JUNOS 4.3 (Jan. 2001)
Juniper M20	330-MHz Pentium Pro	768M bytes	JUNOS 4.4 (Jun. 2001)

TABLE I. Configuration of Tested Routers

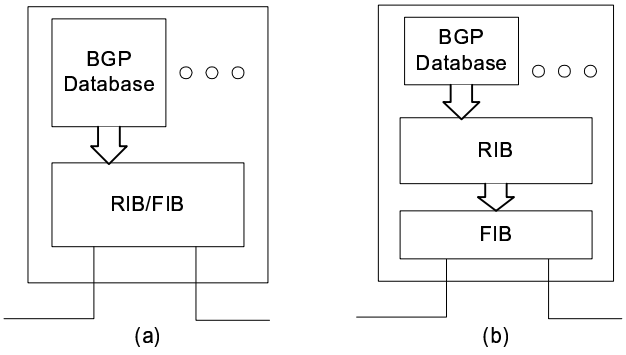

Fig. 3. Idealized models of routing databases in (a) Cisco routers and (b) Juniper router tested in this work.

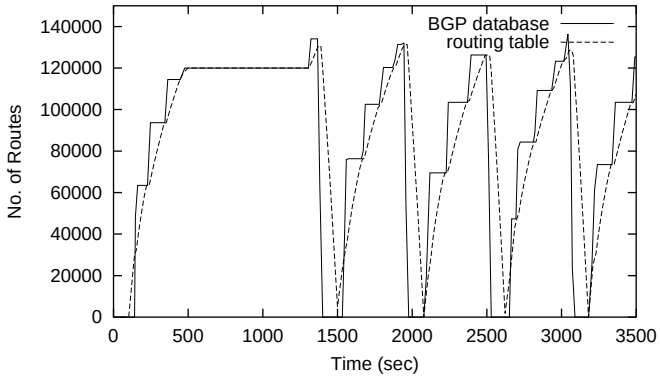

Fig. 4. Cisco 7000 under malloc failure. Initially, A_1 and A_2 announce 70k and 50k routes, respectively. After A_3 announces 20k routes, the router has a malloc failure (at time 1374) and resets all routing processes repeatedly.

router's memory capacity. These experiments are designed to understand the failure modes of a single router in isolation and to serve as a basis for multi-router studies. The first experiment is conducted with a Cisco 7000, the second with a Cisco 12008 GSR, and the third with a Juniper M20. We note that the details of our experiments depend on the exact memory configurations of the routers. However, we believe our qualitative conclusions about behavior of each router would be unchanged if we repeat our experiments with different memory configurations of these routers. Overall, we believe both vendors make very reasonable implementation decisions, given the protocol specifications.

A. Cisco 7000

We conduct this experiment with the Cisco 7000 using IOS 11.1. This router is capable of storing about 130,000 BGP routes. To understand how the router treats different peers that advertise different numbers of routes we configure the advertisers to exceed this capacity. Initially, A_1 and A_2 announce 70,000 and 50,000 routes, respectively. After these 120,000 routes have been processed in the router, A_3 announces 20,000 routes to the router. This last set of advertisements exceeds the router's memory capacity.

Figure 4 shows the number of routes stored in the router's databases as time proceeds. Our monitoring program extracts this information from the console of R_1. We first focus on the period of initial advertisement (from 101 seconds to 498 seconds). As expected, the received routes are first imported to the BGP database, then get installed into the routing table. However, the growth patterns of these two databases are somewhat unexpected. The BGP database grows stepwise instead of continuously, and waits for routing table to catch up before growing. This lag represents the time taken to process the routes (apply policies, insert into the tree structure, install routes into the line card). We will discuss its impact shortly.

Now we focus on the moment of overloading. At time 1306, A_3 starts announcing routes and R_1's BGP database size increases accordingly. When the BGP database size exceeds the memory capacity, R_1 declares a "malloc failure" and *all BGP peering sessions are closed*. Thus, even though R_1 has enough capacity to handle routes from each individual peer, and even though in our experiment it is advertiser A_3's advertisements that cause R_1 to exceed its memory, it tears down the BGP sessions with A_1, A_2, and L as well. We observe that the router clears the dynamic routes in the routing table and resets all other routing processes including the intra-domain routing processes. Only static routes and directly connected networks (direct routes) survive in the routing table across the reset.

This is an interesting failure mode. As we show below, not all router models fail in the same way. We note that the BGP specification itself does not specify what a router should do when it runs out of memory. However, the stateful design of BGP, in which a BGP speaker is required to remember routing information it learned from its peers, precludes a graceful degradation mode where the router can selectively "shed" excess routes. (A router can selectively "shed" peers; we shall see later that this is how other routers behave).

Continuing on with our analysis of Figure 4, we notice that R_1 then attempts to *re-establish BGP peering sessions* with all its neighbors! The BGP specification [5] doesn't mandate a particular behavior. This particular router implementation chooses to immediately reset all routing processes and re-establish BGP peering sessions. When the connections are re-established, all three advertisers start announcing routes almost simultaneously.

Since the router has re-established connections to all four peers, the load of 140,000 routes from A_1, A_2, and A_3 is processed and results in *another* "malloc failure". This sequence

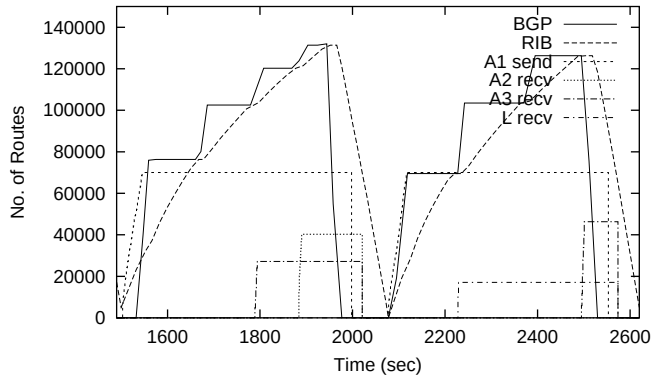

Fig. 5. Route installation and re-advertising in Cisco router with default MinRouteAdver (30 seconds).

can repeat indefinitely, causing a repetitive router failure as the entire routing table *oscillates*. Operator intervention is required to terminate this oscillation. To our knowledge, this phenomenon has not been previously analysed in the literature, although we assume it has been observed in the field.

Now we turn our attention from the router's internal state to its external behavior, namely the sending and receiving of routing protocol messages. Focusing on the moment of failure, we observe that no route withdrawal messages or error notification messages are issued by the router to the neighbors. The router simply tears down the BGP peering session. The neighbor detects the disconnection only when the router sends an Open Message to re-establish the connection, or when the Holddown timer for the connection expires. In other words, the neighbors have no idea about what type of failure causes the teardown.

Figure 5 shows how the route announcements propagate through the router during two up-down cycles of the router failure. Here we try to understand the temporal relationship between different routing events, namely, the advertisers announcing routes, the router importing the routes into BGP database, the router installing the routes into its routing table, and neighbors receiving routes re-advertised by the router. For ease of exposition, we show only A_1's announcements, but the result is the same for other advertisers' announcements. In the figure, "A_1 send" represents the cumulative number of routes announced by A_1 during the lifetime of each BGP session between it and the router. Similarly, "A_2 recv" represents the cumulative number of routes re-advertised by the router to A_2. "A_3 recv" and "L recv" are similarly defined. As in Figure 4, the lines "BGP" and "RIB" represent the total numbers of routes in BGP database and routing table, respectively.

We study the number and the frequency of re-advertisement since this determines the impact on the router's neighbors. The result shows that the router re-advertises the routes only when the routing table is consistent with the BGP database. This complies with the BGP specification. As we indicated before, instead of processing the routes one by one, the router will import a fraction of received initial advertisements into BGP database, then gradually install them to the routing table. This may reflect the implementation decision of accelerating the route-selection process by batch execution. As a result, however, re-advertisements to a particular neighbor are batched, and the intervals between batches depends the processor speed, the number of received initial advertisements, and (as we show below) the minimum route advertising interval (MinRouteAdver) timer.

As shown in the figure, the re-advertisements to A_2, A_3, and L don't happen in every failure cycle. In first cycle, only A_3 and A_2 get the re-advertisements, while in second cycle only L and A_3 receive the re-advertisements. We conduct other experiments by reducing MinRouteAdver to 10 and 1 seconds, and observe that this does make the router re-advertise to all peers more frequently. Specifically, the router will re-advertise to all peers 2 or 3 times in each failure cycle when the MinRouteAdver becomes less than 10 seconds. In Section V, we illustrate the impact of these propagating routes on a larger topology.

B. Cisco GSR

With the Cisco GSR running IOS 12.0, our experiences are different. We setup the same scenario as the experiment in Section IV-A. Since this router (with 64M memory) can store about 16,000 BGP routes, we have A_1, A_2, and A_3 announce 10,000, 5,000, and 2,000 routes, respectively. When receiving excess routing announcements, the router permanently stops responding to the interface where it peers with the advertisers. Thus, even though it is advertiser A_3's advertisements that causes the router to exceed its memory, the BGP sessions with A_1 and A_2 are also disconnected due to the interface failure. We omit the graph of this router behavior here, but it resembles the period from 0 to 1500 seconds of Figure 4. Although the router is still able to process the console commands, we need to manually reboot the router to bring it back to normal operation.

Thus, unlike the Cisco 7000 running IOS 11.1 which resets its routing processes, the Cisco GSR simply "freezes". It does not leak routes to its neighbors. Both the router and the advertisers detect the link failure when the Hold-Down timers expire. This different failure mode has less impact on the infrastructure, in the sense that it doesn't exhibit table-size oscillations that can cause route flaps at neighboring routers. However, this mode *requires operator intervention* in order to restore connectivity; even if the advertising routers no longer announce large routing tables, R_1 has no way of re-establishing the BGP connection until the operator intervenes.

C. Juniper M20

When we repeat our experiment on a Juniper M20 with JUNOS 4.3, we find that the result is similar to that of the Cisco GSR. Unlike the Cisco routers, where a large routing table will overrun the BGP database before the routing table, our Juniper router has much more capacity in the BGP database than the forwarding table. As a consequence, a large routing table (about 500,000 routes for this router) overruns the forwarding table before the BGP database. The router completely stops responding to the interface where it peers with the advertisers. We need to unplug the cable from the interface card and re-plug the cable in order to resume the router's normal operation. We report this bug to the vendor who then releases a new version of the OS with a fix.

After consulting with Juniper support personnel we upgrade the router to a new OS release (JUNOS 4.4) and the router handles the failure more gracefully. When we send it the same num-

ber of routes, the forwarding engine generates a "malloc failure" error message indicating that some prefixes can not be installed into the forwarding table but the interface remains operational. Thus, it appears that our experiment exceeds the Juniper's line card memory but not the route processor memory. Even though the FIB capacity is exceeded, the router *continues to route packets for all prefixes*. We are not sure how it continues to route packets; possibly it fills the line-card FIB on demand, or possibly it forwards packets not handled in the FIB through the RIB.

Because we have access to the Juniper only for a very limited time we are not able load the router to the point where memory for the the BGP database or RIB would have been exhausted. Had we done so, we believe our results would have been similar to either Cisco 7000 or Cisco GSR, either resetting the connections and oscillating, or freezing.

V. Impact of Large Routing Tables on Simple Topologies

In the previous section we observe the results of injecting a large routing table into a single router. In at least one case, this results in a table-size oscillation. A follow-on question is how would table-size oscillation affect the network topology as a whole? To study this question we examine the behavior of a chain of BGP routers where head of the chain is subjected to a large routing table load. Admittedly, this simple topology is not representative of any realistic ("Internet-like") topology. However, we suggest that this simplified topology captures the essence of how different-sized routers interact and exhibits interesting behavior.

To assess the propagation of these table-size oscillations, we conduct three types of experiments. All experiments use the same topology as shown in Figure 2. However, the three routers in the topology have varying route memory; this allows us to study the behavior both of homogeneous and heterogeneous router chains. For this experiment, we do not have access to a heterogeneous collection of routers (our experimental testbed contains only Cisco 7000s). Due to the limited flexibility on the memory configuration of Cisco 7000, it's impossible to configure the three routers with three different capacity memory modules. Instead, we *emulate the effects* of different physical memory capacities by preloading some of the routers with different numbers of static routes, and configuring the router not to re-advertise the static routes. By consuming some memory through software, less memory is left for BGP-learned routes. While this is a contrived way of deriving experimental configurations with differing memory, we expect that our results would be similar if our experiments were conducted on mix of routers with different physical memory.

A. Routers with the same capacity

In this experiment, all three routers are configured with the same amount of memory. This scenario is intended to study the impact of table-size oscillation in relatively homogeneous sections of the Internet topology, such as routers within a backbone network.

Figure 6 shows the changes in the BGP database size for the three routers when advertisers A_1 and A_2 overload R_1. (A_1 sends 80,000 routes at time 215 and A_2 sends 70,000 at time

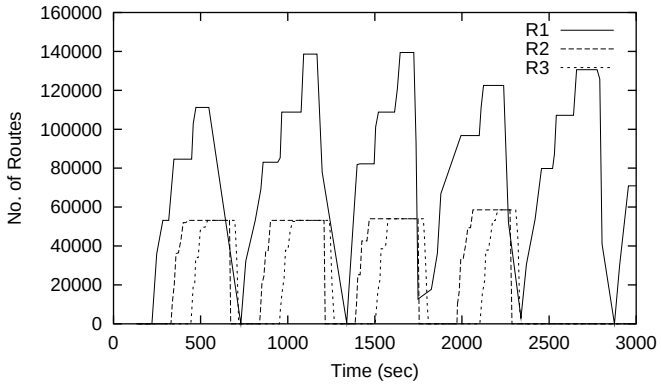

Fig. 6. Simple topology experiment: routers have the same capacity and R_1 is overloaded. We show the BGP database sizes of R_1, R_2, and R_3.

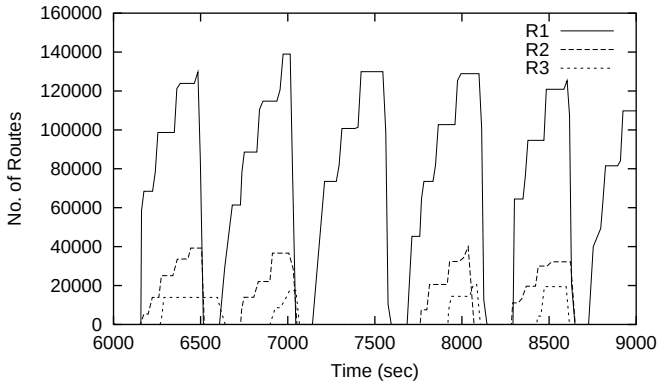

Fig. 7. Simple topology experiment: routers have decreasing capacities. R_1 (with large capacity) is overloaded and the failure extends to R_2 (with medium capacity).

274.) Before failing, the router (R_1) propagates some routes to R_2 which, in turn, propagates a subset of these to R_3. In this case, then, routers see their routing table fluctuate over time. R_1 periodically fails, and R_2 and R_3 see their routing tables change as R_1's routes are advertised and withdrawn over time. Thus, each router downstream of R_1 see table-size oscillations, albeit smaller in amplitude and not resulting in router failures.

Clearly, for those routes never advertised by R_1, the observer L will never be able to reach to the applied prefixes. However, for those routes repeatedly advertised by R_1, L observes routing flaps for the applied prefixes. As indicated in Section IV-A, the number of propagated routes and the frequency of these fluctuations are determined by the speed of route processing in R_1 and R_1's MinRouteAdver timer for the BGP session to R_2.

Also, we observe that sometimes routes are propagated down the chain of routers, but sometimes they are not. For example, the last peak in Figure 6 (for 2300 to 2900 seconds) does not propagation routes. We believe that this failure to propagate routes is due to an interaction between R_1's batched advertising mechanism and its MinRouteAdver timer. We discuss its impact further in Section VI-B.

B. Large-Medium-Small

In this experiment, the routers are configured with decreasing memory capacities (R_1 is largest, R_2 is medium, and R_3 is

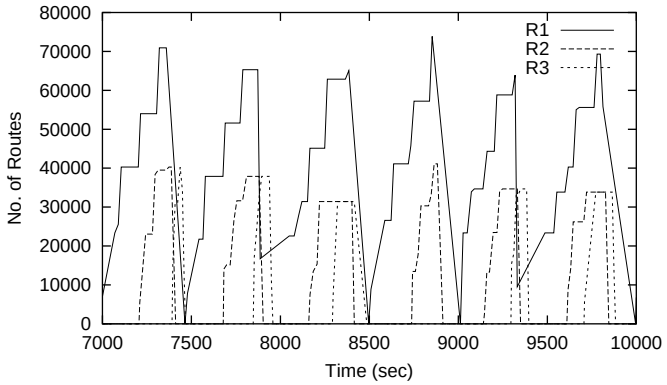

Fig. 8. Simple topology experiment: routers have increasing capacities and R_1 (with small capacity) is overloaded.

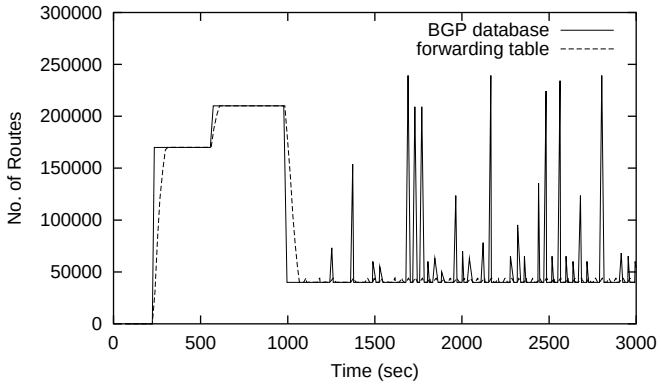

Fig. 9. Prefix limiting in Juniper M20 tears down the offending session. This results in table-size oscillations.

smallest) and we inject a large routing table into R_1. This scenario is meant to mimic a misconfiguration in a large ISP, and its effects on customers and smaller downstream ISPs.

As in previous experiment, the failed router R_1 advertises some routes to medium router which then propagates to small router. Due to the batched advertising implementation in these routers, the number of routes advertised to R_2 is much less than the number of routes received by R_1. Depending on the memory configuration of the medium and small routers, we can observe an interesting phenomenon that we call a *cascading failure*. In Figure 7 we show an extreme experiment where R_1 is capable of dealing with 130,000 routes while R_2 and R_3 can only deal with 40,000 and 20,000 routes, respectively. We overload R_1 with 150,000 routes. Before R_1 fails, it advertises nearly 60,000 routes to R_2 which then also fails. Similarly, before R_2 fails, it propagates about 20,000 routes to R_3, sometimes causing R_3 to fail. This interesting phenomenon might suggest an explanation for some of the earlier Internet failures caused by large routing table leakages [4].

Note that in this case, powering down the medium or large router for some time *cannot alleviate* these failures because upstream routers or the advertisers will recreate the problem. That is, a downstream ISP can't really eliminate the cascades by local actions such as power cycling equipment or installing filters (although installing filters can prevent the cascade from spreading downstream). The failure condition can only be eliminated by reconfiguring the advertisers to not exceed all routers' capacities.

C. Small-Medium-Large

Finally, for completeness, we study a scenario where the routers are configured with increasing memory capacities and the router with the least memory R_1 is subjected to a large routing table load. This scenario is meant to mimic the origination of a failure in a small (leaf) ISP, and is meant to study its propagation towards the core.

Figure 8 shows an experiment where the small router R_1 is capable to storing near 70,000 routes. Upon failure, R_1 advertises some routes to medium router R_2 which then propagates to big router R_3. As with the previous scenarios, we observe the propagation of the table-size oscillation from the smaller router to its neighbors. However, the amplitude of these oscillations are smaller than with homogeneous routers (Section V-A) and we do not observe the cascading failures from decreasing capacity routers (Section V-B). In this sense, this scenario is the least disruptive of the three.

VI. Impact of Resource Control Mechanisms

The table-size oscillations described in Section IV-A are caused by a resource overrun (in this case, memory). At least two kinds of relevant resource control mechanisms are currently deployed in routers in the Internet: *prefix limiting* and *route-flap damping*. In this section, we examine their impact on route-table oscillations and router cascading failures.

A. Prefix Limiting

Prefix limiting is a mechanism that places a configured limit on the number of prefixes that a router will accept from a given BGP peer. Typically, this feature is implemented using two thresholds, each of which can be defined on a per-peer basis: a *warning threshold* and a *teardown threshold.* When the number of prefixes announced by a particular peer reaches the warning threshold, the router generates a warning message to its logging facility. When the number exceeds the teardown threshold, the router tears down the BGP peering session with its neighbor. Clearly, this feature prevents router memory overrun (and its attendant resetting of all BGP peerings (Section IV-A)) caused by a single large routing table infusion from a peer. When a peer exceeds its prefix limit, only the router's BGP session to that peer is affected; all other peering sessions are kept alive, and the router continues to forward traffic to the unaffected prefixes. Prefix limiting is not defined in any draft or standard BGP specification, but both vendors in our study implement this feature.

In order to understand the impact of prefix limiting on table-size oscillations and on cascading failures, we repeat the experiment of Figure 1 on our Juniper M20 running JUNOS 4.4 with prefix limiting. For this experiment, we configure the teardown threshold for all peers to 200,000. At time 250 we configure advertiser A_1 to announce 170,000 routes and, at time 500, A_2 announces 40,000 routes. After the router R_1 processes these 210,000 routes, at time 1000, A_1 send routing updates consisting of additional 40,000 routes.

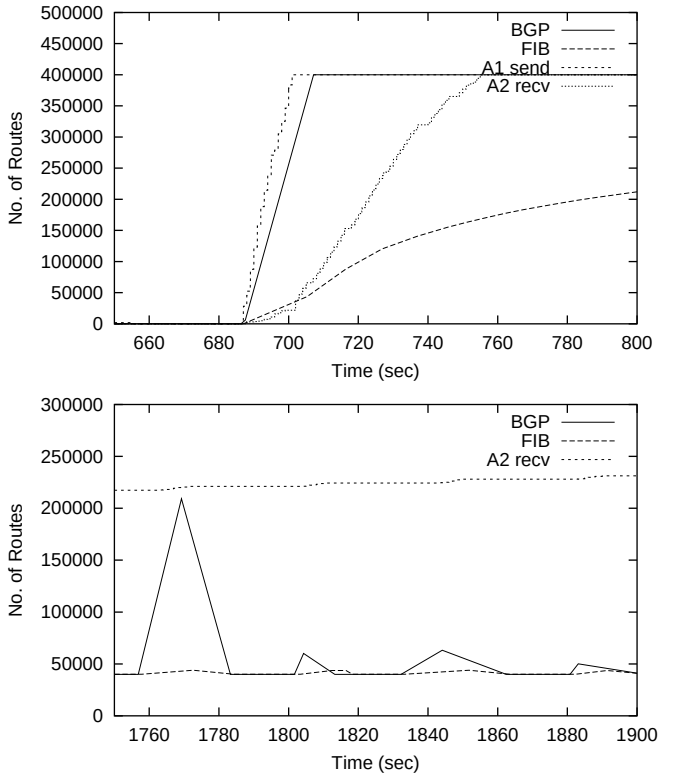

Fig. 10. Route installation and re-advertising in Juniper router during (a) initial BGP updates, and (b) repetitive BGP connection restarts.

Figure 9 shows the result of this experiment. At time 1000, the total number of routes announced by A_1 exceeds the configured limit, and so R_1 closes the peering session to A_1. A_1 re-establishes the peering session with R_1 and re-advertises the 210,000 routes. Again, R_1 tears down the BGP peering. Thus, this particular implementation of prefix limits *still exhibits table-size oscillations* that can only be stopped by operator intervention (Section IV-A). In that experiment, however, the router resets all BGP peering sessions; here, only the session that causes the router to exceed its limits is torn down.

To more closely understand the mechanics of prefix limiting we examine how the Juniper responds to a large routing table infusion in the absence of prefix limits. Figure 10(a) shows the results of a route-loading experiment where no prefix limit is set and only one advertiser, A_1, announces 400,000 routes. Figure 1 shows the topology for this experiment. The figure illustrates four routing events: A_1 advertising, the router R_1's BGP database growing, R_1's forwarding table growing, and A_2 receiving advertisements from R_1. Unlike the Cisco router, the Juniper's BGP database grows without waiting for the forwarding table to catch up. Another difference is that R_1 re-advertises the routes immediately after the BGP database receives them, even though most routes do not seem to have been installed in forwarding table. We refer to this implementation as allowing *premature route advertisements*. Although these early announcements accelerate initial exchange of routing information between two routers, they raise the danger of transient routing black hole.

This phenomenon of premature advertisements explains an interesting aspect of the prefix limit experiment. Figure 10(b) illustrates microscopically how the prefix-limiting works. R_1 tears down the BGP peering immediately after it detects the peer announcing too many routes. However, before it tears down the connection, it still propagates a small number of routes to other neighbors. The line "A_2 recv" represents the cumulative number of route announcements received by A_2. From time 1750 to 1900, it increases by 13,742. In other words, there are thousands of route announcements propagated to A_2 each time the connection between the router and A_1 is re-established. When the connection is disconnected, R_1 withdraws these routes from A_2. This results in routing flaps in A_2. In a larger topology, these *table-size oscillations* can propagate beyond neighboring routers, and may impose significant processing load on many routers in the network. The number of propagated routes depends on the time the advertiser A_1 takes to finish the initial advertisement. This in turn depends on the processor speed of A_1, the bandwidth of the peering link, and the processor speed of the router R_1.

We also conduct the same experiment on Cisco GSR with IOS 12.0. The result is a little different. When R_1 detects an advertiser exceeding the threshold, it permanently denies further connection attempts from that advertiser. Operator intervention is required to re-establish the peering session. Thus, no table-size oscillations are observed. However, until R_1's operator intervenes, its customers cannot reach prefixes advertised by A_1, even if A_1 is no longer injecting a large routing table prefix.

B. Route Flap Damping

In Section V-B we describe how a large-medium-small router configuration will propagate table-size oscillations and can result in cascading failures. *Route-flap damping* is a deployed mechanism in Internet routers that is used to limit the rate of propagation of unstable routing information [6]. In this section, we investigate its efficacy in preventing the spread of cascading failures.

Route flap damping works as follows. Associated with each route in the routing table is a number that represents the instability history of the route. Each time a route changes, its instability value increases linearly. When the value exceeds a configurable *suppress threshold*, the route is deemed to be unstable and the router stops advertising that route. The instability value decays exponentially over time with a configurable *half-life*. When the decayed instability value falls below a configurable *reuse threshold*, or after a route has been deemed unstable for a *maximum suppress time*, the router deems that route stable and reinstates that route in its routing table.

We conduct experiments to investigate the effectiveness of this mechanism on damping the two types of route flaps described in previous sections: route flaps injected by a failed Cisco 7000 (Section IV-A), and route flaps injected by a Juniper M20 using prefix-limiting (Section VI-A). In all experiments, we use the vendor supplied default route flap damping parameters: a half-life of 15 minutes, a reuse threshold of 750, a suppress threshold of 2000, and a maximum suppress time of 60 minutes.

For the route flaps generated as in Section IV-A, route flap damping cannot detect and suppress flaps. This is because in this

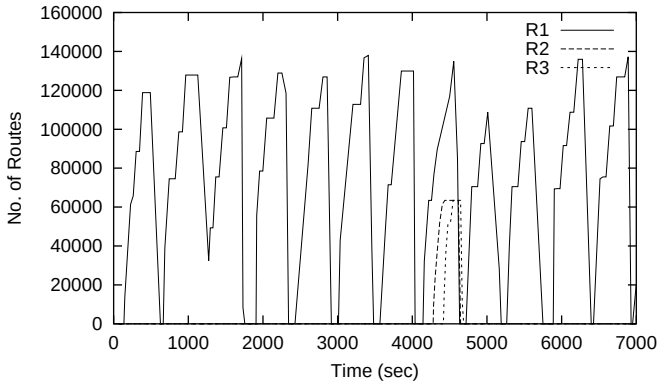

Fig. 11. Same experiment as in Section V-A but enabling the route flap damping in all three routers with default damping parameters. The routing table flaps still exist in R_2 and R_3.

case all BGP connections are terminated and the router loses all state, including the instability history associated with each route. To alleviate the routing flaps, all the router's peers have to have damping enabled. Even so, route flaps can't still be completely eliminated in some cases. Figure 11 shows an example where we conduct the same experiment as in Section V-A but enabling the route flap damping in all three routers with default damping parameters. In this case, as described in Section V-A, R_1 does not propagate routes to R_2 periodically. Thus, in R_2, the instability values associated with the applied routes decay exponentially during the periods that no route readvertisements are received. When the period is long enough to have the instability values fall below the reuse threshold and R_1 propagates the routes again, R_2 reinstates these routes in its routing table (as shown in the figure from time 4280 to time 4640).

However, for the kind of route flapping caused by prefix limiting (Section VI-A), we find that route flap damping can be effective. By using prefix-limiting, a router can keep around instability history for prefixes advertised by the offending peer. When the peer repeatedly re-advertises its table, the router can suppress these routes. This is an apparently attractive reaction to the failure, in that it confines the impact of route flapping to the immediate peers of a router. Note however, that route flap damping is not without its drawbacks. Because of the way damping works, for several tens of minutes after the peer stops injecting a large routing table, the router continues to suppress these routes. This effectively ensures that the corresponding prefixes are unreachable for extended periods of time.

VII. Summary, Discussion and Conclusions

A. Summary of results

Table II summarize our experimental results. A large routing table can overload the router's BGP database, routing table, or forwarding table in the line card depending on the router's memory architecture and the memory capacities of these data structures. The router responds to it by *resetting* the routing processes, *freezing* the network interface, or possibly *degrading* the performance of traffic forwarding. These failure modes impose different kinds of problems on the routing system. Degrading the forwarding performance causes no changes as to the routing topology. Freezing the network interface results in a link failure. As a consequence, the peerings (including BGP and IGP) through this link are closed and the routing topology is changed. Unlike other failure modes where the router can resume the normal operation when the routing overload disappears, interface failure requires manual operator intervention for repair. The intervention may involve a router reboot. Resetting the routing processes impose most disruptive impact on the routing system—it can result in persistent routing flaps and even cascading failures.

Prefix limiting is the one mechanism currently deployed to prevent the memory overloads. However, prefix limiting can not completely eliminate this problem since it is configured on per-peer basis and may overcommit router resources. Consider the scenario that the router is capable of loading 200,000 routes, and the prefix limits for all peers are set to 80,000. When three peers announce 70,000 routes separately, each conforms to its limit, but the aggregate routing load of 210,000 routes will exceed the router's capacity.

Prefix limiting implemented by different vendors imposes different penalties on the offending BGP session. Cisco routers permanently deny further connection, even after the peer decreases the number of announced prefixes to under the limit. On the other hand, Juniper routers allow further connection which results in repetitive connection establishing and tearing down. This produces the similar phenomenon of table-size oscillation as the "reset" failure mode. Since Juniper router's policy is to prematurely advertise routes, some route announcements that will be withdrawn are leaked to all other neighbors, causing routing flaps.

For those router responses that generate route flaps, *route-flap damping* is a deployed mechanism to limit their negative impact on the routing system. For the route flaps caused by prefix limiting, enabling damping in the overloaded router can effectively detect and suppress the flaps. However, for the route flaps generated by routing process reset, enabling this mechanism in the overloaded router has no effect. Even when all its peers enable this mechanism, in some cases, the route flaps can't be completely suppressed.

Our results reveal a surprising fact that current router implementations are quite vulnerable to routing overload. We expect that this vulnerability is because operational guidelines strongly encourage that routers be equipped with several times more memory than is required. However, we believe that exploring this aspect of router performance is important because misconfiguration is still possible and improper failure handling in even one router can impose substantial burden on the entire routing system. Also, although our experiments are conducted for BGP, we believe that our finding suggests the same vulnerability may exist in other stateful protocols.

B. Possible remedies and future directions

Here we discuss the possibility of a graceful way to deal with the routing table overload. First we note that the BGP routing overload may result from the misconfiguration of routers in neighboring Autonomous Systems. In other words, the sources of overloading may not be controlled by local Autonomous System. This makes trouble-shooting more difficulty since multiple

Router	Table Overloaded	Response	Impact	Peers Affected	Damping Efficacy
7000 (IOS 11.1)	BGP database	reset	route flap	all	sometimes
GSR (IOS 12.0)	BGP database	freeze	link failure	some	N/A
M20 (JUNOS 4.3)	forwarding table	freeze	link failure	some	N/A
M20 (JUNOS 4.4)	forwarding table	degrade	low performance	no	N/A
GSR with prefix limits	still possible	deny	no peering	one	N/A
M20 with prefix limits	still possible	reset	route flap	all	all times

TABLE II. Summary of Experimental Results

administration authorities are involved.

Second, as we indicated before, the stateful design of BGP precludes a graceful degradation mode where the router can unilaterally and incrementally shed excess routing load, that is, discard portion of prefixes learned from neighbors. This limitation is different from route filtering where, even after the router filter and don't propagate a route learned from a neighbor, the router still remember that route and the fact that the neighbor advertises the route. However, simply discarding excess routes would not do that, hence violate the BGP semantics.

Thus, we come up with some goals for designing a graceful failure mode. We don't suggest the completeness of this list.

- Don't violate the BGP protocol, especially its stateful design.
- Avoid persistent route flaps.
- Avoid failure modes that require manual operator intervention.
- Keep as many routes active as long as possible under overload.
- Provide predictable behavior to make trouble-shooting as easy as possible.

A simple design may be a degradation mode followed by a variant of the prefix limit mechanism. Specifically, in addition to configuring the prefix limits on per-peer basis, we can set the prefix limit for all routing databases. This setting requires knowledge of memory capacity of the router. Thus, the vendor may preconfigure the limits for various memory configurations that the router supports. When the number of learned routes exceeds this limit, the router then temporarily tears down the peering that most recently announces new routes. The period of teardown can be controlled by a configurable timer, so if the peer repairs itself, things can completely recover without operator intervention. Obviously, in configuring this timer, we need to take account of the possibility of route flaps.

This degradation mode meets the first three goals listed above. It has an advantage over the original prefix limiting mechanism, that is, it can effectively prevent the resource exhaustion. To accomplish this it sacrifices is one BGP peering. We suggest that this partial failure is preferable to either freezing the interface or resetting all connections. We suggest terminating the peering session that pushes the router over the edge because, as the session that changed most recently it seems suspect as the source of overloading. This approach is not perfect, since in practice the overload may be generated elsewhere[2] and the allowed capacity may be subject to specific service-level agreements. Because "proper" allocation may be ultimately governed by business agreements, definition of a fair policy is outside the scope of this paper.

A second direction of future work is to understand how the set of relevant router configurations (route-flap damping, prefix limits, and other knobs) are best tuned for safe operation in practice for wider range of router and peer configurations.

[2]For example, given peers A_1 and A_2, each with 50,000 routes, and R_1 with a capacity of 200,000, A_1 might advertise 149,999 routes followed by A_2 advertising 50,002.

REFERENCES

[1] BGP Daily Statistics, `http://www.telstra.net/ops/bgp` and `bgp-stats@lists.apnic.net`.

[2] "News: BGP router failures caused by misconfiguration," May 2000, `http://www.unixathome.org/adsl/2000_05/0013.html`.

[3] "Report: Internet core router test," LightReading.com, Mar. 2001, `http://www.lightreading.com/document.asp?site=lightreading&doc_id=4009`.

[4] R. Barrett, S. V. Haar, and R. Whitestone, "Routing snafu snips net service," Apr. 1997, `http://www.zdnet.com/zdnn/content/inwk/0413/inwk0032.html`.

[5] Y. Rekhter and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 1771, Mar. 1985.

[6] C. Villamizar, R. Chandra, and R. Govindan, *BGP Route Flap Damping*, RFC 2439, Nov. 1998.

[7] S. Ramachandra, Y. Rekhter, R. Fernando, J. G. Scudder, and E. Chen, *Graceful Restart Mechanism for BGP*, draft-ietf-idr-restart-01.txt, Dec. 2000.

[8] Craig Labovitz, G. R. Malan, and Farnam Jahanian, "Origins of internet routing instability," in *Proceedings of the IEEE INFOCOM*, 1999.

[9] Craig Labovitz, G. R. Malan, and Farnam Jahanian, "Internet routing instability," in *Proceedings of the ACM SIGCOMM*, 1997.

[10] Aman Shaikh, Lampros Kalampoukas, Rohit Dube, and Anujan Varma, "Routing stability in congested networks: Experimentation and analysis," in *Proceedings of the ACM SIGCOMM*, 2000, pp. 163–174.

[11] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," Tech. Rep. 96-631, University of Southern California, Computer Science Department, Sept. 1997, `ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/96-631.ps.Z`.

[12] Timothy G. Griffin and Gordon Wilfong, "An analysis of bgp convergence properties," in *Proceedings of the ACM SIGCOMM*, 1999, pp. 277–288.

[13] Timothy G. Griffin and Gordon Wilfong, "A safe path vector protocol," in *Proceedings of the IEEE INFOCOM*, 2000, pp. 490–499.

[14] Craig Labovitz, Abha Ahuja, Abhijit Abose, and Farnam Jahanian, "An experimental study of delayed internet routing convergence," in *Proceedings of the ACM SIGCOMM*, 2000, pp. 175–187.

[15] Merit GateD Consortium, `http://www.gated.org`.