Delayed Internet Routing Convergence [*]

Craig Labovitz
Microsoft Research
One Microsoft Way
Redmond, WA 98052-63999

labovit@microsoft.com

Abha Ahuja, Abhijit Bose
Farnam Jahanian
University of Michigan
Department of Electrical Engineering and
Computer Science
1301 Beal Avenue
Ann Arbor, MI 48109-2122

{ahuja,abose,farnam}@umich.edu

ABSTRACT

This paper examines the latency in Internet path failure, failover and repair due to the convergence properties of inter-domain routing. Unlike switches in the public telephony network which exhibit failover on the order of milliseconds, our experimental measurements show that inter-domain routers in the packet switched Internet may take tens of minutes to reach a consistent view of the network topology after a fault. These delays stem from temporary routing table oscillations formed during the operation of the BGP path selection process on Internet backbone routers. During these periods of *delayed convergence*, we show that end-to-end Internet paths will experience intermittent loss of connectivity, as well as increased packet loss and latency. We present a two-year study of Internet routing convergence through the experimental instrumentation of key portions of the Internet infrastructure, including both passive data collection and fault-injection machines at major Internet exchange points. Based on data from the injection and measurement of several hundred thousand inter-domain routing faults, we describe several unexpected properties of convergence and show that the measured upper bound on Internet inter-domain routing convergence delay is an order of magnitude slower than previously thought. Our analysis also shows that the upper theoretic computational bound on the number of router states and control messages exchanged during the process of BGP convergence is factorial with respect to the number of autonomous systems in the Internet. Finally, we demonstrate that much of the observed convergence delay stems from specific router vendor implementation decisions and ambiguity in the BGP specification.

1. INTRODUCTION

In a brief number of years, the Internet has evolved from an experimental research and academic network to a commodity, mission-critical component of the public telecommunication infrastructure. During this period, we have witnessed an explosive growth in the size and topological complexity of the Internet and an increasing strain on its underlying infrastructure. As the national and economic infrastructure has become increasingly dependent on the global Internet, the end-to-end availability and reliability of data networks promises to have significant ramifications for an ever-expanding range of applications. For example, transient disruptions in backbone networks that previously impacted a handful of scientists may now cause enormous financial loss and disrupt hundreds of thousands of end users.

Since its commercial inception in 1995, the Internet has lagged behind the public switched telephone network (PSTN) in availability, reliability and quality of service (QoS). Factors contributing to these differences between the commercial Internet infrastructure and the PSTN have been discussed in various literature[22, 16]. Although recent advances in the IETF's Differentiated Services working group promise to improve the performance of application-level services within some networks, across the wide-area Internet these QoS algorithms are usually predicated on the existence of a stable underlying forwarding infrastructure.

The Internet backbone infrastructure is widely believed to support rapid restoration and rerouting in the event of individual link or router failures. At least one report places the latency of inter-domain Internet path failover on the order of 30 seconds or less based on qualitative end user experience [13]. These brief delays in inter-domain failover are further believed to stem mainly from queuing and router CPU processing latencies [3, (message digests 11/98, 1/99)]. In this paper, we show that most of this conventional wisdom about Internet failover is incorrect. Specifically, we demonstrate that the Internet does **not** support effective inter-domain failover and that most of the delay in path restoral stems solely from the unexpected interaction of configurable routing protocol timers and specific router vendor protocol implementation decisions during the process of delayed BGP convergence.

---

[*] Partially supported by National Science Foundation Grants NCR-971017 and NCR-961276.

The slow convergence of distance vector (DV) routing algorithms is not a new problem [20]. DV routing requires that each node maintain the distance from itself to each possible destination and the vector, or neighbor, to use to reach that destination. Whenever this connectivity information changes, the router transmits its new distance vector to each of its neighbors, allowing each to recalculate its routing table.

DV routing can take a long time to converge after a topological change because routers do not have sufficient information to determine if their choice of next hop will cause routing loops to form. The count-to-infinity problem [20] is the canonical example used to illustrate the slow convergence in DV routing. Numerous solutions have been proposed to address this issue. For example, including the entire path to the destination, known as the *path vector* approach, is used in the Border Gateway Protocol (BGP), the inter-domain routing protocol in the Internet. Other attempts to solve the count-to-infinity problem or accelerate convergence in many common cases include techniques such as split horizon (with poison reverse), triggered updates, and the diffusing update algorithm [7].

Although the theoretical aspects of the delayed convergence problems associated with DV protocols are well known, this paper is the first to our knowledge to investigate and quantitatively measure the convergence behavior of BGP4 deployed in today's Internet. In [4], the authors showed that in the worst case, the original Bellman-Ford distance vector algorithm requires $O(n^3)$ iterations to find the shortest path lengths for a network with n nodes. However, we are not aware of any published result of a similar bound for path vector algorithms. The adoption of the path vector is widely and incorrectly believed to provide BGP with significantly improved convergence properties over traditional DV protocols, including RIP [12].

A number of recent studies, including Varadhan et al. [23] and Griffin and Wilfong [8] have explored BGP routing *divergence*. As we describe in the next Section, BGP allows the administrator of an autonomous system to specify arbitrarily complex policies. In BGP divergence, Griffin and Wilfong show that it is possible for autonomous systems to implement "unsafe," or mutually unsatisfiable policies, which will result in persistent route oscillations. Griffin et al. in [9] and Rexford et al. in [6] also describe modifications to BGP policies which guarantee that the protocol will not diverge. The authors of all these papers note that BGP divergence remains a theoretical finding and has not been observed in practice. Our work explores a complimentary facet of BGP routing – the convergence behavior of safe, or satisfiable routing policies. As we describe in the next Section, deployed Internet routers default to a constrained shortest path first route selection policy. We show that even with this constrained policy, the theoretical upper-bound on complexity for BGP convergence is factorial with respect to the number of autonomous systems.

Bhargavan et al. in [4] provide a stricter upper bound on the convergence of RIP. The authors account for implementation details of RIP including poison reverse, triggered updates, and split-horizon, which provide for improved convergence behavior over previous analyses of Bellman-Ford algorithms. In [24], the authors simulated the convergence behaviors of several algorithms including a distributed Bellman-Ford and present metrics for comparing the convergence properties of these different protocols. In this work, we similarly focus on both measuring the convergence latencies of BGP and developing theoretical upper and lower bounds.

In [17], Labovitz et al. describe significant levels of measured Internet routing instability. The authors show that most Internet routing instability in 1997 was pathological and stemmed from software bugs and artifacts of router vendor implementation decisions. In a later paper, Labovitz and his co-authors show in [18] that once ISPs deployed updated router software suggested by [17], the level of Internet routing instability dropped by several orders of magnitude. Finally, in [16], Labovitz et al. measured the rate of network failure, repair and availability. In this work, we present a complimentary study of both the impact and the rate at which inter-domain repair and failure information propagates through the Internet. We also measure the impact of Internet path changes on end-to-end network performance. Specifically, our major results include:

- Although the adoption of the path vector by BGP eliminates the DV count-to-infinity problem, the path vector exponentially exacerbates the number of possible routing table oscillations.
- The delay in Internet inter-domain path failovers averaged three minutes during the two years of our study, and some percentage of failovers triggered routing table oscillations lasting up to fifteen minutes.
- The theoretical upper bound on the number of computational states explored during BGP convergence is $O(n!)$, where n is the number of autonomous systems in the Internet. We note that this is a theoretical upper bound on BGP convergence and is unlikely to occur in practice.
- If we assume bounded delay on BGP message propagation, then the lower bound on BGP convergence is $\Omega((n-3)*30)$ seconds, where n is the number of autonomous systems in the Internet.
- The delay of inter-domain route convergence is due almost entirely to the unforeseen interaction of protocol timers with specific router vendor implementation decisions.
- Internet path failover has significant deleterious impact on end-to-end performance – measured packet loss grows by a factor of 30 and latency by a factor of four during path restoral.
- Minor changes to current vendor BGP implementations would, if deployed, reduce the lower bound on inter-domain convergence time complexity from $\Omega((n-3)*30)$ to $\Omega(30)$ seconds, where n is the number of autonomous systems in the Internet.

The remainder of this paper is organized as follows: Section 2 provides additional background on BGP. Section 3 provides a description of our experimental measurement infrastructure. In Section 4, we present the results of our two year study of Internet routing convergence. We describe the measured convergence latencies of both individual ISPs and the Internet as a whole after several categories of injected routing faults. In Section 5, we present a simplified model of delayed BGP convergence and discuss the theoretical upper and lower bounds on the process. In Section 6, we provide analysis of our experimental data based on our model of BGP convergence. Finally, we conclude with a discussion of specific modifications to vendor BGP implementations which, if deployed, would significantly improve Internet convergence latencies.

2. BACKGROUND

Autonomous systems (ASes) in the Internet today exchange inter-domain routing information through BGP. We assume that the reader is familiar with Internet architecture and the BGP routing concepts discussed in [21, 10]. We provide a brief review of the more salient attributes of BGP related to the discussion in this paper.

Unlike interior gateway protocols, which periodically flood an intra-domain network with all known topological information, BGP is an incremental protocol that sends update information only upon changes in network topology or routing policy. Routing information shared among BGP speaking peers has two forms - announcements and withdrawals. A route announcement indicates that a router has either learned of a new network attachment or has made a policy decision to prefer another route to a network destination. Route withdrawals are sent when a router makes a new local decision that a network is no longer reachable via any path. Explicit withdrawals are those associated with a withdrawal message. Implicit withdrawals occur when an existing route is replaced by an announcement of a new, more preferred route without an intervening withdrawal message. We define route *failover* as the implicit withdrawal and replacement of a route with one having a different ASPath. For purposes of our discussion, we define a *steady-state network* as one where no BGP monitored peer sends updates for a given prefix for 30 minutes or more. We choose the 30 minute time period as an upper bound on short-term routing table oscillations based on results described in [16].

BGP limits the distribution of a router's reachability information to its peer, or neighbor routers. As a path vector protocol, BGP updates include an ASPath, or a sequence of intermediate autonomous systems between source and destination routers that form the directed path for the route. The default BGP behavior uses the ASPath for both loop detection and policy decisions. Upon receipt of a BGP update, each router evaluates the path vector and invalidates any route which includes the router's own AS number in the path.

Although not specified in the BGP standard [21], most vendor implementations ultimately default to the best path selection based on ASPath length. The number of ASes in the path is used in a manner similar to the metric count attribute in the RIP protocol. While BGP allows for path selection based on policy attributes, including local preference and multi-exit discriminator values, a review of BGP logs, discussions with Internet network operators, and a survey of policies registered in the Internet Routing Registry (IRR) indicates that the majority of ISP policies default to the selection of the route with the shortest path. In the remainder of this paper, we base our analysis on the default behavior of BGP, or constrained shortest path first policies.

Internet providers commonly constrain path selection and subsequent advertisements to peers through the use of ingress and egress filtering. Most commercial routers include configurable filter-lists which support the rejection or acceptance of route advertisements based on prefix or ASPath pattern matching. Common provider filtering practices include the rejection of customer route advertisements outside the address space owned by that customer, and the filtering of non-customer and non-transit route advertisements to peers.

The BGP standard also includes a minimum route advertisement interval timer, abbreviated in this paper as *MinRouteAdver*, which specifies a minimum amount of time that must elapse between advertisement of routes to a particular destination from a given BGP peer. This timer provides both a rate limiter on BGP updates as well as a window in which BGP updates with common attributes may be bundled into a single update for greater protocol efficiency. In order to achieve a minimum of MinRouteAdver between announcements, the specification calls for this rate-limiter to be applied as a jittered interval on a (prefix destination, peer) tuple basis.

The standard further specifies that MinRouteAdver only applies to BGP announcements and not explicit withdrawals. This distinction stems from the goal of avoiding the long-lived "black holing" of traffic to unreachable destinations. Due to the delay introduced by MinRouteAdver on announcements throughout the Internet, BGP withdrawals are commonly (and incorrectly) believed to propagate and converge more quickly.

3. METHODOLOGY

We base our analysis on data collected from the experimental instrumentation of key portions of the Internet infrastructure. Over the course of two years, we injected over 250,000 routing faults into geographically and topologically diverse peering sessions with five major commercial Internet service providers. We then measured the impact of these faults through both end-to-end measurements and logging ISP backbone routing table changes.

Figure 1 shows a simplified diagram of our RouteViews measurement and fault injection infrastructure. We measured the impact of injected faults via both active and passive probe machines deployed at major US exchange points, as well as on the University of Michigan campus. Our passive instrumentation included several "RouteViews" probe machines, which maintained default-free peering with over 25 Internet providers. These RouteViews machines timestamped and logged all BGP updates received from peers to disk.

We injected faults consisting of BGP update messages including route transitions (i.e. announcements and withdraws) for both /19 and /24 prefix-length addresses. Al-

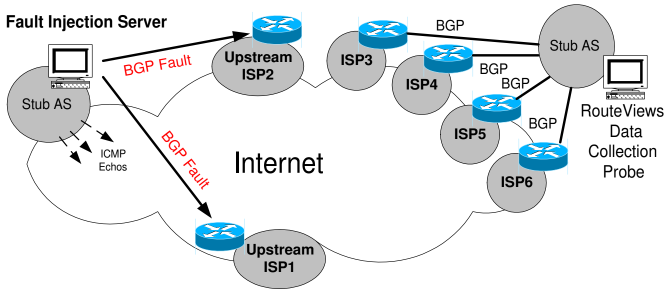


Figure 1: Diagram of the fault injection and measurement infrastructure.

though we injected faults from a number of diverse probe locations, we simplify the discussion in this paper by presenting data only from faults injected at the Mae-West exchange point and from the University of Michigan campus. We note that data from other probe locations exhibited similar behaviors. As we only injected routing information for addresses assigned to our research effort, these faults did not impact routing for commodity ISP traffic with the exception of the addition of some minimal level of extra routing control traffic. We generated faults over a two year period to provide statistical guarantees that our analysis was based on deliberately injected faults rather than normally occurring exogenous Internet failures, which the authors in [16] found occur on the average of once a month.

Software from the MRT and IPMA projects [1, 2] running on both FreeBSD PCs and Sun Microsystems workstations was used to generate BGP routing update messages at random intervals of roughly a two-hour periodicity. The faults simulated route failures, repairs and multi-homed failover. In the case of failover, we announced both a primary route for a given prefix with a short ASPath to one upstream BGP neighbor, and a longer ASPath route for the same prefix to a second provider. The announcement of two routes of different ASPath length represents a common method of customer multihoming to two Internet providers. In an effort to ensure that the downstream peers would always prefer the primary route if it existed, we prepended the long ASPath route announcement with three times the average number of AS numbers observed in steady-state path lengths. We then periodically failed the shorter ASPath route while maintaining the longer backup path.

While the RouteViews probes monitored the impact of BGP faults on core Internet routers, our active measurements monitored the impact on end-to-end performance. We configured these probe machines with a virtual interface addressed within the prefix blocks included in the injected BGP faults. These probe machines sent 512 byte ICMP echo messages to 100 randomly selected web sites once a second. We randomly selected the web site IP addresses from a major Internet cache log of several hundred thousand entries.

We then correlated the data between our NTP synchronized fault injection probe machines and both our RouteViews and end-to-end measurement logs. These correlations provided data on the number of update messages generated for a particular route announcement and withdrawal, as well as the convergence delay for a particular ISP, and all ISPs to reach steady state after a fault.

We also simulated routing convergence using software from the MRT project [2]. The MRTd daemon supports the configuration of multiple BGP autonomous systems and associated routing tables within a single workstation process. As a complete routing protocol implementation, the software supports the generation of BGP update packets and the application of arbitrary BGP policies similar to those available on commercial routers. In simulation mode, the daemon exchanges packets internally and does not forward updates to the network. By programmatically introducing delay in message propagation and processing, we were able to simulate both the average and upper bound on BGP convergence for networks of varying degree and topology.

4. EXPERIMENTAL RESULTS

In this section we present data collected with the experimental measurement infrastructure described in the previous section. We first provide a taxonomy for describing the four categories of routing events injected into the Internet during our study:

Tup A previously unavailable route is announced as available. This represents a route repair.

Tdown A previously available route is withdrawn. This represents a route failure.

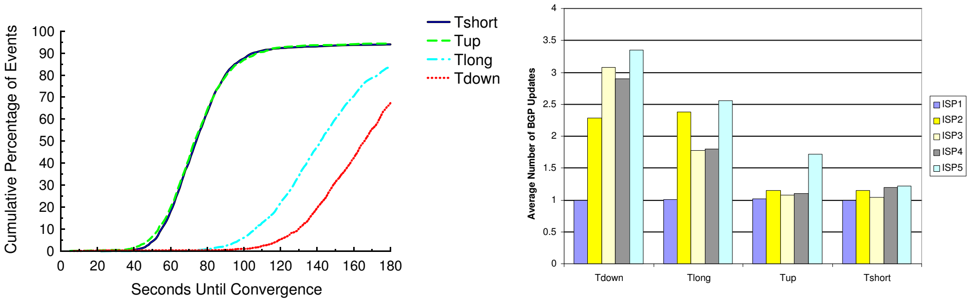
Tshort An active route with a long ASPath is implicitly replaced with a new route possessing a shorter ASPath. This represents both a route repair and failover.

Tlong An active route with a short ASPath is implicitly replaced with a new route possessing a longer ASPath. This represents both a route failure and failover.

We define the *latency* of each injected event as the time between the injection of the fault and the routing tables of a given ISP, or all ISPs, we monitored to reach steady state for the injected prefix. In the following two subsections, we present data from our both our passive routing and active end-to-end measurements.

4.1 Routing Measurements

We first explore the differences in latency among the four categories of routing events. Figure 2(a) shows the convergence latency for a cumulative percentage of Tdown, Tup, Tshort and Tlong events over all monitored ISPs. The horizontal axis represents the number of seconds from injection of the fault until all ISPs' BGP routing tables reach steady state for that prefix; the vertical axis shows the cumulative percentage of all such events. For clarity we limit the horizontal axis to 180 seconds. All four events exhibited a long-tailed distribution of convergence latencies extending up to fifteen minutes for a small, but tangible percentage of events. Significantly, Figure 2(a) shows more than twenty percent of Tlong and forty percent of Tdown events oscillated for more than three minutes. We note that these observed latencies



(a) Latency

(b) Messages

Figure 2: Convergence latency of cumulative percentage of Tup, Tshort, Tlong and Tdown events and average number of BGP updates from 5 ISPs triggered by Tdown, Tlong, Tup, Tshort events for all monitored ISPs over course of our two year study. Both data sets are for faults injected at the Mae-West exchange point.

are an order of magnitude longer than those reported in [3, 13].

We also observe in Figure 2 that (Tlong, Tdown) and (Tshort, Tup) form equivalence classes based on their similar distribution of convergence latencies. Both Tdown and Tlong converged more slowly than Tup or Tshort: Tup and Tshort events converged within 90 seconds while only five percent of Tdown and Tlong events converged within 90 seconds, and twenty percent of Tdown/Tlong required longer than two minutes to converge. We note that the cumulative percentage curves for Tup and Tshort match closely while Tlong and Tdown share similar curves separated by an average of 20 seconds. We posit a likely explanation for both the equivalence classes and the differences between Tlong and Tdown curves in Section 6.

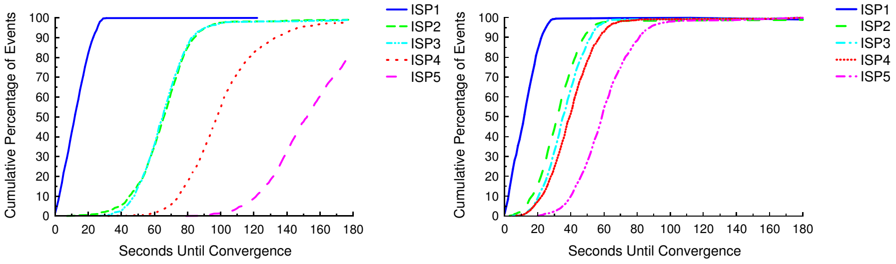
We next examine the volume or number of BGP routing updates triggered by each injection of a routing event. We observe that the injection of a single routing event may trigger the generation of multiple route announcements and withdrawals from each ISP. In Figure 2(b), we show the average number of update messages generated by five ISPs for each category of routing event over the two year course of our study. Although we monitored the BGP routing tables of 25 ISPs, we graph only five ISPs in Figure 2(b) for clarity. We note that data from the other monitored providers exhibited similar behaviors.

The most salient observation we make from Figure 2(b) is that both Tdown and Tlong events on average triggered more than two times the number of update messages than both Tup and Tshort events. As we observed in Figure 2(a), (Tlong, Tdown) and (Tup, Tshort) appear to form equivalence classes with respect to both convergence latency and the number of update messages they trigger. We note significant variation in the average number of updates generated by individual ISPs within each equivalence class. For example, we see that for ISP3, Tdown triggered twice the number of messages as Tlong. In contrast, Tlong events triggered more messages in ISP2 than Tdown. In all categories, ISP1 generated an average of only one BGP update. Finally, we note strong correlations between the relative number of update messages generated per equivalence class in Figure 2(b) and the convergence latencies of each category in Figure 2(a). We provide probable explanations for these behaviors later in Section 6.

We now look at the latency for two categories of injected events on a per ISP basis. Figure 3 shows the convergence latency of a cumulative percentage of both Tdown (a) and Tup (b) events for five ISPs. The horizontal axis represents the delay in one second bins between the time of event injection and the BGP routing tables in each ISP reach steady state for that prefix. The vertical axis shows the cumulative percentage of all such events. As before, we present data from only five ISPs and limit the horizontal axis to 180 seconds for clarity of presentation.

We observe significant variation in the convergence latencies of the five ISPs in both graphs of Figure 3. The variations appear most pronounced in Figure 3(a) where a three-minute gap separates 80% of ISP1 converged events from ISP5. In our analysis, we looked for correlations between the convergence latencies of an ISP and both the geographic and network distance of that ISP. We define *network distance* as the steady-state number of traceroute hops or BGP ASPath entries from the point of fault injection to the peer border router interface of a ISP. In Figure 2 and Figure 3, ISP1 represents a special case – the only ISP into which we both injected events and monitored the convergence latencies. As one of the ISPs into which we also injected faults, the routing table of ISP1 did not exhibit EBGP route oscillations. As we explain in Section 6, at all times ISP1 either had the shortest ASPath route, or ignored updates from neighbor ISPs after detection of an ASPath loop.

With the exception of ISP1, our data shows no correlation between convergence latency and geographic or network distance. For example ISP3, which is a national backbone in Japan, converged more quickly for both Tdown and Tup than a Canadian provider, ISP5. We show in Section 6 that convergence latencies are likely primarily dependent on topological factors including the number of adjacent BGP



(a) Tdown

(b) Tup

Figure 3: Convergence latency of a cumulative percentage of Tdown and Tup events injected at the Mae-West exchange point for five major ISPs

peers and upstream provider transit policies.

We also looked for temporal correlations between convergence delay and the time of day or week. In [17], Labovitz et al. describe a direct relationship between the hourly rate of routing instability and the diurnal bell curve exhibited by Internet bandwidth consumption and the corresponding load on backbone routers. Our analysis, however, found no such temporal relationship with failover latency. This result suggests that the factors contributing to Internet fail-over delay are largely independent of network load and congestion.

4.2 End-to-End Measurements

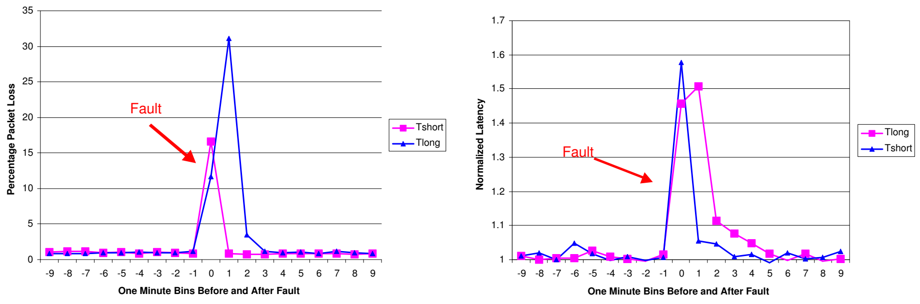
We now turn our attention from the convergence latencies of backbone routing tables to the impact of delayed convergence on end-to-end network paths. We show that even moderate levels of routing table oscillation will lead to increased packet loss, latency and out of order packets. These performance problem arise as routers drop packets for which they do not have a valid next hop, or queue packets while awaiting the completion of forwarding table cache updates. We expect end-to-end active measurements to provide a better measure of the application-level impact of routing convergence as not all routing table changes affect the forwarding path, and external BGP routing table measurements do not the capture delays introduced by the convergence of smaller, stub ISPs or interior routing protocol communication.

In Figure 4(a), we show packet loss averaged over one minute intervals between our fault injection machine and 100 randomly selected web sites. The horizontal axis shows one-minute bins for the ten minutes both proceeding and immediately following the injection of both a Tlong and Tshort failover event. Time 0 is the point of fault injection. The vertical axis represents the percentage loss for each one-minute bin averaged both over all web sites and each corresponding bin in every ten-minute fault injection period. We see in Figure 4(a) less than one percent average packet loss throughout the ten-minute period before each fault. Immediately following the fault, the graphs for Tlong and Tshort events show a sharp rise to 17 and 32 percent loss, respectively, followed by sharply declining loss over the next three minutes. The wider curve of Tlong with respect to Tshort corresponds to the relative speeds of routing table convergence for both events shown in Figure 2. Specifically, Tlong exhibits a two minute period where loss exceeds twenty percent and Tshort a one minute period of greater than fifteen percent loss. These loss trends support the data in Figure 2, where eighty percent of Tlong and Tshort events converged within the same respective periods.

We also examine the impact of convergence on end-to-end path latency. Figure 4(b) shows the average normalized round-trip latency of ICMP echos in ten-minute bins before and after a Tlong and Tshort event. Time 0 represents the instant of fault injection. We normalize the latency of echos on a per destination basis by dividing the latency of each echo by the average delay to that destination. As with the analysis of packet loss, we see that route failover has significant impact on end-to-end latencies. For both Tlong and Tshort, latencies more than tripled in the three minutes immediately following both categories of failover. Although Tshort exhibited an initially higher increase in latency, the curve for Tlong appears broader, extending for five minutes after the event. We note that the variation in end-to-end latency between Tup and Tdown corresponds with routing table convergence data presented in Figure 3.

Finally, we analyze the end-to-end speed of repair, or Tup, by measuring the rate at which ICMP echos first began consistently returning from each web site after a repair. Although we omit the graph of Tup end-to-end behavior for brevity, we note that the majority (over 80%) of web sites began returning ICMP echoes within 30 seconds, and all web sites returned echos within one minute. These results correspond with the routing convergence latencies reported in Figure 3 for Tup events.

We also note that our end-to-end and routing table measurements correspond to observations by other researchers. Delayed convergence provides a likely explanation for both the temporary routing table oscillations observed by Paxson in [19] as well as some of the instabilities observed by Labovitz et al. in [18].



(a) Loss (b) Latency

Figure 4: Average percentage end-to-end loss and normalized latency of 512 byte ICMP echos sent to 100 web sites every second during the ten minutes immediately proceeding and following the injection of a Tshort and Tlong events at the Mae-West exchange point.

5. BGP CONVERGENCE MODEL

In this Section, we present a simplified model of the delayed BGP convergence process. We provide examples and analysis of both the theoretic upper and lower computational bound on BGP convergence. We will use this model later in Section 6 as the basis for our analysis of the BGP convergence behaviors we observed. We base our model on the BGP specification [21], simulation results, and the previously described experimental measurements.

We simplify our analysis by modeling each AS as a single node. In practice, most ASes encompass dozens or even hundreds of border and internal routers. These routers may exchange routing information through a myriad of protocols, including intra-domain BGP communication (IBGP), route reflectors, confederations and interior routing protocols [10]. We exclude the delay and additional states generated by these ancilliary protocols in our model as our experimental results show these do not add significant latency with respect to the overall BGP convergence delays.

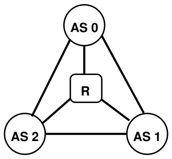
We further simplify our analysis by choosing a full mesh, or complete graph of autonomous systems as our model of the Internet (i.e. each node has $n-1$ adjacencies). In addition, we exclude the impact of ingress and egress filters on BGP route propagation. In practice, the Internet retains some level of hierarchy and most providers implement some degree of customer route filtering. We note, however, that the choice of a full mesh reflects current trends in the evolution of the Internet towards less hierarchy and a more meshed topology[14, 13]. We show in Section 5.1 that a complete graph in the absence of ingress/egress filters provides the worst-case complexity of BGP convergence and, as such, significantly overestimates the average case. Current research, including our ongoing work and [6], has begun to explore the effect of incomplete topologies and more restrictive policies on BGP convergence.

Since BGP does not place bounds on the delay of update propagation or processing, discussions of time complexity are only constructive if we assume bounded delays. We initially exclude the impact of MinRouteAdver and associated timers on convergence. We will discuss time complexity and the impact of these timers in SubSection 5.2. Given the lack of bounds on message propagation, we initially assume messages may arrive in non-deterministic order subject only to the constraint that FIFO ordering is preserved between any pair of autonomous system peers. This unbounded delay model will provide the basis of our calculation for the upper bound on BGP convergence later in this Section. In practice, the link latency and router processing delay for most BGP messages is significantly less than the MinRouteAdver interval.

Finally, we model BGP processing as a single linear, global queue. All messages (both announcements and withdrawals) are placed in a global queue after transmission, and only one set of messages from a single node to each of its peers is processed at a time. We refer to the processing of a single set of messages from a node and the resultant possible state changes and message generation as a *stage*. Such 'serialization' of the BGP algorithm may arise in practice if there are long link delays in a network. In Section 5.2, we extend our taxonomy of BGP convergence to include a set of stages which form a round. We define a *round* as the set of all contiguous stages which process BGP paths at a given length within a single MinRouteAdver timer interval.

In Figure 5, we provide an example of BGP convergence involving a complete graph of a three node system where all nodes are initially directly connected to route R. The "Routing Tables" column shows the routing table of each autonomous system at each computational stage. For each AS, we provide the matrix of current paths through each of its neighbors. We denote the active route with an asterisk and a withdrawn, or invalid path with a dash and/or ∞ symbol. So, for example, we see at step 0 from $1(0R, *R, 2R)$ that $AS1$ has one primary route (directly connected) and two backup paths (via $AS0$ and $AS2$) to R.

The "Message Processing" column in Figure 5 provides the messages processed at each stage. The last "Messages Queued" column shows the global queue of outstanding messages in the system. We process messages in serial fashion from this



Stage	Time	Routing Tables	Messages Processing	Messages Queued in System
0	N/A	steady state 0(*R, 1R, 2R) 1(0R, *R, 2R) 2(0R, 1R, *R)		
1	N/A	R withdraws its route 0(-, *1R, 2R) 1(*0R, -, 2R) 2(*0R, 1R, -)	R -> 0 W R -> 1 W R -> 2 W	0 -> 1 01R 0 -> 2 01R 1 -> 0 10R 1 -> 2 10R 2 -> 0 20R 2 -> 1 20R
2	N/A	1 and 2 receive new announcement from 0 0(-, *1R, 2R) 1(-, -, *2R) 2(01R, *1R, -)	0 -> 1 01R 0 -> 2 01R	1 -> 0 10R 1 -> 2 10R 2 -> 0 20R 2 -> 1 20R 1 -> 0 12R 1 -> 2 12R 2 -> 0 21R 2 -> 1 21R
3	N/A	0 and 2 receive new announcement from 1 0(-, -, *2R) 1(-, -, *2R) 2(*01R, 10R, -)	1 -> 0 10R 1 -> 2 10R	2 -> 0 20R 2 -> 1 20R 1 -> 0 12R 1 -> 2 12R 2 -> 0 21R 2 -> 1 21R 0 -> 1 02R 0 -> 2 02R 2 -> 0 201R 2 -> 1 201R
4	N/A	0 and 1 receive new announcement from 2 0(-, -, -) 1(-, -, *20R) 2(*01R, 10R, -)	2 -> 0 20R 2 -> 1 20R	1 -> 0 12R 1 -> 2 12R 2 -> 0 21R 2 -> 1 21R 0 -> 1 02R 0 -> 2 02R 2 -> 0 201R 2 -> 1 201R 0 -> 1 W 0 -> 2 W 1 -> 0 120R 1 -> 2 120R
5	N/A	0 and 2 receive new announcement from 1 0(-, *12R, -) 1(-, -, *20R) 2(*01R, -, -)	1 -> 0 12R 1 -> 2 12R	2 -> 0 21R 2 -> 1 21R 0 -> 1 02R 0 -> 2 02R 2 -> 0 201R 2 -> 1 201R 0 -> 1 W 0 -> 2 W 1 -> 0 120R 1 -> 2 120R 0 -> 1 012R 0 -> 2 012R
6	N/A	0 and 1 receive new announcement from 2 0(-, *12R, 21R) 1(-, -, -) 2(*01R, -, -)	2 -> 0 21R 2 -> 1 21R	0 -> 1 02R 0 -> 2 02R 2 -> 0 201R 2 -> 1 201R 0 -> 1 W 0 -> 2 W 1 -> 0 120R 1 -> 2 120R 0 -> 1 012R 0 -> 2 012R 1 -> 0 W 1 -> 2 W
(steps omitted)				
48	N/A	steady state 0(-, -, -) 1(-, -, -) 2(-, -, -)		

Figure 5: Example of BGP bouncing problem.

global queue subject only to the constraint that the FIFO ordering of messages is preserved between BGP peers. We use the following notations to represent messages: an announcement of a new path by node i sent to its neighboring node j is given as $i \rightarrow j$ $[path]$ where path is the set of nodes starting with node i. Similarly a withdrawal message originated at node i is represented by $i \rightarrow j$ $[\infty]$. We also represent a withdrawal message, or the absence of a valid path with a W. So, for example, "$0 \rightarrow 1$ $01R$" at stage 1 denotes that $AS0$ has sent a route announcement to $AS1$ with the path $01R$. Similarly, $R \rightarrow 0$ W at stage 1 indicates that R has sent a withdrawal to $AS0$.

As the full example includes over forty stages, we present only the first six stages and the last stage in Figure 5 for clarity. The main goal of the example is to illustrate the exploration of ever increasing ASPath lengths and the generation of large numbers of update messages during convergence. At stage 0, Route R is withdrawn following a fault. All three ASes in stage 1 then invalidate their directly connected paths of length 1, and choose secondary paths: ASO selects $1R$, $AS1$ selects $0R$ and $AS2$ select $0R$. The three ASes also announce these new active routes to each of their neighbors. In the next stages (2 through 4), $AS0$ detects a looped path from $AS1$ and $AS2$, and invalidates both of these routes. Lacking a valid route to R, $AS0$ then sends out withdrawal messages to both neighbors. Upon receipt of this withdraw, $AS1$ and $AS2$ again failover to secondary routes ($AS1$ via $20R$, and $AS2$ via $10R$). In the final stages of the example, $AS1$ and $AS2$ detect the mutual route dependency through each other via the exchange of looped BGP ASPaths. Finally, at stage 48 the system converges with all routes withdrawn.

The intuition behind the large number of messages generated in this example is that adoption of the path vector in BGP exponentially exacerbates the bouncing problem [5]. We note that the loop detection mechanism in BGP resolves the RIP routing table looping problem where a given node reuses information in a new path that the node itself originally initiated. The ASPath mechanism, however, does not prevent an AS from learning of a new, invalid path from a neighbor. For example, in stage 3 of Figure 5 AS2 processes the queued $1 \rightarrow 2$ $10R$ message from $AS1$ and selects this invalid route as a new active path. $AS2$ then appends its own AS number and propagates the new invalid $210R$ path to each of its neighbors.

Intuitively, the most significant difference between the convergence behavior of traditional DV algorithms and BGP is that DVs are strictly increasing, whereas BGP is monotonically increasing. Traditional DVs will explore one, and only one route associated with each distance metric value. In contrast, BGP has $n!$ possible paths in a network of n nodes. We show in the next SubSection that in the worst case, long link/queuing or processing delays can result in an ordering of messages such that BGP will explore all possible paths of all possible lengths. We note that such an ordering represents the upper bound on BGP convergence and is unlikely to occur in practice.

5.1 Upper Bound on Convergence

In this Section, we provide an upper bound on the convergence time for a network of n BGP autonomous systems. As discussed earlier, we initially assume unbounded delay on message propagation. We begin with several observations:

Observation 1: For a complete graph of n nodes, there exist $O((n-1)!)$ distinct paths to reach a particular destination.

To show this, we note that there exists a total of $(n-1)$ paths of length 1 to reach a particular destination in a complete graph. Any other path of length greater than 1 must use one of these $(n-1)$ paths as the last hop in order to reach that destination. For example, there are exactly $(n-1)*(n-2)$ paths of length 2 in a complete graph. Therefore, the sum of all paths can be written as a series sum:

$$P[n] = (n-1) + (n-1)(n-2) + ... + (n-1)!$$

The above expression can be rewritten as :

$$P[n] = (n-1)!\,[1 + 1/2! + 1/3! + ... + 1/(n-2)!]$$

which is closely approximated by $p[n] = O((n-1)!)$. This is an upper bound on the number of all possible paths to any destination in a compete graph of size n.

Observation 2: When a particular route is withdrawn, a path vector algorithm attempts to find an alternate path by iterating on the available paths of equal or increasing length. We refer to this as a k-level iteration of the algorithm. At the k-th iteration, the algorithm looks at paths spanning at most k edges of the graph.

Observation 3: The conditions necessary for the worst case convergence are :

i A complete graph, i.e. all nodes have a degree of $(n-1)$.

ii All messages (both announcements and withdrawals) are processed in sequence i.e. only one message is allowed to be processed at a time. Such 'serialization' of the BGP algorithm may arise in practice if there are long link delays in a network.

iii The messages generated in each k-level iteration are reordered at the beginning of each iteration. Those messages that invalidate the currently installed path at each node are favored and processed ahead of the others.

With these definitions, it is straightforward to construct a sequence of messages between any two nodes i and j for each k-level iteration. Consider the routing table at node i of a network at time t: $(*013, 103, \infty, \infty)$. In this case, node i has two possible paths to the destination via its two neighboring nodes 0 and 1 respectively. Let us assume that node i receives a new announcement from its neighbor, node 1: $1 \rightarrow i[1i3]$. Since this newly announced path creates a routing loop, node i rejects it and also deletes path 103 from its routing table. The only effect of the announcement is the deletion of an alternative path from the routing table. No new update is generated at node i for its neighbors. We consider such announcements a necessity for rapid convergence of a network following the withdrawal of a route since the removal of path 103 prevents it from being propagated during the next k-level ($k = 4$) iteration as a new path $i103$.

On the other hand, suppose that node i receives an announcement from a different neighbor, node 0 (instead of node 1): $0 \rightarrow i[0i3]$. This time, however, path 013 is withdrawn and a new path $i103$ is announced by node i. This leads to more iterations of the shortest path algorithm until every possible path containing $i103$ has been explored.

The above discussion points out an important characteristic of BGP. In the absence of a fixed timer such as MinRouteAdver, the order in which announcements are processed at a node influences the rate of convergence for a path-vector algorithm.

Observation 4: If the conditions in **Observation 3** are applied to all new announcement messages generated at any k-level, the algorithm will continue until all possible paths have been explored. Once the set of all possible paths is exhausted, the algorithm will stop after processing the final withdrawal messages. This is the basis of our conjecture that the complexity for the worst case is $O((n-1)!)$.

Observation 5: The communication complexity, or the number of announcements and withdrawals, are much larger than the bound on the number of states $O((n-1)!)$. Each announcement of a new path is forwarded to all $(n-1)$ neighbors of an AS, thereby generating $(n-1)O((n-1)!)$ messages until convergence. The number of initial withdrawals is $(n-1)$ and in the worst case, the final iteration (i.e. $k = n-1$) generates $(n-1)!$ messages, each of which ends in a withdrawal. Depending on the implementation details of BGP, this may result in $O(n-1)!)$ withdrawals for the worst case. Therefore, for the worst-case BGP model, the number of messages (both withdrawals and announcements) grows faster than exponentially with n.

We present an algorithm that provides an ordering of messages as per condition (iii) (in Observation 3) while preserving the essential features of BGP in the Appendix of [15]. The algorithm forces the path-vector algorithm to explore all $k = 1, 2, ...(n-1) - length$ paths until convergence and results in the worst-case behavior of BGP. As pointed out in a later Section, the best case convergence for BGP can be achieved in $O(n)$ stages. Since the Internet is not a complete graph and the link delays vary widely, the convergence behavior in practice will be in between these two bounds. We describe an artificially severe worst-case algorithm in this Section and [15] to provide a loose upper bound on BGP convergence and demonstrate the vulnerability of the BGP protocol to long or unbounded message delays. We believe our study fills an important gap in the analysis of path-vector algorithms.

5.2 Lower Bound on Convergence

We now examine BGP convergence under the assumption of bounded message delay. Although BGP does not place bounds on message propagation time, operator experience has shown that the vast majority of BGP messages propagate between two peers within several seconds. As noted earlier, the assumption of bounded delay limits the re-ordering of messages that may occur (as demonstrated in Figure 5) and provides a more realistic model of BGP convergence.

Figure 6 provides an example of BGP convergence for a four node full mesh topology. As in the previous example, all nodes are initially directly connected to a route R. At stage 0, Route R is withdrawn and all four nodes fail-over to secondary paths ($AS0$ to $1R$, $AS1$ to $0R$, $AS2$ to $0R$, and $AS3$ to $0R$). Unlike Figure 5, however, this example converges within 13 stages due to the synchronization added by the MinRouteAdver timers. We provide insight into the behavior of MinRouteAdver and its effect on the overall convergence of BGP in the next several observations.

We now show that with the adoption of MinRouteAdver timer, the lower bound on convergence for BGP requires at least $(n-3)$ rounds of the MinRouteAdver timer in a complete graph, where n is the number of autonomous systems. We again refer to the graph of five nodes shown in Figure 6.

Observation 1: The best case algorithm with MinRouteAdver when applied to a complete graph of size n results in complete withdrawal of at most one node at the end of the first round.

The following example illustrates the above observation in the event of a withdrawal of a route R which is initially directly connected to every node in the graph. The initial routing table at each node is represented in stage 0 of Figure 6.

In the event of a withdrawal message from node R, every node in the system, except node 0 will choose the path $0R$ as the active route; node 0 will announce path $1R$. Under the MinRouteAdver timer, node 0 will receive $(n-2)$ announcements from its neighbors and will try to replace its alternate paths (i.e. paths $1R$, $2R$, $3R$ etc.) with the newly received information. However, each of these new updates results in a loop and therefore, node 0 removes all these paths. Node 0 then sends a withdrawal message to all its neighbors, as it no longer has a valid path to R.

Since the direct path of length one from any node, if available, is the best route to reach R, the above sequence of route withdrawal at a single node applies to any complete graph of size n, i.e. one of the nodes will always be withdrawn irrespective of the size of the graph.

Observation 2: The primary effect of a MinRouteAdver timer is to impose a monotonically increasing path metric for successive k-level iterations.

This is the most important contribution of the MinRouteAdver timer and also helps to intuitively explain rapid convergence of general graphs in the event of a route failure. By 'monotonically increasing' paths, we mean that at the end of a MinRouteAdver round, only the next higher level paths (i.e. longer paths) will be announced. Consecutively, under MinRouteAdver, there should be no pending path announcements of length k for a network when a $(k+1)$-length path has already been announced by any node. Under a MinRouteAdver timer, a node must process **all** $(n-1)$ announcements from its neighbors before it can send out a new update. The order in which it processes each announcement does not matter since it receives only one message from each of its neighbor and must wait for the MinRouteAdver timer to expire before announcing a new path. A newly received path from a neighbor may either result in a loop or replace the existing path to that neighbor. If it replaces an existing path, we need to show that the path being replaced is a shorter path than the path replacing it. If this is true for all nodes, each of the nodes will send out a longer path in the next MinRouteAdver timer. This will then ensure that only longer and longer ASPaths will be announced under MinRouteAdver. To see this, let us consider the 4-node example again.

Upon receiving the withdrawals from node R, twelve messages are generated as shown in stage 1 of Figure 6. Let us consider the messages waiting to be processed at node 1. Its routing table currently consists of paths of length two: $1(*0R,\infty,2R,3R)$. However, each of the arriving messages at node 1 replaces the corresponding 2-length path with a 3-length path. As a result, once all $(n-1)$ messages have been processed at node 1 under the MinRouteAdver timer, its routing table now has the following entries: $1(\infty,\infty,*20R,30R)$. A new longer $(k=4)$ path $120R$ is therefore announced to its neighbors in the next iteration at the end of stage 5. Let us contrast this situation with the case when no MinRouteAdver timer is allowed. In this case, node 1 will process **only one** message before it announces a new path. If the particular message $0 \rightarrow [01R]$ was processed (without the MinRouteAdver timer), the routing table at node 1 would become: $1(\infty,\infty,*2R,3R)$ resulting in the same-length path $12R$ to be announced to its neighbors.

The overall convergence of BGP under MinRouteAdver is as follows: as shown above, the very first round of the timer results in announcements of paths of length 2 which cause one of the nodes to delete all paths in its routing table. In the next round, paths of length 3 are announced. These messages will result in a different node being completely withdrawn. The process continues until the longest path (of length $(n-1)$) is announced from each of the remaining nodes, resulting in all nodes being withdrawn. The important observation here is that for a complete graph of size n, an announcement for a path of length k will cause a routing loop at $(k-1)$ nodes in the graph. The role of MinRouteAdver in a complete graph is to ensure that all newly announced paths of length k are processed and loops at $(k-1)$ nodes are detected so that in the next round, only paths of longer path are announced.

By following the routing tables at other nodes in the example graph, one can confirm the same observation as above, i.e. only increasingly longer paths will be announced under the MinRouteAdver timer. Therefore, the effect of the MinRouteAdver timer is to impose a global state synchronization which results in deletion of all k-length paths before a new longer $k+1$ path is announced by any node.

Observation 3: Since $k_{max} = n-1$ and each MinRouteAdver timer deletes paths of length k at the k-th iteration, there will be at least $(n-1)$ MinRouteAdver rounds for the best-case algorithm when applied to a complete graph of size n. (This follows readily from **Observation 2**.)

Observation 4: The above estimate for the number of MinRouteAdver rounds can be further reduced to $(n-3)$ for a complete graph of size n greater than 3. This result follows from the observation that for complete graphs of size $n \leq 3$, BGP converges within a single MinRouteAdver period in the event of a route withdrawal.

We re-emphasize that the above observations are valid when the best-case algorithm with the MinRouteAdver timer is applied to a complete graph. The degree to which MinRouteAdver preserves the monotonicity of each k-level iteration in incomplete graphs is a topic of our current research.

6. ANALYSIS OF RESULTS

Armed with a model of BGP convergence, we now return to the results presented in Section 4. Why do Tup/Tshort converge more quickly than Tdown/Tlong? The explanation lies in the observation that, like the comparison between DV algorithms and BGP, Tup/Tshort are strictly in-

Stage	Time	Routing Tables	Messages Processing	Messages Queued in System
0	N/A	steady state 0(*R, 1R, 2R, 3R) 1(0R, *R, 2R, 3R) 2(0R, 1R, *R, 3R) 3(0R, 1R, 2R, *R)		steady state
1	N/A	R withdraws its route 0(-, *1R, 2R, 3R) 1(*0R, -, 2R, 3R) 2(*0R, 1R, -, 3R) 3(*0R, 1R, 2R, -)	R -> 0 W R -> 3 W R -> 1 W R -> 2 W	0 -> 1 01R 1 -> 0 10R 2 -> 0 20R 3 -> 0 30R 0 -> 2 01R 1 -> 2 10R 2 -> 1 20R 3 -> 1 30R 0 -> 3 01R 1 -> 3 10R 2 -> 3 20R 3 -> 2 30R
2	N/A	announcement from 0 0(-, *1R, 2R, 3R) 1(-, -, *2R, 3R) 2(01R, *1R, -, 3R) 3(01R, *1R, 2R, -)	0 -> 1 01R 0 -> 2 01R 0 -> 3 01R	1 -> 0 10R 2 -> 0 20R 3 -> 0 30R 1 -> 2 10R 2 -> 1 20R 3 -> 1 30R 1 -> 3 10R 2 -> 3 20R 3 -> 2 30R
3	N/A	announcement from 1 0(-, -, *2R, 3R) 1(-, -, *2R, 3R) 2(*01R, 10R, -, 3R) 3(*01R, 10R, 2R, -)	1 -> 0 10R 1 -> 2 10R 1 -> 3 10R	2 -> 0 20R 3 -> 0 30R 2 -> 1 20R 3 -> 1 30R 2 -> 3 20R 3 -> 2 30R
4	N/A	announcement from 2 0(-, -, -, *3R) 1(-, -, 20R, *3R) 2(01R, 10R, -, *3R) 3(*01R, 10R, 20R, -)	2 -> 0 20R 2 -> 1 20R 2 -> 3 20R	3 -> 0 30R 3 -> 1 30R 3 -> 2 30R
Min Route Timer expires 5	30	announcement from 3 0(-, -, -, -) 1(-, -, *20R, 30R) 2(*01R, 10R, -, 30R) 3(*01R, 10R, 20R, -)	3 -> 0 30R 3 -> 1 30R 3 -> 2 30R	0 -> 1 W 1 -> 0 120R 2 -> 0 201R 3 -> 0 301R 0 -> 2 W 1 -> 2 120R 2 -> 1 201R 3 -> 1 301R 0 -> 3 W 1 -> 3 120R 2 -> 3 201R 3 -> 2 301R
6	N/A	withdrawal from 0 0(-, -, -, -) 1(-, -, *20R, 30R) 2(-, *10R, -, 30R) 3(-, *10R, 20R, -)	0 -> 1 W 0 -> 2 W 0 -> 3 W	1 -> 0 120R 2 -> 0 201R 3 -> 0 301R 1 -> 2 120R 2 -> 1 201R 3 -> 1 301R 1 -> 3 120R 2 -> 3 201R 3 -> 2 301R
7	N/A	announcement from 1 0(-, -, -, -) 1(-, -, *20R, 30R) 2(-, -, -, *30R) 3(-, 120R, *20R, -)	1 -> 0 120R 1 -> 2 120R 1 -> 3 120R	2 -> 0 201R 3 -> 0 301R 2 -> 1 201R 3 -> 1 301R 2 -> 3 201R 3 -> 2 301R
8	N/A	announcement from 2 0(-, -, -, -) 1(-, -, -, *30R) 2(-, -, -, *30R) 3(-, 120R, *201R, -)	2 -> 0 201R 2 -> 1 201R 2 -> 3 201R	3 -> 0 301R 3 -> 1 301R 3 -> 2 301R
Min Route Timer expires 9	60	announcement from 3 0(-, -, -, -) 1(-, -, -, -) 2(-, -, -, *301R) 3(-, *120R, 201R, -)	3 -> 0 301R 3 -> 1 301R 3 -> 2 301R	1 -> 0 W 2 -> 0 2301R 3 -> 0 3120R 1 -> 2 W 2 -> 1 2301R 3 -> 1 3120R 1 -> 3 W 2 -> 3 2301R 3 -> 2 3120R
10	N/A	withdrawal from 1 0(-, -, -, -) 1(-, -, -, -) 2(-, -, -, *301R) 3(-, -, *201R, -)	1 -> 0 W 1 -> 2 W 1 -> 3 W	2 -> 0 2301R 3 -> 0 3120R 2 -> 1 2301R 3 -> 1 3120R 2 -> 3 2301R 3 -> 2 3120R
11	N/A	announcement from 2 0(-, -, -, -) 1(-, -, -, -) 2(-, -, -, *301R) 3(-, -, -, -)	2 -> 0 2301R 2 -> 1 2301R 2 -> 3 2301R	3 -> 0 3120R 3 -> 1 3120R 3 -> 2 3120R
Min Route Timer expires 12	90	announcement from 3 0(-, -, -, -) 1(-, -, -, -) 2(-, -, -, -) 3(-, -, -, -)	3 -> 0 3120R 3 -> 1 3120R 3 -> 2 3120R	2 -> 0 W 3 -> 0 W 2 -> 1 W 3 -> 1 W 2 -> 3 W 3 -> 2 W
13	N/A	process withdrawals 0(-, -, -, -) 1(-, -, -, -) 2(-, -, -, -) 3(-, -, -, -)	2 -> 0 W 2 -> 1 W 2 -> 3 W 3 -> 0 W 3 -> 1 W 3 -> 2 W	

Figure 6: Example of BGP bouncing problem with MinRouteAdver.

creasing while Tdown/TLong are monotonically increasing. Intuitively, once a node receives an update during Tup and selects an active path, the node will never choose a route with a longer path. In contrast, since the Tdown implicit metric of infinity is longer than all possible ASPaths, each node will failover to secondary paths until all paths have been eliminated. If we assume bounded delays, then Tup has a computational complexity of $O(1)$ and Tdown of $O(n)$ for a network of n autonomous systems.

Unlike Tup/Tshort, Figure 2(a) shows a slight variation between the relative latencies of Tlong and Tdown. Due to the effects of MinRouteAdver, we might expect Tlong to converge at the same rate or slower than Tdown. Analysis of the data, however, shows that if the prepended ASPath associated with a Tlong is not sufficiently long, then this route might be preferred over shorter paths at some point during convergence. In effect, these Tlongs would resemble both Tshort and Tdown and represent the average of the two. In our experiments, we observed a small number of paths with lengths four times the steady-state average following Tdown and Tlong events. As described in Section 3, we only associated a path of only three times the steady-state average with the injected Tlongs.

Although we did not associate a sufficiently long ASPath with Tlong to render Tshort completely indistinguishable from Tup, or Tdown indistinguishable from Tlong, Tshort/Tup enjoy the property that routing information associated with the shortest ASPath will usually propagate faster than routing information associated with longer paths. This speed advantage arises because in the absence of pre-pending policies which create artificially long paths, ASPaths by definition are formed by routing information traveling through more BGP autonomous systems, each of which adds some additional latency. Although convergence following Tshort theoretically may have introduced added oscillations over Tup as the system explored ASPaths longer than Tlong, such oscillations are unlikely in practice.

In Figure 3, we described significant variations between the convergence latencies of five ISPs. We noted that these differences were independent of both geographic and network distance. As we showed in Section 5.2, if the Internet were truly a complete mesh we would expect all ASes to exhibit the same convergence behaviors. Instead, analysis of the data shows that these variations directly relate to a number of topological factors, including the length and number of possible paths between an AS and a given destination. The number of available paths is a factor of peering relationships, transit policies/agreements and the implementation of filters by both the AS and downstream ASes.

Nodes	Time	States	Messages
4	N/A	12	41
5	N/A	60	306
6	N/A	320	2571
7	N/A	1955	23823

(a) Unbounded

Nodes	Time	States	Messages
4	30	11	26
5	60	26	54
6	90	50	92
7	120	85	140

(b) MinRouteAdver

Nodes	Time	States	Messages
4	30	11	26
5	30	23	54
6	30	39	92
7	30	59	140

(c) Modified

Figure 7: Simulation results for convergence with unbounded delay, MinRouteAdver, and modified MinRouteAdver.

Analysis of Figure 3(a) also shows that the Tdown convergence times of between 0 and 180 seconds directly relate to the number of MinRouteAdver rounds. Our data shows a strong correlation between the average ASPath length during Tdown events and convergence latency. Specifically, as the point of injection ISP1 always announced routes of length one; ISP3 averaged 2.6, and ISP5 averaged ASPaths of length 6. These results corresponds with our $30(n-3)$ lower bound on MinRouteAdver convergence times.

Finally, we examine the 0 to 30 second convergence latencies exhibited in Figure 3(b). As described earlier, Tup events are strictly increasing and do not typically generate multiple announcements. Figure 2 shows that most ISPs average one update message following a Tup event. Since MinRouteAdver does not impact the first announcement of a route, we might expect Tup latencies to be significantly less than 30 seconds, as they would reflect only the network latency and router processing delays along a single path. Discussion with a major router vendor, however, indicates that at least one widely deployed router implements MinRouteAdver on a per peer basis instead of the (destination prefix, peer) tuple. We emphasize that this implementation choice is in accordance with the BGP specification [21] and may improve router memory utilization. A per peer timer, however, introduces some portion of the MinRouteAdver delay to Tup/Tshort updates. If a router has previously sent any update to a given peer within the last 30 seconds, then a new Tup announcement destined for the same peer will also be delayed until the expiration of the per-peer MinRouteAdver timer.

In general, while MinRouteAdver significantly reduces the computational and communication complexity of BGP convergence, the timer also artificially creates multiple thirty-second rounds which delay end-to-end failover in most cases. As we showed in Section 5.2, these rounds form due to the delay in the exchange of path vectors containing mutually dependent routes. Although the BGP specification describes ASPath loop detection, [21] does not specify where the detection should occur. Analysis of our data and discussions with vendors indicates that most commercial routers only perform loop detection upon the receipt of a route update. We distinguish receiver-side loop detection from the route inspection and invalidation performed by a sender before the origination of a looped update.

Figure 6 illustrates the delay introduced by receiver-side only loop detection. At stage 4, $AS0$ and $AS3$ share mutually dependent routes: $AS0$ has an active route via $3R$ and $AS3$ has an active route via $01R$. At the end of stage 4, $AS3$ delays sending the new $01R$ path to all three of its neighbors due to the operation of its MinRouteAdver timer. Only after its MinRouteAdver timer expires, will $AS3$ send the "$AS3 \rightarrow AS0$ $301R$" BGP update message. Upon receipt of this looped path in stage 5, $AS0$ will invalidate the path via $AS3$ and send BGP withdrawals to each of its neighbors. The example encounters a similar mutual dependency between $AS2$ and $AS3$ at the end of stage 8.

We note that if loop detection is performed on both the sender and receiver side, in the best case all mutual dependencies will be discovered and eliminated within a single round. Again returning to Figure 6, we observe that $AS3$ at the end of stage 4 could invalidate the "$AS3 \rightarrow AS0$ $301R$" message and send an explicit withdrawal to $AS0$. Since withdrawals are not impacted by MinRouteAdver according to the standard [21], $AS3$ and $AS0$ would learn of their mutual dependency within a single MinRouteAdver round.

Figure 7(c) provides simulation results of MinRouteAdver modified to perform sender-side loop detection. We note that for all node sizes, modified MinRouteAdver converges within a single thirty second round. We also observe that although the communication complexity remains the same, modified MinRouteAdver exhibits improved state complexity over unmodified MinRouteAdver.

We discussed this proposed modification to MinRouteAdver with a number of router vendors, and at least one indicated that all future versions of a widely deployed router will include both sender and receiver-side ASPath loop detection. The elimination of rounds, however, requires that the router does not apply MinRouteAdver to withdrawals as specified in [21]. At least one major router vendor has made an implementation decision to apply MinRouteAdver to both announcements and withdrawals. A discussion of the motivation and engineering tradeoffs for applying MinRouteAdver to withdrawals is outside the scope of this paper and remains an active area of our current research.

7. CONCLUSION

As the national and economic infrastructure become increasingly dependent on the global Internet, the availability and scalability of IP-based networks will emerge as among the most significant problems facing the continued evolution of the Internet. This paper has argued that the lack of interdomain failover due to delayed BGP routing convergence will potentially become one of the key factors contributing to the "gap" between the needs and expectations of today's data networks. In this paper, we demonstrated that multihomed failover now averages three minutes, and may trigger oscillations lasting as long as fifteen minutes. Further, we showed that these delays will grow linearly with the addition of new autonomous systems to the Internet in the best case, and exponentially in the worst. These results suggest a strong need to reevaluate applications and protocols, including emerging QoS and VoIP standards [11], which assume a stable underlying inter-domain forwarding infrastructure and fast IP path restoral.

This paper also suggested specific changes to vendor BGP implementations which, if deployed, would significantly improve Internet convergence latencies. But even with our suggested changes to ASPath loop detection, BGP path changes will still trigger temporary oscillations and require many seconds longer than the current PSTN restoral times. We can certainly improve BGP convergence through the addition of synchronization, diffusing updates [7] and additional state information [5], but all of these changes to BGP come at the expense of a more complex protocol and increased router overhead. The extraordinary growth and success of the Internet is arguably due to the scalability and simplicity of the underlying protocols. The implications of this trade-off between the scalability of wide-area routing protocols and the growing need for fault-tolerance in the Internet is an active area of our current research.

Acknowledgments

We wish to thank Anish Arora, Randy Bush, Tony Li, Pedro Marques, Susan R. Harris, Bert Rossi, John Scudder and Bob Stovall for their comments and helpful insights. We also thank members of the Internet service provider community for their comments, support and willingness to permit our instrumentation of their networks. Finally, we thank the SIGCOMM 2000 anonymous referees for their feedback and constructive criticism.

8. REFERENCES

[1] Internet Peformance Measurement and Analysis Project (IPMA). http://www.merit.edu/ipma.

[2] Multithreaded Routing Toolkit (MRT) Project. http://www.mrtd.net.

[3] North American Network Operators Group (NANOG). http://www.nanog.org.

[4] K. Bhargavan, D. Obradovic, and C. Gunter. Formal Verification of Distance Vector Routing Protocols. *International Conference on Theorem Proving and Higher Order Logics*, Aug. 2000.

[5] C. Cheng, R. Riley, S. Kumar, and J. G. Aceves. A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect. *Proc. of the ACM SIGCOMM*, pages 224–236, Aug. 1989.

[6] L. Gao and J. Rexford. Stable Internet Routing Without Global Coordination. *Proc. of ACM SIGMETRICS*, June 2000.

[7] J. Garcia-Luna-Aceves. Loop-free Routing Using Diffusing Computations. *IEEE/ACM Transactions on Networking*, Feb. 1993.

[8] T. Griffin and G. Wilfong. An Analysis of BGP Convergence Properties. *Proceedings of the ACM SIGCOMM*, Aug. 1999.

[9] T. Griffin and G. Wilfong. A Safe Path Vector Protocol. *Proc. of IEEE Infocom*, Mar. 2000.

[10] B. Halabi. *Internet Routing Architecture.* Cisco Press, 1997.

[11] D. Hampton, H. Salama, and D. Shah. The IP Telephony Border Gateway Protocol (TBGP), June 1999. draft-ietf-iptel-glp-tbgp-01.txt.

[12] J. Hawkinson. Cisco Routing FAQ. http://www.faqs.org/faqs/cisco-networking-faq.

[13] M. Kaat. Overview of 1999 IAB Network Layer Workshop, Nov. 1999. http://www.ietf.org/internet-drafts/draft-ietf-iab-ntwlyrws-over-01.txt.

[14] C. Labovitz. *Scalability of the Internet Backbone Routing Infrastructure.* PhD thesis, University of Michigan, Aug. 1999.

[15] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. A Study of Delayed BGP Convergence. Technical Report MSR-TR-2000-08, Microsoft Research, Feb. 2000.

[16] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental Study of Internet Stability and Wide-area Network Failures. *Proc. International Symposium on Fault-Tolerant Computing*, June 1999.

[17] C. Labovitz, G. Malan, and F. Jahanian. Internet Routing Instability. *IEEE/ACM Transactions on Networking*, Aug. 1997.

[18] C. Labovitz, G. Malan, and F. Jahanian. Origins of Pathological Internet Routing Instability. *Proc. of the IEEE INFOCOM*, Mar. 1999.

[19] V. Paxson. End-to-End Internet Packet Dynamics. *Proc. of the ACM SIGCOMM*, 1997.

[20] R. Perlman. *Interconnections, Second Edition.* Addison-Wesley, Reading Massachusetts, 1999.

[21] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP4), Sept. 1999. draft-ietf-idr-bgp4-09.txt.

[22] F. Schneider, S. Bellovin, and A. Inouye. Building Trustworthy Systems: Lessons from the PTN and Internet. *Internet Computing*, pages 64–72, Nov. 1999.

[23] K. Varadhan, R. Govindan, and D. Estrin. Persistent Route Oscillations in Inter-Domain Routing. Technical Report USC CS TR 96-631, Department of Computer Science, University of Southern California, Feb. 1996.

[24] W. Zaumen and J. J. G.-L. Aceves. Dynamics of Distributed Shortest-Path Routing Algorithms. *Proc. of the ACM SIGCOMM*, Aug. 1991.