A Study of BGP Path Vector Route Looping Behavior *

Dan Pei
UCLA
peidan@cs.ucla.edu

Xiaoliang Zhao
USC/ISI
xzhao@isi.edu

Dan Massey
USC/ISI
masseyd@isi.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

Abstract

Measurements have shown evidences of inter-domain packet forwarding loops in the Internet, but the exact cause of these loops remains unclear. As one of the efforts in identifying the causes, this paper examines how transient loops can be created at the inter-domain level via BGP, and what are the major factors that contribute to duration of the routing loops. As a path-vector routing protocol, BGP messages list the entire AS path to each destination and the path information enables each node to detect, thus break, arbitrarily long routing loops involving itself. However, delays due to physical constrains and protocol mechanisms slow down routing updates propagation and the routing information inconsistencies among the nodes lead to loop formation during convergence. We show that the duration of transient BGP loops match closely to BGP's routing convergence time and the looping duration is linearly proportional to BGP's Minimum Route Advertisement Interval Timer (MRAI) value. We also examine four BGP routing convergence enhancements and show that two enhancements effective in speeding up routing convergence are also effective in reducing routing loops.

1 Introduction

Measurements [11, 6, 17] have shown that packet forwarding loops exist in the Internet. However due to the scale and complexity of the global routing infrastructure, the exact causes behind routing loops remain unclear. The Internet is composed of thousands of interconnected Autonomous Systems (ASes), also called *domains*. Each domain deploys an intra-domain routing protocol, such as OSPF[10], IS-IS [7], or RIP[9], to compute the internal routes, and BGP serves as the inter-domain routing protocol which exchanges reachability information among the ASes. Routing loops could potentially form due to behaviors in any of these protocols under dynamic topological and policy changes, the interactions between the intra- and inter-domain routing protocols, or even due to subtle implementation details such as the delay between the routing table change and the update to the forwarding table which is used to forward packets. A complete understanding of packet looping in the Internet requires an understanding of all the above components and their interactions. As a first step in identifying the causes of packet looping in the Internet, this paper solely focuses on understanding routing loops in *path vector* routing protocols in general, and in BGP in particular.

Path-vector routing algorithms were designed as an improvement over previous distance vector routing algorithms. One of BGP's primary reasons for adopting a path-vector approach is to eliminate routing loops. BGP routing messages include the entire AS path to each destination and, according to the BGP specification, "This (entire path) information is sufficient to construct a graph of AS connectivity from which routing loops may be pruned" [15, 16]. However, prior to the development of BGP there was no in-depth study on the performance of path-vector routing during topological changes. Our recent network simulation studies show that transient routing loops exist in a network using BGP as the only routing protocol [12], and that in the absence of traffic congestion, packet looping during routing convergence is the primary cause of packet losses. In this paper, we examine how routing loops are created in a path-vector routing protocol such as BGP, and what are the dominant factors that contribute to the duration of the routing loops.

In recent years, there have been several research efforts on improving BGP performance. Most of these studies focused on improving BGP convergence time and reducing message overhead. While these enhancements do not directly address packet looping, we are interested in understanding the impact these convergence enhancements may have on packet looping. Thus in addition to studying the standard BGP, we also analyzed and simulated the following four proposed BGP enhancement mechanisms: Sender Side loop detection [8, 5], Withdrawal Rate Limiting(WRATE) [8, 5], Assertion approach [13], and Ghost Flushing [1]. As an additional benefit, this study provides the first side by side comparison of these proposed BGP convergence improvements.

Our analysis and simulation show that a topology (or

*This work is partially supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No DABT63-00-C-1027, by National Science Foundation (NSF) under Contract No ANI-0221453, and by a research grant from Cisco Systems. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the DARPA, NSF, or Cisco Systems.

policy) change can lead to *inconsistent* routing state among network nodes, and the duration of such inconsistent state is determined by both physical constraints such as message processing time and propagation delay, as well as protocol mechanisms such as BGP's Minimum Route Advertisement Interval (MRAI) timer. During the routing convergence period, transient forwarding loops may occur. Our simulation results show that, in a network running standard BGP, packet looping may persist throughout the routing convergence period, and the majority of the packets sent during this period may encounter loops. For example, in a 110-node Internet-derived topology, BGP experienced a convergence time of 527 seconds and 86% of the packets sent during this time encountered transient loops. Our results also show that both the Assertion and Ghost Flushing approaches are effective in speeding up routing convergence and reducing transient loops, however WRATE enhancement may significantly lengthen transient loop duration compared to the standard BGP without WRATE.

The remainder of the paper is organized as follows. Section 2 reviews the previous studies on routing loops. Section 3 illustrates the loop formation and analyzes the dominant factors that contribute to the duration of looping in path vector algorithms. Section 4 presents simulation results. Section 5 examines the impact of four BGP convergence enhancement mechanisms on routing loops. Finally, Section 6 concludes the paper.

2 Previous Studies on Routing Loops

Paxson [11] analyzed the end-to-end trace-route measurements collected in 1994 and 1995, and detected a few transient loops. The author conjectured that these transient loops were caused by link failures, without pinning down more precisely which component in the global routing infrastructure might have contributed to the loop formation after the link failure.

Transient looping is known to occur in both link state and distance vector routing protocols. Hengartner *et al* [6] illustrated that transient loops can form in link state protocols, and used off-line analysis of packet traces to detect loops on a backbone ISP who runs IS-IS, a link state routing protocol, internally. They observed that forwarding loops were rare, that packets which encountered and escaped a loop were delayed by an additional 25 to 1300 msec, and that 30% of loops on a subset of the links lasted longer than 10 seconds. In addition, more than half of the loops involved only two nodes. Sridharan *et al*[17] used measurement data from the same backbone ISP to correlate packet loops with IS-IS and BGP events. They observed that IS-IS updates were seldom correlated with packet loops, but there was a strong *temporal* correlation between packet loops and BGP updates for the destination prefixes of the looped packets.

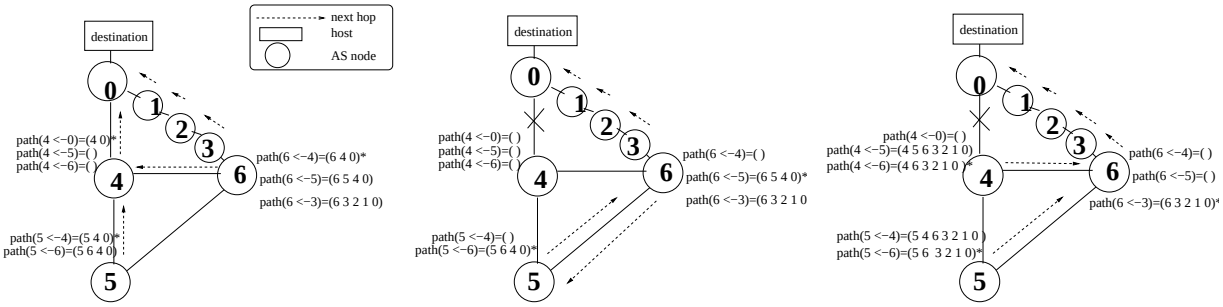
In distance vector routing protocols such as RIP, link failures may lead to counting-to-infinity [7] which results in transient looping. Several mechanisms have been proposed to improve distance vector routing by using *path finding* techniques.The *path finding* algorithms [2, 4] attach the second-to-last hop information to routing update messages, which allows a node to reconstruct the full path to a destination through iterative queries. While path finding algorithms can provide significant improvement over pure distance vector protocols in terms of loop detection, they do not *eliminate* transient loops [4]. True loop freedom can be achieved through inter-nodal coordination algorithm such as the DUAL algorithm [3] used by EIGRP [7]. In this approach, backup paths are pre-calculated. After a link failure, necessary checking is conducted to make sure that the backup paths do not depend on the failed link before they can be used for data forwarding. These algorithms pay an overhead and delay cost for the coordination needed to achieve the loop-freedom; packet flow is stopped while backup paths are being verified. This delay can be non-negligible and may lead to packet drops. [12] argues that overall packet delivery can be improved by quickly selecting (even non-optimal) paths.

BGP is the only path vector routing protocol in use today. Unlike link state and distance vector routing protocols, potential looping behavior of path vector protocols has not been investigated. In BGP each node announces to its neighbors the *full path* to each destination. However previous simulation work in [12] shows that BGP's full path information does not eliminate transient loops. This paper focuses on the understanding of routing loops in *path vector* routing protocols in general, and in BGP in particular.

3 Loop Formation, Resolution and Duration in BGP

BGP uses TCP for reliable update delivery. Each BGP node announces to its neighbors its best *path* to each destination and keeps a copy of the most recent paths received from *each* of its neighbors. The route to each destination is advertised only once; subsequent updates are sent *only* upon route changes. BGP also uses a Minimum Route Advertisement Interval (*MRAI*) timer to space out consecutive updates for the same destination by M seconds (default value 30) with a small jitter interval. When a router notices its current path to a destination, D, is no longer available, it first attempts to find an alternative path to D by looking through all the saved paths it learned from its other neighbors previously. If no alternative path is found, it sends an explicit path withdrawal message to its neighbors. Following the BGP specification in RFC 1771 [15], in this study the MRAI timer is not applied to withdrawal messages, except when we simulate the Withdrawal Rate limiting (WRATE) enhancement mechanism. We note that the latest BGP specification update draft [16] has adopted WRATE as the standard behavior.

For clarity of description, all the examples and simula-



(a) Before the link failure (b) Loop is formed (c) Loop is resolved

Figure 1. Formation of Transient Loops in BGP

tions in this paper assume a shortest-path routing policy, and the smaller node ID is used for tie-breaking between equal length paths. Figure 1(a) shows the BGP routes known by nodes 4, 5 and 6 to reach a destination connected to node 0; the best path to the destination is marked with a star, and the packet forwarding directions (next hops) between nodes are shown in dashed lines. The path information is used to detect potential loops. When node 4 receives the path advertisement of $(6\ 4\ 0)$ from neighbor node 6, the path is discarded because it contains node 4. Similarly, node 4 also discards routes $(5\ 4\ 0)$ or $(5\ 6\ 4\ 0)$ when they are received from node 5. More generally, node v discards any path that includes itself. We call this feature *path-based Poison Reverse*. The poison reverse scheme in distance vector protocols, such as RIP [9], can only detect 2-node routing loops. The *path-based Poison Reverse* allows a node to detect arbitrarily long loops involving itself.

3.1 Loop Formation in BGP

Figures 1(a), 1(b), and 1(c) illustrate how a transient loop can occur in BGP. Assume that all packets are destined to the destination connected to node 0. In Figure 1(a), nodes 5 and 6 forward packets to node 4 and node 4 forwards the packets directly to node 0. When link $[4\ 0]$ fails, node 4 sends withdrawal messages to both node 5 and node 6. Node 5 consults its routing table, finds a new path $(5\ 6\ 4\ 0)$, and starts forwarding data packets to node 6. It will also advertise its new path to its neighbors. Similarly, node 6 chooses a new path $(6\ 5\ 4\ 0)$, starts forwarding packets to node 5 and will also advertise the new path to its neighbors. As a result, data packets start looping between nodes 5 and 6 immediately, as shown in Figure 1(b). Meanwhile, the new route advertisements from both node 5 and node 6 are undergoing message processing delay, propagation delay, and the MRAI timer delay. These delays in routing update exchanges leads to inconsistent routing information between nodes 5 and 6: 5 does not know 6's next hop has changed to 5, and 6 does not know 5's next hop has changed to 6. As soon as node 6 receives node 5's new path $(5\ 6\ 4\ 0)$, it will switch to path $(6\ 3\ 2\ 1\ 0)$, thus breaking the loop, as shown in Figure 1(c). This example illustrates how the transient routing state inconsistency resulted in a simple 2-node loop. In a network as large as the Internet, more complicated inconsistency scenarios can arise and create routing loops of various sizes.

Although temporary path inconsistency is inevitable in a distributed routing protocol due to inherent physical constrains such as processing time and propagation delay, BGP's MRAI timer's impact on delaying routing information exchange is far more significant than all the other factors. [1] shows that the MRAI delay is at least an order of magnitude larger than the normal nodal delay of a routing message. However [5] shows that the MRAI timer is necessary in order to suppress large amount of update messages during BGP convergence. A large network using BGP as the routing protocol inevitably faces route inconsistency and thus transient loops. Note also that while the term MRAI timer is specific to BGP, routing protocols typically have some damping timer similar to the MRAI timer to assure certain minimum delay between updates.

3.2 Loop Resolution and Duration

We have illustrated how delays can result in inconsistent path information. Furthermore, after losing its current path to a destination, a node may explore several paths in sequence, and each of the new paths may be subject to the MRAI timer delay. As a result, after a *single* failure *multiple* routing loops may form. Furthermore, a path update which may help break a loop can be delayed by up to M seconds at a node, and such scenarios have been observed in simulations (e.g. [12]). In this section, we use an analysis example similar to the one used in [4] to show the impact of the MRAI timer on the resolution of a single routing loop.

In Figure 2, at time t node c_1 changes its path from $path(c_1, old)$ to $path(c_1, new)$ with a new next hop c_2, and an m-node loop consisting of node $c_1, c_2, \cdots, c_m$ is formed. Once this loop forms, we will have $nexthop(c_i) = c_{i+1}, 1 \leq i \leq m-1$, and $nexthop(c_m) = c_1$. We define $nexthop(c_i, new)$ as the next hop of c_i at time

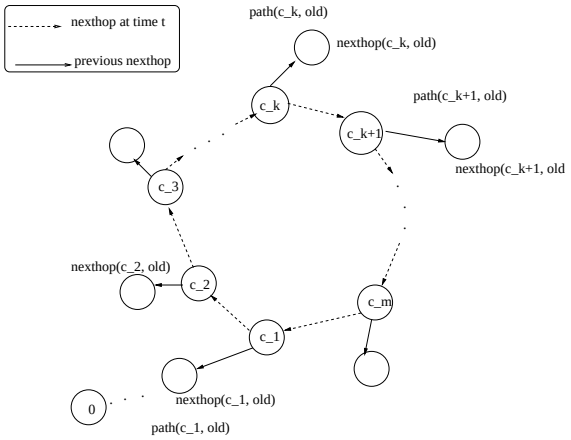


Figure 2. Loop Formation at time t: c_1 chooses c_2 as the new next hop, and the loop consisting of $c_1, c_2, \cdots, c_m$ is formed.

t, and $nexthop(c_i, old)$ as the previous next hop of c_i, and corresponding paths are defined as $path(c_i, new)$, and $path(c_i, old)$. We now consider how this loop can be resolved.

We say that there must exist a $k(2 \leq k \leq m)$ such that $path(c_1, new)$ takes the format of $(c_1\ c_2\ \cdots\ c_k) \cdot path(c_k, old)$, where "$\cdot$" is the concatenation operator for paths. After some delay, c_1 sends $path(c_1, new)$ to all its neighbors, including c_m. If $path(c_1, new)$ is not the best path available to c_m, c_m changes its path, switches to a next hop different from c_1, and the loop is resolved. The loop will also be resolved if $path(c_1, new) = (c_1\ c_2\ \cdots\ c_k) \cdot path(c_k, old)$ happens to include c_m (i.e. $path(c_k, old)$ includes c_m) since then c_m will discard this path and choose another next hop, or declare the destination unreachable.

If the loop is not resolved at c_m, c_m's new best path becomes $(c_m) \cdot path(c_1, new)$ and c_m will propagate this new path to c_{m-1}, and so on. In the worst case, nodes $c_{k+1}, ..., c_m$ all use $path(c_1, new)$, and c_{k+1}'s new path will be $(c_{k+1}\ \cdots\ c_m) \cdot path(c_1, new) = (c_{k+1}\ \cdots\ c_m\ c_1\ c_2\ \cdots\ c_k) \cdot path(c_k, old)$. Eventually when this new path is propagated to node c_k, the loop is resolved. During this process, a routing message has to travel $m - k + 1$ hops and can be delayed at each hop by up to M seconds due to the MRAI timer. In the worst case, $k = 2$ and the resolution of an m-node loop can take up to $(m - 1) \times M$ seconds.

Note that in the worst case, the loop will not be resolved until $path(c_1, new)$ has propagated counterclockwise through the loop shown in Figure 2. However, the loop can also be resolved sooner because of messages triggered by other nodes. For example, before $path(c_1, new)$ is propagated to c_{k+1}, c_{k+1} might send a new path to node c_k, causing c_k to change to a new next hop other than node c_{k+1}, and the loop is resolved. Nevertheless, at least one message must be sent out by one of the nodes in the loop in order to resolve the loop, and each message can be delayed by the MRAI timer.

Also note that resolution of the loop consisting of $c_1, c_2, \cdots, c_m$ shown in Figure 2 could result in another (but different) loop. This new loop might consist of some of the nodes in $c_1, c_2, \cdots, c_m$, and these overlapping nodes might be involved in looping for longer durations than that of a single loop. In any case, we emphasize that the MRAI timer delays the propagation of information needed for loop resolution, and in the worst case a single m-node loop's duration can be as long as $(m - 1) \times M$ seconds.

3.3 Remarks on Loop Detection and Prevention

We have shown that the *path-based Poison Reverse* enabled by BGP allows node v to discard any $path(u)$ of arbitrary length from neighbor u if $path(u)$ contains v. However, BGP's full path information does not prevent packet loops from *occurring*. Figure 2 has shown that a node c_1 can pick a backup path which does not include itself, such as $(c_2\ \cdots\ c_k\ ) \cdot path(c_k, old)$, even when the validity of that path has been obsoleted by the latest topology change. Selecting such an obsolete path can thus lead to transient loops as shown in Figure 2. Furthermore, after a failure it takes time for a node to receive neighbors' new path information and detect whether a loop has been created. Meanwhile, packets may already be in the forwarding loops. For example, in Figure 2, when c_k detects the loop after receiving $(c_{k+1}\ \cdots\ c_m\ c_1\ \cdots\ c_k) \cdot path(c_k, old)$, this loop may have lasted for up to $(m - k + 1) \times M$ seconds and a large number of packets may have been forwarded around the loop during this time.

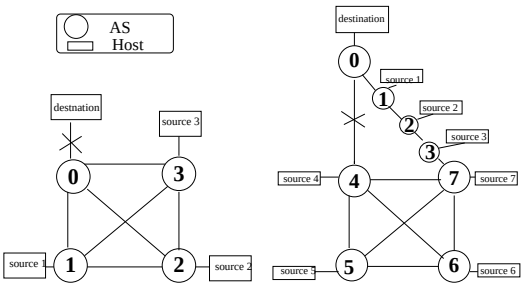
Existing loop prevention algorithms, such as the DUAL algorithm [3], *avoid* using any previously obtained information after a failure until the information is verified. However, the verification step delays the use of any backup path, causing all incoming packet being dropped in the meanwhile. We are exploring new directions for solutions that minimize both looping and packet losses.

4 Simulation Results For Standard BGP

In this section, we use simulation to further explore BGP's transient looping behavior.

4.1 Simulator Topologies and Settings

We use the SSFNET [18] simulator to measure both BGP looping and data delivery. In each topology, we choose one AS to contain a destination host, and every other AS has one host that sends a constant rate IP packet stream to the destination. We then inject a topology change event to trigger BGP routing adaptation. In a T_{down} event, the destination AS becomes unreachable from the rest of the network. Although the destination is unreachable, packets sent and trapped in the transient loops can continue to consume network resources during T_{down} convergence



(a) CLIQUE of size 4 (b) B-CLIQUE of size 4

Figure 3. CLIQUE, and B-CLIQUE Topologies

period. In a T_{long} event, a link in the network fails, which does not disconnect the destination AS but forced the rest of the network to use less preferred paths to reach the destination. Looping during T_{long} convergence can introduce losses and long delays to packets. In all our simulations, the $MRAI$ timer is implemented on a per (destination, neighbor) pair base, and its value is configured to be 30 seconds with a random jitter, unless specified otherwise.

Clique, B-Clique and Internet-derived network topologies were used in our simulations. Clique (full-mesh) topologies, shown in Figure 3(a), are frequently used in literature [8, 5, 1] as a simple basis for analysis and comparison for T_{down} convergence. A B-Clique topology of size n, shown in Figure 3(b), consists of $2n$ nodes. Nodes $0, \cdots, n-1$ constitute a chain topology of size n, and nodes $n, \cdots, 2n-1$ constitute a Clique topology of size n. Node 0 is connected to node n, and node $n-1$ is connected to node $2n-1$. This topology is used to model an edge network (node 0) that has a direct link and a long backup path (the chain) to the well-connected Internet core (a Clique topology). In our simulation, AS 0 is chosen as the destination AS and the link between AS 0 and n is failed during simulation to induce a T_{long} event. To represent an Internet topology, we used 29-node and 110-node topologies which were derived from actual Internet routing tables as described in [14][1]. Following the same algorithm in [14], we also generated two more Internet-like topologies with 75 nodes and 48 nodes, respectively. In the Internet topologies, the destination AS was randomly chosen among the nodes with the lowest degrees and in T_{long} one of its link is randomly chosen to fail. The simulation were repeated for a number of times with different destination ASes and failed links.

[1]Due to memory requirements, SSFNET can support relatively small topologies, and topologies generated by power-law generators are not suitable for small topologies [19].

4.2 Simulation Metrics

We use *TTL exhaustion* (i.e. a packet's TTL is decrementing to zero) as a simple indication of routing loops. The TTL is set such that if a packet is dropped due to TTL exhaustion, there must exist a routing loop in the network around the time when this packet is dropped. However if the network convergence time is very short, a packet involved in a loop might escape from the loop before it is dropped due to TTL exhaustion, hence we may see no TTL exhaustion even though loops exist. To help counter this, we set the following parameters to insure that some packets involved in a loop will be caught by TTL exhaustion.

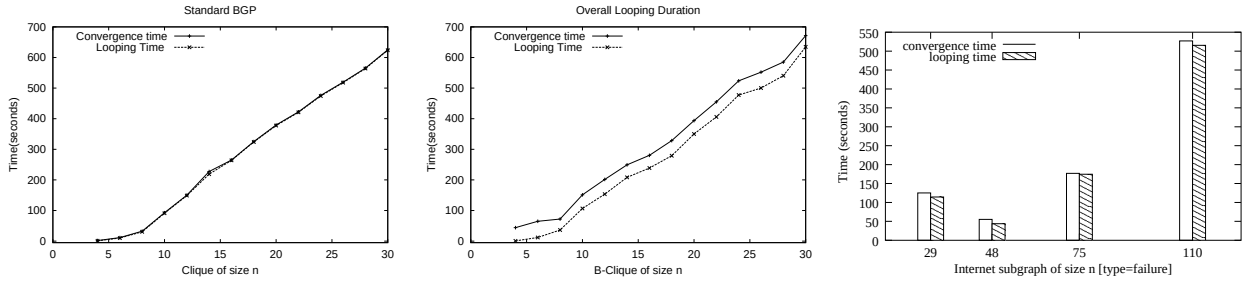
We set the link delay to 2 milliseconds and the routing message processing delay (uniformly distributed between 0.1 second and 0.5 second) to be two orders of magnitude larger than the link propagation delay. With an initial TTL value of 128, a data packet will have a lifetime of $128 \times 2ms = 256ms$ before TTL exhaustion. In some sense, the impact of message processing delay on routing loops is emphasized in our study and the impact of propagation delay on routing loops is negligible. We intentionally set a slow data packet rate of 10 packets per second to avoid congestion and make packet queueing delay negligible. The 100ms inter-packet time also assures that at least some packets are sent during the transient loops that last longer than 256ms.

Note that the simulated TTL is decremented by one at each node, which represents an AS in our simulation. In the Internet, a single AS normally contain many routers and a packet traversing an AS may have its TTL decremented by a large value (or an intra-AS loop could even decrement the TTL to zero). However because our main purpose is to use TTL exhaustion as an indication of the existence of BGP routing loops at the inter-domain level, this inaccuracy in TTL decrement should not affect our simulation results.

We measure the BGP routing loops by the following metrics. First, *Overall Looping Duration* starts when the first TTL exhaustion occurs and ends when the last TTL exhaustion occurs. This is a rather coarse estimate on loop duration, and it only reflects the length of the period during which loops existed. Second, *Convergence Time* starts when the link failure happens, and ends when the last BGP update message is sent. Third, we count the *Number of TTL Exhaustions*, which reflects the aggregated effect of the frequency and duration of individual loops. Finally, to better compare the network with different numbers of source hosts and MRAI timer settings, we also measure *Looping Ratio*, the ratio of "Number of TTL exhaustions" to "Number of packets sent during convergence time". This metric can be considered as the the probability that a packet sent during routing convergence encounters looping.

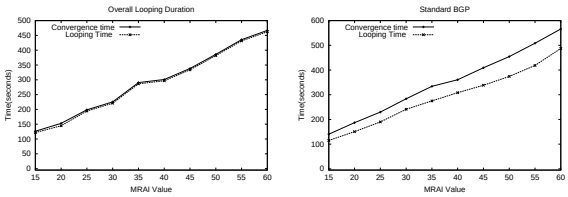
4.3 Simulation Results

Figure 4 shows the comparison of overall looping duration and convergence time with various network topolo-



(a) T_{down} in Clique topologies (b) T_{long} in B-Clique topologies (c) T_{down} in Internet-Derived topologies

Figure 4. Overall Looping Duration and Convergence Time with various network sizes



(a) T_{down} in Clique (b) T_{long} in B-Clique

Figure 5. Overall Looping Duration and Convergence Time with various MRAI timer values

gies. Figure 5 shows this comparison with various MRAI values. In Figure 4(a) and 4(c), T_{down}'s overall looping duration is only a few seconds shorter than the convergence time, showing that looping happens through the T_{down} convergence. Figure 4(b) shows that the overall looping duration in T_{long} is typically 30 to 45 seconds shorter than the convergence time[2].

These results demonstrate that looping indeed happens during BGP convergence, and even worse, it happens almost throughout the convergence period.

Observation 1 *The overall looping duration is closely coupled with the convergence time and the overall looping duration is linearly proportional to the MRAI value.*

As we discussed in Section 3, MRAI timer is the major theoretical factor contributing to the loop duration, and an m-node loop can last for $(m-1) \times M$ seconds. Figure 5(a) shows that T_{down} *convergence time* in Clique is linearly proportional to the MRAI value, confirming the results from [5][3]. Furthermore, our results in Figure 5(b) shows that the convergence time of B-Clique is also linearly proportional to the MRAI value. Given that the convergence time and the overall looping duration are closely coupled, it is not surprising that the overall looping duration, also shown in Figures 5(a) and 5(b), is also linearly proportional to the MRAI timer value.

Figure 6 shows the number of TTL exhaustions (left Y-axis) and looping ratio (right Y-axis) using various network sizes. Figure 7 shows the results using various MRAI values.

Observation 2 *In T_{down} with Clique topologies and T_{long} with B-Clique topologies, the number of TTL exhaustions is linearly proportional to the MRAI timer value, while the packet looping ratio stays almost constant.*

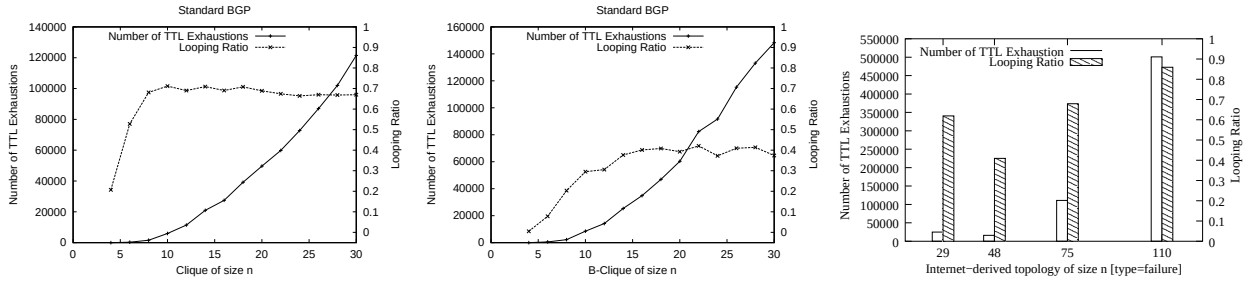
In Section 3, we argued that the MRAI timer is the major contributing factor of the duration of each individual loop. Since packet generation rate is constant in our simulation, the number of TTL exhaustions caused by one specific loop is determined by the duration of the loop, which, in turn, is linearly proportional to the MRAI value. Therefore, it is not surprising that the aggregation of TTL exhaustions for all the individual loops during the convergence, as shown in Figure 7, is also linearly proportional to the MRAI value.

The looping ratio is more than 65% for T_{down} in Clique of size 15 or larger and more than 35% for T_{long} in B-Clique of size 15 or larger.[4] These ratios are lower than ratio of "overall looping duration" to "convergence time" shown in Figures 4 and 5. This shows that not every node is involved in a loop at a given time and there is not always a loop during the "overall looping duration". On the

[2]The final update sent in T_{down} is a withdrawal, which is not delayed by MRAI timer. In T_{long}, on the other hand, even after a loop is resolved, a node may not be able to send out its best path immediately due to MRAI timer, which delays the convergence time(as measured by the time the last message is sent). However in either case this final update does not trigger any further routing changes, indicating that the routing state at all the nodes are already consistent, i.e. loop free, at the time of this last update message. This explains why gap between convergence and looping time is different for T_{down} and T_{long}.

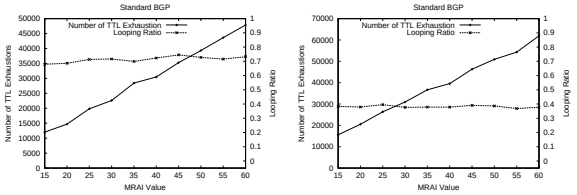
[3][5] shows that this property holds only when the MRAI value is larger than a topology-specific optimal value, which is a value large enough for a node to process the messages received from all the neighbors.

[4]Note that in T_{long} convergence not every node in the network will be affected. For example, in B-Clique topology, nodes $2, \cdots, n/2$ are not affected by link failure $[n\ 0]$, therefore the packets sent by them will not encounter a loop.



(a) T_{down} in Clique topologies (b) T_{long} in B-Clique topologies (c) T_{down} in Internet-derived topologies

Figure 6. Number of TTL Exhaustions and Looping Ratio with Various Network Topologies



(a) T_{down} in Clique (b) T_{long} in B-Clique

Figure 7. Number of TTL exhaustions and looping ratio with various MRAI values

other hand, the looping ratio does reflect the probability of looping (e.g. 65% in T_{down} in Clique topologies) when a packet is sent to network during the convergence.

We observed almost constant looping ratio when varying MRAI values in Clique and B-Clique topologies and this observation warrants more detailed discussion. As we discussed before, the MRAI timer is the dominant factor contributing to the duration of each individual loop. However, varying MRAI timer itself introduces little randomness and does not change the patterns and frequency of the loops formed during convergence. The major impact of varying MRAI timer value is simply a change in the duration of each individual loop. On the other hand, we have shown that the convergence time is linearly proportional to the MRAI timer value. Given the constant packet rate, and duration of the each individual loop is linearly proportional to the MRAI timer value, the looping ratio (defined as the number of TTL exhaustions divided by the number of packets sent during convergence) remains constant. In other words, the constant looping ratio is a result of the dominance of the MRAI timer on duration of individual loops.

5 Convergence Enhancement Mechanisms

In Section 4, we observed that the overall looping duration in BGP is closely coupled with the routing convergence time. To further understand the relation between routing loops and convergence improvement, we simulated the following four BGP convergence enhancements: Sender Side loop detection (SSLD) [8, 5], Withdrawal Rate Limiting (WRATE) [8, 5], Assertion approach [13], and Ghost Flushing[1]. SSLD and WRATE are built-in in the SSFNET simulator, and we implemented Assertion approach and Ghost Flushing in SSFNET according to the description in [13] and [1], respectively. To the best of our knowledge, our work provides the first study on the impact of these enhancement mechanisms on transient routing loops. This work is also the first comparative simulation study of these convergence enhancements. Figure 8 shows the results for T_{down} events, and Figure 9 shows the results for T_{long} events.

Observation 3 *Both Assertion and Ghost Flushing are effective in speeding up route convergence and reducing transient loops, while SSLD and WRATE are not.*

- *Assertion approach consistently improves convergence time and reduces looping, but the magnitude of the improvement depends on the details of topology. In Clique and B-Clique topologies, Assertion approach is most effective among the four enhancement mechanisms in reducing both packet looping and the convergence time, however the improvements are much less pronounced in Internet-derived topologies.*

- *Ghost Flushing consistently reduces both BGP convergence time and looping (by 80% in large topologies), and gives the best results among the four enhancement mechanisms for Internet-derived topologies. However its improvement is reduced in large size Clique and B-Clique Topologies due to the high overhead of flushing withdrawal messages after each failure.*

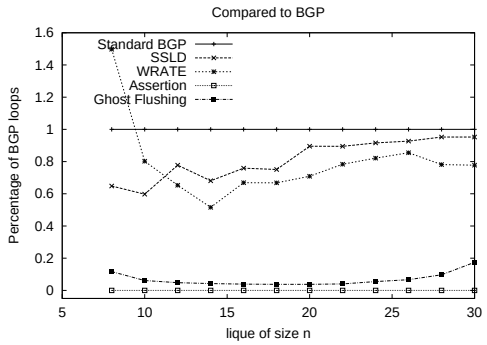
- *SSLD can also reduce BGP convergence delay and looping losses, but only by a modest amount.*

- *WRATE reduces packet looping in Clique and B-Clique topologies, however for Internet-derived topologies, it* increases *packet looping by at least 20% in T_{down} and by an order of magnitude in T_{long}. It also slightly* increases *the T_{long} convergence time in B-Clique topologies.*

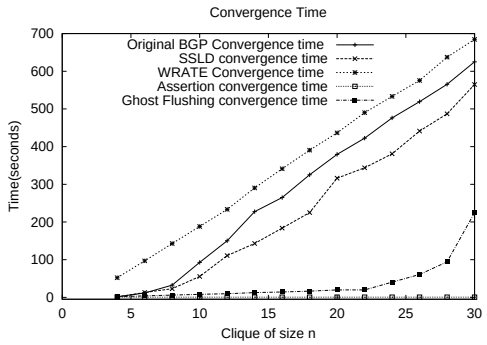
The *Assertion* approach proposed in [13] removes inconsistent routes by fully utilizing locally available information. For example in Figure 1(b), when node 5 receives a withdrawal message from node 4, it will also remove the backup path $(5\ 6\ 4\ 0)$ since the path goes through node 4. More generally, when node v receives a path $path(u, new)$ from neighbor u , v removes any backup paths that include u and contain a sub-path different from $path(u, new)$. In the Clique topologies, all other nodes are directly connected to node 0, and thus can achieve immediate convergence after receiving the withdrawal from node 0. However, in the Internet-derived topologies, it is unlikely that all the other nodes are directly connected to the origin AS, thus the assertion checking is less likely to detect obsolete paths than in the case of Clique topologies (similar in T_{long} convergence). Overall, Assertion provides each node increased ability to detect obsolete paths, hence reducing the chance of packet looping, with an effectiveness degree depending on topological properties such as the degree of origin AS or the AS closest to the failure.

Ghost Flushing[1] requires that a node immediately send a withdrawal when the node changes to a longer path when the new path announcement is delayed by the MRAI timer. "Withdrawal flushing" can quickly flush out obsolete path information, such as $path(c_k, old)$ or $path(c_1, old)$ in Figure 2. A quick flush of $path(c_k, old)$ reduces the chance that node c_1 would use $path(c_k, old)$, and flushing $path(c_1, old)$ immediately resolves the loop. Only message processing and propagation delay affect the resolution and duration of the loops. In our simulation the message processing time is set relatively large (0.1 to 0.5 seconds), therefore in the cases of Clique and B-Clique of size 26 and higher the message containing the latest path information is delayed by the processing of a large number of withdrawal flushes[5]. Ghost Flushing reduces convergence time and looping duration for T_{down} in both Clique and Internet-derived topologies. It also reduces the convergence and looping duration for T_{long} in B-Clique and Internet-derived topologies, although the convergence time reduction in the latter case is less dramatic (since standard BGP's convergence time is already below 65 seconds). Overall, Ghost Flushing reduces packet looping by at least 80% in Clique topologies and Internet-derived topologies for both T_{down} and T_{long}.

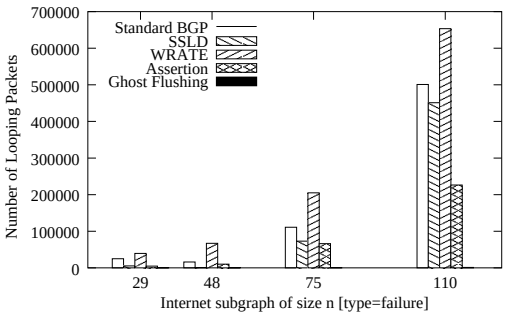
[5]The *exact* turning point depends on the message processing time. Our large processing time setting helps showing the trend of Ghost Flushing's performance as the node degree increases.



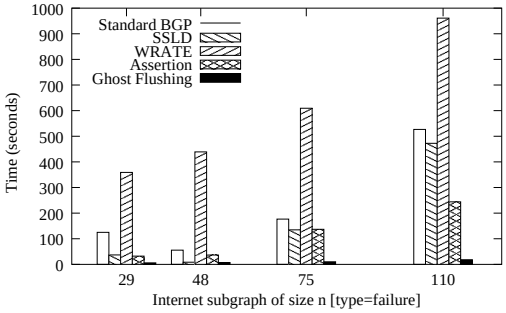
(a) TTL Exhaustion Normalized by Standard BGP in Clique



(b) Convergence Time in Clique

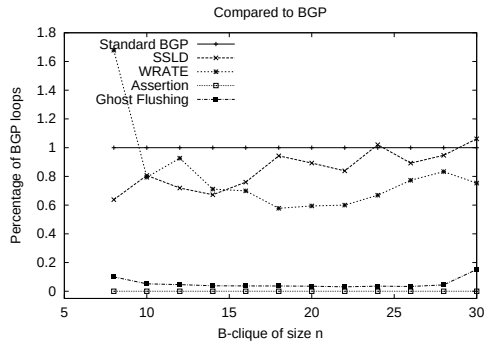


(c) TTL Exhaustion in Internet-derived Topologies

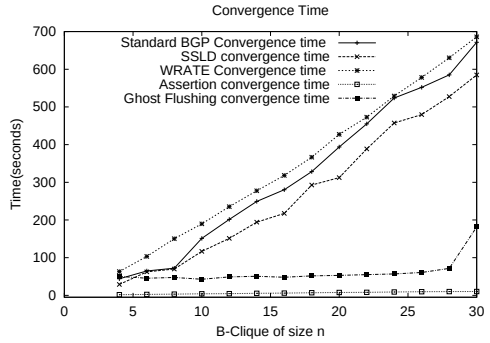


(d) Convergence Time in Internet-derived Topologies

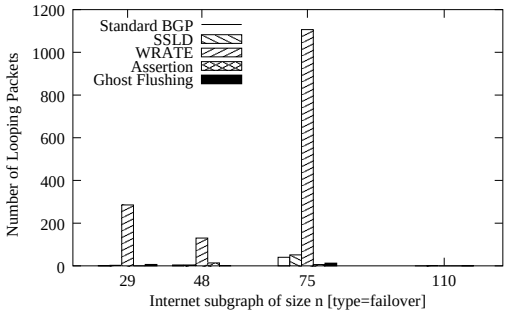
Figure 8. T_{down} in Clique and Internet-derived Topologies



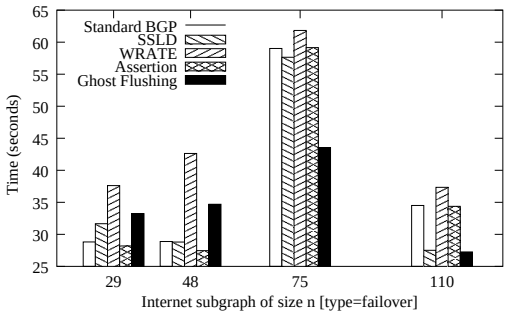
(a) TTL Exhaustion Normalized by Standard BGP in B-Clique Topologies



(b) Convergence Time in B-Clique Topologies



(c) TTL Exhaustion in Internet-derived Topologies



(d) Convergence Time in Internet-derived Topologies

Figure 9. T_{long} in Backup-Clique and Internet-derived Topologies

Although Ghost Flushing is effective in reducing both routing convergence delay and packet looping, it provides fast propagation of failure information without propagating the new reachability information at the same speed. Thus nodes that lost their current path to the destination due to the failure end up dropping packets, as opposed to continuing forwarding packets based on the old reachability information. Had these packets not been dropped, they may have been delivered to the destination by following a longer path or escaping a transient loop. Future research efforts are needed to develop a full understanding of routing convergence algorithms that can simultaneously minimize both packet looping and packet drops caused by the lack of reachability.

The *Sender Side Loop Detection (SSLD)*[8] applies the path loop detection rule at the sender. Before sending a path, a node checks whether the receiver is present in the path; if so, the sender knows the path will be discarded by the receiver. For example, with SSLD, node 5 in Figure 1(b) knows that node 6 will discard path $(5\ 6\ 4\ 0)$. Instead of sending this path (which is subject to MRAI timer delay), node 5 will send a withdrawal message to node 6 (which is not limited by the MRAI timer). Had the path $(5\ 6\ 4\ 0)$ been sent to node 6, it would only serve as a *path-based poison reverse* message; instead by sending a withdrawal SSLD allows this *path-based poison reverse* information to arrive faster. With SSLD, the loop duration in Figure 1(b) is determined only by the processing time and propagation delay. However, when a loop consists of more than two nodes such as the one in Figure 2, SSLD applies only when $path(c_1, new)$ includes c_m, or $path(c_m, new)$ includes c_{m-1}, and so on. The chance of such loop resolution is low. Our simulation results show that SSLD reduces packet looping by less than 20% when the topology is larger than 15 nodes, and improves convergence time only modestly, which confirms the results by [5]. SSLD consistently reduces BGP convergence delay and packet looping, albeit with a rather modest effectiveness.

Withdrawal rate limiting (WRATE) requires that the MRAI timer be applied to withdrawal messages as well. It has been implemented by at least one router vendor [8, 5], and the latest BGP specification draft [16] has adopted WRATE as the standard behavior. A withdrawal message can sometimes lead to inconsistent routing state (e.g. a router loses its current route to a destination and picks an obsolete alternative route), and WRATE "hopes" to reduce loops by propagating withdrawal and new reachability messages at the same speed. However, WRATE can delay a withdrawal that could have resolved a loop, thus lengthening the looping duration as a result. There has been no quantitative analysis to show how much WRATE can help reduce routing loops in general. It has been shown in [5] that WRATE improves the T_{down} convergence time in Clique topologies $only$, and makes T_{down} convergence time longer in other topologies. Our results further show that WRATE slightly increases the T_{long} convergence de-

lay in both B-Clique topology and Internet-derived topologies. WRATE also reduces packet looping in T_{down} for Clique and T_{long} in B-Clique by less than 20-30%. In the Internet-derived topologies, WRATE consistently worsens both the convergence time and packet looping; in particular, WRATE makes packet looping in T_{long} one order of magnitude worse than the standard BGP. Further examination on packet forwarding performance is needed to understand the overall impact of a change in BGP specification such as WRATE.

6 Summary

In any distributed routing protocol, topology (or policy) changes can lead to *inconsistent* routing state among network nodes. Whether this inconsistency results in transient forwarding loops depends on the ability of each node to avoid potential loops while selecting an alternative path. Link state protocols typically propagate updates fast to reduce the duration of inconsistency, but transient loops can still form since delays are inevitable. For distance vector protocols, poison-reverse can be used to detect two-node loops but fails to detect longer loops.

A path vector routing protocol extends the effectiveness of poison-reverse to the entire path by enabling each of the nodes in the path immediately detect loops involving itself. In other words, when node v has lost its current route to a destination, v can avoid those alternate paths that include itself. However as we demonstrated,this form of *path-based poison reverse* does not *eliminate* routing loops; in the worst case, a loop may not be detected *until* all the path updates triggered by a topology change have reached every node in the loop. Consequently, a routing loop can last as long as the routing convergence time period, and BGP's MRAI timer is the major contributing factor to the looping duration. Our simulation results show that overall looping duration are linearly proportional to the MRAI timer value. For a network using an Internet-derived 110-node topology, BGP may experience a convergence time of 523 seconds in T_{down} event; during this time packets in the network may encounter looping with a probability up to 86%.

Furthermore, we show through analysis and simulation that both the Assertion and Ghost Flushing approaches are effective in both speeding up routing convergence and reducing transient loops. Our results also show that the WRATE enhancement, recently adopted by BGP specification, may significantly lengthen the duration of transient loops compared to the standard BGP without WRATE.

To the best of our knowledge, this paper is the first effort to systematically examine the creation and duration of transient routing loops under a path-vector routing protocol such as BGP. As a first step our investigation started with a few simple simulation cases and used aggregate metrics such as overall looping duration and looping ratio to measure the severity of transient loops. As our next steps, we plan to examine route change traces to measure the statistics of individual loops such as the loop size and duration.

References

[1] A. Bremler-Barr, Y. Afek, and S. Schwarz. Improved BGP Convergence via Ghost Flushing. In *Proceedings of the IEEE INFOCOM*, April 2003.

[2] C. Cheng, R. Riley, S. Kumar, and J. Garcia-Lunes-Aceves. A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect. In *Proceedings of ACM Sigcomm*, pages 224–236, August 1989.

[3] J. J. Garcia-Luna-Aceves. A unified approach to loop-free routing algorithm using distance vectors or link states. In *Proceedings of ACM Sigcomm*, September 1989.

[4] J. Garcia-Lunes-Aceves and S. Murthy. A Path-Finding Alogirthm for Loop-Free Routing. *IEEE/ACM Transaction On Networking*, 5(1), February 1997.

[5] T. Griffin and B. Premore. An Experimental Analysis of BGP Convergence Time. In *Proceedings of ICNP*, November 2001.

[6] H. Hengartner, S. Moon, R. Mortier, and C. Diot. Dection and Analysis of Routing Loops in Packet Traces. In *Proceedings of ACM IMW 2002*, October 2002.

[7] C. Huitema. *Routing in the Internet*. Prentice-Hall, 2000.

[8] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet Routing Convergence. In *Proceedings of ACM Sigcomm*, August 2000.

[9] G. Malkin. Routing Information Protocol Version 2. RFC 2453, SRI Network Information Center, November 1998.

[10] J. Moy. OSPF Version 2. RFC 2328, SRI Network Information Center, September 1998.

[11] V. Paxson. End-to-End Routing Behavior in the Inthernet. *IEEE/ACM Transactions on Communications*, 5(5):610–615, 1997.

[12] D. Pei, L. Wang, D. Massey, S. F. Wu, and L. Zhang. A study of packet delivery performance during routing convergence. In *IEEE DSN*, June 2003.

[13] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, F. S. Wu, and L. Zhang. Improving BGP Convergence Through Assertions Approach. In *Proceedings of the IEEE INFOCOM*, June 2002.

[14] B. Premore. Multi-as topologies from bgp routing tables. http://www.ssfnet.org/Exchange/gallery/asgraph/index.html.

[15] Y. Rekhter and T. Li. Border Gateway Protocol 4. RFC 1771, SRI Network Information Center, July 1995.

[16] Y. Rekhter, T. Li, and S. Hares. Border Gateway Protocol 4. http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp4-20.txt, April 2003.

[17] A. Sridharan, S. Moon, and C. Diot. On the correlation between route dynamics and routing loops. In *Proceedings of ACM IMC 2003*, October 2003.

[18] The SSFNET Project. http://www.ssfnet.org.

[19] H. Tangmuarunkit, R. Govindan, S. Jamin, and W. Willinger. Network topology generators: Degree-based vs. structural. In *Proceedings of ACM Sigcomm*, August 2002.