BGP-RCN: Improving BGP Convergence Through **R**oot **C**ause **N**otification

Dan Pei, Matt Azuma, Nam Nguyen, Jiwei Chen, Dan Massey, and Lixia Zhang

Technical Report TR-030047
UCLA Computer Science Department
Oct 31st, 2003

***Abstract*— This paper presents a new mechanism, called BGP with Root Cause Notification (BGP-RCN), that provides an upper bound of $O(d)$ on routing convergence delay for BGP, where d is the network diameter. In our approach, each routing update message carries the information about the specific cause which triggered the update message. Once a node v receives the first update message triggered by a link failure, v can avoid using any paths that are obsolete due to the same failure. Our approach is applicable to path vector routing protocols in general, and our analysis and simulation show that BGP-RCN can achieve substantial reduction in both BGP convergence time and the total number of intermediate route changes.**

Keywords: *Path-vector routing protocols, Routing convergence, System design, Simulations*

I. Introduction

The Internet is composed of thousands of Autonomous Systems (ASes) and the Border Gateway Protocol (BGP) [1], [2], is used to exchange reachability information among the ASes. BGP routers adapt dynamically to changes in network topology and routing policy, but measurements in [3] showed that on average, it can take 3 minutes for the whole Internet to switch from failed routes to valid ones for a given destination. In some cases, it may take up to 15 minutes for the routing tables in all routers to stabilize. Previous studies [3], [4], [5], [6] have analyzed and addressed some aspects of path vector routing protocol convergence.

As a path vector routing protocol, BGP's routing update messages include the entire AS path to each destination. After a topology change (e.g. link or node failure) or policy change that invalidates a current best path, the router will select a new best path. The router, however, may mistakenly choose and propagate a path that has been obsolete due to the very same topology (or policy) change. This obsolete path may, in turn, be chosen by other nodes as their "new" best path, resulting in an invalid path being propagated throughout the network. Furthermore, BGP uses a *Minimum Route Advertisement Interval timer (MRAI timer)* to space out consecutive updates but it can also delay the propagation of *valid* reachability information [4], [7], [6]. Although some recent methods, such [5] and [6], can significantly reduce BGP convergence delays in many cases, they cannot prevent all the invalid paths from propagating out, and are rendered ineffective under certain topological conditions.

In this paper we present a new design, called *BGP with Root Cause Notification* (BGP-RCN), which piggybacks information about the root cause on each routing update message triggered by the root cause. Our complexity analysis and simulations show that BGP-RCN achieves significant improvement in both convergence time and message overhead. In the case of a single link failure, each update carries sufficient information to invalidate all other routes that depended on the failed link. Therefore, routers will never select an invalid route, and the convergence time is limited only by the physical propagation time and BGP's *MRAI* timer setting. For a 110-AS Internet-derived topology, when a destination becomes unreachable (i.e. a T_{down} event defined later), BGP-RCN reduces the convergence time from 715.3 seconds to 1.3 seconds, and reduces number of update messages from 30483 to 926. When a destination becomes reachable through a $longer$ path (i.e., a T_{long} event defined later), in 95% of the cases BGP converges in 56 seconds or less, and BGP-RCN reduces this time to 32 seconds.

The remainder of the paper is organized as follows. Section II presents a protocol model for the standard BGP and summarizes the existing complexity results from literature. Section III presents the BGP-RCN algorithm and provides the complexity analysis for BGP-RCN. Section IV discusses

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 article are those of the authors and do not necessarily reflect the views of the DARPA, NSF, or Cisco Systems

Dan Pei, Matt Azuma, Nam Nguyen and Lixia Zhang are with UCLA CSD; email:{peidan,mazuma,songuku,lixia}@cs.ucla.edu. Jiwei Chen is with UCLA EE Department; email:cjw@ee.ucla.edu. Dan Massey is with USC/ISI; email:masseyd@isi.edu.

implementation and deployment issues. Section V presents the simulation results. Section VI reviews previous work. Finally, Section VII concludes this paper.

II. A Simple Model for the Standard BGP

In this section we provide a simplified model for the standard BGP, the *Simple Path Vector Protocol (SPVP)*. This model is slightly different from that in [8], in that we model important BGP features such as the MRAI timer. We use this protocol model to define concepts of BGP convergence and summarize the complexity results from previous studies. We formalize our BGP-RCN approach as SPVP-RCN.

A. The SPVP Model

We model the Internet as a simple directed connected graph $G = (V, E)$, where $V = V_N \cup V_P$ and $E = E_N \cup E_P$. $V_N = \{0, 1, \cdots, n-1\}$ represents the set of n nodes that run SPVP protocol. The nodes in V_N are not considered destinations in network G, and correspond roughly to the Internet ASes. Nodes in V_N are connected by links in E_N. The destinations are prefixes attached to the nodes in V_N. For each node $i \in V_N$, i connects through links in E_P to a set of prefixes ($P^i = \{p^i_j\}$) (P^i could be empty). $V_P = \cup_i\{P^i\}$ is the set of all the destinations in the network G. Without loss of generality, we consider only a single destination p and assume it is connected only to node 0 (in BGP terms, p is "single-homed"). A path to destination p is an ordered sequence of nodes $P = (v_k v_{k-1} \ldots v_0 p)$ such that link $[v_i\ v_{i-1}] \in E_N$ and $v_i \in V_N$ for all $i, 0 \leq i \leq k$, and $v_0 = 0$. We say $v_i \in P, \forall i, 0 \leq i \leq k$; $[v_i\ v_{i-1}] \in P, \forall i, 1 \leq i \leq k$; and define $Length(P) = k + 1$.

SPVP is a single path routing protocol, in which each node advertises *only* its best route to neighboring nodes. For node v, the latest route received from neighbor u is stored in $rib\_in(v \Leftarrow u)$. After the initial route announcement, further updates are sent *only* if the best route changes (i.e. there are no periodic route announcements). A node v selects its best route, denoted $rib(v)$, according to some routing policy (or ranking function). Note that while the standard BGP allows arbitrary route selection policies, some policies can lead to persistent route oscillation [8]. Although SPVP and SPVP-RCN can work with any routing policies, for clarity, this paper considers only a shortest-path policy (which has been proven to converge [9]).

Nodes in V_N and links in E can fail and recover, and we assume that nodes u and v can detect failure or recovery of link $[u\ v]$ within limited time. The failure or recovery of link $[0\ p]$ can also be detected by node 0 within limited time. In response to either a link status change or a received routing update message, node v recomputes it best route $rib(v)$. If v's best route has changed, it will send the new $rib(v)$ to its neighbors. If the link status change or update message results in no available route to the destination, $rib(v) = \epsilon$ and a *withdrawal message* carrying an ϵ aspath is sent to the neighboring nodes.

SPVP includes an *MRAI timer* that guarantees any two (non-withdrawal) update messages from v to w will be separated by at least a *Minimum Route Advertisement Interval*. We use $mrai(v \Rightarrow w)$ to denote this timer, whose default value is 30 seconds, and we use the M to denote its default value. The MRAI timer is usually not applied to withdrawal messages, according to [1].

B. SPVP Convergence

[3], [4], [6] divide route convergence events into four classes with different properties. We briefly review these classes, map each class into the context of our SPVP model, and review the potential causes of each event. Note that our choice of a shortest path routing policy helps simplify the possible causes. For clarity, our analysis and simulations focus on the impact of single failure in a topology where each node has degree at least 2. Multiple failures are discussed later in the paper.[1]

1) T_{down}: a previously reachable destination becomes unreachable. Here, a link failure results in a network partition, and all nodes partitioned from the destination withdraw their routes to it. Note that, under the conditions discussed above, a T_{down} event can only occur when node 0 detects a failure in the $[0\ p]$ link.

2) T_{up}: a previously unreachable destination becomes reachable. This event can occur as the result of a link recovery that eliminates a partition, and all nodes in the previously disconnected graph learn routes to the destination. Again, in the case above, a T_{up} event can only occur when node 0 detects that the $[0\ p]$ link has recovered from a previous failure.

3) T_{long}: the current route to the destination becomes unavailable, and the affected nodes switch to a less-preferred alternate path. This event can occur when a node v uses link $[v\ u]$ to reach destination p, and v detects a failure of this link. As v's alternate path propagates, it may cause other nodes downstream of v to also select and advertise less-preferred alternate paths.

4) T_{short}: the current route to the destination is replaced by a more-preferred path. This occurs when a node v detects a recovery of link $[v\ u]$, which causes node v to learn and select a new, more-preferred path to destination p. As this change propagates, it may cause other nodes downstream of v to select and advertise more-preferred paths.

Although in practice, a new triggering event could occur before the previous convergence event finishes, routing convergence is typically defined relevant to *one* of these four events.

***Definition** 1: Converged State*: we say a node v is in a converged state if and only if $rib(v)$ will not change until the *next* triggering event.

[1] The RCN approach applies equally well in multiple link failures and sparse topologies, but the analysis becomes more complex to present.

***Definition** 2: Network Convergence Delay*: the network convergence delay starts when a triggering event T occurs and ends when each node reaches the converged state again.

[4] has shown the convergence time for T_{up} and T_{short} are both bounded by $M \cdot d$, where M is the MRAI timer value (i.e. 30 seconds). [4], [10] proved that the convergence of T_{down} is bounded by $M \cdot n$. Furthermore, since at most one advertisement can be sent in a particular direction on each link every M seconds, the message overhead of SPVP is bounded at $(|E_N| \cdot \frac{M \cdot n}{M}) = |E_N| \cdot n$. Like BGP-Assertion and BGP-GF, our RCN algorithm focuses on improving T_{down} and T_{long} convergence.

III. SPVP with Root Cause Notification

This section describes the RCN algorithm in the context of SPVP, analyzes the convergence time and message complexity, and demonstrate RCN's significant reduction in routing convergence time.

A. SPVP-RCN Algorithm

We first note that since SPVP has no periodic advertisements, an update sequence can be triggered only by a change in the physical link status (either a link failure or recovery), or a policy change that alters the link availability. When a link's status changes, the nodes adjacent to the link will detect the change, and at most one adjacent node may change its route as a result. We call this node the *root cause node (RCN)*. The root cause node will attach its name to the update it sends out after the status change, and this RCN is copied into and propagated along in all subsequent SPVP updates caused by the status change. In this way, any given node can learn the unique root cause node which spawned any given update messages it receives. Different from flooding used in link-state protocols (e.g. OSPF[11]), SPVP-RCN *piggybacks* the root cause in the routing updates, and thus only the affected nodes and their direct neighbors are notified.

In addition, each node v in SPVP-RCN maintains a sequence number, $ts(v)$, which is incremented by 1 upon each change in $rib(v).aspath$. This sequence number is used for both the aspath and RCN. An SPVP-RCN update message is defined as $r = \{r.aspath, r.ts, r.rcn\}$, where $r.aspath$ is the SPVP aspath; $r.ts = \{ts(u) | u \in r.aspath\}$ is a list of sequence numbers, which correspond in a one-to-one fashion with the nodes in $r.aspath$; and $r.rcn = \{c, ts(c)\}$ is the node ID and sequence number of the "root cause node".

To detect invalid transient routes, each node v keeps a copy of the highest sequence number for node x, denoted by $seqnum(v, x)$, that it has ever received. Upon receiving an update r, node v updates $seqnum(v, x)$ if either

1) $x \in r.aspath$ and $r.ts(x) > seqnum(v, x)$
2) $x = r.rcn.c$ and $r.rcn.ts(c) > seqnum(v, x)$

After any change in $seqnum(v, x)$, node v verifies all routes in its rib_in tables. If $x \in rib\_in(v \Leftarrow u).aspath$ and $rib\_in(v \Leftarrow u).ts(x) < seqnum(v, x)$, the route $rib\_in(v \Leftarrow u)$ is outdated and is removed (replaced with ϵ). This allows v to rapidly remove obsolete routes, improving convergence time.

B. An Example

Figure 1 illustrates how SPVP-RCN works. Figure 1(a) shows the SPVP routing tables prior to a failure. Each node's best path to the destination is marked with a star and the sequence numbers appear in brackets next to each node. Node 5's sequence number cache is shown in the box, and initially all sequence numbers in the cache are 0. In Figure 1(b), the link between node 2 and node 0 fails, so node 2 chooses backup path $(1[0]0[0]p)$, increases $ts(2)$ to 1, and includes $rcn = \{c, ts(c)\} = \{2, 1\}$ in the update message to neighbors 3 and 4.

Now suppose node 2's announcement reaches node 3 first. Node 3 will change to path $(2[1]1[0]0[0]p)$, increase $ts(3)$ to 1, and announce this new path to node 5 using $rcn = \{2, 1\}$ to indicate that the *root cause* of this announcement is a change in node 2. Without the use of RCN, node 5 would learn its current route via node 3 is invalid and node 5 would select the (invalid) alternate route via node 4 and advertise this path to its neighbors. With RCN, however, node 5 marks $rib\_in(5 \Leftarrow 4) = (4[0]2[0]0[0]p)$ invalid, since this route lists node 2's sequence number as 0, but the most recent update indicates node 2 has increased its sequence number to 1. Node 5 avoids selecting and further propagating the invalid route via node 4, resulting in the routing table shown in Figure 1(b).

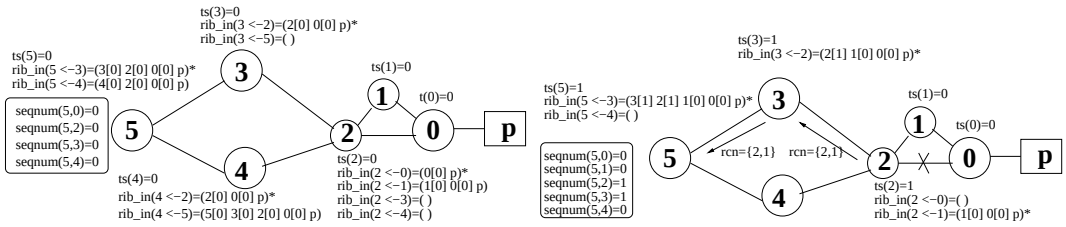
Note that SPVP-RCN not only prevents node 5 from adopting an invalid alternate route, but also allows node 5 to rapidly propagate the new information about node 2. After receiving the update from node 3, node 5 will change its route and send an update to node 4. This update from 5 to 4 lists the new sequence number for node 2. Node 4 learns that its route via node 2 is invalid when either the update from node 2 arrives or the update from node 5 arrives. In other words if $update(x, y)$ denotes the time to send an update from x to y, the time required for node 4 to learn its route is invalid is $min(update(2, 4), update(2, 3) + update(3, 5) + update(5, 4))$.

C. SPVP-RCN T_{down} Convergence Complexity

We now consider the behavior of single node v, and provide an upper bound on its convergence time. Figure 2 summarizes the symbols used throughout this section. For each node v, define $rib(v)^{old}$ as v's route before triggering event T, and $rib(v)^{new}$ as v's new route after v adapts to event T and returns to a converged state.

***Theorem** 1:* If $conv(v)$ denotes the convergence time of node v after a T_{down} event, then $l \cdot d \leq conv(v) \leq \mu \cdot d$.

Proof: By construction, node 0 is the only node directly connected to the destination p and, without loss of generality, we let $ts(0) = 0$ prior to event T. At the start of event T,



(a) Before the link failure

(b) After node 5 receives the first update from node 3

Fig. 1. SPVP-RCN Example For T_{long}

h	average nodal delay: the time it takes for a message to traverse one AS hop, including both processing delay and propagation delay (about 2 seconds according to [6])
$l(u,v)$	lower bound on the nodal delay from u to v if $link[u\ v] \in E_N$; otherwise $l(u,v) = \infty$.
l	$l = min_{link[u\ v]\in E_N}\{l(u,v)\}$
$\mu(u,v)$	upper bound of the nodal delay from u to v if $link[u\ v] \in E_N$; otherwise $\mu(u,v) = \infty$.
μ	$\mu = max_{link[u\ v]\in E_N}\{\mu(u,v)\}$
$d(u,v)$	the shortest aspath length from u to v
d	network diameter, $d = max_{u,v\in V}\{d(u,v)\}$

Fig. 2. Symbols used in Section III

node 0 detects that link $[0\ p]$ has failed. Node 0 immediately converges, and $conv(0) = 0$. We label the nodes in V_N according to their convergence time such that $conv(v_i) \geq conv(v_{i-1})$, for $1 \leq i \leq n-1$. Node v_1 converges as soon as it receives the first message from v_0 since the RCN sequence number carried by this message increases $ts(0)$ to 1 and invalidates all paths that v_1 currently has and might later receive during the T_{down} convergence. Thus for node v_1, $l(v_0,v_1) + conv(v_0) \leq conv(v_1) \leq \mu(v_0,v_1) + conv(v_0)$.

We now proceed by induction and show that $\forall v_i \in V_N$, $min_{0\leq j<i}\{l(v_j,v_i) + conv(v_j)\} \leq conv(v_i) \leq min_{0\leq j<i}\{\mu(v_j,v_i) + conv(v_j)\}$. Suppose the hypothesis holds true for v_i. Since every update contains an RCN that sets $ts(0) = 1$, any message received by v_{i+1} will invalidate all routes that v_{i+1} currently has and might later receive during the T_{down} convergence. Thus v_{i+1} converges after receiving the first message from any of the already-converged nodes $(v_0,\cdots,v_i)$. v_{i+1} will receive this first message no later than $min_{0\leq j<i+1}\{\mu(v_j,v_{i+1}) + conv(v_j)\}$ and no sooner than $min_{0\leq j<i+1}\{l(v_j,v_{i+1}) + conv(v_j)\}$. The hypothesis holds true and we have $l \cdot d(0,v) \leq conv(v) \leq \mu \cdot d(0,v) \leq \mu \cdot d$.

In other words, node v converges no later than the time it takes for a message to propagate along the shortest path from 0 to v with the maximal nodal delay μ, and no sooner than the time it takes for a message to propagate along the shortest path from 0 to v with the minimal nodal delay l.

***Corollary** 1.1:* If we model $h = l = \mu$, then SPVP-RCN's T_{down} convergence is bounded above by $h \cdot d$.

***Theorem** 2:* The message overhead for T_{down} convergence in SPVP-RCN is *bounded* by $|E_N|$

Proof: Each node will converge immediately after it receives its first message, and will send out just *one* withdrawal message to each neighbor. Since the number of directed links is $|E_N|$, so the message overhead is bounded by $|E_N|$.

Note that in some particular implementation, the number of messages might be smaller. For example, some time $before$ the current T_{down} event happens, node v chooses node u as the next hop. In some implementation, at that time node v will send a withdrawal message to node u(similar to poison reverse [12] in distance vector protocol) since node v knows u will discard v's announcement anyway due to BGP's loop detection mechanism. Therefore, when node v receives the first message during T_{down} convergence, send a withdrawal message to each neigbhor except the $current$ next hop u(the next hop before processing $first$ message). Since each of nodes $1,\cdots,n-1$ has a next hop, the total number of messages is $|E_N| - N + 1$ in such particular implementation.

Note that this message overhead is for T_{down} only, and the message overhead for T_{long} will be discussed in next section.

D. SPVP-RCN T_{long} Convergence

***Theorem** 3:* If $conv(v)$ denotes the convergence time of node v after a T_{long} event, then $conv(v) \leq d(2\mu + M)$.

Proof: Let $V_{stable} = \{v \in V_N | rib^{old}(v).aspath = rib^{new}(v).aspath\}$. In other words, V_{stable} is the set of nodes whose paths have not changed after convergence event T. Similarly, let $V_{affected} = V_N - V_{stable} = \{v \in V_N | rib^{old}(v).aspath \neq rib^{new}(v).aspath\}$. In

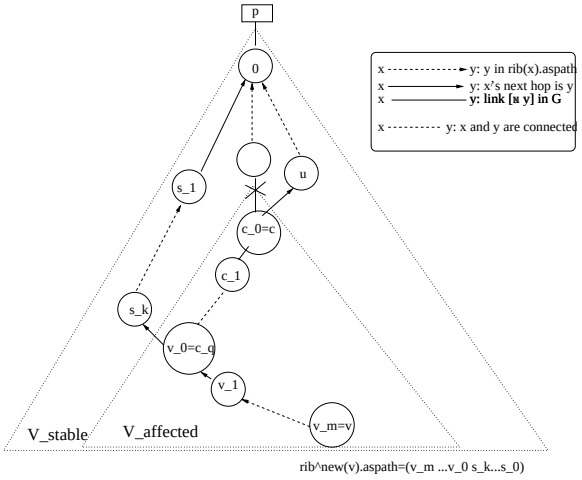


Fig. 3. Routing Trees after T_{long} Convergence

other words, $V_{affected}$ is the set of nodes whose paths have changed as a result of the convergence after event T. For a node $v \in V_{affected}$, the new path $rib^{new}(v).aspath$ must have the form $(v_m \dots v_0 s_k \dots s_0)$ where $v = v_m$, $v_i \in V_{affected}(0 \le i \le m)$, $s_j \in V_{stable}(0 \le j \le k)$, and $s_0 = p$.

The T_{long} convergence of node v is divided into two stages as illustrated in Figure 3. Node $c = c_0$ is the root cause node. During the first stage, node v_0 converges. By definition $rib^{new}(v_0) = (v_0 s_k \cdots s_0) = (v_0 \cdot rib\_in^{old}(v_0 \Leftarrow s_k))$. Since $s_k \in V_{stable}$, the path from s_k is already available in v_0's rib_in tables before the T_{long} event happens and this path will not change during convergence. Any path P strictly shorter than $Length(rib^{new}(v_0))$ must include the root cause node, c, (otherwise P would become the preferred alternate path after the convergence ends). Since every update in the T_{long} event includes the root cause node (and signals an increase in the sequence number for the root cause node), all shorter paths are marked invalid by v_0 as soon as it receives the first message after T_{long} event. The convergence time for v_0 is no later than the longest time it takes a message to propagate from $c = c_0$ to v_0 and no sooner than the shortest time it takes a message to propagate from $c = c_0$ to v_0 . More precisely, $d(c_0, c_q) \cdot l = q \cdot l \le conv(v_0) \le d(c_0, v_0) \cdot \mu = q \cdot \mu \le d\mu$.

The *second stage* starts when v_0 converges and ends when $v = v_m$ converges. We now show this is equivalent to a T_{short} event with a root cause v_0. After node v_0 converges, its path will be propagated along $v_1, \cdots v_{m-1}$ to $v_m = v$. As soon as v_{i+1} receives v_i's update, v_{i+1} learns the shortest route that does not include the root cause node (since $rib^{new}(v_{i+1}).aspath = v_{i+1} \cdot rib^{new}(v_i).aspath$). In addition, the new update allows v_{i+1} to immediately discard any shorter routes that contain the root cause node (since the update carries a new sequence number for the root cause node). Thus, the convergence time of a node v_i in stage 2 is no longer than the time it takes for a message to propagate along the path from v_0 to v_i with the longest nodal delay plus MRAI timer. The convergence time of node v_i in stage 2 is no shorter than the time it takes for a message to propagate along the path from v_0 to v_i with the shortest nodal delay. More precisely, $d(v_0, v_i) \cdot l = q \cdot l \le conv(v) \le d(v_0, v_m) \cdot (\mu + M) = m \cdot (\mu + M) \le d(\mu + M)$ and thus node v's T_{long} convergence time is upper bounded at $(\mu \cdot d + (\mu + M)d) = (2\mu + M)d$.

***Theorem** 4:* The message overhead for T_{long} convergence in SPVP-RCN is $|E_N| d \frac{(2\mu+M)}{M}$.

Proof: There can be only one message sent every M seconds on each of the $|E_N|$ directed links during T_{long} convergence. Since convergence lasts at most $d(2\mu + M)$ seconds, there can be at most $|E_N| \cdot d(2\mu + M)/M = |E_N| d \frac{(2\mu+M)}{M}$ messages sent.

***Corollary** 4.1:* If we model $\mu = l = h$, then SPVP-RCN's T_{long} convergence time's upper bound is $(2 \cdot h + M)$, message overhead is $|E_N| d \frac{(2h+M)}{M}$.

E. Complexity with Per-Neighbor MRAI Timer and WRATE

Up to this point, we have assumed a distinct MRAI timer is kept for each (neighbor, destination) pair. In practice, however, the MRAI timer is typically implemented only for each neighbor, and all destinations advertised to that neighbor share the same MRAI timer [3], [4], [6]. In this case, an update regarding prefix p will turn on the per-neighbor timer and delay updates regarding other prefixes p'. [4] shows that the MRAI is almost always on, and the waiting time for a newly arrived update message was observed to be uniformly distributed between 0 and M seconds. Thus, even the first message for a given destination can be delayed for 15 seconds, on average, because some previous update message for a different prefix has already triggered the per-neighbor MRAI timer. In this case, SPVP-RCN's upper bound for T_{down} remains unchanged since the MRAI timer does not apply to withdrawal messages. But for T_{long}, the upper bound on the convergence time would increase to $d \cdot (2h + 60)$, and the message upper bound would increase to $|E_N| d \frac{2h+2M}{M}$.

In addition, certain BGP implementations use Withdrawal Rate Limiting (WRATE), and the latest BGP draft [2] proposes the use of WRATE. In WRATE, nodes apply the MRAI timer to withdrawals as well as advertisements [3], [4], [7]. In this scenario, the SPVP-RCN T_{down} convergence time would be bounded above by $(h + M) \cdot d$, but the message overhead still remains $|E_N| - n + 1$. Regardless of the MRAI timer implementation, the SPVP-RCN algorithm still has the time complexity of $O(d)$, and only constant factors change with the per-neighbor MRAI timer and/or WRATE.

F. SPVP-RCN Storage Overhead

SPVP-RCN requires that each router store the sequence number associated with each AS in its rib_in cache. In the

RCN algorithm presented in this section, each router needs $O(|V_N| = n)$ storage overhead for each prefix. Therefore, it needs $O(n \cdot |V_P|)$ storage overhead for all the prefixes in the network. In practice, however, the storage overhead can be much less. The sequence number for node x is stored only after x has appeared in some path received by node v, and the number of such x is typically less than the number of nodes in the network. In addition, BGP-RCN stops nodes from exploring many invalid paths currently tried in the standard BGP, and this also helps to reduce the number of nodes whose sequence numbers must be cached.

IV. Implementation and Deployment

Section III presented the basic design of RCN, this section briefly discusses a number of implementation and deployment issues.

A. Transmission of Sequence Numbers and RCN

In order to transmit the sequence numbers and the RCN, we define a new community attribute [13] and include this attribute in BGP update messages. The community attribute is a 32-bit value normally associated with route advertisements, and is used to convey routing policy information. All sequence numbers and the RCN can be encoded into a sequence of community attributes. The encoding specification is omitted for brevity.

This approach is compatible with existing BGP implementations. If a router does not implement BGP-RCN, it will not attach a sequence number community attribute for itself, but since the community attribute is an optional transitive attribute, the router will further propagate the community attributes learned from a neighbor, according to the operation of the standard BGP [1], [2].

B. Handling the Absence of RCN

The discussions in Section III assume that either a node v detects a link change and sets v itself as the root cause node in the outgoing updates, or v propagates the RCN carried in the incoming update message that triggered the $rib(v).aspath$ change. It is possible, however, that an incoming update has no RCN. If such an update triggers a $rib(v).aspath$ change, node v should set itself as the RCN in outgoing update messages.

This approach allows incremental deployment of RCN in a network. For example, suppose node u is the root cause node, but has not implemented RCN. Updates from u will not contain an RCN, but the first RCN-capable node, v, that acts on the update will set itself as a root cause. In other words, the RCN-capable nodes closest to the "real" root cause will set themselves as the root cause. Although the full power of RCN may not be achieved in such a partial deployment case, any invalid paths containing v can still be quickly removed by other RCN-capable nodes.

This approach also handles "policy withdrawals". In a policy withdrawal, node u may decide to stop announcing reachability for prefix p to neighbor v, but u has not changed its AS path to prefix p, and thus the sequence number $ts(u)$ has not changed either. In this case, u sends a "policy withdrawal" to v, which contains no RCN. Node v treats such a withdrawal as a failure of link $[v\ u]$, and following the rule above, sets itself as the root cause.

C. Sequence Number Issues

There are several issues common to any approach that uses sequence numbers. A node might lose its current sequence number as the result of a crash, sequence numbers can wrap around, or a fault (or intentional attack) could introduce erroneous sequence numbers. In particular, an attacker who has compromised a node v can launch a Denial-of-Service attack on destination p by sending a withdrawal with $rcn = \{0, ts'(0)\}$, where $ts'(0) > ts(0)$ and $ts(0)$ is node 0's latest sequence number. As a result, nodes who believe the false sequence number will remove their valid paths to node 0.

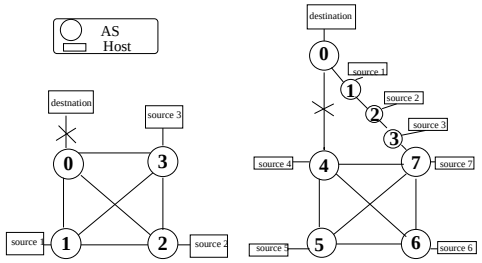
There is considerable prior work on managing sequence numbers. Techniques proposed in [14] can be used to deal with these issues in face of arbitrary failures. Alternatively, timestamps could be used instead of sequence numbers to solve the wrapping around and rebooting problems. Using timestamps would also assist in protecting nodes against false sequence numbers by allowing nodes to apply a sanity check, ensuring that the timestamps are within a reasonable range. Cryptography can also be used to protect the sequence number, as in [15], when adding the sequence number to the origin AS. Furthermore, OSPF [11] is widely used and addresses sequence number problems; we primarily borrow techniques from this approach.

D. Modeling an AS as a Node

To present the RCN design, we modeled each AS as a single node, and assume in the SPVP model that each node announces the same paths to all its neighbors. In reality, each AS is a collection of routers, and some large ASes may announce different paths to different neighbors [5]. Like [5], [6], we can divide each AS into multiple "logical ASes" such that each logical AS announces the same paths to all its neighbors. A logical AS is represented as $(ASN, entry\text{-}router)$, and each router in this logical AS uses the path learned from the $entry\text{-}router$. For our RCN approach, we require each router in the same logical AS to use the sequence number announced by the $entry\text{-}router$.

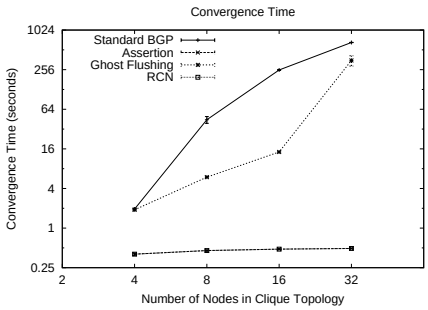
E. Multiple Failures Overlapping in Time

For simplicity, Section III discusses how SPVP-RCN works in the case of a *single link failure*. But, in a network as large as the Internet, it is likely that multiple failure events

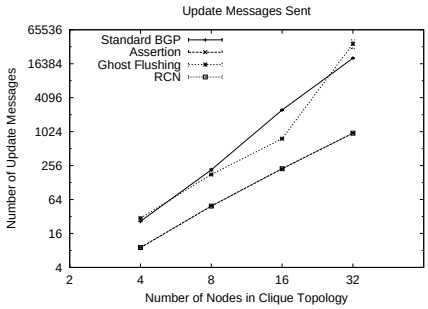


(a) Clique of size 4 (b) B-Clique of size 4

Fig. 4. Clique and B-Clique Topologies



(a) Convergence Time



(b) Number of Update Messages

Fig. 5. Results for T_{down} Convergence in Clique Topologies

could occur concurrently and affect routes to the same destination. For example, suppose link $[v\ u]$ quickly flaps between the $down$ and up status. Before the convergence triggered by the first $down$ event has ended, the up event has begun, and timing could result in different views of the link at different nodes. A node v may receive $rib\_in(v \Leftarrow u)$ from u such that $rib\_in(v \Leftarrow u)$ contains nodes x and y, where $ts(x) > seqnum(v,x)$, but $ts(y) < seqnum(v,y)$. In this case, $seqnum(v,x)$ is updated to the newer sequence number $ts(x)$, and $rib\_in(v \Leftarrow u)$ is invalidated and removed.

It is possible for the $rcn.ts(c)$ in $rib\_in(v \Leftarrow u)$ to be smaller than its counterpart $seqnum(v,c)$. In this case, it is not entirely straightforward how we set v's outgoing RCN, since we know that node c has increased its sequence number to at least $seqnum(v,c) > rcn.ts(c)$. We choose to use an easy-to-understand approach, and set v's outgoing RCN as $rcn = \{c, seqnum(v,c)\}$ Thus, although v's outgoing RCN might not be the "real" root cause, it still achieves the effect of quickly removing any paths that contain c and have a sequence number smaller than $seqnum(v,c)$.

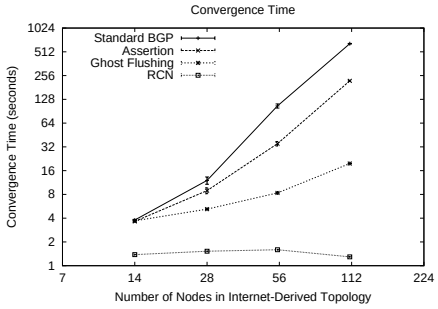
V. SIMULATION RESULTS

We used the SSFNET [16] simulator to evaluate BGP performance and data delivery. For a comparative evaluation, in addition to BGP-RCN we also simulated the standard BGP and two of the previously proposed BGP convergence speedup approaches, BGP-Assertion and BGP-GF [5], [6] which are described in delay in Section VI. SSFNET has a built-in BGP simulator, we added the implementation of BGP-RCN as well as the implementations of BGP-Assertion and BGP-GF according to [5], [6], respectively. A third-party package [17] was incorporated and modified to enable tracing packet forwarding paths in SSFNET.

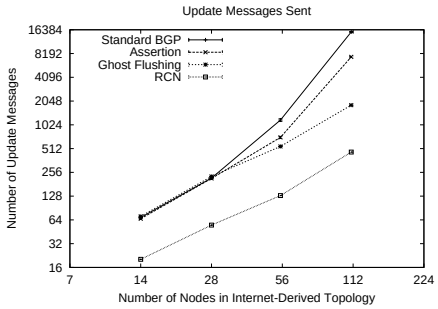
A. Simulation Setting

We used Clique, Backup-Clique (or B-Clique in short) and Internet-derived network topologies to evaluate the performance of different BGP variants. Clique (full-mesh) topologies, as shown in Figure 4(a), are frequently used in literature [3], [7], [18], [6] as a simple basis for analysis and comparison. A B-Clique topology of size n, as shown in Figure 4(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. Furthermore, 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); it has been used by [6] to study T_{long} convergence. To derive a simulation topology that resembles the Internet topology, we first generated a 110-node topology based on BGP routing tables, using the algorithm described in [19]. Following the same algorithm, we randomly removed some links and selected the largest connected sub-graph. In this sub-graph, we merged two non-adjacent nodes with the smallest degrees, and which shared no neighbors. This merging was repeated until all nodes in the sub-graph had degree 2 or greater. We used this method to generate two 55-node topologies. Similarly, we generated four 28-node topologies and eight 14-node topologies.

In all our simulations, the $MRAI$ timer value was set to 30 seconds plus a random jitter. The link propagation delay was 2 milliseconds, and the processing delay of each routing message was chosen randomly between 0.1 and 0.5 seconds.



(a) Convergence Time



(b) Number of Update Messages

Fig. 6. Results for T_{down} Convergence in Internet-Derived Topologies

The bandwidth of each network interface was 10Mbps.

To evaluate the performance of the standard BGP and three of its proposed variants, we measured not only network routing convergence time and number of routing update messages, but also data packet losses during routing convergence for T_{long} event. As shown in [20], a short convergence time does not necessarily imply minimal packet losses. We believe that maximizing packet delivery should be one of the design priorities for all routing protocols. In all of our T_{long} simulations, there is a data source attached to each AS node in the network except the origin AS. Each data source generated packets at the rate of one packet per second. The packet size was 24 bytes and carried a TTL (Time-To-Live) value of 128, the default setting in SSFNET. Given each link has a bandwidth of 10Mbps, there is no congestion-induced packet losses during simulation.

B. T_{down} Simulation Results

Clique and Internet-derived topologies were used in simulation to evaluate the performance of BGP-RCN. For Clique topologies, we chose node 0 as the only origin AS which advertised a destination prefix, and simulated the T_{down} event by marking node 0 down. The simulation results are based on 100 simulation runs which used different random seeds. Figure 5 shows the convergence delay and the number of update messages averaged over 100 runs with a 95% confidence interval. For Internet-derived topologies, one node x was chosen as the only origin AS which advertised a destination prefix, and we simulated the T_{down} event by marking this node x down. We repeated the simulation with five random seeds. We then repeated that set of simulations for each node in each topology. Therefore, we have $1 \times 110 \times 5 = 550$ runs for the 110-node topology, and $2 \times 55 \times 5 = 550$ runs for the two 55-node topologies. Figure 6 shows the T_{down} convergence results averaged over 550 runs with a 95% confidence interval for Internet-derived topologies. Note that both the x and y-axis are in log scale.

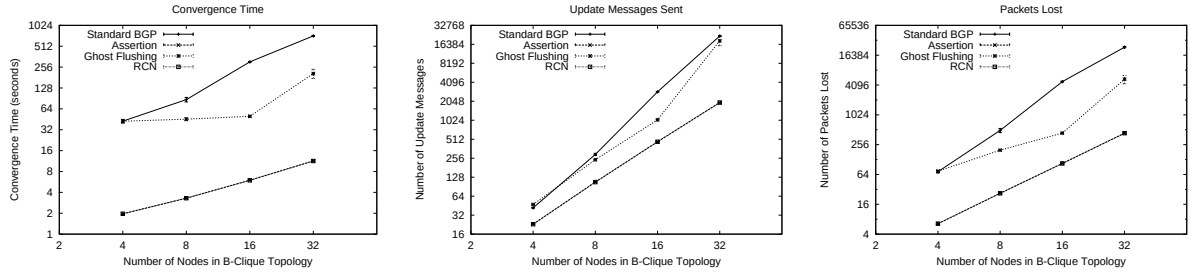
Our results show that, compared with the standard BGP, BGP-RCN can reduce BGP's T_{down} convergence time by 2 to 3 orders of magnitude, and reduce the total number of routing update messages by 1 to 2 orders of magnitude. For the 110-node Internet-derived topology, the T_{down} convergence time was reduced from 648.4 seconds to 1.3 seconds, and number of messages was reduced from 15387 to 463. For Clique topologies of size 32, the convergence time was reduced from 662.1 seconds to 0.5 seconds, and the number of messages was reduced from 20533 to 961. BGP-RCN's dramatic improvement is consistent with our analysis in Section III.

C. T_{long} Simulation Results

B-Clique and Internet-derived topologies were used in simulation to evaluate the T_{long} performance of BGP-RCN.

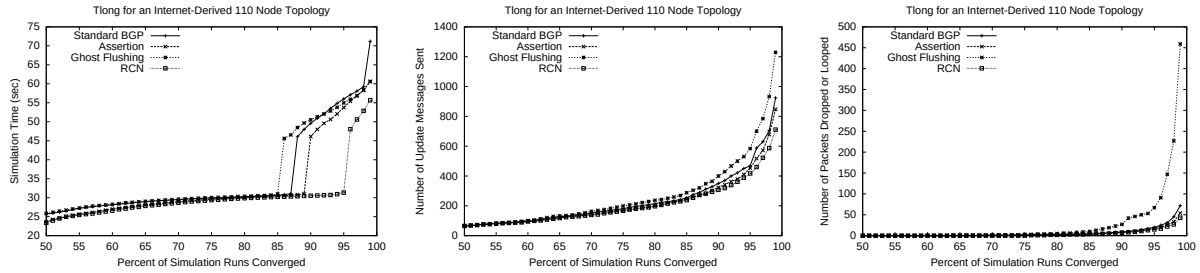
1) B-Clique Topologies: To simulate the T_{long} event in B-Clique topologies, we chose node 0 as the origin AS and marked the link $[0\ n]$ down after the simulation started. The average T_{long} convergence results over 100 runs with 95% confidence interval are shown in Figure 7. Note that both the x and y-axis are in log scale. As the figure shows, BGP-RCN can reduce BGP's T_{long} convergence time and number of messages in B-Clique topologies by 1 to 2 orders of magnitude. For the B-Clique topology of size 32, the convergence time was reduced from 720.0 seconds to 11.3 seconds, and the number of messages was reduced from 22211 to 1955. This improvement is less dramatic than in the T_{down} case because BGP-RCN's convergence time upper bound is $d(2h + 30)$ seconds, a function of the value of MRAI timer which has a big impact during the second stage of T_{long}'s convergence. Nevertheless, BGP-RCN significantly improves packet delivery performance by reducing the number of packets lost by 1 to 2 orders of magnitude, from 23730 in standard BGP to 438 in BGP-RCN.

2) Internet-Derived Topologies: For Internet-derived topologies, one node x was chosen as the only origin AS which advertised a destination prefix, and we simulated the T_{long} event by marking down one of x's links, l. We repeated the simulation with five random seeds. We then repeated that set of simulations for each l and x for each topology. There are 286 links in our 110-node topology and each



(a) Convergence Time (b) Number of Update Messages (c) Number of Packet Losses

Fig. 7. Results for T_{long} Convergence in B-Clique Topologies



(a) Convergence Time (b) Number of Update Messages (c) Number of Packet Losses

Fig. 8. Results for T_{long} Convergence in Internet-Derived Topologies

link was failed twice, thus $1 \times 2 \times 286 \times 5 = 2860$ simulation runs were conducted for the 110-node topology. Unlike in B-Clique topologies where the failure of the *current* route to node 0 forces most of the other nodes to move to a new route, relatively few nodes have to change their route to 0 after a single link failure. In particular, our Internet-derived topologies include a number of nodes that have a large number of links to other nodes, resembling the ASes of large Internet service providers. For example, 5 nodes in the 110-node topology connect to 20 or more other nodes each, and another 7 nodes have 10 or more links each. When a link attached to one of these well-connected nodes fails, very few nodes in the network need to readjust their routes. Furthermore, when those affected nodes try to find the next best path to the same destination, due to the rich connectivity, not only do most of the nodes have multiple alternative routes to each destination, but also, when a node naively picks a next best path, the new path has a good chance of not depending on the failed link, even without knowing which link failed.

Our simulations of the 110-node Internet-derived topology show that more than 50% of our T_{long} simulation runs had short convergence delays and very small numbers of update messages, even under the standard BGP. Therefore, instead of using the average, we choose to show the percentile curve of T_{long} convergence results. A point (x, y) in Figure 8(a) means that x% of the link-failures resulted in a convergence delay of no longer than y seconds. Notice that, for up to 85% of the links, the convergence delay is similar for all the four variants of BGP; above that point, the convergence delay reduction by BGP-RCN becomes more pronounced. For example, 95% of link failures converged in 56 seconds or less in the standard BGP, and BGP-RCN improved this number to 32 seconds. BGP-RCN does not bring significant reduction in message count, not because BGP-RCN did not perform well, but rather it is because the rich connectivity in the simulated topology enabled good performance for the other three BGP variants.

VI. Comparison with Previous Work

In this section, we compare our BGP-RCN design with other convergence improvement mechanisms. Figure 9 summarizes the upper bound of the standard BGP, BGP-RCN, BGP-Assertion [5] and BGP-GF (Ghost Flushing [6]), as well as the empirical data for the parameters used.

A. Assertion Approach

In BGP-Assertion [5], when a node v receives $rib\_in(v \Leftarrow u)$ from a neighbor node u, it checks $rib\_in(v \Leftarrow w)$ it has received from each node w among its neighbors. If

Protocols	T_{down} Conv. Delay	T_{down} Messages	T_{long} Conv. Delay	T_{long} Messages
the standard BGP	$M \cdot n$	$\|E_N\| \cdot n$	$M \cdot n$	$\|E_N\| \cdot n$
BGP-Assertion[5]	N/A	N/A	N/A	N/A
BGP-GF[6]	$h \cdot n$	$2\|E_N\|n\frac{h}{M}$	N/A	N/A
BGP-RCN	$h \cdot d$	$\|E_N\| - n + 1$	$d(2h + M)$	$\|E_N\|d\frac{(2h+M)}{M}$

(a) Upper Bound

data from [21], [6]
$n \approx 15000$
$d \approx 11$
$\|E_N\| \approx 80000$
$h \approx 2\ seconds$

(b) Empirical Data

Fig. 9. Convergence time and message upper bound.M is $MRAI$ timer value. n is the number of ASes in the network, d is the network diameter, $|E_N|$ is the number of directed AS-Level links in the network. h is the average delay for a BGP update message to traverse an AS hop.

$u \in rib\_in(v \Leftarrow w).aspath$ but $rib\_in(v \Leftarrow w)$ is inconsistent with $rib\_in(v \Leftarrow u)$, then $rib\_in(v \Leftarrow w)$ is marked as infeasible. The effectiveness of BGP-Assertion is sensitive to the specific topological connectivity of a network; it cannot eliminate the propagation of invalid paths under certain topological conditions. For example, when BGP-Assertion is used in the topology shown in Figure 1(b), after node 5 receives node 3's new update, it will choose the invalid backup path $(5\ 4\ 2\ 0\ p)$ and further propagate the path to other neighbors. In BGP-RCN, node 3's update message carries a root cause $\{2, 1\}$ which invalidates all the paths that is obsolete because of the root cause (that is, any path includes node 2 but with $ts(2) < 1$, e.g. $(4[0]2[0]0[0]p)$). For T_{down} convergence in Clique topologies and T_{long} convergence in B-Clique topologies, BGP-Assertion's performance is as good as that of BGP-RCN, and 2 orders of magnitude better than BGP-GF (Figures 5 and 7). For T_{down} convergence in Internet-derived topologies, however, its performance is 2 orders of magnitude worse than BGP-RCN, and 1 order of magnitude worse than BGP-GF (Figure 6), because both BGP-RCN and BGP-GF avoid the impact of MRAI from T_{down} convergence, while Assertion does not. BGP-Assertion does improve T_{long} convergence with the Internet-derived topology, but only to a limited extent.

B. BGP-GF: Ghost Flushing

In BGP-GF, when a node u has changed its path to a less preferred one, but cannot propagate it due to the MRAI timer, u will first send a "flushing" withdrawal message to flush out the path previously advertised by u. Thus, u's neighbor v can avoid using u's invalidated path. Because BGP withdrawal messages are not subject to the MRAI timer delay, invalid paths can potentially be quickly deleted from the entire network. Like BGP-Assertion, BGP-GF does not eliminate the propagation of *all* invalid paths. For example, when BGP-GF is used in Figure 1(b), after the link $[2\ 0]$ fails and node 2 sends the update, node 3 will send a flushing withdrawal to node 5 to remove the old path $(3\ 2\ 0\ p)$. Node 4 will also send a flushing withdrawal to node 5, but this flushing withdrawal might arrive at node 5 after the node 5 processes the flushing withdrawal from node 3, chooses an obsolete backup path $(5\ 4\ 2\ 0\ p)$ and propagates it further. According to [6], BGP-GF's T_{down} convergence time is bounded at $h \cdot n$ seconds and the message overhead is bounded at $2|E_N|n\frac{h}{M}$. No complexity results for T_{long} is provided in [6]. Because BGP-RCN eliminates all the invalidated paths, its convergence time is upper bound by $O(d)$ in T_{down} convergence, where d is the network diameter, which is about 3 orders of magnitude smaller than n in today's Internet topology, as shown in Figure 9.

Figure 6 shows that BGP-GF can reduce T_{down} convergence time by 1 order of magnitude in the Internet-derived topologies. Figure 5(b), however, shows that in a densely connected topology such as a Clique of size 32, the additional withdrawals sent by BGP-GF lead to higher message overhead than in the standard BGP. This causes a flood of messages to process, and an invalid path P can be propagated out by a node v before v can process the latest updates (either a new advertisement or a flushing withdrawal) that would have invalidated P. Therefore, the improvement in convergence time at size 32 is much less than at size 16 (Figure 5(a)). The additional withdrawal messages also reduce the improvement of T_{long} convergence in B-Clique of size 32 (Figure 7), and even increases the T_{long} convergence time in Internet-derived topologies (Figure 8).

Because BGP-GF quickly removes invalidated paths, but does not necessarily speed up the propagation of alternative ones, it causes significantly more packet losses than the standard BGP during T_{long} convergence in Internet-derived topologies (Figure 8(c)). BGP-Assertion and BGP-RCN, on the other hand, both reduce packet losses. This observation concurs with [20] that minimizing convergence time in T_{down} does not necessarily lead to minimal packet losses in T_{long} convergence.

BGP-GF has an advantage that it does not change the format of the standard BGP message. But, it does require that the last path sent to each neighbor be recorded, leading to a storage overhead which may be of practical concern in implementation cost [22], [23], [16]. Furthermore, the flushing withdrawals sent by BGP-GF are incompatible with the recent adoption of WRATE [2], and can result in penalty by the route damping mechanisms [24], [18]. BGP-RCN re-

quires changes to the standard BGP update packet format, but does not generate *additional* updates and has no conflict with WRATE or route damping.

C. Other related work

To speed up the convergence of path-finding algorithm for distance vector protocols, [25] proposed stamping each routing update with the triggering link failure. [26] proposed explicitly signaling the T_{down} failure, and their approach can improve the T_{down} convergence time significantly. These two approaches did not address the issues of multiple failures overlapping in time, as discussed in Section III-E. The issues of multiple failures is mitigated by the "BGP-Cause Tag (BGP-CT)" approach [27], [28] using timing heuristic. BGP-CT also explicitly signals the link failure, but it marks those suspected paths as infeasible, and avoids choosing and propagating infeasible path. BGP-CT assigns a timer for infeasible paths and re-uses the infeasible path when the timer expires. Our BGP-RCN approach can safely remove all the paths that are invalidated by RCN, due to the use of sequence numbers. This distinguishes BGP-RCN from previous root cause notification approaches.

Sequence numbers have been used before by [15] to improve the security of BGP. In this approach, each origin AS maintains a sequence number for each prefix it originates, and increases the sequence number when it withdraws or reannounces the prefix. If a withdrawal can carry the sequence number, this approach can be considered a sub-case of our RCN approach. as BGP-RCN does. Similar approaches have been proposed for distance vector protocols in [29] and AODV [30]. Our BGP-RCN approach lets every AS maintain sequence number for each prefix in the network, and improves both T_{long} convergence and T_{down} convergence.

VII. Conclusion

As evidenced by previous measurement and simulation efforts [4], [7], [5], [6], both the convergence time and message overhead of standard BGP can increase quickly as the network topology becomes larger in size and denser in its connectivity. Previous work [4], [10] proved that standard BGP T_{down} and T_{long} convergence time has an upper bound of $O(n)$, where n is the number of AS nodes in the network, and a message overhead upper bound of $|E_N| \cdot n$, where $|E_N|$ is number of directed AS-level links.

Our proposed BGP-RCN design reduces BGP's convergence time upper bound to $O(d)$, where d is the network diameter. By piggybacking the root cause on each update message, the first update message after a link failure allows a node to invalidate all the paths that are obsolete due to the same failure; this includes both obsolete paths currently in the routing table as well as obsolete paths that could be received in the future. Our simulation results show that the convergence time for a T_{down} event is reduced by at least 2 orders of magnitudes in both Clique and Internet-derived topologies. For T_{long} events, the elimination of invalid paths enables BGP-RCN to propagate only valid reachability information. As a result, simulations on B-Clique topologies showed substantial reduction in the convergence time, update messages, and packet losses after a connectivity change. Simulations of the Internet-derived topology also showed an improvement by BGP-RCN over standard BGP in all the three measurements, although the improvement is moderate in most cases. This is not because BGP-RCN did not perform well, rather, the rich connectivity in our simulated topologies enabled the other three protocols to also perform well. When a link failure occurs close to the network edge, BGP-RCN offers more pronounced improvement.

In addition to routing convergence improvements, we believe that the root cause information carried in BGP-RCN has potential to help diagnose Internet routing performance. When a routing change occurs in today's Internet, it is often difficult to infer the cause and origin of the change. We plan to leverage the root cause information in our future efforts in understanding global routing dynamics.

VIII. Acknowledgments

We would like to thank Eli Gafni, Songwu Lu, S. Felix Wu for invaluable discussions, Xiaoliang Zhao for his help on simulation setting package, and Beichuan Zhang, Mohit Lad, Vasilis Pappas for their comments on an earlier version of this paper.

References

[1] Y. Rekhter and T. Li, "Border Gateway Protocol 4," RFC 1771, SRI Network Information Center, July 1995.
[2] 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.
[3] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet Routing Convergence," in *Proceedings of ACM Sigcomm*, August 2000.
[4] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The Impact of Internet Policy and Topology on Delayed Routing Convergence," in *Proceedings of the IEEE INFOCOM*, April 2001.
[5] 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.
[6] A. Bremler-Barr, Y. Afek, and S. Schwarz, "Improved BGP Convergence via Ghost Flushing," in *Proceedings of the IEEE INFOCOM*, April 2003.
[7] T. Griffin and B. Premore, "An Experimental Analysis of BGP Convergence Time," in *Proceedings of ICNP*, November 2001.
[8] T. Griffin, F. B. Shepherd, and G. Wilfong, "The stable path problem and interdomain routing," *IEEE/ACM Transactions on Networks*, vol. 10, no. 2, 2002.
[9] T. Griffin and G. Wilfong, "A Safe Path Vector Protocol," in *Proceedings of IEEE INFOCOMM*, March 2000.
[10] D. Obradovic, "Real-time Model and Convergence Time of BGP," in *Proceedings of the IEEE INFOCOM*, June 2002.
[11] J. Moy, "OSPF Version 2," RFC 2328, SRI Network Information Center, September 1998.
[12] Christian Huitema, *Routing in the Internet*, Prentice-Hall, 2000.
[13] R. Chandra, P. Traina, and T. Li, "BGP Communities Attribute," RFC 1997, SRI Network Information Center, August 1996.
[14] Radia Perlman, *Network Layer Protocols with Byzantine Robustness*, Ph.D. thesis, MIT Lab. for Computer Science, 1988.

[15] B. R. Smith, S. Murphy, and J. J. Garcia-Luna-Aceves, "Securing the border gateway routing protocol," in *Global Internet'96*, November 1996.
[16] "The SSFNET Project," http://www.ssfnet.org.
[17] Michael Liljenstam, "Ssf.os.trace - a record route mechanism for ssfnet ip, v 0.1," http://www.cs.dartmouth.edu/ mili/research/ssf/trace/Trace.html.
[18] Z. Mao, R. Govindan, G. Varghese, and R. Katz, "Route Flap Damping Exacerbates Internet Routing Convergence," in *Proceedings of ACM Sigcomm*, August 2002.
[19] B.Premore, "Multi-as topologies from bgp routing tables," http://www.ssfnet.org/Exchange/gallery/asgraph/index.html.
[20] 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.
[21] G. Huston, "BGP Table Data," http://bgp.potaroo.net/.
[22] C. Labovitz, G. Malan, and F. Jahanian, "Origins of Internet Routing Instability," in *Proceedings of the IEEE INFOCOM*, march 1999.
[23] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu, and L. Zhang, "Observation and Analysis of BGP Behavior under Stress," in *Proceedings of the ACM IMW 2002*, October 2002.
[24] C. Villamizar, R. Chandra, and R. Govindan, "BGP Route Damping," RFC 2439, SRI Network Information Center, May 1998.
[25] C. Cheng, R. Riley, S. Kumar, and J.J. Garcia-Lunes-Aceves, "A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect," in *Proceedings of ACM Sigcomm*, August 1989, pp. 224–236.
[26] J. Luo, J. Xie, R. Hao, and X. Li, "An Approach to Accelerate Convergence for Path Vector Protocol," in *Proceedings of IEEE Globecom*, Nov. 2002.
[27] C. Labovitz and A. Ahuja, "Modeling inter-domain routing protocol dynamic," http://www.caida.org/outreach/isma/0012/talks/labovitz/, December 2000.
[28] R. Wattenhofer, "Slow internet routing convergence," http://www.inf.ethz.ch/s̈chlude/webalgs/BGP_slides.pdf, December 2002.
[29] B. R. Smith, S. Murphy, and J. J. Garcia-Luna-Aceves, "Securing distance-vector routing protocol," in *Global Internet'96*, Februray 1997.
[30] C. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (aodv) routing," http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt, February 2003.