ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Networks xxx (2004) xxx–xxx

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

BGP-RCN: improving BGP convergence through root cause notification ☆

Dan Pei [a,*], Matt Azuma [a], Dan Massey [b], Lixia Zhang [a]

[a] *Department of Computer Science, UCLA, Los Angeles, CA 90095, United States*
[b] *Department of Computer Science, Colorado State University, Fort Collins, CO 80523-1873, United States*

Received 2 April 2004; received in revised form 19 August 2004; accepted 28 September 2004

Responsible Editor: E. Ekici

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 as measured by the number of AS hops. BGP-RCN lets each routing update message carry 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 have been obsoleted by the same failure. The basic approach in BGP-RCN is applicable to path vector routing protocols in general. 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.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Routing; BGP; Routing protocol convergence

1. 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. However measurements in [3] showed that on average, it can take 3 min for the whole Internet to switch from failed

☆ 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.
* Corresponding author. Tel.: +1 310 825 4838.
E-mail addresses: peidan@cs.ucla.edu (D. Pei), matt_azuma@earthlink.net (M. Azuma), massey@cs.colostate.edu (D. Massey), lixia@cs.ucla.edu (L. Zhang).

1389-1286/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2004.09.008

routes to valid ones for a given destination. In some cases, it may even take up to 15 min for the routing tables in all routers to stabilize.

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 obsoleted by 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, however *MRAI* timer can also delay the propagation of *valid* reachability information [4–6]. Although a few recently proposed approaches, such as [7,6], can significantly reduce BGP convergence delays in many cases, they do not prevent all the invalid paths from propagating out, and are rendered ineffective under certain topological conditions.

This paper presents a new mechanism, called BGP with root cause notification (BGP-RCN), that provides an upper bound of O(d) on BGP convergence delay, where d is the network diameter as measured by the number of AS hops, a much tighter bound on the convergence delay compared to the existing results in literature. In BGP-RCN, 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, the root cause information is sufficient to enable v to identify all other routes that depend on the failed link. The basic approach in BGP-RCN is applicable to path vector routing protocols in general. 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. In the simulation of 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 s to 1.3 s, and reduces the number of BGP update messages from 30,483 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 s or less, and BGP-RCN reduces this time to 32 s.

The rest of the paper is organized as follows. Section 2 presents a protocol model for the standard BGP and summarizes the existing complexity results from literature. Section 3 presents the BGP-RCN algorithm and provides the complexity analysis for BGP-RCN. Section 4 discusses implementation and deployment issues. Section 5 presents the simulation results. Section 6 compares BGP-RCN with previous work. Finally, Section 7 concludes this paper.

2. A simple model for the standard BGP

In this section we define *simple path vector protocol* (SPVP), a simplified model of BGP. 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 later in Section 3.

2.1. The SPVP model

We model the Internet as a simple directed connected graph $G = (V, E)$, where $V = V_N \cup V_P$ and $E = E_N \cup E_P$. $V_N = \{0, 1, \ldots, n-1\}$ represents the set of n nodes that run SPVP protocol. A node in V_N corresponds roughly to an Internet AS. [1], and is not considered a destination in network G. Nodes in V_N are connected by links in E_N. V_P is the set of all the destination nodes (prefixes) in the network G. Without loss of generality, we consider only a single destination p which is connected to node 0 only (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, $0 \leqslant i \leqslant k$, and

[1] In practice, there can be multiple routers per AS and multiple physical links between two neighbor ASes. These issues will be discussed later in Section 4.

$v_0 = 0$. We say $v_i \in P, \forall i, 0 \leqslant i \leqslant k$; $[v_i v_{i-1}] \in P, \forall i, 1 \leqslant i \leqslant k$; 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. A node v selects its best route, denoted $rib(v)$, according to some routing policy (or ranking function). BGP allows arbitrary route selection policies, and some policies can lead to persistent route oscillation [8]. For clarity, the SPVP model considers only a shortest-path policy (which has been proven to converge [9]) unless specified otherwise. When two paths have the same length, the one received from the neighbor with a lower ID is preferred.

Nodes in V_N and links in E can fail and recover, and we assume that both nodes u and v can detect failure or recovery of link $[uv]$ within limited time. The failure or recovery of link $[0p]$ can also be detected by node 0 within limited time. After node v detects the failure of link $[uv]$, $rib\_in(v \Leftarrow u)$ is changed to ϵ, and after node v detects the recovery of link $[uv]$, node v sends its $rib(v)$ to node u. 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 each neighboring node.

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 M to denote its default value. According to [1] the MRAI timer is usually not applied to withdrawal messages.

2.2. SPVP convergence definitions

In [3,4,6,10], BGP routing events are categorized into four classes:

- T_{up}. A previously unavailable destination is announced as available.
- T_{down}. A previously available destination is announced as unavailable.
- T_{short}. An existing path is replaced by a more preferred path.
- T_{long}. An existing path is replaced by a less preferred path.

For clarity, our analysis and simulations focus on the impact of a *single* link failure event. Note that in our model, T_{down} happens when the link $[0p]$ fails, and T_{up} happens when node 0 detects that the $[0p]$ link has recovered from a previous failure. T_{long} events can be triggered by the failure of any link other than $[0p]$, and T_{up} can be triggered by the recovery of any link other than $[0p]$. [2]

Definition 1. *Converged state*. A node v is in a converged state iff $rib(v)$ will not change unless some new event occurs.

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. Multiple failures can be treated as multiple independent single failures, and details will be discussed later in Section 3.

The convergence time associated with an event is defined as follows:

Definition 2. *Network Convergence Delay*, denoted $time(T)$, starts when a triggering event T occurs and ends when all the nodes in the network are converged.

Labovitz et al. [4] have 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.

[2] In some extreme cases, after a T_{long} event triggered by one *single* link failure $[vu]$, the network can be partitioned into two parts. One part, say G_v, is disconnected to destination p, and the other part, say G_u, is still connected to destination p. In this case, the analysis and simulation will be equivalent to a T_{down} event in G_v where a destination p' is connected to node v. For clarity of presentation, we ignore such T_{long} event in the rest of the paper, and consider the topology where each node in E_N has at least two neighbors in E_N, a condition which guarantees the network is not partitioned by any *single* link failure.

30 s) and d is the network diagmeter. Labovitz et al. [4] and Obradovic [11] proved that the convergence of T_{down} is bounded by $M \cdot n$, where n is the number of nodes in the network. 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 $(|E_N| \cdot \frac{M \cdot n}{M}) = |E_N| \cdot n$. Our RCN design focuses on improving T_{down} and T_{long} convergence delay.

3. SPVP with root cause notification

In this section we describe the RCN algorithm in the context of SPVP, prove its correctness, and analyze SPVP-RCN's bounds of convergence delay and message count for T_{down} and T_{long} events. Then we consider issues related to multiple failures overlapping in time and Section 4 discusses implementation issues. Our correctness result hold for arbitrary (convergent) policies, but the analysis of delay bounds requires detailed modeling of the policy and our delay bound analysis is based on shortest path policies only.

3.1. SPVP-RCN algorithm

Because *SPVP* model has no periodic advertisements, an update message can be triggered only by a change in connectivity, that is a status change of some *link*. [3] When a link's status changes, the two nodes adjacent to the link will detect the change. For a given destination, at most one of the two adjacent node may change its route as a result. We call this node the *root cause node*. The root cause node attaches its ID to the update message(s) sent that result from this link status change, and this root cause information is copied into, and propagated along, all subsequent SPVP updates caused by the same link status change. Thus any node in the network can learn the unique root cause node which spawned the update messages it receives. Different from flooding used in link-state protocols (e.g. OSPF[12]), SPVP-RCN *piggybacks* the root cause in the routing updates, thus only the affected nodes and their direct neighbors are notified.

Because routing updates triggered by the same root cause may propagate along multiple paths at different speeds, one must be able to distinguish which update represents the *latest* link status changes. For example, after a link flapping (i.e. a link goes down and then comes up quickly), the link down notification might arrive at a node later than the path announced in the link up event. Therefore, when the link down notification arrives, the node can mistakenly remove a valid path that announced in the link up event. SPVP-RCN solves this problem by letting each node v maintain a sequence number, $t(v)$, which is incremented by 1 upon each change of node v's route to prefix p. In SPVP-RCN, a route is defined as $r = \{r.aspath, r.ts\}$, where $r.aspath$ is the SPVP aspath; $r.ts = \{ts(u) | u \in r.aspath\}$ is a list of sequence numbers, which correspond one-to-one with the nodes in $r.aspath$. Any change of $r.ts$, or $r.aspath$, or both should result in an increment of $t(v)$, and an update should be scheduled for each neighbor, subject to delay required by the MRAI timer. In SPVP-RCN, an update is defined as $update = \{update.r, update.rcn\}$, where $update.r$ is a route and $update.rcn = \{c, ts(c)\}$ represents the node ID and sequence number of the "root cause node".

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

- $x \in update.r.aspath$ and $update.r.ts(x) > seqnum\,(v, x)$, or
- $x = update.rcn.c$ and $update.rcn.ts(c) > seqnum\,(v, x)$

Note that $ts(x)$ in any routes, updates, root cause, or $seqnum(v, x)$ are just copies of $t(x)$, thus could be outdated. That is, we always have $t(x) \geqslant r.ts(x)$ for any

[3] The status of a link can change due to either a policy change or a physical failure that has occurred within an AS or between ASes. A node failure can be modeled by failing all its attached links. If two ASes are connected by multiple links, the failure of one of them may not result in a link status change in SPVP model. These issues are further discussed in Section 4.

routes r in *rib_in*, *rib* or *updates* in the network, and $t(x) \geqslant rcn.ts(x)$ for any *rcn* propagated in the network, and $t(x) \geqslant seqnum(v, x)$ for any node v.

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 we claim that this route will be withdrawn by u during the convergence. Therefore, node v can safely remove this route (replace it with ϵ). This allows v to rapidly remove obsolete routes, improving convergence time. We now prove this claim on the "correctness" of the SPVP-RCN algorithm.

Theorem 1. *If at time t, $x \in rib\_in(v \Leftarrow u).aspath$ and $rib\_in(v \Leftarrow u).ts(x) < seqnum(v, x)$, then u must send an update by the end of convergence that explicitly or implicitly withdraws the current $rib\_in(v \Leftarrow u)$.*

Proof. Let $P = rib\_in(v \Leftarrow u).aspath = (x_i x_{i-1} x_1 x \ldots 0)$ at time t, where $x_i = u$. Suppose at the end of the convergence, the route that v learned from u is $rib\_in'(v \Leftarrow u)$. If $rib\_in'(v \Leftarrow u).aspath \neq P$, clearly, $rib\_in(v \Leftarrow u) \neq rib\_in'(v \Leftarrow u)$, i.e., the $rib\_in(v \Leftarrow u)$ at time t is replaced by a route with a different aspath.

If $rib\_in'(v \Leftarrow u).aspath = P$, we claim $rib\_in(v \Leftarrow u).ts \neq rib\_in'(v \Leftarrow u).ts$ and thus $rib\_in(v \Leftarrow u)$ at time t is withdrawn after t. Since $rib\_in(v \Leftarrow u).ts(x) < seqnum(v, x)$, it is clear that node x has changed its route, and incremented $t(x)$ to at least $seqnum(v, x)$. Let $T(x)$ be the $t(x)$ value at the end of the convergence, and it is clear that $T(x) \geqslant seqnum(v, x) > rib\_in(v \Leftarrow u).ts(x)$. The SPVP-RCN algorithm requires that node x must send an update to x_1 with a $T(x) > rib\_in(v \Leftarrow u).ts(x)$. Note that SPVP-RCN algorithm requires that any change of $r.ts$, or $r.aspath$, or both result in an increment of a node's sequence number. Therefore after node x_1 receives the update from node x, it must change its own sequence number $ts(x_1)$, and send an update to x_2. Similarly, x_{j-1} must send an update to x_j, for all $1 \leqslant j \leqslant i-1$. For the same reason, node $u = x_i$ must receive an update from node x_{i-1}, increases its sequence number, and send an update to node v, resulting in $rib\_in'(v \Leftarrow u).ts \neq rib\_in(v \Leftarrow u).ts$. The claim is true, and so is the theorem. □

Note that Theorem 1 is true even when multiple failures overlap in time, guaranteeing a node only removes a path that will anyway be withdrawn later. Furthermore, above proof did not make any policy assumptions other than the route eventually converges and Theorem 1 is true for a general policy, provided this policy does not lead to persistent oscillation.

3.2. An example

Fig. 1 illustrates how SPVP-RCN works in T_{long}, and the algorithm operation is the same in T_{down}. Fig. 1(a) shows the 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 table is shown in the box, and initially all sequence numbers in the table are 0. In Fig. 1(b), the link between node 2 and node 0 fails, so node 2 chooses backup path $(1[0]0[0]p)$,

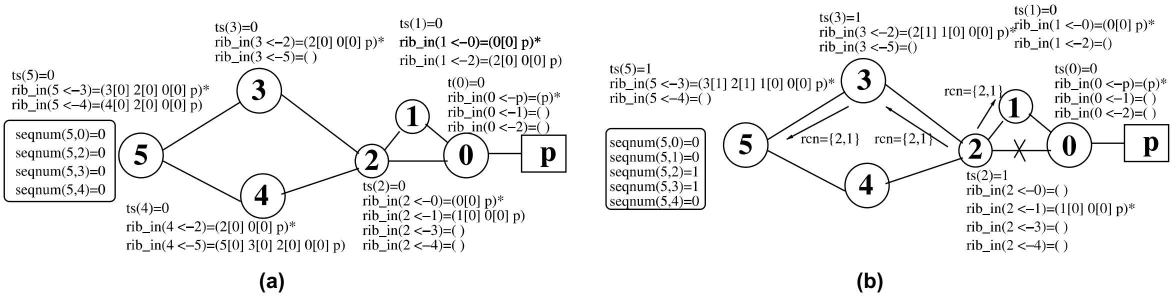


Fig. 1. SPVP-RCN Example For T_{long}: (a) before the link failure and (b) after node 5 receives the first update from node 3.

h	average nodal delay: the time it takes for a message to traverse one AS hop, including both processing delay and propagation delay
	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. Notations used in Section 3.

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 $rib\_in(5 \Leftarrow 4) = (4[0]\,2[0]\,0[0]\,p)$ and advertise this path to its neighbors. [4] 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 Fig. 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 $min(update(2,4),\ update(2,3) + update(3,5) + update(5,4))$.

[4] Here, we assumed this advertisement is sent out before node 4's update arrives at node 5.

3.3. SPVP-RCN T_{down} convergence bounds

We now consider the behavior of a single node v, and provide an upper bound on its convergence time. Fig. 2 summarizes the symbols used throughout this section. For each node v, define $rib(v)^{old}$ as v's route before the triggering event T, and $rib(v)^{new}$ as v's new route after v adapts to event T and returns to a converged state. Note that these convergence bounds apply to shortest-path policy only, and we mainly focus on the worst case.

Theorem 2. *If $conv(v)$ denotes the convergence time of node v after a T_{down} event, then $l \cdot d \leqslant conv(v) \leqslant \mu \cdot d$.*

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

$conv(v_{i-1})$, for $1 \leqslant i \leqslant n-1$. 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, $l(v_0, v_1) + conv(v_0) \leqslant conv(v_1) \leqslant \mu(v_0, v_1) + conv(v_0)$.

We now prove by induction and show that $\forall v_i \in V_N$, $\min_{0 \leqslant j<i}\{l(v_j, v_i) + conv(v_j)\} \leqslant conv(v_i) \leqslant \min_{0 \leqslant j<i}\{\mu(v_j, v_i) + conv(v_j)\}$. 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 $(v_0, \ldots, v_i)$. v_{i+1} will receive this first message no later than $\min_{0 \leqslant j<i+1}\{\mu(v_j, v_{i+1}) + conv(v_j)\}$ and no sooner than $\min_{0 \leqslant j<i+1}\{l(v_j, v_{i+1}) + conv(v_j)\}$. The hypothesis holds true and we have $l \cdot d(0, v) \leqslant conv(v) \leqslant \mu \cdot d(0, v) \leqslant \mu \cdot d$. □

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

It is clear that in the worst case, SPVP-RCN improves the convergence time from $h \cdot n$ to $h \cdot d$. Actually, given a fixed h, this is the theoretical ideal result of any path vector protocol, thus SPVP-RCN always improves the T_{down} convergence time.

Theorem 3. *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 at most *one* withdrawal message to each neighbor. Since the number of directed links is $|E_N|$, so the message overhead is bounded by $|E_N|$. □

3.4. SPVP-RCN T_{long} convergence

Theorem 4. *If $conv(v)$ denotes the convergence time of node v after a T_{long} event, then $conv(v) \leqslant d(2\mu + M)$.*

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 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 $(v_m \ldots v_0 s_k \ldots s_0)$ where $v = v_m$, $v_i \in V_{affected} (0 \leqslant i \leqslant m)$, $s_j \in V_{stable} (0 \leqslant j \leqslant k)$, and $s_0 = p$.

The T_{long} convergence of node v is divided into two stages as illustrated in Fig. 3. Node $c = c_0$ is the root cause node. During the first stage, node v_0 converges. By definition $rib^{new}(v_0) = (v_0 s_k \ldots s_0) = (v_0 \cdot rib\_in^{old}(v_0 \Leftarrow s_k))$. 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, $d(c_0, c_q) \cdot l = q \cdot l \leqslant conv(v_0) \leqslant d(c_0, v_0) \cdot \mu = q \cdot \mu \leqslant d\mu$.

The *second stage* starts when v_0 converges and ends when $v = v_m$ converges. After node v_0 converges, its path will be propagated along $v_1, \ldots v_{m-1}$ 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

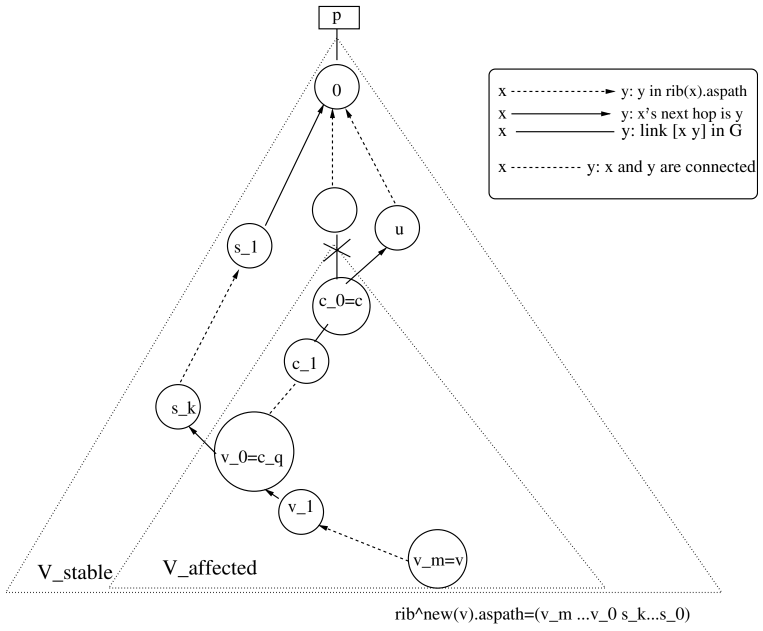


Fig. 3. Routing trees after T_{long} convergence.

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, $d(v_0, v_i) \cdot l = q \cdot l \leqslant conv(v) \leqslant d(v_0, v_m) \cdot (\mu + M) = m \cdot (\mu + M) \leqslant d(\mu + M)$ and thus node v's T_{long} convergence time is upper bounded at $(\mu \cdot d + (\mu + M)d) = (2\mu + M)d$. □

Similar to T_{down}, it is clear that in the worst case, SPVP-RCN improves the T_{long} convergence time and SPVP-RCN always improves the first stage of T_{long} convergence. However, SPVP-RCN might worsen the second stage of T_{long} convergence in some rare cases. We leave more detailed analysis on this topic for future work.

Theorem 5. *The message overhead for T_{long} convergence in SPVP-RCN is $|E_N|d(2\mu + M)/M$.*

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 $|E_N|d(2\mu + M)/M$ messages sent. □

Corollary 5.1. *If we model $\mu = l = h$, then SPVP-RCN's T_{long} convergence time's upper bound is $(2h + M)d$, message overhead is $|E_N|d(2h + M)/M$.*

3.5. Multiple failures overlapping in time

For clarity of presentation, so far we have described SPVP-RCN in the case of a single link failure. In a large network multiple failures may overlap in time. These overlapping failures can be treated as independent events, each with its own root cause. We leave as future work a more detailed analysis of the convergence delay for overlapping failures since it requires a detailed modeling of the timing of the overlapping failures. Instead, we focus on the correctness of SPVP-RCN, which has been proven in Theorem 1, and on how to maximize RCN's improvement in convergence delay in face of multiple failures, discussed below. Differences in update propagation delays could result in the following scenarios when there are multiple failures. 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, although $rib\_in(v \Leftarrow u)$ is invalidated, $ts(x)$ is indeed the latest information about node x thus $seqnum(v,x)$ is updated to the newer sequence number $ts(x)$.

Even with multiple root causes overlapping in time, a node v just copies the *rcn* of the incoming update that has triggered $rib(v)$ changes. However, it is possible that the $rcn.ts(c)$ in the incoming update $rib\_in(v \Leftarrow u)$ is smaller than its counterpart $seqnum(v,c)$. An outdated $rcn.ts(c)$ means there are still nodes in the network that have not learned latest sequence number of node c. Instead of further propagating information known to be obsolete, node v sets its outgoing root cause as $rcn = \{c, seqnum(v,c)\}$ to help quickly removing any paths that contain c and have a sequence number smaller than $seqnum(v,c)$.

3.6. Delay bounds with per-neighbor MRAI timer and WRATE

Up to this point, we have assumed that a distinct MRAI timer is kept for each (neighbor, prefix) pair, so that the first message sent by a node v during the convergence is not delayed by the MRAI timer at node v. The proof of Theorem 2 and the first stage of Theorem 4 have taken advantage of this assumption. As we mentioned earlier, however, in practice the MRAI timer is typically implemented on a per neighbor basis. When an update regarding prefix p turns on the per-neighbor timer, future updates regarding other prefixes p' may be delayed. Thus even the first message sent by a node v during the convergence can be delayed by up to M seconds. In this case, for T_{long}, SPVP-RCN's the upper bound on the convergence time of the first stage would increase to $d \cdot (h + M)$. The total T_{long} convergence would increase to $d \cdot (2h + 2M)$, and the upper bound for the total number of messages would increase to $|E_N|d2h+2M/M$. On the other hand, SPVP-RCN's upper bound for T_{down} remains unchanged because the MRAI timer does not apply to withdrawal messages, and all the messages sent during SPVP-RCN T_{down} convergence are withdrawal messages.

However 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 announcements [3–5]. In this scenario, the SPVP-RCN T_{down} convergence time would be bounded above by $(h + M) \cdot d$, but the message overhead is still bounded by $|E_N|$.

In practice routers might also use different MRAI values. In this case, we need to replace the M in our delay bound with the maximum *MRAI* value in the network. Regardless of the details in the MRAI timer implementation, the SPVP-RCN algorithm has the same time complexity of O(d), and only constant factors change with the per-neighbor MRAI timer, heterogeneous settings of MRAI value, and/or WRATE.

4. Implementation and deployment

Section 3 presented the basic design of SPVP-RCN, this section discusses implementation and deployment issues. The RCN approach is deployable in the current infrastructure. However, a design trade-off occurs in associating a sequence number with a node (as presented here) or associating a sequence number with a link and, due to space limitations, we first focus on this trade-off. We then address other issues such as transmission of the RCN, and handling the absence of an RCN (i.e. partial deployment), sequence number wrapping, and the protection of the sequence number.

In the SPVP-RCN design, we use node sequence number to represent the root cause. The node sequence number is incremented each time the AS path changes. One obvious alternative is to use link sequence number and increment the link sequence number each time the link status changes, as done in an independently developed approach, FESN (forwarding edge sequence number) [13]. In this case, the root cause can be represented by $\{c, b, ts([cb])\}$, where link $[cb]$ is the link that has changed the status. The link sequence number is then attached to the update message, and a path is considered obsolete if it contains the same link with a smaller sequence number. In the context of the SPVP, the effectiveness of these two implementations of the root cause are equivalent and share the same delay bounds. In most of the cases, these two approaches face the same implementation problems, although there exists some differences. We are currently investigating the trade-off of using node sequence number as opposed to link sequence number.

4.1. Storage overhead

Node sequence numbers require that, for each prefix p, node v maintain a table $seqnum(v,p,x)$ for all other nodes x. If s is the number of bytes used to store a sequence number, each node needs at most $s \cdot n$ bytes storage overhead per prefix and needs at most $s \cdot n \cdot |V_P|$ bytes storage overhead for all the prefixes in the network. Comparatively, SPVP-RCN with link sequence number requires that each node v locally store the sequence number associated with each link. As a result, the storage overhead with link sequence number is $s \cdot |E_N| \cdot |V_P|$, a much larger value than the node sequence number implementation.

In practice, however, the storage overhead may be reduced in both approaches. The sequence number for node x is stored at node v only when x has appeared in some path received by v, and the number of such x is typically less than the total number of nodes in the network. Furthermore, because BGP-RCN stops nodes from exploring many invalid paths currently tried in the standard BGP, this also helps reduce the number of nodes whose sequence numbers must be stored. Similar optimization can be used for link sequence number. But even with this optimization, the number of links will typically be much higher than the number of nodes and associating sequence numbers with nodes can achieve a dramatic reduction in storage overhead.

4.2. Logical elements vs. physical elements

For clarity in presentation, the SPVP-RCN design modeled each AS as a single node and each link as a single physical connection. In reality, an logical AS is a collection of routers and each logical link typically consists of multiple physical links. For example, SPVP models a large AS such as Sprint as a single node when in fact the Sprint AS consists of many routers spreading over a large geographic area. Similarly, the link between Sprint and AT&T is modeled as a single logical link when in fact Sprint and AT&T peer in many locations over many different physical links. Note also that this is not simply an over-simplication of SPVP, but the standard BGP protocol also represents the link between Sprint and AT&T as a single logical link. In implementing sequence numbers, we need to address the fact that both nodes (ASes) and links are not simple atomic entities and the sequence number implementation faces two choices.

The first option continues to associate the sequence number with the logical element. In order to implement logical node sequence numbers, the various routers that make up a node must agree on consistent sequence number choices. Similarly, in order to implement the logical link sequence number, the various physical links that make up a logical link must agree on a consistent sequence number for the link. In other words, both node sequence number and link sequence number implementations face a coordination problem. The former requires that routers within the same AS maintain a consistent sequence number, while the latter needs to maintain a consistent sequence number among multiple physical links between two neighboring ASes.

Alternatively, we can avoid coordination by changing the semantics of BGP to represent the more complex physical topology. In the node sequence number approach, we can maintain a sequence number for each border router (who announces routing updates to neighbor ASes), and only increases this sequence number when the border router's path changes. This can limit the effectiveness of RCN since the root cause (a border router and its sequence number) can only invalidate those obsolete paths that go through the same border router. In addition, each router's storage overhead increases to $s \cdot |V'_N| \cdot |V_P|$, where $|V'_N|$ is the total number of border routers in the network. In the link sequence number approach, we can maintain a sequence number for each *physical* link. Again this limits effectiveness since the root cause (physical link sequence number) can only invalidate those obsolete paths that have the same failed *physical* link. Each router's storage overhead increases to $s \cdot |E'_N| \cdot |V_P|$ where $|E'_N|$ is the total number of physical links in the network.

The use of border routers or physical links both change the semantics of BGP and expose a much more detailed view of the local topology to the global system. Exposing this *local* information may greatly increase the number of routing update messages in a *global* scale even when the actual AS path is not affected. Zhao et al. [14] observed an example where the use of a BGP path attribute inadvertently exposed physical link information and triggered a high volume of (unnecessary) BGP updates throughout the Internet. Overall, we believe a key principle is that local changes should be kept local and changing the semantics to expose local information is not a viable operational choice. As a result, we require that the routers within an AS coordinate to select new RCN sequence numbers (or equivalently, physical links connecting two AS coordinate to select new RCN sequence numbers). We are currently investigating the techniques such as timestamp to coordinate the sequence number.

4.3. Handling the absence of RCN

The discussions in Section 3 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.

4.4. Sequence number maintenance and security consideration

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 [15] 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 [16], when adding the sequence number to the origin AS. Furthermore, OSPF [12] is widely used and addresses sequence number problems; we primarily borrow techniques from this approach.

On the other hand, RCN can also help on BGP's security. One of the most important concern that previous BGP security solution such as

Secure-BGP [17] is the demanding CPU overhead needed for cryptography computation for each incoming and outgoing message. Our RCN has reduced the message overhead for T_{down} from $(|E_N|n/|E_N| = n)$ messages per peering session to $(1 - n/|E_N|)$ messages per session, and reduced the message overhead for T_{long} to $d(2h + 30)/30$ messages per session. In either case, the message overhead is reduced by orders of magnitude, and this will make cryptography based security solution such as Secure-BGP [17] more feasible for the Internet, which in return can help improve our RCN approach's security.

5. Simulation results

We used the SSFNET [18] simulator to conduct a comparative evaluation on both routing convergence and packet delivery. In addition to BGP-RCN we also simulated the standard BGP and two of the previously proposed BGP convergence speedup approaches, BGP-Assertion [7] and BGP-GF [6] which are described briefly in Section 6. SSFNET has a built-in BGP model, we added the implementation of BGP-RCN as well as the implementations of BGP-Assertion and BGP-GF according to [7,6], respectively. A third-party package [19] was incorporated and modified to enable tracing packet delivery in SSFNET.

5.1. 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 Fig. 4(a), are frequently used in literature [3,5,20,6] as a simple basis for T_{down} analysis and comparison. A B-Clique topology of size n, as shown in Fig. 4(b), consists of $2n$ nodes. Nodes $0,\ldots,n-1$ constitute a chain topology of size n, and nodes $n,\ldots,2n-1$ 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 connects to the well-connected Internet core (a Clique topology) through

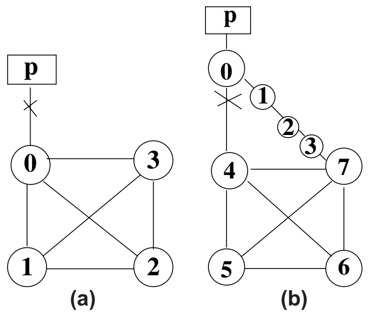


Fig. 4. Clique and B-Clique topologies: (a) Clique of size 4 and (b) B-Clique of size 4.

a direct link and a long backup path (the chain). We use B-Clique to study T_{long} convergence only. [5] To derive a simulation topology that resembles the Internet topology, we first generated a 110-node topology based on BGP routing tables by using the algorithm described in [21]. 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, four 28-node topologies, and eight 14-node topologies.

In all our simulations, the *MRAI* timer value was set to the BGP default of 30 s plus a random jitter. The link propagation delay was 2 ms, and the processing delay of each routing message was chosen randomly between 0.1 and 0.5 s. The bandwidth of each network interface was 10 Mbps.

To evaluate the performance of the standard BGP and three of its proposed variants, we measured not only routing convergence time and the number of routing update messages, but also data packet losses during routing convergence for T_{long} event. Pei et al. [22] shows that a short convergence time does not necessarily imply minimal packet losses and 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

[5] T_{down} result in B-Clique topology is similar to that in Clique topology.

source attached to each AS node in the network except the origin AS. Each data source generates 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 10 Mbps, there is no congestion-induced packet losses during simulation.

5.2. T_{down} simulation results

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. Fig. 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 every node in each topology. In total, there were $1 \times 110 \times 5 = 550$ runs for the 110-node topology, and $2 \times 55 \times 5 = 550$ runs for the two 55-node topologies. Fig. 6 shows the T_{down} convergence results averaged over 550 runs with a

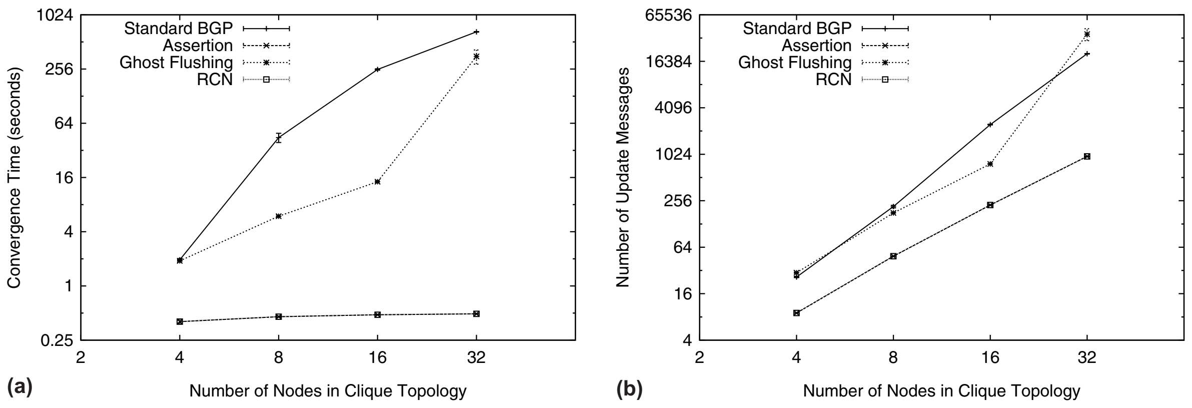


Fig. 5. Results for T_{down} convergence in Clique topologies: (a) convergence time(log–log) and (b) number of update messages(log–log).

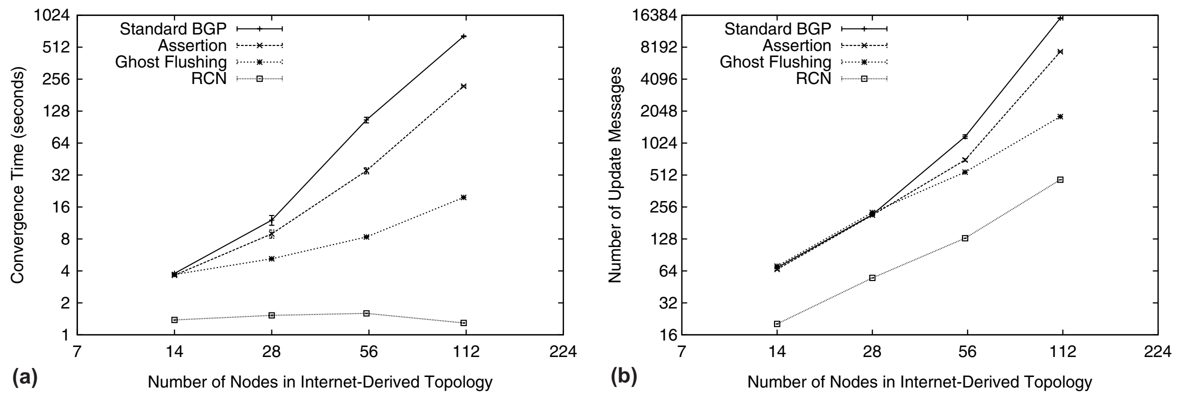


Fig. 6. Results for T_{down} convergence in Internet-derived topologies: (a) convergence time(log–log) and (b) number of update messages(log–log).

95% confidence interval for Internet-derived topologies. Note that both the *X*- and *Y*-axis are in log scale.

Simulation results show that, compared with the standard BGP, BGP-RCN can reduce T_{down} convergence time by 2–3 orders of magnitude, and reduce the total number of routing update messages by 1–2 orders of magnitude. For the 110-node Internet-derived topology, the T_{down} convergence time was reduced from 648.4 s to 1.3 s, and number of messages was reduced from 15,387 to 463. For Clique topologies of size 32, the convergence time was reduced from 662.1 s to 0.5 s, and the number of messages from 20533 to 961. BGP-RCN's improvement is consistent with our analysis in Section 3.

5.3. T_{long} simulation results

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 Fig. 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 1–2 orders of magnitude. For the B-Clique topology of size 32, the convergence time was reduced from 720.0 s to 11.3 s, and the number of messages from 22211 to 1955. This improvement is less dramatic than in the T_{down} case because BGP-RCN's convergence time

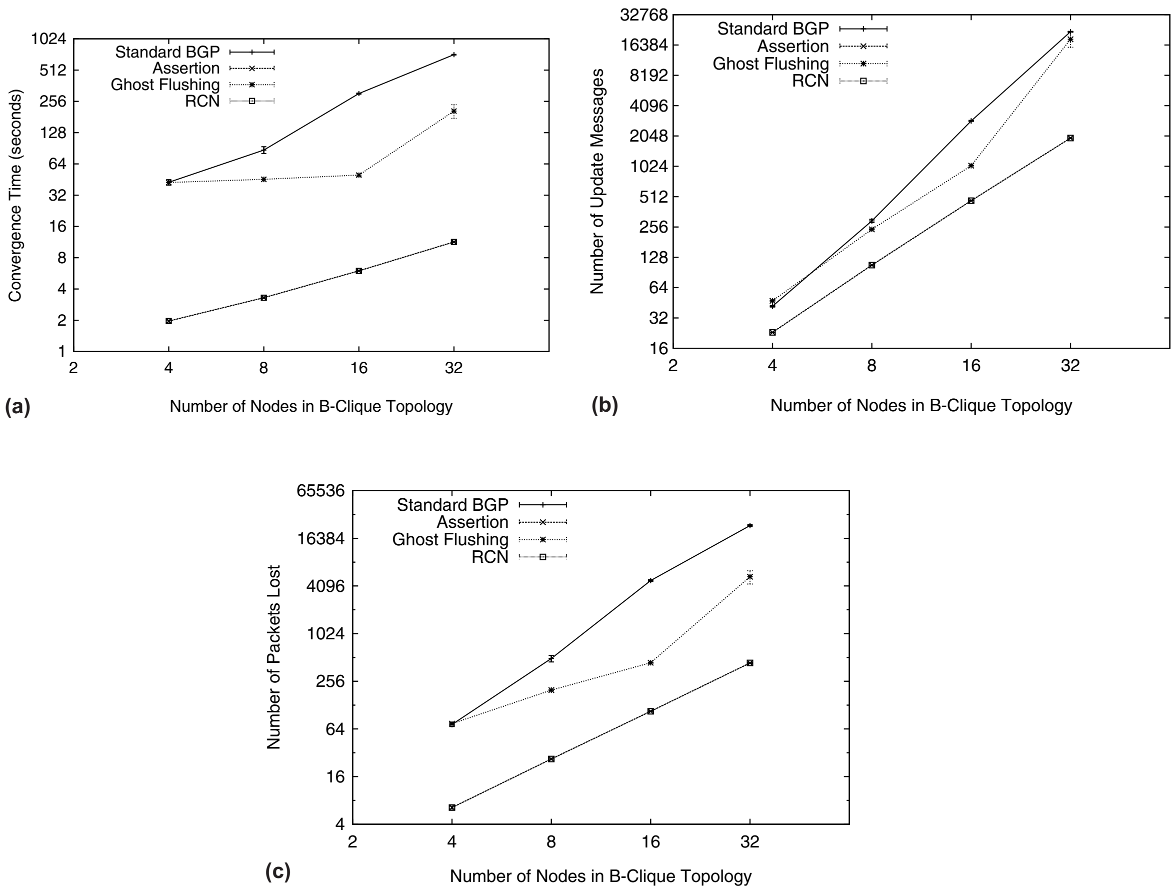
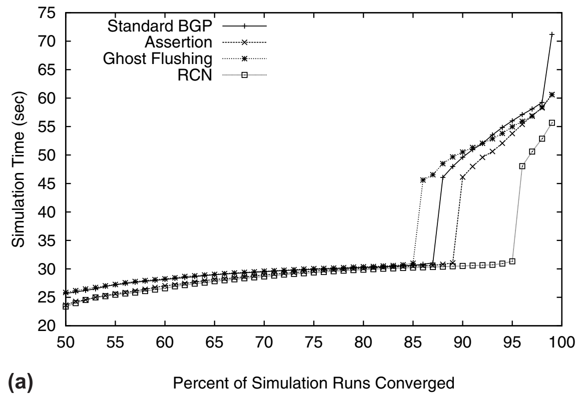
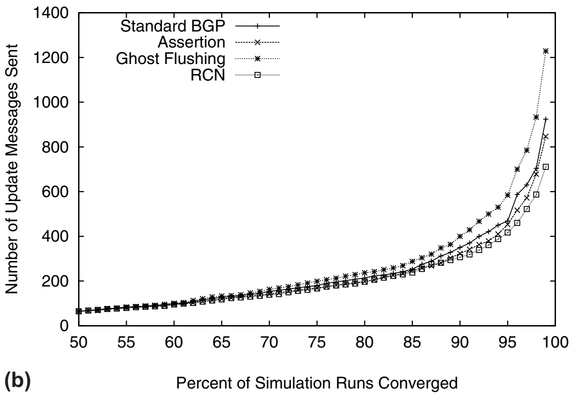


Fig. 7. Results for T_{long} convergence in B-Clique topologies: (a) convergence time(log–log); (b) number of update messages(log–log) and (c) number of packet losses(log–log).

upper bound is $d(2h + 30)$ s, a function of the MRAI timer value 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 losses by 1–2 orders of magnitude, from 23730 in standard BGP to 438 in BGP-RCN.

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 link was failed twice, thus $1 \times 2 \times 286 \times 5 = 2860$ simulation runs were conducted. 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 directly connect to a large number of 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, few nodes in the network need to readjust their routes. In addition, BGP slow convergence only occurs in the local region of affected nodes. Furthermore, when those affected nodes try to find the next best path to the same destination, due to the rich connectivity,





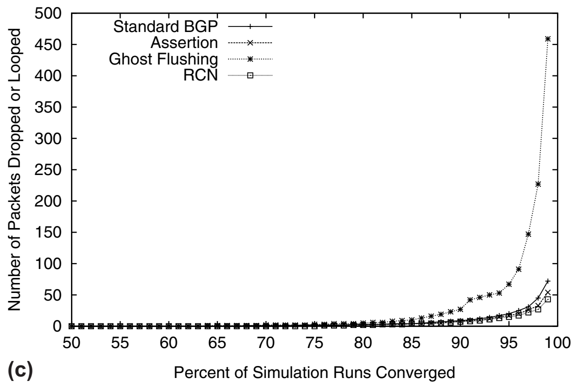


Fig. 8. Results for T_{long} convergence in Internet-derived topologies: (a) convergence time; (b) number of update messages and (c) number of packet losses.

not only do most of the nodes have multiple alternative routes to each destination, but also the newly selected path by the nodes have a good chance of not depending on the failed link, even without the knowledge of which link has failed.

Because of the above factors, the simulation results of the 110-node Internet-derived topology show that, under the standard BGP, more than 50% of our T_{long} simulation runs had short convergence delays and very small numbers of update messages. Therefore, instead of using the average, we use the percentile curve of T_{long} to show convergence results. A point (x, y) in Fig. 8(a) means that x% of the link-failures resulted in a convergence delay of no longer than y s. Fig. 8(a) shows that, for up to 85% of the links, all the four variants of BGP produce similar convergence delay. Above that point, however, the convergence delay reduction by BGP-RCN becomes more pronounced. For example, 95% of link failures converged within 56 s in the standard BGP, and BGP-RCN reduced this number to 32 s. 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.

6. Comparison with previous work

In this section, we compare our BGP-RCN design with other convergence improvement mechanisms through simulation. Fig. 9 summarizes the upper bound of the standard BGP, BGP-RCN, BGP-Assertion [7] and BGP-GF (Ghost Flushing [6]), as well as the empirical data for the parameters used.

6.1. Assertion approach

BGP-Assertion [7] tries to detect the path inconsistency between *rib_in* received from different neighbors, and if an inconsistency is found, the path learned from the direct neighbor is given higher priority in propagation and conflicting paths will also be marked infeasible. BGP-Assertion reduces the chance of choosing or propagating obsolete paths but does not eliminate the propagation of *all* obsolete paths, and its effectiveness is sensitive to the details of topology. For example, when BGP-Assertion is used in the topology shown in Fig. 1(b), after node 5 receives node 3's new update, it will choose the invalid backup path ($5420p$) and will further propagate the path to other neighbors. In BGP-RCN, node 3's update

Protocols	T_{down} Conv. Delay	T_{down} Messages	T_{long} Conv. Delay	T_{long} Messages
the standard BGP	$M \cdot n$	$\lvert E_N\rvert \cdot n$	N/A	N/A
BGP-Assertion[7]	N/A	N/A	N/A	N/A
BGP-GF[6]	$h \cdot n$	$2\lvert E_N\rvert n\frac{h}{M}$	N/A	N/A
BGP-RCN	$h \cdot d$	$\lvert E_N\rvert$	$d(2h+M)$	$\lvert E_N\rvert d\frac{(2h+M)}{M}$

(a)

Empirical data from [23,6]
$n \approx 15000$
$d \approx 11$
$\lvert E_N\rvert \approx 80000$
$h \approx 2$ *seconds*

(b)

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: (a) upper bound and (b) empirical data (from [23,6]).

message carries a root cause $\{2,1\}$ and invalidates all the paths that include node 2 and have $ts(2) < 1$, (e.g. $(4[0]2[0]0[0]p)$ is invalid).

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. In these specialized topologies, the BGP-Assertion algorithm allows node 0 (node 1)'s neighbors in $T_{down}(T_{long})$ to immediately converge. For T_{down} convergence in Internet-derived topologies, however, BGP-Assertion's performance is 2 orders of magnitude worse than BGP-RCN because BGP-RCN avoids 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.

6.2. BGP-GF: ghost flushing

In BGP-GF[6] a node sends a "flushing" withdrawal message when it changes to a less preferred path, but announcement of the less preferred path is delayed due to the MRAI timer. Without BGP-GF, the MRAI timer blocks announcement of the less preferred path and the obsolete path remains in the system until the MRAI timer expires. Because withdrawal messages are not subject to the MRAI timer delay, thus by sending a withdrawal an invalid path can potentially be quickly removed from the entire network. Like BGP-Assertion, BGP-GF does not eliminate the propagation of *all* invalid paths. For example, after the link [2 0] in Fig. 1(b) fails, node 2 sends an update and 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.

BGP-GF has an advantage that it does not change the format of the standard BGP message. According to [6], BGP-GF's T_{down} convergence time is bounded at $h \cdot n$ seconds and the message overhead is bounded at $2\,|E_N|nh/M$. No complexity results for T_{long} is provided in [6]. Because BGP-RCN eliminates all the invalidated paths, its convergence time is upper bounded 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 Fig. 9. In simulations, Fig. 6 shows that BGP-GF can reduce T_{down} convergence time by one order of magnitude in the Internet-derived topologies. Fig. 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. An invalid alternative path P can be propagated out by a node v before v receives the new updates that would invalidate P. Therefore, the improvement in convergence time at size 32 is much less than at size 16 (Fig. 5(a)). The additional processing overhead of withdrawal messages, which can mount to a large number in a dense topology, also reduces the improvement of T_{long} convergence in B-Clique of size 32 (Fig. 7), and even increases the T_{long} convergence time in Internet-derived topologies (Fig. 8).

While BGP-GF quickly removes invalidated paths, it does not necessarily speed up the propagation of alternative ones and it can cause significantly more packet losses than the standard BGP during T_{long} convergence in Internet-derived topologies (Fig. 8(c)). BGP-Assertion and BGP-RCN, on the other hand, both reduce packet losses. As observed in [22], minimizing routing convergence time in T_{down} does not necessarily lead to minimal packet losses in T_{long} convergence. 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 [24,20].

6.3. Other related work

Sequence numbers have been used before by [16] to improve the security of BGP. In [16], each origin AS maintains a sequence number for each prefix it originates, and increases the sequence number when it withdraws or re-announces the prefix. In the case of a withdrawal carrying a sequence number, this approach can be considered a sub-case of BGP-RCN. Approaches similar to [16] have also been proposed for distance vector protocols in [25] and AODV [26]. BGP-RCN lets every AS maintain sequence number for each prefix in the network, and improves both T_{long} convergence and T_{down} convergence. An independently developed approach, FESN (Forwarding Edge Sequence Number) [13], is similar to BGP-RCN but it uses link sequence number instead of node sequence number. We have compared the use of node sequence number and link sequence number in Section 4.

Variations on explicitly signaling a root cause have been proposed in [27–30]. Cheng et al. [27] considered distance vector routing and combined a path finding approach with a system for stamping the triggering link failure (e.g., the identifiers of the two nodes adjacent to the failed link) into each routing update. They provided T_{long} analysis but not T_{down}. [28] proposed explicitly signaling the T_{down} failure in BGP, and their approach can improve the T_{down} convergence time significantly, but is not applicable to T_{long}. "BGP-Cause Tag (BGP-CT)", outlined in two presentations [29,30], uses timing heuristic to deal with overlapping failures. Similar to BGP-RCN, BGP-CT explicitly signals the failure but does not use a sequence number. Any path containing the failed link is marked as "invalidated" and a timer is set. Invalidated paths are retained in the *rib_in*, but cannot be selected as the best path to destination. An invalidated path will either be replaced (by a withdrawal or new advertisement from a neighbor) or will again become available when the invalidation timer expires. The timer ensures a path that was incorrectly marked as invalid will eventually be eligible for use.

However, these approaches do not guarantee "correctness" in the case that multiple failures overlap in time. In all these approaches, link flapping (repeated link failure then link recovery) can cause correctness problems if a newer piece of notification arrives at a node before an older piece of path information. In the first two approaches, a node may mistakenly remove a valid path when an outdated piece of information arrives. BGP-CT's behavior depends on the timer settings. Generally, a path that is incorrectly marked invalid due to outdated information may be re-used when the timer expires. Note that BGP-RCN addresses the multiple failure issue by using a strictly increasing (with wrapping around handling) sequence number to signal the freshness of the root cause, and can safely remove all the paths invalidated by the latest RCN. The performance of BGP-CT depends on the timer settings. In the case of *single* failure, our T_{down} and T_{long} analytical results for BGP-RCN should be similar to BGP-CT with an ideal timer setting. In the case of multiple failures, the correctness and performance of overlapping events in BGP-CT would depend on the timer settings.

7. Conclusion

As evidenced by previous measurement and simulation studies [4,5,7,6], both the convergence time and message overhead of standard BGP can increase quickly as the network topology becomes larger in size and denser in connectivity. Labovitz et al. [4] and Obradovic [11] proved that standard BGP T_{down} 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. This represents a much tighter bound on BGP's convergence delay compared to the existing results in literature. After a link failure, the root cause information carried in each update message enables 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 can be reduced by at least 2 orders of magnitudes in both Clique and Internet-derived topologies. For T_{long} events, BGP-RCN eliminates all the invalid paths and propagates only valid reachability information. As a result, simulations on B-Clique topologies showed substantial reduction in the convergence time, the number of 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, but 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.

This paper focuses on the design of RCN and only compared with other approaches qualitatively through simulations. We are currently developing a general analytical model which can help fill the holes in Fig. 9(a). In particular, we believe that analytical results for T_{long} could be instructive in explaining why the standard BGP performs well and the proposed convergence enhancements offer only insignificant improvement in Internet-like topologies.

In addition to routing convergence improvements, we also believe that the root cause information carried in BGP-RCN can be potentially helpful in Internet routing diagnosis. 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 explore the use of root cause information in understanding global routing dynamics in our future efforts.

Acknowledgement

We are indebted to Eli Gafni for his invaluable suggestions and comments. We thank Nam Nguyen and Jiwei Chen for their contribution on earlier design and simulation of this work. We thank our editor and the anonymous reviewers for their detailed and insightful comments that greatly improved the paper and inspired our future work. We would also like to thank Songwu Lu and S. Felix Wu for helpful 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, T. Li, Border Gateway Protocol 4, Rfc 1771, SRI Network Information Center, July 1995.

[2] Y. Rekhter, T. Li, S. Hares, Border Gateway Protocol 4, Available from <http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp4-22.txt> October 2003.

[3] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, Delayed Internet routing convergence, in: Proceedings of ACM Sigcomm, 2000.

[4] C. Labovitz, R. Wattenhofer, S. Venkatachary, A. Ahuja, The impact of Internet policy and topology on delayed routing convergence, in: Proceedings of the IEEE INFOCOM, 2001.

[5] T. Griffin, B. Premore, An experimental analysis of BGP convergence time, in: Proceedings of ICNP, 2001.

[6] A. Bremler-Barr, Y. Afek, S. Schwarz, Improved BGP convergence via ghost flushing, in: Proceedings of the IEEE INFOCOM, 2003.

[7] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, F.S. Wu, L. Zhang, Improving BGP convergence through assertions approach, in: Proceedings of the IEEE INFOCOM, 2002.

[8] T. Griffin, F.B. Shepherd, G. Wilfong, The stable path problem and interdomain routing, IEEE/ACM Transactions on Networks 10 (2) (2002) 232–243.

[9] T. Griffin, G. Wilfong, A safe path vector protocol, in: Proceedings of IEEE INFOCOMM, 2000.

[10] D. Pei, M. Azuma, N. Nguyen, J. Chen, D. Massey, L. Zhang, BGP-RCN: improving bgp convergence through root cause notification, Technical Report TR-030047, UCLA CSD, Available from <http://www.cs.ucla.edu/peidan/bgp-rcn-tr.pdf> October 2003.

[11] D. Obradovic, Real-time model and convergence time of BGP, in: Proceedings of the IEEE INFOCOM, 2002.

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

[13] J. Chandrashekar, Z. Duan, Z.-L. Zhang, J. Krasky, Limiting path exploration in path vector protocols, Technical Report, University of Minnesota, 2003.

[14] X. Zhao, M. Lad, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, L. Zhang, An analysis BGP multiple origin AS(MOAS) conflicts, in: Proceedings of the IEEE DISCEX, 2003.

[15] R. Perlman, Network layer protocols with byzantine robustness, Ph.D. Thesis, MIT Laboratotry for Computer Science, 1988.

[16] B.R. Smith, J.J. Garcia-Luna-Aceves, Securing the border gateway routing protocol, in: Global Internet'96, 1996.

[17] S. Kent, C. Lynn, K. Seo, Secure border gateway protocol (s-bgp), IEEE JSAC Special Issue on Network Security.

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

[19] M. 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.

[20] Z. Mao, R. Govindan, G. Varghese, R. Katz, Route flap damping exacerbates Internet routing convergence, in: Proceedings of ACM Sigcomm, 2002.

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

[22] D. Pei, L. Wang, D. Massey, S.F. Wu, L. Zhang, A study of packet delivery performance during routing convergence, in: IEEE DSN, 2003.

[23] G. Huston, BGP Table Data, Available from <http://bgp.potaroo.net/>.

[24] C. Villamizar, R. Chandra, R. Govindan, BGP Route Damping, RFC 2439, SRI Network Information Center, May 1998.

[25] B.R. Smith, J.J. Garcia-Luna-Aceves, Securing distance-vector routing protocol, in: Global Internet'96, 1997.

[26] C. Perkins, E. M. Belding-Royer, S. R. Das, Ad hoc on-demand distance vector (AODV) routing, Available from <http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt> February 2003.

[27] C. Cheng, R. Riley, S. Kumar, J. Garcia-Lunes-Aceves, A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect, in: Proceedings of ACM Sigcomm, 1989, pp. 224–236.

[28] J. Luo, J. Xie, R. Hao, X. Li, An Approach to Accelerate Convergence for Path Vector Protocol, in: Proceedings of IEEE Globecom, 2002.

[29] C. Labovitz, A. Ahuja, Modeling inter-domain routing protocol dynamic, Avaiable from <http://www.caida.org/outreach/isma/0012/talks/labovitz/> December 2000.

[30] R. Wattenhofer, Slow internet routing convergence, Available from <http://www.inf.ethz.ch/schlude/webalgs/BGP-slides.pdf> December 2002.

Dan Pei is currently a Ph.D candidate at UCLA Computer Science Department. His current research interests include the fault tolerance and performance of Internet Routing Protocols. He received his Bachelor's and Master's degrees from Tsinghua University.

Matt Azuma received his Master degree from UCLA Computer Science Department in June 2004.

Dan Massey is an assistant professor at Computer Science Department of Colorado State University and is currently the principal investigator on DARPA and NSF funded research projects investigating techniques for improving the Internet's DNS and BGP infrastructures. He received his doctorate from UCLA and is a member of the IEEE, IEEE Communications Society, and IEEE Computer Society. His research interests include fault-tolerance and security for large scale network infrastructures.

Lixia Zhang (SM'95/ACM '84) received her Ph.D. degree from the Massachusetts Institute of Technology. She was a member of the research staff at the Xerox Palo Alto Research Center before joining the faculty of UCLA's Computer Science Department in 1995. In the past she has served on the Internet Architecture Board, Co-Chair of IEEE Communication Society Internet Technical Committee, Vice Chair of ACM SIGCOMM, and editor for the IEEE/ACM Transactions on Networking.