Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 398–421
Computer Networks
www.elsevier.com/locate/comnet

An analysis of convergence delay in path vector routing protocols ☆

Dan Pei [a,*], Beichuan Zhang [a], Daniel Massey [b], Lixia Zhang [a]

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

Received 23 November 2004; received in revised form 27 February 2005; accepted 18 April 2005
Available online 13 July 2005

Responsible Editor: L.G. Xue

Abstract

Path vector routing protocols such as the Border Gateway Protocol (BGP) are known to suffer from slow convergence following a change in the network topology or policy. Although a number of convergence enhancements have been proposed recently, there has been no general analytical framework to assess and compare the various proposed algorithms. In this paper we present such a general framework to analyze the upper bounds of path vector protocols' convergence delay under shortest path routing policy and single link failure. Our framework takes into account important factors such as network connectivity, failure location, and routing message processing delay. It can be used to analyze both standard BGP and *all* the proposed convergence improvement algorithms in the case of shortest path routing policy and single link failure. It enables us to obtain previously unavailable analytical results, including the delay bounds of path fail-over for standard BGP and its convergence enhancements. Our analysis shows that BGP fail-over delay bounds are mainly determined by two factors: (1) the distance between the failure location and the destination, and (2) the length of the longest alternate path to reach the destination after the failure. These two factors are captured formally by our analysis and can explain why existing convergence enhancements often provide only limited improvements in fail-over events. Moreover, explicitly modeling message processing delay reveals insights into the impacts of connectivity richness (i.e., node degree and total number of links in the network), and also the effectiveness of different enhancements. These new results enable one to better understand and compare the behavior of various path vector protocols under different topology structures, network sizes, and message delays.

☆ This work is partially supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. DABT63-00-C-1027 and by National Science Foundation (NSF) under Contract No. ANI-0221453. 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 or NSF.
* Corresponding author. Tel.: +1 310 825 4838.
E-mail addresses: peidan@cs.ucla.edu (D. Pei), bzhang@cs.ucla.edu (B. Zhang), massey@cs.colostate.edu (D. Massey), lixia@cs.ucla.edu (L. Zhang).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.04.013

Keywords: Routing; BGP; Path vector protocol; Routing protocol convergence

1. Introduction

In this paper, we analyze the convergence delays of path vector routing protocols. Once a change happens to the network connectivity, path vector protocols tend to explore a potentially large number of alternative paths before converging on new stable paths. This "slow convergence" problem has been observed on the Internet for the Border Gateway Protocol (BGP) [1–3]. In response, a number of enhancements [2,4–8] have been proposed to speed up BGP's convergence. However, there has been no thorough understanding of BGP and its enhancements' convergence behavior due to several reasons.

First, there is no general analytical model that applies to *all* the path vector protocol variants (i.e., standard BGP and its convergence improvement algorithms). Existing models rely on different assumptions. Though significant convergence speed-up has been demonstrated by some proposed solutions under *certain* conditions, the lack of a common analytical framework makes it difficult to judge the relative merit of each approach in general and make comparisons.

Second, existing analytical results are often incomplete. Particularly, in the case of path failover (i.e., the T_{long} event, where routers switch to less preferred paths), the analytical upper bounds of BGP and its enhancements' convergence delays are not available. As a result, although it is observed in simulations that various enhancements can shorten the T_{long} convergence delay only modestly [6–8], there has been no general explanation for why this should be the case.

Third, existing analysis did not take into account some important factors that may influence a path vector routing protocol's convergence behavior. The impacts of topology size and routing polices were examined in [3,9]. Other factors, including node degree, total number of links in the topology, transmission and processing delays of routing messages, and the locations of failures, have not received any systematic examination regarding their impacts on the routing convergence. Therefore, given a network setting, currently there is no easy way to judge whether the standard path vector protocol would perform adequately; and if not, which (if any) of the proposed solutions would perform best.

In this paper we develop a general framework for analyzing the worst case convergence delay bounds of path vector routing protocols. Under the assumption of shortest path policy and a single link failure, our framework applies to both standard BGP and all the existing convergence improvement algorithms. It takes into account important factors including network connectivity, failure location, and message processing delay. We first apply our general framework to the most commonly used U message processing delay model [2,3,6–9], which assumes that all routing messages are processed within a bounded time independent of network topology, and we present the results in Section 4. In Section 5 we apply the framework to a new message processing model, the Q model, which explicitly takes into account the message queuing delay at each node, and our results reveal important insights not available from the U model. Section 6 reports our simulation results from SSFNET [10], a well known BGP simulator where routing messages are processed through an FIFO queue at each router, which is consistent with the Q model.

1.1. Contribution of this work

Our results advance the analysis of path vector routing protocols in the following aspects:

First, our framework enables us to develop analytical bounds that were not previously available. We developed a two-process approach for T_{long} analysis (presented in Section 3) that allows us to obtain worst case bounds for all existing path vector algorithms under the assumption of shortest path policy and single link failure, enabling us to provide the T_{long} convergence delay bounds for standard BGP, Assertion [4], and Ghost Flushing

[6] for the first time. In addition, we also derived the T_{down} (i.e., the destination becomes unreachable) convergence delay bound for Assertion [4].

Second, our analysis shows that standard BGP's T_{long} delay bound is approximately $\mathscr{M} \cdot (nodediameter(G', 0) - J)$, where $\mathscr{M}$ is the *Minimal Route Advertisement Interval* (typically 30 s), J is the distance between the failure location and the destination, and $nodediameter(G', 0)$ is the length of the longest alternate path used to reach the destination after the failure. This is the first quantitative result of *failure location*'s impact on T_{long} convergence and also reveals the role of $nodediameter(G', 0)$ in determining T_{long} convergence delay. In a well-connected network such as today's Internet, the value of $nodediameter(G', 0)$ is relatively small (around 10 in the Internet [11]). This explains why various enhancement algorithms bring only modest improvement to the convergence delay of T_{long}. In addition, our analysis shows that the delay of T_{long}'s "counting-to-next-best-path" is much shorter than the delay of T_{down}'s "counting-to-infinity." This is in contrast to the previous perception that T_{down} and T_{long} have similar convergence delays [2], a conclusion based on experiments in which the value of $nodediameter(G', 0)$ was artificially exaggerated.

Third, our Q model is the first analytical model that takes into account the queuing delay of routing message.[1] Unlike the previous U model, the Q model reveals insights into the impacts of connectivity richness (node degree and total number of links in the network) and processing delay:

- The Q model provides a quantitative condition under which routing messages will not queue up at a node. The condition depends on the maximal time to process a single message (p_{max}) and the maximum number of messages that can be received by a node during one $\mathscr{M}$ period, which in turn depends on the in-degree of the node.
- The conventional U model cannot explain why some algorithms, including standard BGP and Ghost Flushing, perform differently in topologies of the same size but different connectivity. However, this can be explained by the analytical results based on the Q model since network connectivity plays a role in determining message queuing delay.
- Different protocols react differently to the increase of p_{max} and network connectivity. For example, Ghost Flushing generates additional BGP update messages to speed up routing convergence. Thus it performs much better than standard BGP when a network is sparsely connected and p_{max} is small, but much worse than standard BGP when the network is well connected and p_{max} is large.

[1] Note this paper considers the performance of routing algorithms. In this context, and throughout the rest of the paper, the term messages refers to routing messages.

2. Background, definitions, and algorithms

In this section, we present the *Simple Path Vector Protocol (SPVP)*, which represents a basic path vector routing protocol and corresponds to a simplified version of BGP on the Internet. We also provide convergence definitions that are used throughout the paper, and describe the various enhancements that have been proposed to improve SPVP's convergence time.

A network is modeled as a directed connected graph $G = (V, E)$. $V = \{0, 1, \ldots, N-1\}$ represents the set of N nodes that are connected by links in E and run the SPVP protocol. Without loss of generality, we consider only a single destination node p which is connected to node 0 and $p \notin V$. A path to destination p is an ordered sequence of nodes $r = (v_k, v_{k-1}, \ldots, v_0)$ such that $v_i \in V$ and link $[v_i \leftarrow v_{i-1}] \in E$ for all $i, 1 \leqslant i \leqslant k$, and $v_0 = 0$. By definition of r, $\forall i, 0 \leqslant i \leqslant k$, $v_i \in r$; $\forall i, 1 \leqslant i \leqslant k$, $[v_i \leftarrow v_{i-1}] \in r$; $\forall i, 0 \leqslant i \leqslant k-1$, $(v_i, v_{i-1}, \ldots, v_0) \subset r$. We define $length(r) = k$, and $length(\epsilon) = \infty$ for empty path. Because we consider a single destination node p in the network, any *path* mentioned in the rest of the paper means a path to destination p. This model roughly matches Internet BGP routing: nodes in V correspond to Internet Autonomous Systems and p corresponds to an IP prefix.

The following notations are used throughout the paper:

$indegree(G, v)$	in-degree of node v in G
$outdegree(G, v)$	out-degree of node v in G
$distance(G, v, u)$	shortest distance between v and u
$nodediameter(G, v)$	$= \max_{u \in G}\{distance(G, v, u)\}$
$diameter(G)$	$= \max_{v \in G}\{nodediameter(G, v)\}$

SPVP is a single path routing protocol in which each node advertises *only* its best path to its neighbor nodes. A node v stores the latest path received from each neighbor, selects the best path, $r(v)$, according to its routing policies and ranking functions, and advertises $r(v)$ to its neighbors. In theory, SPVP should be able to work with arbitrary routing policies, however previous studies showed that certain path selection policies can lead to persistent path oscillation [12]. As a first step in deriving a general framework for convergence bounds, this paper only considers shortest path policy[2] which has been proven to converge [13]. In the rest of the paper, we assume shortest path policy in all our analysis, and leave analysis of other policies to future work.

SPVP is an event-driven protocol; after the initial path announcement, further updates are sent *only* if the best path changes. During SPVP operations, links may fail and recover. Both nodes v and u can detect the failure and recovery of link $[v \leftarrow u]$, and node 0 can detect the failure and recovery of link $[0 \leftarrow p]$. If link $[v \leftarrow u]$ changes from *up* to *down*, node v removes the path received from neighbor u from its routing table. If link $[v \leftarrow u]$ changes from *down* to *up*, node u announces its best path to v. Upon detecting a link failure or receiving an update, each node recomputes the best path and sends updates if the best path changes. If link status changes or update messages result in no path to the destination, then $r(v) = \epsilon$ and a *withdrawal* message carrying ϵ as the path is sent to neighbors.

Like BGP, SPVP has a *Minimum Route Advertisement Interval* (*MRAI*) timer which guarantees that any two updates sent from v to u be separated by at least $\mathscr{M}$ seconds.[3] Following the BGP specification [1], the MRAI timer is not applied to withdrawal messages.

2.1. SPVP convergence definitions

Following [2,3,6,7], we categorize all routing events into four classes:

- T_{up}: a previously unavailable destination is announced.
- T_{short}: existing paths are replaced by more preferred paths.
- T_{long}: a link $[v \leftarrow u]$ fails and the nodes relying on this link switch to less preferred paths.
- T_{down}: a destination is no longer reachable and all nodes withdraw their paths to this prefix.

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

Definition 1. Converged State: a node v is in a converged state iff $r(v)$ will not change unless some new event occurs.

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.

Internet measurements [3] showed that in both T_{up} and T_{short} events, the convergence delay is roughly proportional to the network diameter. Convergence problems are commonly associated with T_{down} and T_{long} events [2,3]. For clarity, in the rest of this paper, our analysis and simulations focus on the impact of a *single link* failure event. Node failure analysis is not explicitly included here, but can be done by treating node failure as multiple simultaneous link failures.

In our model, a T_{down} event occurs when the link $[0 \leftarrow p]$ fails, and T_{up} occurs when node 0 detects that the $[0 \leftarrow p]$ link has recovered from a

[2] When two paths have the same length, the path from the neighbor with lower node ID is preferred.

[3] In theory, the MRAI timer is applied to per (neighbor, prefix). In reality, BGP's MRAI timer is often implemented on a per neighbor basis. Since we consider only one destination prefix in this paper, this implementation detail is not essential to our analysis.

previous failure. T_{long} events can be triggered by the failure of any link other than $[0 \leftarrow p]$, and T_{up} can be triggered by the recovery of any link other than $[0 \leftarrow p]$.[4] As is done in all the related work [2,3,6–9], we focus on the worst case upper bound of convergence delay.

2.2. SPVP convergence algorithms

This section reviews existing algorithm enhancements proposed to improve convergence time of SPVP. Due to space limitations, we focus on three representative algorithms: Assertion (SPVP-AS), Ghost Flushing (SPVP-GF), and Route Cause Notification (SPVP-RCN).

SPVP-AS: This algorithm [4] reduces the chance of choosing or propagating obsolete paths by checking path consistency when new updates are received. More specifically, assume that node v receives two paths, r and r', from two neighbors u and w respectively. SPVP-AS states that, if $u \in r'$, then it must be true that $r \subset r'$; otherwise, r' is regarded as invalid and removed. SPVP-AS does not eliminate the propagation of *all* invalid paths, and its effectiveness is sensitive to the topology.

SPVP-Ghost Flushing (*SPVP-GF*): In SPVP-GF [6], if node u changes to a path less preferred and u cannot send the new path to neighbor v immediately due to MRAI delay, u will send a withdrawal message immediately to remove (i.e., "flush out") the path previously advertised to v. Therefore, even though the new path announcement may be delayed by the MRAI timer, the invalid path is still quickly removed from the network. SPVP-GF does not eliminate the propagation of *all* the invalid paths; its effectiveness depends on topological details.

SPVP-RCN: In SPVP-RCN [7], each node maintains a sequence number and increments it by 1 whenever its best path changes. When an event happens, the node that detects the event attaches a *root cause*, defined as the combination of the node's ID and its current sequence number, to the routing update message. If this update message causes other routers to change their paths to the destination, they will send out update messages containing the original root cause information. If a routing event triggers node v to send an update with a root cause $(v, seqnum(v))$, any path containing v but with a sequence number smaller than $seqnum(v)$ is considered invalid and removed. Since every update carries the root cause, once a node receives the first routing message, it can immediately discard all the paths that are invalidated by the link failure.[5]

Other Algorithms: Other convergence algorithms include SSLD, WRATE, RCO, and FESN. In *Sender Side Loop Detection* (*SSLD*) [2], the sender v checks the path r before sending it to the receiver u. If $u \in r$, r will be discarded by u due to loop detection, therefore v will send a withdrawal instead. *Withdrawal rate limiting* (*WRATE*) requires that the MRAI timer be applied to withdrawal messages as well. *Route Change Origin* (*RCO*) is similar to RCN, but not applicable to T_{long}, thus it has the same T_{down} delay bound as RCN, and the same T_{long} delay bound as SPVP. *Forwarding Edge Sequence Number (FESN)* [8] is similar to RCN except that FESN uses link sequence numbers instead of node sequence numbers. FESN has the same T_{down} and T_{long} delay bound as RCN.

3. A framework for convergence analysis

In this section, we present a general framework for analyzing convergence time under the assumptions of shortest path policy and single link failure. We first divide path vector algorithms into two classes, *Implicit Topology-Change Notification* (*ITN*) algorithms and *Explicit Topology-Change Notification* (*ETN*) algorithms. We also develop path classification notations needed later in the

[4] In some extreme cases, after a T_{long} event triggered by one *single* link failure $[v \leftarrow u]$, 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 has at least two neighbors in E, a condition which guarantees the network is not partitioned by any *single* link failure.

[5] Note that, SPVP-RCN treats node failures as multiple simultaneous link failures, thus there would be multiple root causes, and a node can only discard the invalid paths invalidated by the root causes already received.

analysis. We use these classifications to establish a general framework for bounding T_{down} and T_{long} convergence times for arbitrary graphs and all existing path vector algorithms in the case of shortest path policy and single link failure. Our analysis focuses on upper bound (worst case) convergence, but we will show later in Section 6, the insights from delay upper bound helps one understand (average case) simulation results that are otherwise not easy to comprehend.

3.1. Algorithm classification

SPVP and the various enhancement algorithms can be classified as either ITN and ETN algorithms. In Implicit Topology-Change Notification (*ITN*) algorithms, topology changes are signaled implicitly by announcing a replacement path. For example, consider an ITN node v, and suppose that neighboring node v_2 announces that its previous path $(v_2, v_1, v_0 = 0)$ is being replaced by a longer path $(v_2, v_5, v_4, v_3, v_0 = 0)$. Under shortest path policy, the change to a longer path implicitly signals that either link $[v_2 \leftarrow v_1]$ or link $[v_1 \leftarrow v_0]$ has failed. Since link failure (or recovery) is signaled implicitly, a path change received from one neighbor has little or no impact on the validity of the paths received from other neighbors. In SPVP and SPVP-GF, a path r learned from neighbor u can only be invalidated if u withdraws r or advertises a replacement for r. In the above example, path $(v_2, v_1, v_0 = 0)$ will be invalidated only when v_2 explicitly withdraws this path by sending a withdrawal, or when v_2 implicitly withdraws this path by sending a replacement path (as in above example).

SPVP-AS is the only ITN algorithm that attempts to use implicit failure information to invalidate paths. In SPVP-AS, path r learned by v from node u may be invalidated using updates from nodes other than u, provided that one of the nodes in r is a direct neighbor to v and this node sends path information that conflicts with r. Continuing the above example, suppose node v also has neighbor v_6, and the path learned from v_6 is $(v_6, v_2, v_1, v_0 = 0)$ when v receives path $(v_2, v_5, v_4, v_3, v_0 = 0)$ from v_2. In SPVP-AS, node v will remove v_6's path since v_2's new path of $(v_2, v_5, v_4, v_3, v_0 = 0)$ conflicts with v_6's path $(v_6, v_2, v_1, v_0 = 0)$ (the sub-paths from v_2 to v_0 are different). Node v will not consider $(v_6, v_2, v_1, v_0 = 0)$ as potential alternate path and this may speed up convergence.[6] On the other hand, node v in SPVP and SPVP-GF will choose the invalid path $(v_6, v_2, v_1, v_0 = 0)$ as a "new" best path since it is shorter path than $(v_2, v_5, v_4, v_3, v_0 = 0)$, and further propagate this invalid path, increasing the convergence delay. SSLD and WRATE also belong to the ITN class.

In an Explicit Topology-Change Notification (*ETN*) algorithm, every update carries a tag that indicates which link failure (or recovery) triggered this update. SPVP-RCN refers to this link as the "root cause" for the updates. Once a node receives the *first* update during convergence period, it immediately knows the root cause of this routing event and is able to remove all invalid paths, regardless of from which neighbor the paths are received. SPVP-RCN, RCO, and FESN all belong to the ETN class of algorithms.

3.2. Path classification

After an event occurs, some paths may become *invalid*. We say a path is invalid *iff* it contains a failed link. For example, if link $[c \leftarrow b]$ fails then any path that contains link $[c \leftarrow b]$ is invalid. During the convergence period, a node that relies on an invalid path will eventually switch to an alternate path. But in some cases, a node may switch from one invalid path to another invalid path and a large number of invalid paths may be explored before the network finally converges. Algorithm specific rules determine which invalid paths may be (temporarily) explored during convergence. For example, an SPVP node will never explore an invalid path that contains itself (but will explore most of any other invalid paths) while an SPVP-RCN node will never explore any invalid path that contains a failed "root cause" link. Throughout the rest of the paper, we use $R_v(G, A)$ to denote the set of invalid paths that may be explored by node v in topology G under algorithm A. $R_v^l(G, A)$ denotes all invalid paths explored by node v with a length less than or equal to l.

[6] The actual SPVP-AS path selection approach is slightly more complicated due to timing issues and policy withdrawals.

After a node u changes its path, there is some delay before this information is propagated to u's neighbors. During this time period, we say the paths stored at u's neighbors are *obsolete*. More precisely, let u and v be neighboring nodes and let r be a path v learned from node u. We say that path r is obsolete *iff* node u no longer uses path r. Note that *obsolete* is distinct from *invalid*. A path is classified as invalid based solely on the network topology while a path is classified as obsolete based solely on consistency between node u and node v. An obsolete path is not necessarily invalid and an invalid path is not necessarily obsolete.

To analyze convergence, we are interested in the maximum time that may elapse before node v learns its path via neighbor u is *obsolete* and we let $\mathscr{D}(G,[v \leftarrow u])$ denote the upper bound on the time a path can remain obsolete. $\mathscr{D}(G,[v \leftarrow u])$ can include the MRAI delay, transmission delay, propagation delay, queuing delay, and processing delay. For example, suppose node u changes its path at time t_1. In SPVP, node u sends neighbor v an announcement listing the new path. The new path announcement may be delayed by the MRAI timer at u, then incurs some transmission, propagation and queuing delay before being accepted by the processor at v. Finally v takes some time to process the update and update its routing table at time t_2. By definition, $\mathscr{D}(G,[v \leftarrow u]) \geqslant t_2 - t_1$.

In above example, the announcement implicitly obsoletes u's old path and, at the same time, provides a replacement path. However, in the SPVP-GF algorithm there is a subtle but important distinction between the delay in learning a path is *obsolete* and the delay in learning a *replacement* path. An SPVP-GF node u that changes to a less preferred path and has its new path announcement blocked by the MRAI timer can immediately send a "flushing withdrawal". The withdrawal announces the previous path is now *obsolete* but does not announce the replacement path. When the MRAI timer later expires, node u will send an announcement listing the replacement path. In other words, SPVP-GF provides a fast mechanism for *obsoleting* old information and only later sends the replacement path. We use $\mathscr{D}_{\text{replace}}(G,[v \leftarrow u])$ to denote the upper bound on learning the *replacement* path. In algorithms such as SPVP, $\mathscr{D}_{\text{replace}}(G,[v \leftarrow u]) = \mathscr{D}(G,[v \leftarrow u])$. But in SPVP-GF (and future similar algorithms), one can have $\mathscr{D}_{\text{replace}}(G,[v \leftarrow u]) = \mathscr{D}(G,[v \leftarrow u]) + \mathscr{M}$.

$R_v(G,A)$: the set of all the *invalid* paths, starting at node v, in G allowed by algorithm A

$R_v^l(G,A) = \{r | r \in R_v(G,A) \wedge \mathit{length}(r) \leqslant l\}$

$\mathscr{D}(G,[v \leftarrow u])$: maximum time that may elapse between u changes its path and its neighbor v learns its previous path via u is *obsolete*

$\mathscr{D}_{\text{replace}}(G,[v \leftarrow u])$: maximum time that may elapse between u changes its path and its neighbor v learns u's replacement path

---

3.3. T_{down} analysis

In a T_{down} event, the destination is no longer reachable and network converges when all the nodes learn that the destination is unreachable. All (non-empty) paths to the destination are eventually flushed from the network, but intuitively shorter paths are flushed from the network more quickly. The following lemma captures the relationship between path length and the time required to remove a path in a T_{down} event:

Lemma 1. *Given any path* $r = (v_l, v_{l-1}, \ldots, v_0 = 0)$ *of length l that may occur during a* T_{down} *event, the path will be withdrawn by time* $f(r) = \sum_{i=1}^{l} \mathscr{D}(G, [v_i \leftarrow v_{i-1}])$ *and will never be restored.*

Proof. We prove this lemma by induction on l. Consider $l = 1$ and without loss of generality, let path $r = (v_1, v_0)$. At time 0, the failure occurs, v_0 withdraws its path and will never restore it. This information propagates to v_1 and has been processed by v_1 by the time $\mathscr{D}(G,[v_1 \leftarrow v_0])$. The path (v_1, v_0) will be withdrawn. Since a path of length 1 can only be learned from v_0, it will not be restored. Therefore the lemma is true for $l = 1$.

Assume lemma is true for any $r = (v_l, v_{l-1}, \ldots, v_0)$ and consider any path $r' = (v_{l+1}, v_l, v_{l-1}, \ldots, v_0)$. According to the induction hypothesis, v_l has withdrawn path r from its routing table by time $\sum_{i=1}^{l} \mathscr{D}(G,[v_i \leftarrow v_{i-1}])$ and sends a message x to its neighbors. Any earlier updates from v_l to v_{l+1} will have been overwritten by message x, and it takes at most $\mathscr{D}(G,[v_{l+1} \leftarrow v_l])$ for message x to be

processed by v_{l+1}. And v_l will never advertise r again according to the induction hypothesis. Therefore, the hypothesis is true for $l+1$. □

For any path $r=(v_l, v_{l-1}, \ldots, v_0)$ that may occur during a T_{down} event, we call $f(r)=\sum_{i=1}^{l}\mathscr{D}(G,[v_i \leftarrow v_{i-1}])$ the *lifetime* of path r. The lemma proves that after this lifetime, we can be certain the path has been withdrawn from the network and will not be restored later. Using this lifetime, we can derive T_{down} convergence bounds for both ITN and ETN path vector algorithms. We first consider ITN algorithms, including SPVP, SPVP-GF, and SPVP-AS.

Theorem 1. *For any network G and any ITN algorithm A,*
$time(T_{\text{down}}) \leqslant \max_{v\in V, r\in R_v(G,A)}\{f(r)\}$.

Proof. Note that $\max_{v\in V, r\in R_v(G,A)}\{f(r)\}$ is the maximum lifetime of any path in the network. According to Lemma 1, after the maximum lifetime, all paths in the network have been removed and will not be restored. In other words, all nodes must have concluded that the destination is unreachable and the network has converged. □

Explicit Topology-Change Notification (ETN) algorithms converges faster than ITN algorithms, because every message carries a root cause notification. Once the root cause is received, a node will be able to discard all invalid paths. Therefore, the network converges when all nodes receive at least one message.

Theorem 2. *For any network G and any ETN algorithm A,*
$time(T_{\text{down}}) \leqslant \max_{v\in V}\{\min_{r\in R_v(G,A)}\{f(r)\}\}$.

Proof. According to Lemma 1, node v has withdrawn one of its invalid paths by time of $\min_{r\in R_v(G,A)}\{f(r)\}$ (e.g. the result of receiving a message from its neighbor). Therefore v knows the root cause, immediately discards all other paths, and converges.[7] The maximum of this time over all nodes guarantees that all nodes are converged. □

[7] Again, note that this holds true for the convergence triggered by a single link failure, which we assumed in the paper.

3.4. T_{long} analysis

In T_{long} events, a link fails and some paths become invalid, but the destination is still reachable via some less preferred alternate paths. We say a node is *affected* if its path becomes invalid as a result of the failure. Let $[c \leftarrow b]$ denote the link that fails and let J be the distance from c to node 0. All invalid paths have the form $(v_l, \ldots, v_0, t_{J-1}, \ldots, t_0)$, where $v_0 = c$, $t_{J-1} = b$, and $t_0 = 0$. Nodes $t_i\,(0 \leqslant i \leqslant J-1)$ are not affected by the failure and nodes $v_i\,(0 \leqslant i \leqslant l)$ are affected nodes. The affected nodes form a single connected subgraph $G_A(V_A, E_A)$. Affected nodes need to discard invalid paths and converge to the new best paths. The following table summarizes the notations throughout T_{long} analysis and Fig. 1 illustrates the concepts:

$G'(V,E')$	Topology after an event T occurs in $G(V,E)$
V_S	Set of the nodes whose paths have not changed after T
V_A	Set of the nodes whose paths changed after T
E_A	Set of the links where both ends of the link belong to V_A
$G_A(V_A,E_A)$	Sub-graph of G' with V_A and E_A
$r^{\text{old}}(v)$	Node v's best path to destination p in G
$r^{\text{new}}(v)$	Node v's best path to destination p in G'
$[c \leftarrow b]$	The link that triggers the T_{long} convergence
J	$= distance(G,c,0)$, the distance from the failed link $[c \leftarrow b]$ to the destination

From Fig. 1 one can not only observe the subgraph of affected nodes, but also identify the characteristics of the new converged paths. The following lemmas formalize the observations from Fig. 1 and capture the relationship between path length and the time required to remove a path during a T_{long} event:

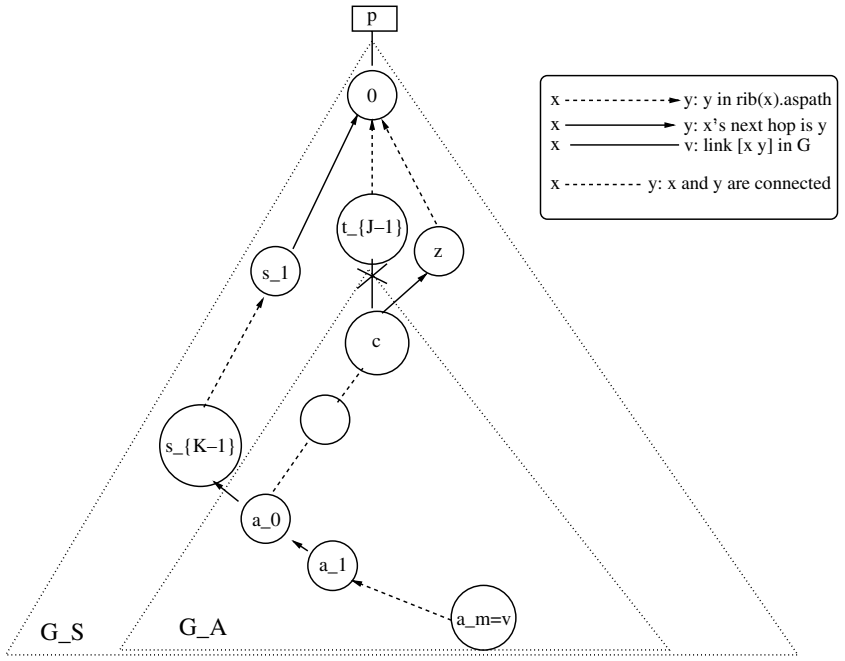


Fig. 1. Routing tree after T_{long} convergence.

Lemma 2. *After T_{long} convergence is complete, the new path of any $v \in V_A$ must have the form $r^{new}(v) = (a_m,\ldots,a_0, s_{K-1},\ldots,s_0)$, where $v = a_m$, $a_i \in V_A$ $(0 \leqslant i \leqslant m)$, $s_i \in V_S$ $(0 \leqslant i \leqslant K-1)$, $s_0 = 0$.*

Proof. Consider any link $[s \leftarrow w] \in r^{\text{new}}(v)$ where $s \in V_S$. Thus we have $r^{\text{new}}(s) = (s, r^{\text{new}}(w))$ and $r^{\text{old}}(s) = (s, r^{\text{old}}(w))$. It must be true that we also have $w \in V_S$. If w was not in V_S, $r^{\text{new}}(w) \neq r^{\text{old}}(w)$ (by definition of V_S) and thus $r^{\text{new}}(s) = (s, r^{\text{new}}(w)) \neq (s, r^{\text{old}}(w)) = r^{\text{old}}(s)$, contradicting the fact $s \in V_S$. □

Note that node c, who triggered the T_{long} convergence, might not necessarily be in $rib^{\text{new}}(v)$. Fig. 1 indeed gives one example in which node c is not in $rib^{\text{new}}(v = a_m)$.

Lemma 3. *During $T_{long}([\, c \leftarrow b\,])$, any invalid path $r = (v_l, v_{l-1},\ldots,v_0 = c, b = t_{J-1},\ldots,t_0 = 0)$ will be withdrawn by time $g(r) = \sum_{i=1}^{l} \mathscr{D}(G, [v_i \leftarrow v_{i-1}])$ and will never be restored later.*

Proof. The proof, similar to Lemma 1, is by induction on l. Consider $l = 1$ and without loss of generality, let path $r = (v_1, v_0 = c, b = t_{J-1},\ldots,t_0 = 0)$. At time 0, the failure occurs, v_0 withdraws this path r and will never restore it. This information propagates to v_1 and has been processed by v_1 by the time $\mathscr{D}(G, [v_1 \leftarrow v_0])$. The path $(v_1, v_0, t_{J-1},\ldots,t_0)$ will be withdrawn. Since an invalid path of length J can only be learned from v_0, it will not be restored and the lemma holds for $l = 1$.

Assume the lemma is true for any $r = (v_l, v_{l-1}, \ldots, v_0, b = t_{J-1},\ldots,t_0 = 0)$ and consider any path $r' = (v_{l+1}, v_l, v_{l-1},\ldots,v_0, t_{J-1},\ldots,t_0 = 0)$. According to the induction hypothesis, v_l has withdrawn path r from its routing table by time $\sum_{i=1}^{l} \mathscr{D}(G, [v_i \leftarrow v_{i-1}])$ and sends a message x to its neighbors. Any earlier updates from v_l to v_{l+1} will have been overwritten by x, and it takes at most $\mathscr{D}(G, [v_{l+1} \leftarrow v_l])$ for message x to be processed by v_{l+1}. v_l will also never advertise r again and the lemma holds for $l + 1$. □

For any invalid path $r = (v_l, v_{l-1}, \ldots, v_0, b = t_{J-1}, \ldots, t_0 = 0)$ that may occur during a T_{long} event, we call $g(r) = \sum_{i=1}^{l} \mathscr{D}(G, [v_i \leftarrow v_{i-1}])$ the *lifetime* of path $r = (v_l, v_{l-1}, \ldots, v_0, b = t_{J-1}, \ldots, t_0 = 0)$. [8] The lemma proves that after this lifetime, we can be certain the path has been withdrawn from the network and will not be restored later.

Using this lifetime, we can derive T_{long} convergence bounds for both ITN and ETN path vector algorithms. We first consider ITN algorithms, including SPVP, SPVP-GF, and SPVP-AS.

Consider the T_{long} convergence triggered by link $[c \leftarrow b]$ failure, illustrated in Fig. 1. According to Lemma 2, for any affected node v, its $r^{\text{new}}(v)$ must have the form $r^{\text{new}}(v) = (a_m, \ldots, a_0, s_{K-1}, \ldots, s_0)$, where $v = a_m$, $a_i \in V_A$ $(0 \leqslant i \leqslant m)$, $s_i \in V_S$ $(0 \leqslant i \leqslant K-1)$, $s_0 = 0$, where 0 is the node connected to the destination p. Therefore, we have the following theorem:

Theorem 3. *Given any network G and any ITN algorithm A, $time(T_{\text{long}}) \leqslant \max_{v \in V_A}\{\max\{wdw(v), ann(v)\}\}$ where*

$$r^{\text{new}}(v) = (a_m, \ldots, a_0, s_{K-1}, \ldots, s_0),$$
$$wdw(v) = \max_{r \in R_{a_m}^{K+m}(G,A)} \{g(r)\},$$
$$ann(v) = \max_{r \in R_{a_0}^{K}} \{g(r)\} + \sum_{i=1}^{m} \mathscr{D}_{\text{replace}}(G, [a_i \leftarrow a_{i-1}]).$$

Proof. In general, T_{long} convergence of node v consists of two processes, the withdrawal of invalid paths and the propagation of new valid paths. $wdw(v)$ is the time necessary for withdrawing invalid paths and $ann(v)$ is the time necessary for propagating new paths. The overall convergence time is the larger of the two times.

The length of $v(= a_m)$'s new best path is $K+m$. According to Lemma 3, all a_m's invalid paths shorter than $K+m$ have been withdrawn by $wdw(v) = \max_{r \in R_{a_m}^{K+m}(G,A)}\{g(r)\}$. After this time, all the shorter invalid paths are no longer available and v will select new best path as soon as it is learned by v.

The time required for the new path to become available consists of the time required for a_0 to establish its new path plus the time spent on propagation from a_0 to a_m. For a_0, the new path is from an unaffected neighbor, s_{K-1}, so it is already in a_0's routing table prior to the failure. Once any a_0's invalid paths with length less than K have been withdrawn, a_0 will converge to the new path, and this time is $\max_{r \in R_{a_0}^{K}(G,A)}\{g(r)\}$. For the new path to propagate from a_0 to a_m, it must "replace" any old paths along the way from a_0 to a_m and each hop can add delay up to $\mathscr{D}_{\text{replace}}(G, [v \leftarrow u])$. Therefore the total propagation time is $\sum_{i=1}^{m} \mathscr{D}_{\text{replace}}(G, [a_i \leftarrow a_{i-1}])$. Combining these two together, we have $ann(v) = \max_{r \in R_{a_0}^{K}}\{g(r)\} + \sum_{i=1}^{m} \mathscr{D}_{\text{replace}}(G, [a_i \leftarrow a_{i-1}])$. □

Explicit Topology-Change Notification (ETN) algorithms again behave differently compared to ITN algorithms. Every ETN message carries a root cause notification that allows a node to immediately discard any short invalid paths and the convergence depends only the announcement of the new best path.

Theorem 4. *Given any network G and any ETN algorithm A, $time(T_{\text{long}}) \leqslant \max_{v \in V}\{\min_{r \in R_{a_0}(G,A)}\{g(r)\} + \sum_{i=1}^{m} \mathscr{D}_{\text{replace}}(G, [a_i \leftarrow a_{i-1}])\}$ where $r^{new}(v) = (a_m, \ldots, a_0, s_{K-1}, \ldots, s_0)$.*

Proof. For ETN algorithms, the first announcement received by $v = a_m$ contains a root cause notification and any invalid path, regardless of length, is immediately discarded. To determine the convergence time, we only need to calculate when the new path arrives at $v = a_m$. Node a_0 (in Fig. 1) converges when it receives the first message by time $\min_{r \in R_{a_0}(G,A)}\{g(r)\}$. The new path then has a maximum propagation time of $\sum_{i=1}^{m} \mathscr{D}_{\text{replace}}(G, [a_i \leftarrow a_{i-1}])$ before reaching $v = a_m$. The overall network convergence time is obtained by simply taking the maximum value over all nodes. □

[8] Note that although the $g(r)$ in T_{long} and $f(r)$ in T_{down} appear to have the same formula $\sum_{i=1}^{l} \mathscr{D}(G, [v_i \leftarrow v_{i-1}])$, they are indeed different due to different representations of invalid path r. In T_{down}, an invalid path r is represented as $r = (v_l, v_{l-1}, \ldots, v_0 = 0)$, while in T_{long}, an invalid path r is represented as $r = (v_l, v_{l-1}, \ldots, v_0 = c, b = t_{J-1}, \ldots, t_0 = 0)$. Therefore, for the same r, the values of l in the formula $\sum_{i=1}^{l} \mathscr{D}(G, [v_i \leftarrow v_{i-1}])$ will be different for T_{down} and T_{long}, thus $f(r)$ and $g(r)$ will have different values.

3.5. Discussion on $\mathscr{D}(G,[v \leftarrow u])$

Having established the general framework for analyzing convergence bounds, we now consider specific delay models and produce algorithm specific results for the standard path vector routing algorithm and the various convergence enhancements. For any delay specific models, to obtain convergence time from the framework, we need to find $\mathscr{D}(G,[v \leftarrow u])$. Generally $\mathscr{D}(G,[v \leftarrow u])$ includes MRAI delay, transmission delay, link propagation delay, routing message queuing delay and processing delay.

The MRAI delay is bounded by the MRAI timer value, $\mathscr{M}$, usually configured with the default value of 30 s with a random jitter. For clarity, this paper assumes that MRAI timer is *exactly* $\mathscr{M}$ seconds without jitter; our results can be easily extended to consider a jittered MRAI timer. We assumed per (neighbor, prefix) based MRAI timer, thus the *first* message sent from u to a neighbor v is not constrained by the MRAI timer, and this has the following implications. For T_{down} events and ETN algorithms, all the messages are withdrawals, thus MRAI timer does not apply. For T_{long} events and ETN algorithms, node a_0 in Fig. 1 converges when it receives the first message (which always carries the root cause) by time $\min_{r \in R_{a_0}(G,A)}\{g(r)\}$ (in Theorem 4), and this process is not delayed by the MRAI timer because the *first* message between two neighbors is not delayed by the MRAI timer. But the propagation of new path from a_0 to a_m may be delayed by the MRAI timer since the first message sent by a_i to a_{i+1} might not be a_i's eventual best path, thus this update would turn on the MRAI timer, and in the worst case, will delay the propagation of $rib^{\text{new}}(a_i)$ from a_i to a_{i+1} by $\mathscr{M}$ seconds. For both T_{down} and T_{long} in ITN algorithms (other than SPVP-GF), suppose $(v_i, v_{i-1}, \ldots, v_0)$ is an invalid path during convergence. The *first* message sent by v_{i-1} to v_i turns on the MRAI timer, but does not necessarily withdraw the path $r' = (v_{i-1}, v_{i-2}, \ldots, v_0)$ since v_{i-1} might only learn this path r' some time later during the convergence. In the worst case, the removal of the path $(v_{i-1}, v_{i-2}, \ldots, v_0)$ can be delayed by $\mathscr{M}$ seconds at each hop because the MRAI timer is on.

Finally, we define $h(G,[v \leftarrow u])$ as the sum of all the delays in $\mathscr{D}(G,[v \leftarrow u])$ except the MRAI delay. The U model and Q model differ in the modeling of $h(G,[v \leftarrow u])$. In U model, $h(G,[v \leftarrow u])$ is a network-wide fixed number h; in Q model, $h(G,[v \leftarrow u])$ considers routing message queuing delay and is a function of node v's in-degree. We first present the U model and its results in the next section.

4. U model and results

In this section, we discuss the U model and its results. The U model, commonly used in the literature [2,3,6–9], assumes that all routing messages are processed within a bounded time, independent of network topology. In other words, the U model assigns the same network-wide fixed upper bound, h, for all $h(G,[v \leftarrow u])$, defined at the end of Section 3. Therefore, depending on the algorithm, either $\mathscr{D} = \mathscr{M} + h$ or $\mathscr{D} = h$, regardless of the topology and node v. In this section, we provide T_{down} and T_{long} convergence time bound under U model, and the results are summarized in Figs. 2 and 3.

$h(G,[v \leftarrow u])$	Sum of all delays in $\mathscr{D}(G,[v \leftarrow u])$ except the MRAI delay
h	A network-wide fixed upper bound of all $h(G,[v \leftarrow u])$ in the network
$\mathscr{M}$	Minimum Route Advertisement Interval
$\mathscr{D}$	The network-wide value of $\mathscr{D}(G,[v \leftarrow u])$ in U model. It equals to either h or $\mathscr{M} + h$
$\mathscr{D}_{\text{replace}}$	The network-wide value of $\mathscr{D}_{\text{replace}}(G,[v \leftarrow u])$ in U model. It equals to $\mathscr{M} + h$

4.1. T_{down} results

Applying the delay models to Theorems 1 and 2, we obtained the following T_{down} delay bounds under U model in Corollaries 1 and 2.

A	$time(T_{down})$
SPVP [3]	$(N-1)\cdot \quad (\mathcal{M}+h)$
*SPVP-AS	$(N-outdegree(G,0))\cdot \quad (\mathcal{M}+h)$
SPVP-GF [6]	$(N-1)\cdot \quad h$
SPVP-RCN [7]	$nodediameter(G,0)\cdot \quad h$

Fig. 2. T_{down} convergence results under U model. The SPVP-AS result was previously unavailable.

Corollary 1. *For any network G and any ITN algorithm A, under U model,*

$$time(T_{\text{down}}) \leqslant \mathscr{D} \cdot \max_{v\in V, r\in R_v(G,A)}\{length(r)\}.$$

Proof. Since $\mathscr{D}(G,[v \leftarrow u]) = \mathscr{D}$, a path r's lifetime becomes $f(r) = \mathscr{D}\cdot length(r)$. The corollary directly follows Theorem 1. □

Considering all possible topologies, the longest path at most can include every node once, therefore $\max_{r\in R(G,A)}\{length(r)\} = N-1$. Different from SPVP and SPVP-GF, SPVP-AS has an additional constraint. Before the failure, node 0's direct neighbor v has a direct path $(v,0)$. During the convergence, the first message v that results in v's path change is a withdrawal from node 0 (because all other paths are longer than $(v,0)$). As a result of assertion checking, v will never choose nor propagate any path containing node 0's other direct neighbors. Therefore, any invalid path during T_{down} convergence can have at most one of node 0's direct neighbors.[9] Thus, for SPVP-AS, $\max_{r\in R(G,A)}\{length(r)\} = N - outdegree(G,0)$. For SPVP and SPVP-AS, $\mathscr{D} = \mathscr{M}+h$, while SPVP-GF has $\mathscr{D} = h$ because the "flushing" withdrawals are not delayed by the MRAI timer.

Corollary 2. *For any network G and any ETN algorithm A, under U model,*

$$time(T_{\text{down}}) \leqslant h\cdot nodediameter(G,0).$$

[9] Similarly, in T_{long} convergence of SPVP-AS, any invalid path can have at most one of node c's direct neighbors in V_A.

Proof. For ETN algorithms, the first update is a withdrawal and all subsequent updates are also withdrawals. Therefore, the MRAI timer does not apply and $\mathscr{D} = h$. By definition and from Theorem 2, $\max_{v\in V}\{\min_{r\in R_v(G,A)}\{length(r)\}\} = nodediameter(G,0)$. □

4.2. T_{long} results

For T_{long} events, the lifetime of r under U model is $g(r) = \mathscr{D}\cdot(length(r)-J)$ when the failure link $[c \leftarrow b]$ is J hops away from node 0.

First consider ITN algorithms. $\mathscr{D} = \mathscr{M}+h$ for SPVP and SPVP-AS and $\mathscr{D} = h$ for SPVP-GF. $wdw(v)$ in Theorem 3 becomes $\mathscr{D}\cdot\min\{K+m-J, \max_{r\in R_{am}(G,A)}\{length(r)\}-J\}$. In SPVP and SPVP-GF, $\max_{r\in R_{am}(G,A)}\{length(r)\}-J = |V_A|-1$, while in SPVP-AS, it is $|V_A| - outdegree(G_A,c)-1$. Similarly, the first half of the $ann(v)$ in Theorem 3, $\max_{r\in R^K_{a_0}(G,A)}\{g(r)\}$, equals to $\mathscr{D}\cdot\min\{K-J,|V_A|-1\}$ for SPVP and SPVP-GF, and $\mathscr{D}\cdot\min\{K-J,|V_A| - outdegree(G_A,c)-1\}$ for SPVP-AS. And $\sum_{i=1}^{m}\mathscr{D}_{\text{replace}}(G,[a_i \leftarrow a_{i-1}) = (\mathscr{M}+h)\cdot m$ for all ITN algorithms. We obtained the results in Fig. 3 by summing these terms and taking the upper bound over all nodes according to Theorem 3. Note that $nodediameter(G',0)$ is the upper bound for $K+m$, and $diameter(G_A)$ for m.

U model results in Fig. 3 show that SPVP T_{long} is (more loosely) bounded by $(M+h)\cdot(nodediameter(G',0)-J)$. Later in Section 6, we show that this (looser) bound provides important insights into the modest improvements seen in simulated T_{long} convergence.

For ETN algorithms, the similar procedure can be repeated, with $\mathscr{D} = h$ and $\mathscr{D}_{\text{replace}} = \mathscr{M}+h$. Note $diameter(G_A)$ is the upper bound of both $distance(G,a_0,c)$ and m.

5. Q model and results

The section presents our Q model, which is the first analytical model that explicitly takes into account the routing message queuing delay at each node, and reveals important insights not previously obtained by the U model.

A	T_{long} results under U model
*SPVP	$(\mathcal{M}+h)\cdot min\{nodediameter(G',0)-J,\lvert V_A\rvert+diameter(G_A)-1\}$
*SPVP-AS	$(\mathcal{M}+h)\cdot min\{nodediameter(G',0)-J,\lvert V_A\rvert+diameter(G_A)-outdegree(G_A,c)\}$
*SPVP-GF	$\mathcal{M}\cdot diameter(G_A)+h\cdot min\{nodediameter(G',0)-J,\lvert V_A\rvert+diameter(G_A)-1\}$
SPVP-RCN[7]	$(\mathcal{M}+2h)\cdot diameter(G_A)$

Fig. 3. T_{long} results under U model. The results for SPVP, SPVP-AS and SPVP-GF were not available before.

The limitation of U model is that it uses the same $h(G,[v \leftarrow u])$ for all nodes, but in fact different nodes may have different $h(G,[v \leftarrow u])$. The U model not only gives coarse estimate of the convergence time, but also fails to reveal important relationships between the convergence time and the network topology. This section introduces the Q model, which incorporates a queuing delay estimate into $h(G,[v \leftarrow u])$ and better reflects BGP implementations. With the Q model, we can obtain tighter bounds on convergence time and new insights into the impact of richness of connectivity and processing delay.

5.1. Queueing delay

ld	Upper bound of the sum of transmission and propagation delay on any link in the network
$p_{\max}$	Maximum message processing time at any node in the network
$h(G,[v \leftarrow u])$	Sum of ld, queuing delay and message processing time when message is propagated from u to v

The Q model uses ld to denote the *network-wide* upper bound on the sum of link delay, transmission delay, and any delay due to retransmitting lost packets. In other words, an update sent by node u will be received by node v within time ld. Note that in reality different links might have different ld values, and in this paper we assume that ld is a network-wide bound in order to simplify our analysis. The Q model assumes a node v processes update messages in FIFO order. If a message arrives while the processor is occupied, the message is placed in an FIFO queue. The queuing delay depends on the number of messages in the FIFO queue at the moment a message arrives. Once the message gets to the processor, it will be fully processed in $[p_{\min},p_{\max}]$ seconds. Again, we assume that $p_{\min}$ and $p_{\max}$ are network-wide bounds in order to simplify our analysis. Thus $h(G,[v \leftarrow u])$ equals to the sum of ld, maximal queuing delay, and maximum processing delay ($p_{\max}$) at node v. Because all nodes in the network have the same ld values and the same $p_{\max}$ values, our model $h(G,[v \leftarrow u])$ is mainly to model the message queuing delay.

If the number of messages that arrive at a node during some interval exceed the number of messages the node can process within the same interval, the node's queue size will increase. If messages keep queuing up at a node, the convergence delay can be very long [14]. In the following, we first derive the quantitative conditions under which messages will not queue up at a node. The MRAI timer (see Section 2) ensures that two *announcements sent* by node u to v must be separated by at least $\mathcal{M}$ seconds. Since withdrawal messages are not restricted by the MRAI timer and our algorithms do not send duplicate updates, during any period of $\mathcal{M}$ seconds, the most updates u can send to v is a sequence of *withdrawal, announcement*, *withdrawal*. That is, we have the following assumption:

Assumption 1. During any $\mathcal{M}$ second interval, node u can *send* at most 3 updates to node v.

More detailed explanation for this assumption is provided in Appendix 1 of this paper. Based on this assumption, we have the following corollary:

Corollary 3. *During any* $(\mathscr{M} - ld)$ *interval, node* v *can receive at most* 3 *updates from node* u.

Proof. Consider any sequence of four updates from u to v, assume the first one is sent at time t_1, received at t_1', and the last one is sent at t_4, received at t_4'. Assumption 1 ensures that $t_4 - t_1 > \mathscr{M}$, and since the link delay is between $(0, ld]$, we have $t_4 < t_4' \leqslant t_4 + ld$ and $t_1 < t_1' \leqslant t_1 + ld$. Therefore, $t_4' - t_1' > \mathscr{M} - ld$. □

This corollary allows us to obtain a bound $h(G, [v \leftarrow u])$.

Lemma 4. *In the* Q *model, if* $\mathscr{M} - ld > 3 \cdot indegree(G, v) \cdot p_{\max}$, *then at any moment* t, *there are at most* $3 \cdot indegree(G, v)$ *messages in* v*'s queue.*

Proof. For a base case, at time $t = 0$, the queue starts with no messages. During the first $(\mathscr{M} - ld)$ seconds, at most three messages can be received from each neighbor according to Corollary 3.

Suppose the Lemma is true for time period $[0, i \cdot (\mathscr{M} - ld))$, $i = 1, 2, 3, \ldots$, we examine the queue at any moment t in time interval $[i \cdot (\mathscr{M} - ld), (i+1) \cdot (\mathscr{M} - ld)]$. At time $t' = t - (\mathscr{M} - ld)$, there are at most $3 \cdot indegree(G, v)$ messages in the queue since t' falls in $[0, i \cdot (\mathscr{M} - ld))$. All these messages are processed within $3 \cdot indegree(G, v) \cdot p_{\max} < \mathscr{M} - ld$ seconds, therefore by time t, they have all left the queue. The number of messages that can arrive within $[t', t]$ is no more than $3 \cdot indegree(G, v)$, thus the hypothesis holds for $(i + 1)$. □

Theorem 5. *In the* Q *model, if* $\mathscr{M} - ld > 3 \cdot indegree(G, v) \cdot p_{\max}$, *then* $h(G, [v \leftarrow u]) \leqslant 3 \cdot indegree(G, v) \cdot p_{\max} + ld$.

Proof. Follows directly from Lemma 4. □

Lemma 4 offers the first quantitative conditions under which messages will not queue up at a node. Using this condition, we note that $\mathscr{M} > 3 \cdot indegree(G, v) \cdot p_{\max} + ld$ is a sufficient condition to provide an upper bound for $h(G, [v \leftarrow u])$ and we assume this condition is true in the rest of this section.[10]

5.2. Delay bounds under Q model

Because ld and $p_{\max}$ are the same for all the nodes in a given network G, $h(G, [v \leftarrow u]) = ld + 3 \cdot p_{\max} \cdot indegree(G, v)$ becomes a function of $indegree(G, v)$. In contrast, the fixed number h in the previous U model is the network-wide bound for $h(G, [v \leftarrow u])$, and it equals the largest $h(G, [v \leftarrow u])$ among all possible nodes v in G, i.e., $\max_{v \in V} h(G, [v \leftarrow u]) = ld + 3 \cdot p_{\max} \cdot \max_{v \in V} indegree(G, v)$. Therefore, the Q model provides tighter bound for each convergence algorithm we have studied, and more insights into how topology affects convergence delay.

Theorem 1 shows that the T_{down} convergence time of ITN algorithms is $time(T_{\text{down}}) \leqslant \max_{v \in G, r \in R_v(G,A)} \{f(r)\}$. Under Q model, the lifetime of path $r = (v_l, v_{l-1}, \ldots, v_0)$ is $f(r) = \sum_{i=1}^{l} (\mathscr{M} + ld + 3p_{\max} \cdot indegree(G, v_i))$ for SPVP and SPVP-AS, and $f(r) = \sum_{i=1}^{l} (ld + 3p_{\max} \cdot indegree(G, v_i))$ for SPVP-GF.

Since SPVP and SPVP-GF do not restrict $R_v(G, A)$ (Section 3), in the worst case an invalid path can include every node. Therefore, for SPVP, $time(T_{\text{down}}) \leqslant \sum_{i=1}^{N-1} (\mathscr{M} + ld + 3p_{\max} \cdot indegree(G, i)) = (N-1) \cdot (\mathscr{M} + ld) + 3p_{\max} \cdot \sum_{i=1}^{N-1} indegree(G, i) = (N-1) \cdot (\mathscr{M} + ld) + 3p_{\max} \cdot (|E| - indegree(G, 0))$; for SPVP-GF, $time(T_{\text{down}}) \leqslant \sum_{i=1}^{N-1} (ld + 3p_{\max} \cdot indegree(G, i)) = (N-1) \cdot ld + 3p_{\max} \cdot (|E| - indegree(G, 0))$. SPVP-AS restricts the invalid path to include only one of node 0's direct neighbors, therefore $time(T_{\text{down}}) \leqslant \sum_{v \in V, \text{where}[v \leftarrow 0] \notin G} (\mathscr{M} + ld + 3p_{\max} \cdot indegree(G, v)) + (\mathscr{M} + ld + 3p_{\max} \cdot \max_{v \in G^0} \{indegree(G^0, v)\}) = (N - outdegree(G, 0)) \cdot (\mathscr{M} + ld) + 3p_{\max} \cdot (|E| - |E^0| + \max_{v \in G^0} \{indegree(G^0, v)\})$, where $G^0 = (V^0, E^0)$

[10] Note that in practice, the default setting of $\mathscr{M}$ is 30 s, and ld in the Internet is at most several hundreds of milliseconds. For an upper bound of $p_{\max} = 0.01$, this assumption is true for topologies with $indegree(G, v) < 1000$. On the other hand, $p_{\max}$ can become large when the background routing load (caused by other prefixes) is heavy.

A	T_{down} results under Q model	
*SPVP	$(N-1)\cdot(\mathcal{M}+ld)+$	$3p_{max}\cdot(\lvert E\rvert - indegree(G,0))$
*SPVP-AS	$(N-outdegree(G,0))\cdot(\mathcal{M}+ld)+$	$3p_{max}\cdot(\lvert E\rvert - \lvert E^0\rvert + max_{v\in G^0}\{indegree(G^0,v)\})$
*SPVP-GF	$(N-1)\cdot ld+$	$3p_{max}\cdot(\lvert E\rvert - indegree(G,0))$
*SPVP-RCN	$nodediameter(G,0)\cdot ld+$	$p_{max}\cdot nodediameter(G,0)$

Fig. 4. Tighter bounds for T_{down} under Q model.

is the subgraph consisting of node 0 and its direct neighbors. These results are summarized in Fig. 4.

The results obtained with the U model (Fig. 2) implied the convergence time bound is proportional to the number of nodes in the network for SPVP, SPVP-GF, and SPVP-AS. However, the Q model reveals that each algorithm also has a term proportional to the number of links in the network, and this is an important factor in understanding the simulation results (Section 6) and the impact of the algorithms in Internet-like topologies.

For SPVP-RCN, since the first message received causes the receiver to converge, queuing delay does not affect the convergence time. Thus $h(G,[v \leftarrow u]) \leqslant ld + p_{\max}$ holds, and according to Theorem 2, $time(T_{\text{down}}) \leqslant nodediameter(G,0)\cdot(ld + p_{\max})$. Compared with the results of SPVP, SPVP-AS, and SPVP-GF, RCN's advantage is more pronounced than in Q model.

For T_{long} convergence, the improvements of convergence algorithms are mainly on removing invalid paths faster ($wdw(v)$ and the first half of $ann(v)$ in Theorem 3). This process is similar to T_{down} thus we can obtain similarly tighter delay bounds for this process under Q model. For brevity, the detailed T_{long} results are not presented in this section, but they can be found in the Appendix 1 of technical report version of this paper [15].

6. Simulation results

We conducted simulations using SSFNET [10]. The SSFNET simulator implements an FIFO queue for incoming messages. Our parameter settings are $\mathcal{M} = 30$ s, $ld = 0.002$ s, $p_{\min} = 0.001$ s and $p_{\max} = 0.01$ s, unless otherwise specified. Each data point represents the average over multiple simulation runs. Although the analysis in earlier sections provides only the upper bound of convergence time, as also done in previous work in the literature [2,3,6–9], the insights from our worst-case analysis help one understand the simulation results (average case) that are otherwise not easy to comprehend.

6.1. T_{down}

We use three different types of topologies to study the impacts of different network properties.

6.1.1. Clique

A $Clique(n)$ is a full-mesh of n nodes, which is commonly used in the literature [2,14,6,7] to study routing protocol's convergence properties. $Clique(n)$ often reflects the worst scenario because of its high connectivity: $indegree(Clique(n),v) = outdegree(Clique(n),v) = n-1$. With $\mathcal{M} = 30$ s and $ld = 0.002$ s fixed, there are two variables: n and $p_{\max}$. We vary both n and $p_{\max}$ together. Fig. 5 shows the Q model T_{down} analytical results

A	T_{down} for $Clique(n)$	
SPVP	$(n-1)(\mathcal{M}+ld)+$	$3p_{max}\cdot(n-1)^2$
SPVP-AS	$ld+$	p_{max}
SPVP-GF	$(n-1)\cdot ld+$	$3p_{max}\cdot(n-1)^2$
SPVP-RCN	$ld+$	p_{max}

Fig. 5. $time(T_{\text{down}})$ for $Clique(n)$.

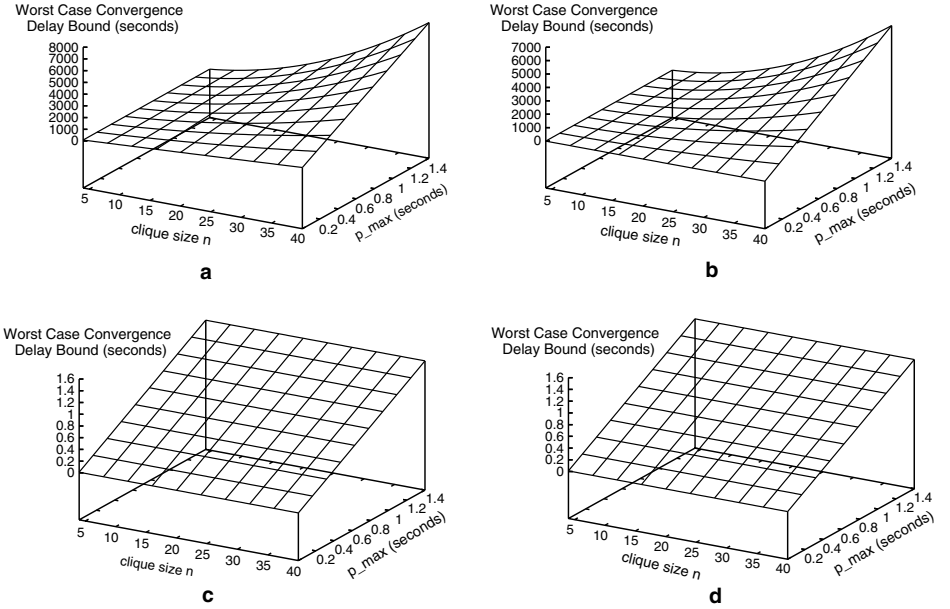


Fig. 6. T_{down} worst case delay bounds with varying n and $p_{\max}$ in *Clique*(n). Assuming no queuing up. (a) SPVP, (b) SPVP-GF, (c) SPVP-AS and (d) SPVP-RCN.

for *Clique*(n)[11] and we graph the analytical worst case in Fig. 6. The simulation results are shown in Fig. 7.

Fig. 8(a) shows the analytical worst case results when $p_{\max}$ is fixed while n varies, and Fig. 8(b) shows the analytical worst case results when n is fixed while $p_{\max}$ varies. Similarly, the simulation results are shown in Fig. 9(a) and (b). Due to the order of magnitude difference in numbers, we use both left and right Y axes in the figures.

The trend of SPVP-AS and SPVP-RCN's convergence time is consistent with the analytical results. They are consistently shorter than the other two, not affected by network size n, but increase linearly with $p_{\max}$.

When routing load ($p_{\max}$) is low and network connectivity (n) is sparse, the trend of SPVP and SPVP-GF's convergence time is consistent with the analytical results, and SPVP-GF outperforms SPVP significantly. However, when $p_{\max}$ and n are large, both protocols have very long convergence time (Fig. 7(a), the right-up corner, and Fig. 7(b), the right half), and SPVP-GF's increase is even more dramatic (Fig. 9(b)). In addition, the comparison between Figs. 7(b) and 6(b) shows that SPVP-GF's simulation results are actually worse than the analytical worst case results! This seemingly strange behavior is explained by Lemma 4. The lemma offers a quantitative condition under which messages will not queue up at a node v: if $\mathscr{M} - ld > 3 \cdot indegree(G, v) \cdot p_{\max}$. In *Clique*($n$), with our setting, this condition becomes $30 > 3(n-1)p_{\max}$. But this is only the *sufficient* condition. The actual turning point where messages start queuing up can be different. On average, each message experiences a processing delay of $(p_{\min} + p_{\max})/2 \approx p_{\max}/2$. The factor 3 comes from Assumption 1, which reflects the worst scenario. Of most cases in simulation, this factor becomes 1 for SPVP, and 2 for SPVP-GF since SPVP-GF sends *extra* withdrawal messages. Therefore, the

[11] If we just plug the *Clique*(n) parameters into SPVP-AS's result in Fig. 4 we would get $(ld + \mathscr{M}) + 3p_{\max} \cdot (n-1)$. However, because the link delay is fixed at ld, a node will always receive the withdrawal message from the origin earlier than from other nodes, thus there is no queuing delay, and the actual bound becomes $(ld + p_{\max})$.

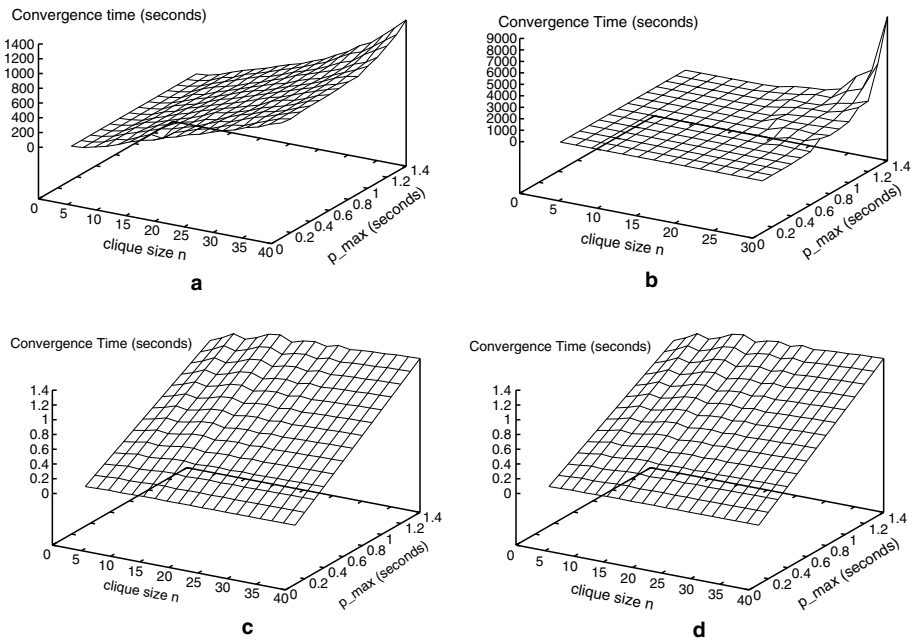


Fig. 7. T_{down} simulation results with varying n and $p_{\max}$: (a) SPVP.largest: 1282, (b) SPVP-GF.largest: 8039, (c) SPVP-AS and (d) SPVP-RCN .

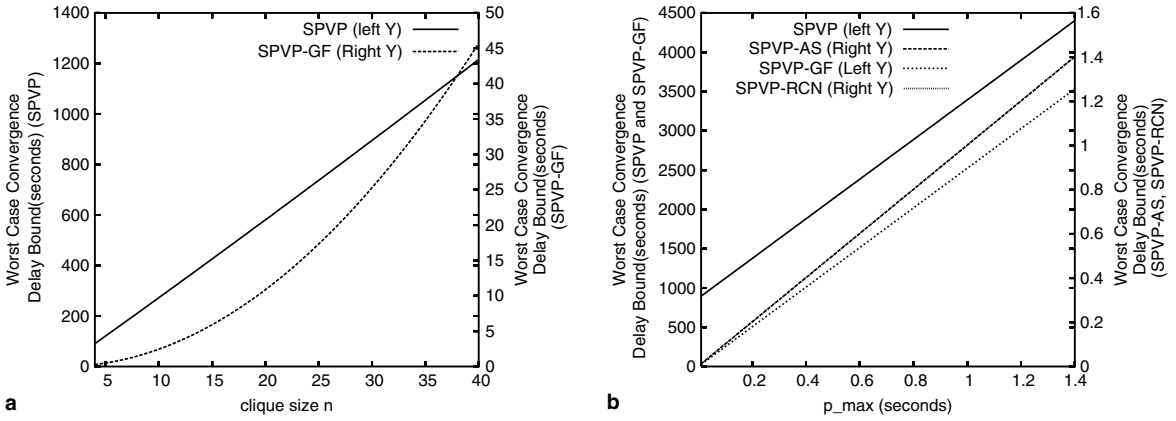


Fig. 8. T_{down} worst case delay bounds in *Clique*(n). (a) n varies, $p_{\max} = 0.01$, convergence times of SPVP-AS and SPVP-RCN are constant (0.012 seconds); (b) $p_{\max}$ varies, $n = 30$, the curves of SPVP-AS and SPVP-RCN collapse into one.

condition of messages being queued up in simulation is approximately $(n-1)\cdot p_{\max} \approx 60$ for SPVP, and $(n-1)\cdot p_{\max} \approx 30$ for SPVP-GF. Once messages start being queued up in routers, the convergence time will increase dramatically. Since SPVP-GF hits the turning point earlier, its convergence time becomes longer than that of SPVP when routing load is high (Fig. 9(b)).

6.1.2. Grid

A *Grid*(n,d) is a two-dimensional n by n grid, whose nodes have the same in-degree and out-

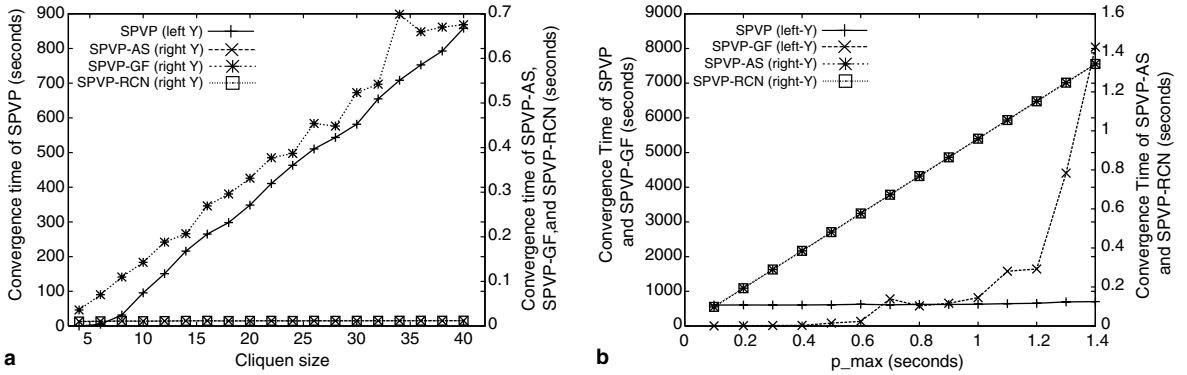
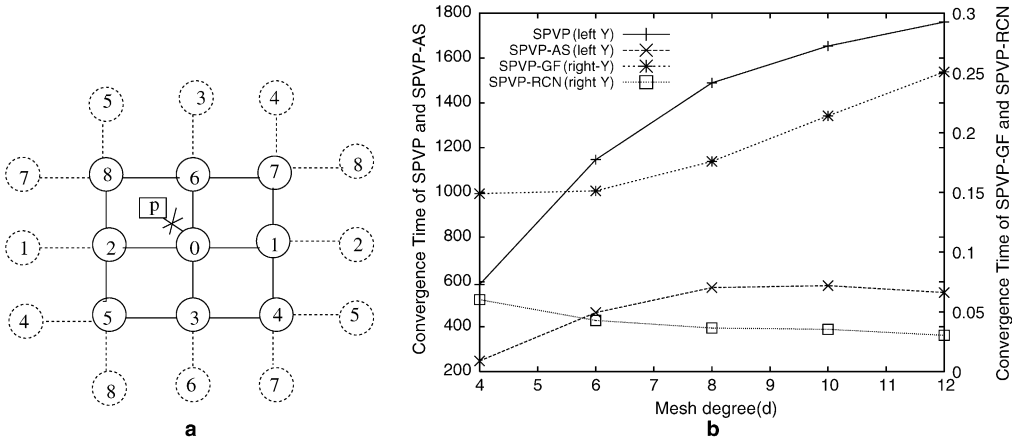


Fig. 9. T_{down} in *Clique*(n): (a) n varies, $p_{\max} = 0.01$ and (b) $p_{\max}$ varies, $n = 30$.



A	T_{down} for $Grid(n,d)$
SPVP	$(n^2-1)(ld+\mathcal{M})+3d(n^2-1)p_{max}$
SPVP-AS	$(n^2-d)(ld+\mathcal{M}+3dp_{max})$
SPVP-GF	$(n-1)ld+3d(n^2-1)p_{max}$
SPVP-RCN	$diameter(Grid(n,d))(ld+p_{max})$

c

Fig. 10. T_{down} in *Grid*(n,d): (a) T_{down} for *Grid*(3,4), (b) *Grid*(10,d) and (c) *time*(T_{down}) for *Grid*(n,d) under Q model.

degree of d. Fig. 10 shows a sample *Grid*(n,d) topology, Q model analytical results, and simulation results with $n = 10$ while d varies. As the node degree increases, the convergence time of *SPVP* and *SPVP-GF* increases, *SPVP-RCN decreases*, and *SPVP-AS* increases first but decreases later. These are all consistent with the Q model analytical results. The U model (Fig. 2) would expect the convergence time fixed for all protocols since the network size $N = 100$ does not change. This demonstrates the improved explanatory power of Q model because it takes into account the richness of the network connectivity.

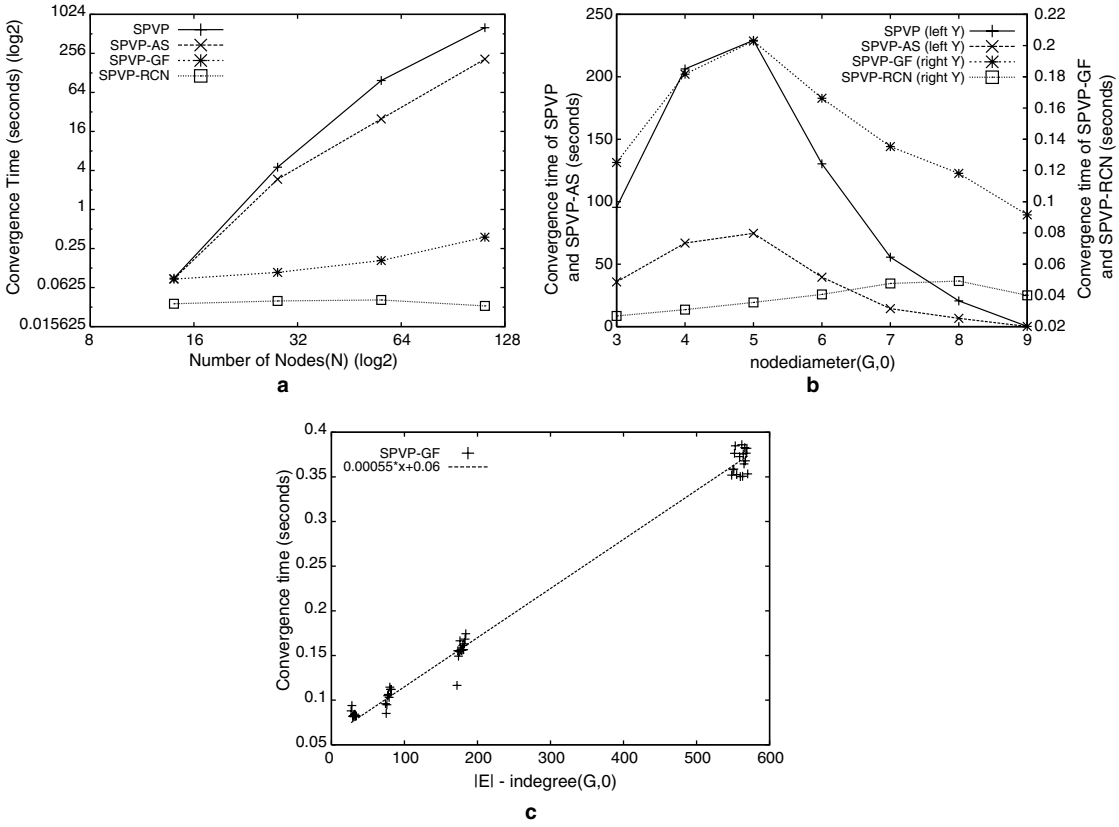


Fig. 11. Simulation results for T_{down} convergence time in Internet-like topologies: (a) network size, (b) *nodediameter*$(G,0)$ and (c) $|E| - \mathit{indegree}(G,0)$ in SPVP-GF.

6.1.3. Internet-like topology

To further understand T_{down} convergence, we simulate Internet-like AS-level topologies. To derive a simulation topology that resembles the Internet topology,[12] we first generated a 110-node AS-level topology based on BGP routing tables from RouteViews [16] by using the algorithm described in [17]. 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.

One node x is chosen as the only origin AS that advertises a destination prefix, and we simulate T_{down} event by marking x down. We repeat simulations for each node in each topology. The Q model analytical results (Fig. 4 the second column) show that N, $|E| - \mathit{indegree}(G,0)$ and *nodediameter*$(G,0)$ are important factors, so we are interested in their impact as well as the comparison among different protocols. From the network size (N) point of view, Fig. 11(a) shows that the convergence time of SPVP-RCN and SPVP-GF are 2–3 order of magnitudes better than that of SPVP and SPVP-AS; this is because SPVP-RCN and SPVP-GF do not have $\mathscr{M}$ in their T_{down} convergence time. This performance difference is also

[12] Due to SSFNET's well-known simulation speed problem and demanding memory requirements when simulating large network topologies [14], we can only simulate relatively small AS-level topologies.

confirmed by the results from *nodediameter*$(G,0)$ point of view in Fig. 11(b). In addition, the trend of linearly increase of SPVP-RCN is expected from its worst case *nodediameter*$(G,0)(ld + p_{\max})$. For SPVP-GF, its worse case is $(N-1)ld + 3(|E| - indegree(G,0))p_{\max}$, and Fig. 11(c) confirms that its convergence time is indeed approximately proportional to $(|E| - indegree(G,0))$. These results demonstrate that although our analytical results considered the upper bound, insights obtained from this analysis can help us understand the average case.

6.2. T_{long}

Prior to this work, there were questions about the T_{long} convergence time that had not been answered. Early Internet experiments [2] claimed that T_{long} and T_{down} have similar convergence time due to path exploration. However, later algorithms such as SPVP-RCN and SPVP-GF improved T_{down} significantly by reducing path exploration, but only improved T_{long} modestly in simulations [6,7]. For example, Fig. 12(a) shows the averaged T_{long} convergence time versus the network size N in some Internet-like topologies. The results are averaged over various origin nodes and failure links, while J is kept fixed at 1. It is worth to noting that SPVP performs well even in large network size, and none of SPVP-AS, SPVP-GF, or SPVP-RCN provides significant improvement.

Our analysis enables us to provide the first quantitative explanation to the above phenomena. The analytical results of T_{long} delay under Q model are presented in Appendix 1 of the extended version of this paper [15]. They are similar to the results under U model (Fig. 3) in that the dominant factor in T_{long} convergence time is *nodediameter*$(G',0)$ (Fig. 12(b)). *nodediameter*$(G',0)$, the longest distance to the destination after the failure, is usually a small value in a well connected network, e.g., $\leqslant 9$ in our simulation topologies. Therefore, the room for T_{long} improvement by any algorithm is far less than the room for improvement in T_{down}. In the real Internet, *nodediameter*$(G',0)$ is likely to be a little bit more than 10 [11], a relatively small value. Previous experiments [2] injected synthesized backup path with length around 30, which artificially increased the *nodediameter*$(G',0)$ about three times, resulting in very long T_{long} convergence time.

Our analysis in Fig. 3 shows that another important factor in T_{long} delay bound is J, the distance from the failure to the destination. The larger J is, the smaller the delay bound. However, this factor has been implicitly ignored in previous studies. To study the impact of J, we simulated SPVP T_{long} in *Grid*$(n,4)$ topologies, while varying both *nodediameter*$(G',0) = n$ and J. The results in Fig. 13 show that the convergence time indeed is proportional to $-J$.

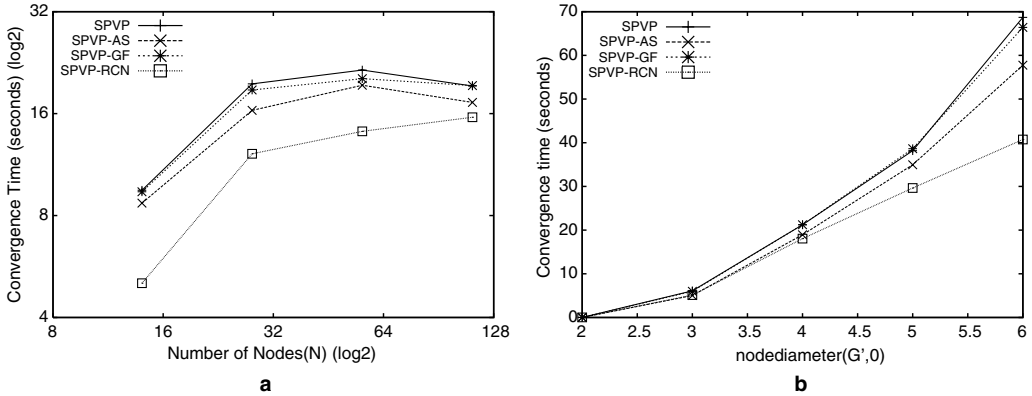


Fig. 12. T_{long} in Internet-like topologies: (a) network size (log–log), (b) *nodediameter*$(G',0)$ in 110-node topology.

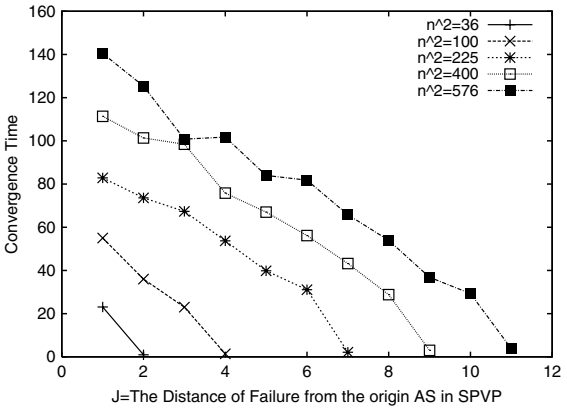


Fig. 13. SPVP T_{long} in *Grid*$(n,4)$, n and J vary.

7. Related work

There are several previous efforts in analyzing convergence delay in BGP (or SPVP). Labovitz et al. [2] analyzed the T_{down} convergence delay bound by using a synchronous model of BGP and observed that *Clique*(n)'s convergence time is bounded by $(n-1)\mathscr{M}$ seconds. Further analysis by Labovitz et al. in [3] showed that T_{down} convergence delay is upper bounded by $(p \cdot \mathscr{M})$, where p is the length of the longest possible backup path. The above results were obtained using U model, ignoring the routing message queuing delay. Obradovic [9] developed a real-time BGP model which takes into account an edge delay similar to the definition of $\mathscr{D}(G,[v \leftarrow u])$. Based on this real-time model, the author showed that the T_{down} convergence time bound for the shortest path policy is ωp where p is defined above and ω the largest edge delay. The author did not specify how to calculate the edge delay or model the MRAI delay. Our analytical framework is more general than these three works, and provides the T_{long} analysis results which are missing in the above works. Our Q model also provides more accurate and insightful results.

The analysis of Ghost Flushing [6], RCN [7] and FESN [8] uses the U delay model. Our analysis with Q model provides tighter delay bounds than those provided by these three works. In addition, our general analytical framework allows us to provides T_{long} results for SPVP-GF, which were missing previously. Simulation studies using SSFNET by Griffin et al. [14] found that for each network topology there is an optimal $\mathscr{M}$ during which messages received from each neighbor can be "consumed." Our work provides a sufficient condition under which the messages can be consumed (Lemma 4 and Theorem 5).

Ansari et al. [18] proposed an efficient and reliable approach for disseminating link-state information in a network. Traditionally link-state protocols (e.g., OSPF) flood link-state information to the entire network, which incurs significant communication overhead. A low-overhead alternative is to disseminate the information only along a spanning tree, but it will break if any link on the spanning tree fails. Given a network topology graph, Ansari et al. [18] proposed to construct a sub-graph which has fewer links than the original graph, but does not introduce any one-link minimum edge cut. Disseminating the link-state information over this sub-graph has less communication overhead than flooding the entire network, and at the same time it can tolerate some link failures on the dissemination paths.

However, Ansari et al. [18]'s approach cannot be directly applied to path vector protocols like BGP due to fundamental difference between path vector and link-state routing. In link-state routing, the information being propagated is the origin link's state. This information does not change during the propagation, and as long as a router receives this information, it does not matter from which neighbor it receives. However, in path vector routing, the information being propagated is the sender's "best path," which will change after each hop. A router must know all of its neighbors' best paths in order to pick its own best paths. Therefore restricting the routing exchange on a subset of links like [18] will affect the correctness of routing decision.

8. Conclusion and future work

This paper presents a general framework for deriving and analyzing convergence delay bounds in path vector routing protocols. To the best of our knowledge, our framework is the first one that can be used to analyze *all* the existing path vector protocol variants (both standard BGP and its con-

vergence improvements) under the assumption of the shortest path policy and single link failure. It quantifies the impacts of important factors, including network connectivity, failure location, and message processing delay. We believe that our framework can also be used to analyze new improvements of path vector routing protocols, should they occur.

Our framework enabled us to develop analytical bounds that did not exist previously, i.e., T_{long} delay bounds of standard BGP, Assertion, and Ghost Flushing as well as the T_{down} delay bound for Assertion. Our analysis also shows that the dominant factor in BGP's T_{long} delay bound is the $\textit{nodediameter}(G', 0) - J$, where J is the distance between the failure and the destination and $\textit{nodediameter}(G', 0)$ is the length of the longest alternate path used to reach the destination after the failure. The value of this term is relatively small in a well-connected network such as today's Internet, which explains why various proposed convergence improvement algorithms only shorten the convergence delay of T_{long} to a modest degree. Furthermore, by taking into account the message processing delay, the Q model reveals insights into the impacts of topological connectivity richness and message processing delay on convergence delay, and explains why different protocols react differently to the increase of routing message load and network connectivity.

8.1. Future work

We believe that the framework developed in this paper can be extended in the following ways. First, as with all existing analysis in the literature, our current framework is not directly applicable to multiple failures overlapping in time. Modeling overlapping failures would require factoring in detailed timing of the overlapping failures. Furthermore, not all the existing convergence algorithms provide details of how to treat multiple link failures, and different details could lead to different convergence results. In our on-going work, our first step is to analyze the case of single node failure of ETN algorithms, and we assume that simultaneous root causes will be sent out by the neighbors of the failed node. This assumption can help simplify the analysis. For example, in T_{down} a node's convergence starts when the node failure happens and ends when the last piece of root cause information is received. In addition to this on-going work, we also plan to analyze the node failure and multiple overlapping failures in general.

Second, as stated in the BGP specification [1], multiple routers within the same AS should behave consistently and appear to be one node to the outside. Therefore, as with all the existing BGP convergence studies, we model each AS as a node even though there may be multiple routers within one AS. This omission of detail does not affect our general framework in Section 3 or U model analysis results in Section 4, but does have an impact on the accuracy of message processing delay h in our Q model since queuing delay may occur at each of multiple routers. For future work we plan to extend the "message processing delay" in our Q model to cover the case of multiple-router ASes.

Third, for simplification, we assumed shortest path policy in our framework. However, a path vector protocol can adopt many types of policies, including the shortest path policy and the "no-valley" policy [19]. We note that our analysis provides comparative evaluation for different path vector protocols under shortest path policy, and that our analysis *technique* should hold in the case of a more general policy, provided that the policy does not lead to policy oscillation as shown in [12]. However, the results can be different, depending on the convergence algorithms and types of failures. For T_{down} convergence under ITN algorithms, the delay bound would be proportional to the length of longest possible path *allowed by the policy*; under no-valley policy, the longest possible path can be smaller than that in case of shortest path policy. Similarly, for T_{down} convergence under ETN algorithm, the delay bound would be proportional to the diameter allowed by the policy (i.e., the length of the longest *shortest path allowed by the policy*). For the T_{long} analysis, the concepts of border-node and two-process convergence will still be the key to the analysis, but the path exploration would not be monotonic in path length as in the shortest path policy case. As our next step, we plan to analyze convergence delay bounds with the no-valley policy.

In summary, the framework presented in this paper provides a solid basis that enables these future works.

Acknowledgements

We thank our editor Professor Guoliang Xue and the anonymous reviewers for their detailed and insightful comments that greatly improved the paper.

Appendix 1. More explanations for Assumption 1

BGP RFC [1] specifies that only one *announcement* can be *sent* out within $\mathscr{M}$ seconds from one node v to a neighbor u, now we show that more than *one* withdrawal can be sent within $\mathscr{M}$ seconds. We then argue that they signal no new information and should cause negligible processing delay if they do occur, thus Assumption 1 is reasonable.

Since the MRAI timer does not apply to *withdrawals*, a node v can send out a withdrawal immediately after $r(v)$ changes to ϵ. Suppose the MRAI timer is turned on at $t = 100$ s, thus no announcement can be sent out within interval $[100, 130)$. Now suppose $r(v) = \epsilon$ at $t = 105$ s, $r(v) \neq \epsilon$ at time $t = 110$ s, and $r(v) = \epsilon$ at time $t = 115$ s. The non-empty path at $t = 110$ cannot be sent out, but the two withdrawals are sent out at $t = 105$, and $t = 115$, respectively.

In the worst case, there can be $y = \lfloor \frac{\mathscr{M}}{p_{\min}} \rfloor$ path changes generated within $\mathscr{M}$ seconds, where $p_{\min}$ is minimum message processing time. On the other hand, according to the protocol definition, $r(v)$ cannot change from ϵ to ϵ, the maximal number of $r(v)$ changes to ϵ is $\lceil \frac{y}{2} \rceil$ withdrawals while the $r(v)$ change series is in a pattern of "*withdrawal*, *announcement*, *withdrawal*, *announcement*, ..., *withdrawal*, *announcement*," or "*announcement*, *withdrawal*, *announcement*, *withdrawal*, ..., *announcement*, *withdrawal*," or "*withdrawal*, *announcement*, *withdrawal*, *announcement*, ..., *withdrawal*, *announcement*, *withdrawal*." In the worst case, each of the $\lceil \frac{y}{2} \rceil$ withdrawals can be sent out within $\mathscr{M}$ seconds.

However, it is not clear whether the above worst case will ever happen in reality. Some forms of duplicate update elimination can help remove part of consecutive withdrawals, and processing time for a *duplicate* withdrawal might be negligible compared to a normal update which needs policy checking and best-path re-computation, and a duplicate withdrawal's contribution to the queuing time is negligible. In addition, a duplicate withdrawal cannot change the receiver's path. In other words, duplicate withdrawals are *ineffective*. Therefore, we have ignored duplicate withdrawals to simplify our analysis.

In addition, the latest BGP standard revision [20] has proposed to apply the MRAI time to withdrawal messages as well (called Withdrawal Rate Limiting, or WRATE). The duplicate withdrawal problem discussed above does not exist if WRATE is used. Furthermore, Assumption 1 should be revised to "During any $\mathscr{M}$ second interval, node u can *send* at most 1 update to node v". Our Q model results in this paper can be easily revised with a different constant factor(replacing "3" with "1") to get the results with WRATE used.

References

[1] Y. Rekhter, T. Li, Border Gateway Protocol 4, RFC 1771, SRI Network Information Center, July 1995.
[2] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, Delayed internet routing convergence, in: Proceedings of ACM Sigcomm, 2000.
[3] 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.
[4] 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.
[5] J. Luo, J. Xie, R. Hao, X. Li, An approach to accelerate convergence for path vector protocol, in: Proceedings of IEEE Globecom, 2002.
[6] A. Bremler-Barr, Y. Afek, S. Schwarz, Improved BGP convergence via Ghost Flushing, in: Proceedings of the IEEE INFOCOM, 2003.
[7] D. Pei, M. Azuma, D. Massey, L. Zhang, BGP-RCN: Improving BGP convergence through root cause notification, Elsevier Computer Networks Journal 48 (2) (2005) 175–194.

[8] J. Chandrashekar, Z. Duan, Z.-L. Zhang, J. Krasky, Limiting path exploration in path vector protocols, in: Proceedings of the IEEE INFOCOM, 2005.
[9] D. Obradovic, Real-time model and convergence time of BGP, in: Proceedings of the IEEE INFOCOM, 2002.
[10] The SSFNET Project, http://www.ssfnet.org.
[11] G. Huston, BGP table data, http://bgp.potaroo.net/.
[12] T. Griffin, F.B. Shepherd, G. Wilfong, The stable path problem and interdomain routing, IEEE/ACM Transactions on Networks 10 (2) (2002).
[13] T. Griffin, G. Wilfong, A safe path vector protocol, in: Proceedings of IEEE INFOCOMM, 2000.
[14] T. Griffin, B. Premore, An Experimental Analysis of BGP Convergence Time, in: Proceedings of ICNP, 2001.
[15] D. Pei, B. Zhang, D. Massey, L. Zhang, An analysis of path-vector routing protocol convergence algorithms, Tech. Rep. TR-040009, UCLA CSD, (March 2004).
[16] The Route Views Project, http://www.antc.uoregon.edu/route-views/.
[17] B. Premore, Multi-as topologies from bgp routing tables, http://www.ssfnet.org/Exchange/gallery/asgraph/index.html.
[18] N. Ansari, G. Cheng, R. Krishnan, Efficient and reliable link state information dissemination, IEEE Communications Letters 8 (2004) 317–319.
[19] L. Gao, On inferring autonomous system relationships in the internet, IEEE/ACM Transactions on Networks 9 (6) (2001).
[20] Y. Rekhter, T. Li, S. Hares, Border Gateway Protocol 4, http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp4-26.txt, October 2004.

Dan Pei is currently a Ph.D. candidate at UCLA Computer Science Department. He received his B.E. and M.S. in Compute Science from Tsinghua University, China. His primary research interest is in networking. In particular, he is interested in protocol design and analysis, fault-tolerance, and security of various networks.

Beichuan Zhang is a postdoctoral researcher in the Computer Science Department at UCLA. He received his Ph.D. in Computer Science from UCLA in 2003 and spent a year at USC/ISI. His research interests include Internet routing, overlay networks, multicast, network measurement and performance evaluation.

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 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.