An Analysis of Convergence Delay in Path-Vector Routing Protocols

Dan Pei [1,*] Beichuan Zhang [1] Daniel Massey [2] Lixia Zhang [1]

Abstract

Path vector routing protocols such as BGP are known to suffer from slow convergence. In response a number of convergence enhancements have been proposed, but convergence dynamics have yet to be well understood and there has been no general framework to assess and compare the various improvement algorithms. In this paper we present a general framework to analyze the convergence delay bounds of path vector routing protocols, under the assumption of shortest path policy and single link failure. Our framework takes into account important factors including network connectivity, failure location, and message processing delay. It applies to *all* path vector protocol variants(standard path vector protocol and convergence improvement algorithms) and allows us to develop analytical bounds that were not previously available, such as the delay bounds for path fail-over for BGP and most of the convergence enhancements. Our analysis shows that BGP fail-over delay bounds are determined largely by a combination of two factors: 1) the distance between the failure and the prefix origin and, 2) the length of the longest alternate path used to reach the destination after the failure. These factors are captured formally and can explain why existing convergence enhancements often provide only limited improvements in fail-over events. Furthermore, explicitly modeling message processing delay reveals insights into the impacts of topology structure (e.g. richness in connectivity) and different effectiveness of different enhancements. These new results allow one to better understand the behavior of various path vector protocols under given topology structure, network size, and message delays.

Key words: Routing, BGP, Path Vector Protocol, Routing Protocol Convergence

1 Introduction

In this paper, we present an analytical framework to study the convergence properties of path vector routing algorithms. After a change in network connectivity, path vector routing protocols tend to explore a potentially large number of alternative paths before converging on new stable paths. The resulting "slow convergence" problems have been observed in the Internet's BGP[1] path vector protocol [2][3]. In response, a number of enhancements ([2][4][5][6][7][8]) have been proposed. However, a thorough understanding of convergence behavior for BGP and its enhancements has not yet been obtained due to the following reasons: First, there is no general analytical model that applies to *all* the existing path vector protocol variants(standard BGP and convergence improvement algorithms). Existing models make different assumptions. Even though dramatic improvements have been demonstrated by some of the 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.

Second, the existing analytical results are often incomplete. In particular, BGP and most of the enhancements do not have an analytical bound on the convergence delay for path fail-over (T_{long}, where routers switch to less preferred paths). In addition, it is commonly observed in simulation experiments [6][7][8] that various proposed convergence improvement algorithms result in only modest improvements to T_{long} convergence delay, but there has been no general explanation for why this is so.

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

⋆ 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
Email addresses: peidan@cs.ucla.edu (Dan Pei), bzhang@cs.ucla.edu (Beichuan Zhang), massey@cs.colostate.edu (Daniel Massey), lixia@cs.ucla.edu (Lixia Zhang).

[1] UCLA Computer Science Department, Los Angeles,CA 90095

[2] Computer Science Department, Colorado State University, Fort Collins, CO 80523-1873

This paper develops a general framework in analyzing the convergence delay bounds of path vector routing protocols under the assumption of shortest path policy and a single link failure. The framework applies to *all* the existing path vector protocol variants(standard BGP and convergence improvement algorithms), and 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 in [2][3][9][6][7][8]. The U model assumes that all routing messages are processed within a bounded time, independent of network topology, and the results are presented in Section 4. Section 5 presents the Q model, which explicitly takes into account the message queueing delay at each node and reveals important insights not previously obtained by the U model. In Section 6, we report our simulation results from SSFNET[10], a well known BGP simulator in which a router processes messages through FIFO queue, 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 allows us to develop analytical bounds that were not previously available. We develop a two-process approach for T_{long} analysis (presented in Section 3) that allows us to obtain bounds for arbitrary path vector algorithms and we provide the first T_{long} convergence delay bounds for standard BGP, Assertion [4], and Ghost Flushing [6]. In addition, we provide the first T_{down} (where the destination becomes unreachable) convergence delay bound for Assertion [4].

Second, our analysis shows that standard BGP's T_{long} delay bound is approximately $\mathcal{M} \cdot (nodediameter(G', 0) - J)$, where M is the *Minimal Route Advertisement Interval* (typically 30 seconds), J is the distance between the failure and the prefix origin, 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 the various enhancement algorithms bring only modest improvement to the convergence delay of T_{long}. In addition, our analysis show that the delay of T_{long}'s "counting-to-new-best-path" is much quicker than the delay of T_{down}'s "counting-to-infinity". This result is in contrast to the previous perception that T_{down} and T_{long} have similar convergence delays [2], a conclusion based on experiments where the value of $nodediameter(G', 0)$ was arbitrarily exaggerated.

Third, the Q model is the first analytical model that takes into account the message queueing delay. Unlike the previous U model, our Q model reveals insights into the impacts of topology structure (e.g. richness in connectivity) and processing delay:

- The Q model provides a quantitative condition under which messages will not queue up at a node. The condition depends on the maximal time to process a single message (p_{max}), as well as the maximum number of messages that can be received by a node during one $\mathcal{M}$ period, which in turn depends the degree of the node.
- 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.
- 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. In the Q model, network connectivity plays a role in determining message queueing delay and the analytical results obtained using the Q model can better explain simulation results that could not previously be explained.

2 Background, Definitions, and Algorithms

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

The network is modeled as a directed connected graph $G = (V, E)$. $V = \{0, 1, \cdots, N-1\}$ represents the set of N nodes that run SPVP protocol, and they are connected by links in E. 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 link $[v_i \leftarrow v_{i-1}] \in E$ and $v_i \in V$ for all $i, 0 \leq i \leq k$, and $v_0 = 0$. We say $v_i{\in}r, \forall i, 0 \leq i \leq k$; $[v_i \leftarrow v_{i-1}]{\in}r, \forall i, 1 \leq i \leq k$; $(v_i, v_{i-1}, \cdots, v_0) \subset r, \forall i, 0 \leq i \leq k-1$. We define $length(r) = k$, and $length(\epsilon) = \infty$ for empty path. 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:

$degree(G, v)$	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 path vector routing protocol in which each node advertises *only* its best path to its neighbor nodes. A node v stores the latest path received from all the neighbors, 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. Previous studies showed that some path selection policies can lead to persistent path oscillation [12]. As a first step, this paper only considers shortest-path policy(when two paths have the same length, the path from the neighbor with lower node ID is preferred) 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 for other policies as one of our future works.

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 $[v \leftarrow u]$, 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 all paths received from neighbor u from its routing table. If link $[v \leftarrow u]$ changes from $down$ to *up*, node u announces its best routes 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 is separated by at least $\mathcal{M}$ seconds, and it is per (neighbor, prefix) based.[3] Following the BGP specification [1], the MRAI timer is not applied to withdrawal messages.

2.1 SPVP Convergence Definitions

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

In [2][3][6][7], BGP routing events are categorized into four classes:

- T_{up}: a previously unavailable prefix is announced.
- T_{short}: an existing path is replaced by a more preferred path.

[3] In reality, the MRAI timer can also be implemented on a per neighbor based.

- T_{long}: a link $[v \leftarrow u]$ fails and the nodes relying on this link switch to a less preferred path.
- T_{down}: a prefix is no longer reachable and all nodes withdraw their paths to the destination p

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

Definition 2 *Converged State: a node v is in a converged state iff $r(v)$ will not change unless some new event occurs.*

Definition 3 *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 can be done by treating node failure as multiple simultaneous link failures, but we leave a detailed analysis as our future work.

Note that in our model, T_{down} happens when the link $[0\ p]$ fails, and T_{up} happens when node 0 detects that the $[0\ p]$ link has recovered from a previous failure. T_{long} events can be triggered by the failure of any link other than $[0\ p]$, and T_{up} can be triggered by the recovery of any link other than $[0\ p]$. [4] In addition, as is done in all the related work[2][3][9][6][7][8], we focused on the upper bound of convergence delay.

2.2 SPVP Convergence Algorithms

This section reviews existing convergence algorithms proposed to improve convergence time of the basic SPVP. Due to page limit, 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

[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.

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 obsolete and removed. SPVP-AS does not eliminate the propagation of all obsolete paths, and its effectiveness is sensitive to the topology. b

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, the obsolete path is quickly removed from the network. SPVP-GF does not eliminate the propagation of all the obsolete 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. Suppose 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 obsolete and removed. Since every update carries the root cause, once a node receives the first routing message, it can immediately discard all the obsolete paths in the convergence triggered by a single link failure. [5]

Other Algorithms 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. RCO (Route Change Origin) 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. FESN (Forwarding Edge Sequence Number) [8] is similar to RCN except that FESN uses link sequence number instead of node sequence number. Therefore FESN has the same T_{down} and T_{long} delay bound as RCN.

[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 obsolete paths invalidated by the root causes already received.

3 A Framework for Convergence Analysis

In this section, we present a general framework for analyzing convergence time under the assumption 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 and develop path classification notations needed later in the analysis. We then use these classifications to establish a general framework for bounding T_{down} and T_{long} convergence times for arbitrary graphs and arbitrary path vector algorithms. [6]

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, an ITN node announces that path (v_2, v_1, v_0) is being replaced by longer path $(v_2, v_5, v_4, v_3, v_0)$. Under shortest path policy, the change to a longer path implicitly signals that either link $[v_2 \leftarrow v_1]$ or link $[v_1 \leftarrow v_0]$ fails. Since link failure (or recovery) is signaled implicitly, a path change received from one neighbor has little or no impact on the validity of 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. SPVP-AS is the only ITN algorithm that attempts to use implicit failure information to invalidate paths. In the SPVP-AS, route r learned from node u may be invalidated using updates from other nodes provided that one of the nodes in r is a direct neighbor to v and this node sends path information that conflicts with r. 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.

[6] In addition to the upper bound of convergence delay, ideally one would also like to analyze the *average* convergence delay, which could be compared with simulation results and measurement results. However, average case analysis is much more difficult, as also realized in previous work in the literature [2][3][9][6][7][8], and we will leave it as our future work. On the other hand, as we will show later in Section 6, the insights from delay upper bound help undertand the simulation results that are otherwise not easy to comprehend.

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, 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 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 any other invalid path) 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.

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 neighbor v learns the path via neighbor u is *obsolete* and we let $\mathcal{D}(G, [v \leftarrow u])$ denote the upper bound on the time a path can remain obsolete. $\mathcal{D}(G, [v \leftarrow u])$ can include the MRAI delay, transmission delay, propagation delay, queueing 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 queueing 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, $\mathcal{D}(G, [v \leftarrow u]) \geq t_2 - t_1$.

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 $\mathcal{D}_{replace}(G, [v \leftarrow u])$ to denote the upper bound on learning the *replacement* path. In algorithms such as SPVP, $\mathcal{D}_{replace}(G, [v \leftarrow u]) = \mathcal{D}(G, [v \leftarrow u])$. But in SPVP-GF (and future similar algorithms), one can have $\mathcal{D}_{replace}(G, [v \leftarrow u]) = \mathcal{D}(G, [v \leftarrow u]) + \mathcal{M}$.

$R_v(G, A)$: the set of all the *invalid* paths, starting at node v, in G allowed by algorithm A
$R_v^l(G, A) = \{r \mid r \in R_v(G, A) \wedge length(r) \leq l\}$
$\mathcal{D}(G, [v \leftarrow u])$: maximum time that may elapse between u changes its path and its neighbor v learns the path via u is *obsolete*.
$\mathcal{D}_{replace}(G, [v \leftarrow u])$: 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 $r = (v_l, v_{l-1}, \cdots, v_0)$ of length l that may occur during a T_{down} event, the path will be withdrawn by time $f(r) = \sum_{i=1}^{l} \mathcal{D}(G, [v_i \leftarrow v_{i-1}])$ 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 $\mathcal{D}(G, [v_1 \leftarrow v_0])$. 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 $r = (v_l, v_{l-1}, \cdots, v_0)$ and consider any path $r' = (v_{l+1}, v_l, v_{l-1}, \cdots, v_0)$. According to the induction hypothesis, v_l has withdrawn path r from its routing table by time $\sum_{i=1}^{l} \mathcal{D}(G, [v_i \leftarrow v_{i-1}])$ 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 $\mathcal{D}(G, [v_{l+1} \leftarrow v_l])$ 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 $r = (v_l, v_{l-1}, \cdots, v_0)$ that may occur during a T_{down} event, we call $f(r) = \sum_{i=1}^{l} \mathcal{D}(G, [v_i \leftarrow v_{i-1}])$ the path ***lifetime***. 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,*

$$time(T_{down}) \leq \max_{v, r \in R_v(G,A)} \{f(r)\}$$

Proof: Note that $\max_{v, r \in R_v(G,A)} \{f(r)\}$ 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,*

$$time(T_{down}) \leq \max_{v \in V} \{ \min_{r \in R_v(G,A)} \{f(r)\}\}$$

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. ■

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 that link 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 $v_0 = c$ and $t_{J-1} = b$. Nodes $t_i(0 \leq i \leq J - 1)$

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

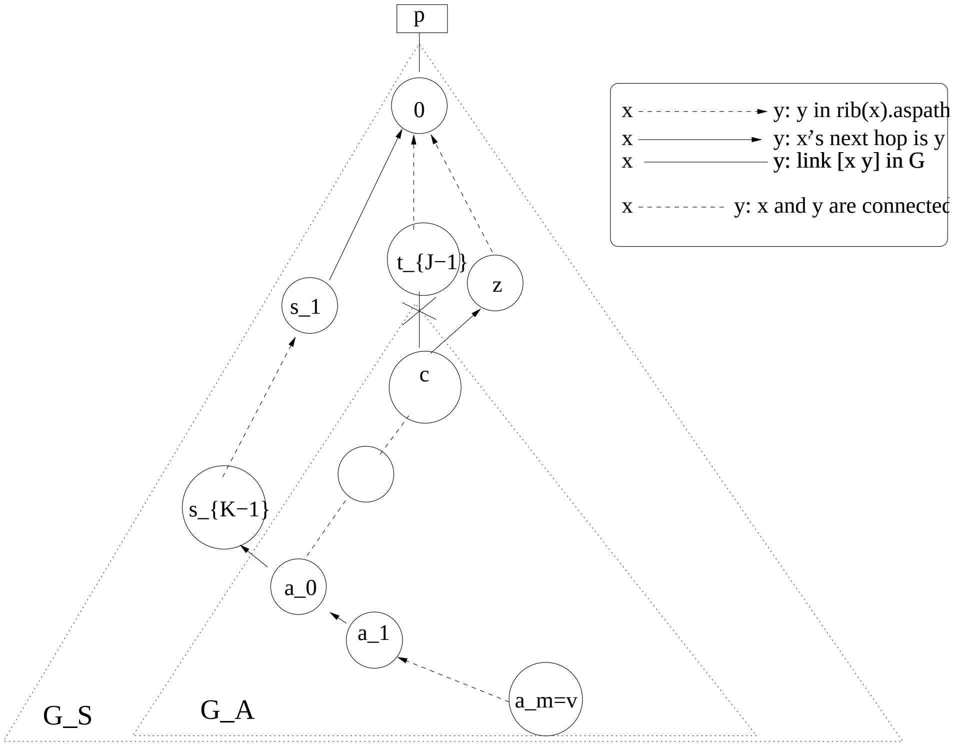


Fig. 1. Routing Tree after T_{long} Convergence

are not affected by the failure and nodes $v_i (0 \le i \le l)$ 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.

$G'(V, E')$	Topology after an event T occurs in $G(V, E)$
V_S	Nodes whose paths have not changed after T
V_A	Nodes whose paths changed after T
E_A	Links whose both ends belong to V_A
$G_A(V_A, E_A)$	Sub-graph of G' with V_A and E_A
$r^{old}(v)$	node v's best path in G
$r^{new}(v)$	node v's best path in G'
J	$= distance(G, c, 0)$, in $T_{long}([c \leftarrow b])$

The table summarizes the notation used in T_{long} analysis and Figure 1 illustrates the concepts. Note from the figure, one can observe not only the subgraph of affected nodes, but also identify the characteristics of the new converged paths. The following lemmas formalizes the observations from Figure 1 and captures the relationship between path length and the time required to remove a path during a T_{long} event:

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

m), $s_i \in V_S (0 \leq i \leq K-1)$.

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

Lemma 3 *During $T_{long}([c \leftarrow b])$, any invalid path $r = (v_l, v_{l-1}, \cdots, v_0 = c, b = t_{J-1}, \cdots, t_0)$ will be withdrawn by time $g(r) = \sum_{i=1}^{l} \mathcal{D}(G, [v_i \leftarrow v_{i-1}])$ 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 $r = (v_1\ \ v_0\ \ t_{J-1} \cdots t_0)$. 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 $\mathcal{D}(G, [v_1 \leftarrow v_0])$. The path $(v_1, v_0, t_{J-1}, \cdots, t_0)$ 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 $r = (v_l, v_{l-1}, \cdots, v_0, t_{J-1}, \cdots, t_0)$ and consider any path $r' = (v_{l+1}, v_l, v_{l-1}, \cdots, v_0, t_{J-1}, \cdots, t_0)$. According to the induction hypothesis, v_l has withdrawn path r from its routing table by time $\sum_{i=1}^{l} \mathcal{D}(G, [v_i \leftarrow v_{i-1}])$ 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 $\mathcal{D}(G, [v_{l+1} \leftarrow v_l])$ 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 $r = (v_l, v_{l-1}, \cdots, v_0, t_{J-1}, \cdots, t_0)$ that may occur during a T_{long} event, we call $g(r) = \sum_{i=1}^{l} \mathcal{D}(G, [v_i \leftarrow v_{i-1}])$ the path ***lifetime***. 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.

Theorem 3 *Given any network G, and ITN algorithm A, $time(T_{long})$*
$\leq \max_{v \in V_A}\{max\{wdw(v), ann(v)\}\}$ where
$r^{new}(v) = (a_m, \cdots, a_0, s_{K-1}, \cdots, s_0)$
$wdw(v) = \max_{r \in R_{a_m}^{K+m}(G,A)}\{g(r)\}$
$ann(v) = \max_{r \in R_{a_0}^{K}}\{g(r)\} + \sum_{i=1}^{m} \mathcal{D}_{replace}(G, [a_i \leftarrow a_{i-1}])$

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 $\mathcal{D}_{replace}(G, [v \leftarrow u])$. Therefore the total propagation time is $\sum_{i=1}^{m} \mathcal{D}_{replace}(G, [a_i \leftarrow a_{i-1}])$. Combining these two together, we have $ann(v) = \max_{r \in R_{a_0}^K}\{g(r)\} + \sum_{i=1}^{m} \mathcal{D}_{replace}(G, [a_i \leftarrow a_{i-1}])$ ■

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,* $time(T_{long}) \leq$
$$\max_{v \in V}\{\min_{r \in R_{a_0}(G,A)}\{g(r)\} + \sum_{i=1}^{m} \mathcal{D}_{replace}(G, [a_i \leftarrow a_{i-1}])\}$$
where $r^{new}(v) = (a_m, \cdots, a_0, s_{K-1}, \cdots, s_0)$

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 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 $\sum_{i=1}^{m} \mathcal{D}_{replace}(G, [a_i \leftarrow a_{i-1}])$ before reaching $v = a_m$. The overall network convergence time is obtained by simply taking the maximum value over all nodes. ■

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

Having established the general framework for computing convergence bounds, we now consider specific delay models and produce algorithm specific results for the base 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 $\mathcal{D}(G, [v \leftarrow u])$. Generally $\mathcal{D}(G, [v \leftarrow u])$ includes MRAI delay, transmission delay, link propagation delay, queueing delay and processing delay.

The MRAI delay is bounded by the MRAI timer value, $\mathcal{M}$, usually configured with the default value of 30 seconds with a random jitter. In this paper we assume that MRAI timer is *exactly* $\mathcal{M}$ seconds without jitter; our results can be easily extended to consider 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 ETN algorithms, in T_{down}, all the messages are withdrawals, thus MRAI timer does not apply anyway; in T_{long}, node a_0 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 are not delayed by the MRAI timer. But for propagation of new path from a_0 to a_m in ETN algorithms' T_{long} convergence in Theorem 4, the first message sent by a_i to a_{i+1} might not be a_1's eventual best path, thus this update would turn on the MRAI timer, and in the worst case, will delay the propagation of $rib^{new}(a_i)$ from a_i to a_{i+1} by $\mathcal{M}$ seconds. For both T_{down} and T_{long} in SPVP and SPVP-AS algorithms, suppose an invalid path is $(v_0v_1\cdots v_{i-1}v_i\cdots)$, the first message sent by v_{i-1} to v_i during convergence (which turns on the MRAI timer anyway) does not necessarily withdraw the path $(v_0v_1\cdots v_{i-1})$, because v_{i-1} might only learn this path later during the convergence. Therefore, in the worst case, the MRAI timer is on at each hop during the withdrawal of the path $(v_0v_1\cdots v_{i-1}v_i\cdots)$.

Finally, we define $h(G,[v\leftarrow u])$ as the sum of all the delays except MRAI delay. The U model and Q model differ in the modeling of $h(G,[v\leftarrow u])$. 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][9][6][7][8], assumes that all routing messages are processed within a bounded time, independent of network topology. In other words, the U model assigns a fixed upper bound, h, for each $h(G,[v\leftarrow u])$, defined at the end of Section 3. Therefore, depending on the algorithm, either $\mathcal{D}=\mathcal{M}+h$ or $\mathcal{D}=h$, regardless of the topology and node. In this section, we provide T_{down} and T_{long} convergence time bound under U model, and the results are summarized in Fig. 2 and Fig. 3.

$h(G,[v\leftarrow u])$	sum of all delays except MRAI delay
h	a fixed upper bound of $h(G,[v\leftarrow u])$
$\mathcal{M}$	Minimum Route Advertisement Interval

4.1 T_{down} Results

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

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

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.

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

$$time(T_{down}) \leq \mathcal{D} \cdot \max_{v,r \in R_v(G,A)} \{length(r)\}$$

Proof: Since $\mathcal{D}(G,[v \leftarrow u]) = \mathcal{D}$, a path r's lifetime becomes $f(r) = \mathcal{D} \cdot length(r)$. 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 receives is a withdrawal from node 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. Similarly, in T_{long} convergence, any invalid path can have at most one of node c's direct neighbors in V_A. Thus, for SPVP-AS, $\max_{r \in R(G,A)}\{length(r)\} = N - degree(G,0)$. For SPVP and SPVP-AS, $\mathcal{D} = \mathcal{M} + h$, while SPVP-GF has $\mathcal{D} = h$ 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,*

$$time(T_{down}) \leq h \cdot nodediameter(G,0)$$

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 $\mathcal{D} = h$. By definition and from Theorem 2, $\max_{v \in V}\{\min_{r \in R_v(G,A)}\{length(r)\}\} = nodediameter(G,0)$. ■

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	$(\mathcal{M}+h)\cdot min\{nodediameter(G',0)-J, \lvert V_A\rvert+diameter(G_A)-degree(G_A,c)\}$
*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	$(\mathcal{M}+2h)\cdot diameter(G_A)$

Fig. 3. T_{long} results under U model

4.2 T_{long} Results

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

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

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

For ETN algorithms, the similar procedure can be repeated, with $\mathcal{D} = h$ and $\mathcal{D}_{replace} = \mathcal{M}+h$. 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 message queueing delay at each node, and reveals important insights not previously obtained by the U model.

The limitation of U model is that it uses the same $h(G,[v \leftarrow u])$ for all nodes,

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

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

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 queueing delay estimate into $h(G, [v \leftarrow u])$ and reflects BGP implementations better. With the Q model, we can obtain tighter bounds of convergence time and new insights into topology structure and message processing delay's impact.

5.1 Queueing Delay

ld	upper bound of the sum of transmission and propagation delay on one link
p_{max}	maximum message processing time
$h(G, [v \leftarrow u])$	sum of ld, queueing delay and message processing time

The Q model uses ld to denote the 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. 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 queueing 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. Thus $h(G, [v \leftarrow u])$ equals to the sum of ld, queueing delay and processing delay.

If the message arrival rate is persistently higher than p_{max}, the queue will increase and result in very long delays [14]. 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 $withdrawal$, $announcement$, $withdrawal$. This observation allows us to obtain a bound $h(G, [v \leftarrow u])$.

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

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

Proof: Consider any sequence of 4 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 $t_4 - t1 > \mathcal{M}$, and since the link delay is between $(0, ld]$, we have $t_4 < t'_4 \leq t_4 + ld$ and $t_1 < t'_1 \leq t_1 + ld$. Therefore, $t'_4 - t'_1 > \mathcal{M} - ld$. ■

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

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

Suppose the Lemma is true for time period $[0, i \cdot (\mathcal{M} - ld))$, $i = 1, 2, 3...$, we examine the queue at any moment t between $[i \cdot (\mathcal{M} - ld), (i + 1) \cdot (\mathcal{M} - ld))$. At time $t' = t - (\mathcal{M} - ld)$, there are at most $3 \cdot degree(G, v)$ messages in the queue since t' falls in $[0, i \cdot (\mathcal{M} - ld))$. All these messages are processed within $3 \cdot degree(G, v) \cdot p_{max} < \mathcal{M} - ld$ 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 $3 \cdot degree(G, v)$, thus the hypothesis holds for $(i + 1)$. ■

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

Proof: Follows directly from Lemma 4. ■

Lemma 4 offers the first quantitative conditions under which messages will not queue up at a node. $\mathcal{M} > 3 \cdot degree(G, v) \cdot p_{max} + ld$ 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. [8]

[8] Note that in practice, the default setting of $\mathcal{M}$ is 30 seconds, 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 $degree(G, v) < 1000$. On the other hand, p_{max} can become large when the background routing load due to other prefixes is heavy.

By using per node $h(G, [v \leftarrow u]) = ld + 3 \cdot p_{max} \cdot degree(G, v)$ instead of a fixed number h, the Q model provides tighter bound for each convergence algorithm we have studied, and more insights of how topology affects convergence time.

Theorem 1 shows that the T_{down} convergence time of ITN algorithms is $time(T_{down}) \leq \max_{v, r \in R_v(G,A)}\{f(r)\}$. Under Q model, the lifetime of path $r = (v_l, v_{l-1}, \cdots, v_0)$ is $f(r) = \sum_{i=1}^{l}(\mathcal{M} + ld + 3p_{max} \cdot degree(G, v_i))$ for SPVP and SPVP-AS, and $f(r) = \sum_{i=1}^{l}(ld + 3p_{max} \cdot degree(G, v_i))$ 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, $time(T_{down}) \leq \sum_{i=1}^{N-1}(\mathcal{M} + ld + 3p_{max} \cdot degree(G, i)) = (N-1) \cdot (\mathcal{M} + ld) + 3p_{max} \cdot \sum_{i=1}^{N-1} degree(G, i) = (N-1) \cdot (\mathcal{M} + ld) + 3p_{max} \cdot (|E| - degree(G, 0))$; for SPVP-GF, $time(T_{down}) \leq \sum_{i=1}^{N-1}(ld + 3p_{max} \cdot degree(G, i)) = (N-1) \cdot ld + 3p_{max} \cdot (|E| - degree(G, 0))$. SPVP-AS restricts the invalid path to include only one of node 0's direct neighbors, therefore $time(T_{down}) \leq \sum_{v, where\ [v \leftarrow 0] \notin G}(\mathcal{M} + ld + 3p_{max} \cdot degree(G, v)) + (\mathcal{M} + ld + 3p_{max} \cdot max_{v \in G^0}\{degree(G^0, v)\}) = (N - degree(G, 0)) \cdot (\mathcal{M} + ld) + 3p_{max} \cdot (|E| - |E^0| + max_{v \in G^0}\{degree(G^0, v)\})$, where $G^0 = (V^0, E^0)$ is the subgraph consisting of node 0 and its direct neighbors. These results are summarized in Figure 4.

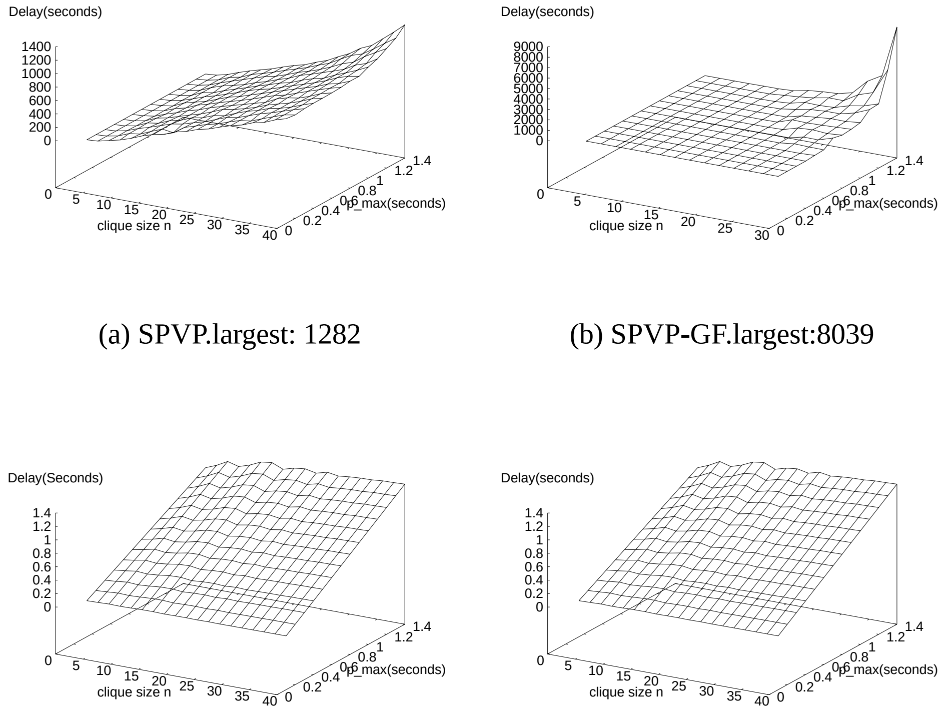
The results under U model (Fig. 2) imply the convergence time bound is proportional to the number of nodes in the network for SPVP, SPVP-GF, and SPVP-AS. However, the Q model reveals that each algorithm also has a term proportional to the number of links in the network, and this is an important hint in understanding the simulation results (Section 6).

For SPVP-RCN, since the first message received makes the receiver converged, queueing delay does not affect the convergence time. Thus $h(G, [v \leftarrow u]) \leq ld + p_{max}$ holds, and according to Theorem 2, $time(T_{down}) \leq nodediameter(G, 0) \cdot (ld + p_{max})$. 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 of technical report version of this paper [15].

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

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



(a) SPVP.largest: 1282

(b) SPVP-GF.largest:8039

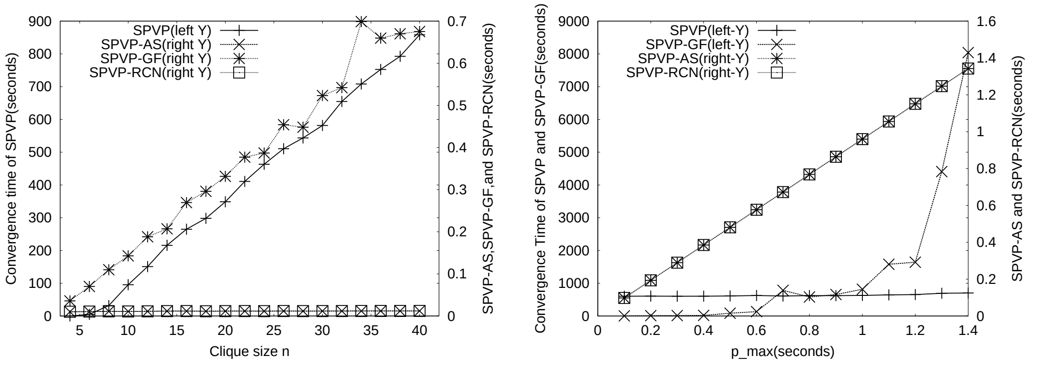
(c) SPVP-AS

(d) SPVP-RCN

Fig. 6. T_{down} simulation results with varying n and p_{max}.

6 Simulation Results

We conducted simulations using SSFNET [10]. The SSFNET simulator implements FIFO queue for incoming messages, which makes it suitable to verify our analytical results under Q model. Our parameter setting is $\mathcal{M} = 30s$, $ld = 0.002s$, $p_{min} = 0.001s$ and $p_{max} = 0.01s$, unless otherwise specified. Each data point represents the average over multiple simulation runs. Although our analysis provides only the upper bound of convergence time, as also done in previous work in



(a) n varies, $p_{max} = 0.01$

(b) p_{max} varies, $n = 30$

Fig. 7. T_{down} in $Clique(n)$

the literature [2][3][9][6][7][8], the insights from it help understand the simulation results 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 literature ([2][14] [6][7]) to study routing protocol's convergence properties. $Clique(n)$ often reflects the worst scenario because of its high connectivity: $degree(Clique(n), v) = n - 1$. Figure 5 shows the Q model T_{down} analytical results for $Clique(n)$. [9]

With $\mathcal{M} = 30s$ and $ld = 0.002s$ fixed, there are two variables: n and p_{max} in the analytical results. We vary both n and p_{max} together and show the results in Figure 6. In Figure 7(a), p_{max} is fixed while n varies, and in Figure 7(b) n is fixed while p_{max} varies. 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}.

[9] If we just plug the $Clique(n)$ parameters into SPVP-AS's result in Figure 4, we would get $(ld + \mathcal{M}) + 3p_{max} \cdot (n - 1)$. However, because the propagation 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 queueing delay, and the actual bound becomes $(ld + p_{max})$.

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 (Figure 6(a), the right-up corner, and Figure 6(b), the right half), and SPVP-GF's increase is even more dramatic (Figure 7(b)). This seemingly strange behavior is expected from Lemma 4. The lemma offers a quantitative condition under which messages will not queue up at a node v: if $\mathcal{M} - ld > 3 \cdot degree(G, v) \cdot p_{max}$. 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 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 ((Figure 7(b)).

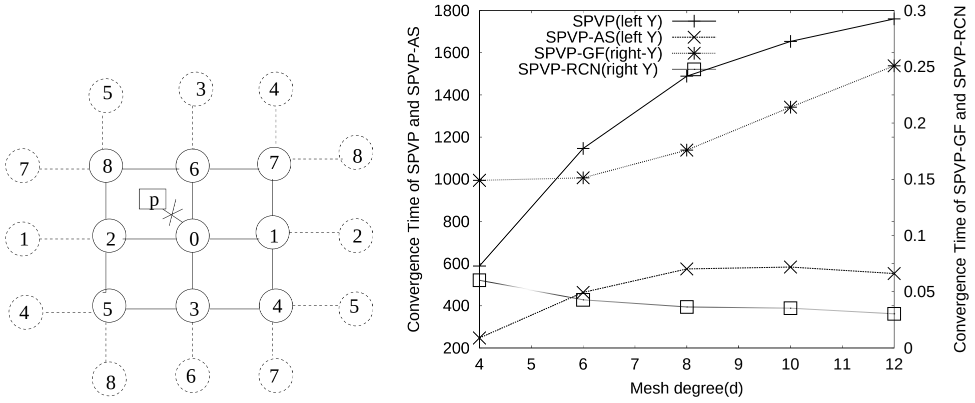
6.1.2 Grid

A $Grid(n, d)$ is a 2-dimensional n by n grid, whose nodes have the same degree of d. Figure 8 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 (Figure 2) would expect the convergence time fixed for all protocols since the network size $N = 100$ does not change. This demonstrates the great explanatory power of Q model because it takes into account how rich the network connectivity is.

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 [10] , 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

[10] Due to SSFNET's well-known simulation speed problem and demanding mememory requirement to simulate large network topology[**?**], we can only simulate a small AS-level topology



(a) T_{down} for $Grid(3,4)$

(b) $Grid(10,d)$

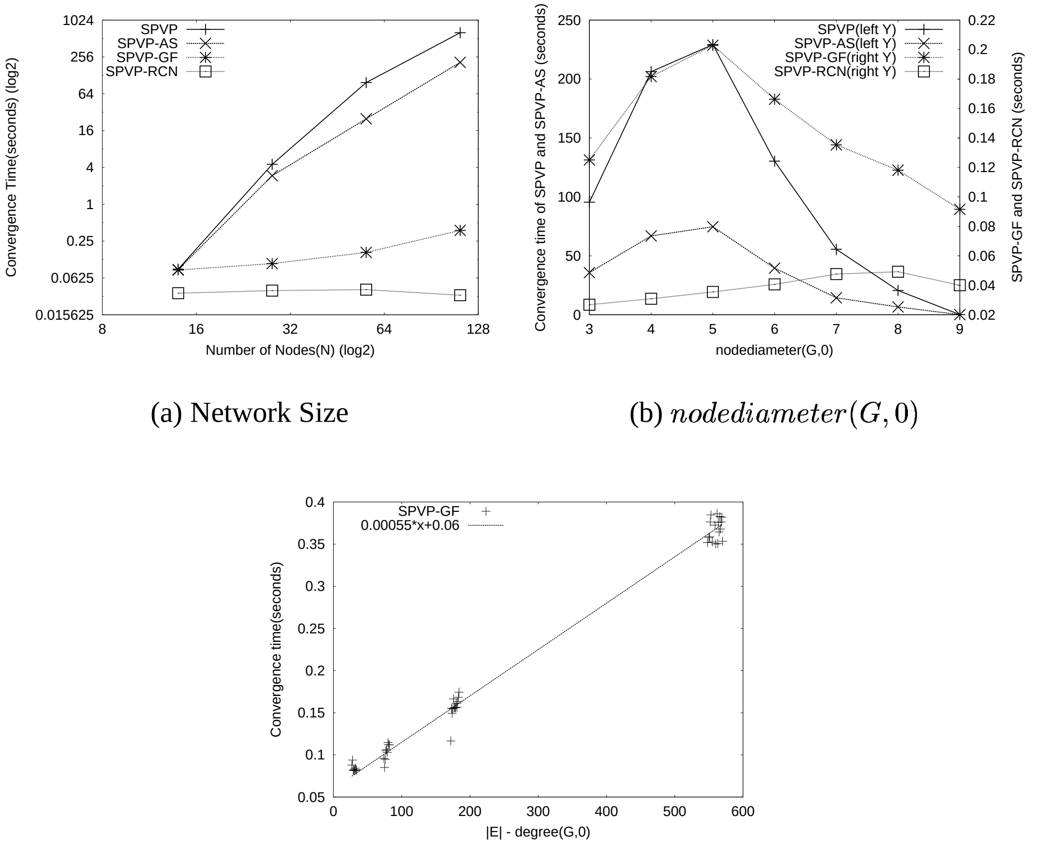
A	T_{down} for $Grid(n,d)$
SPVP	$(n^2-1)(ld+\mathcal{M})+3d(n^2-1)p_{max}$
SPVP-AS	$(n^2-d)(ld+\mathcal{M}+3dp_{max})$
SPVP-GF	$(n-1)ld+3d(n^2-1)p_{max}$
SPVP-RCN	$diameter(Grid(n,d))(ld+p_{max})$

(c) $Time(T_{down})$ for $Grid(n,d)$ under Q model

Fig. 8. T_{down} in $Grid(n,d)$

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 (Figure 4, the second column) show that N, $|E|-degree(G,0)$ 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, Figure 9(a) shows that the convergence time of SPVP-RCN and SPVP-GF are 2 to 3 order of magnitudes better than that of SPVP and SPVP-AS, which is because SPVP-RCN and SPVP-GF do not have $\mathcal{M}$ in their T_{down} convergence time. This performance difference is also confirmed by the results from $nodediameter(G,0)$ point of view in Figure 9(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 $(N-1)ld+3(|E|-degree(G,0))p_{max}$, and Figure 9(c) confirms that its conver-



(a) Network Size

(b) $nodediameter(G, 0)$

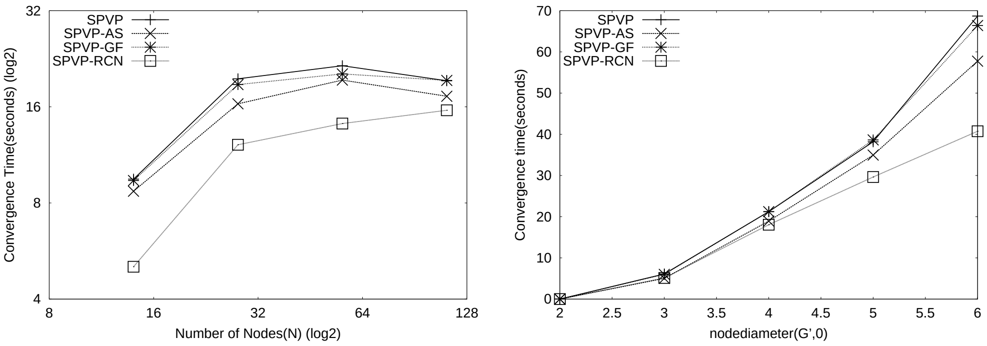
(c) $|E| - degree(G, 0)$ in SPVP-GF

Fig. 9. Simulation Results for T_{down} Convergence Time in Internet-like topologies.

gence time is indeed approximately proportional to $(|E| - degree(G, 0))$. These results demonstrate that although our analytical results are about the upper bound, insights obtained from them can help us understand the average case.

6.2 T_{long}

Prior to this work, there is a question about the T_{long} convergence time that have 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, which improves T_{down} significantly by reducing path exploration, can only improve T_{long} modestly in simulations [6][7]. For example, Figure 10(a) shows the averaged T_{long} convergence time versus the network size N in some Internet-like topologies. The results are average over various origin nodes and failure links, while $J = 1$ is kept. It is worth to note that SPVP performs well even in large network size, and none of SPVP-AS, SPVP-GF, or SPVP-RCN provides significant improvement.



(a) Network Size(log-log)

(b) $nodediameter(G', 0)$ in 110-node topology

Fig. 10. T_{long} in Internet-like Topologies.

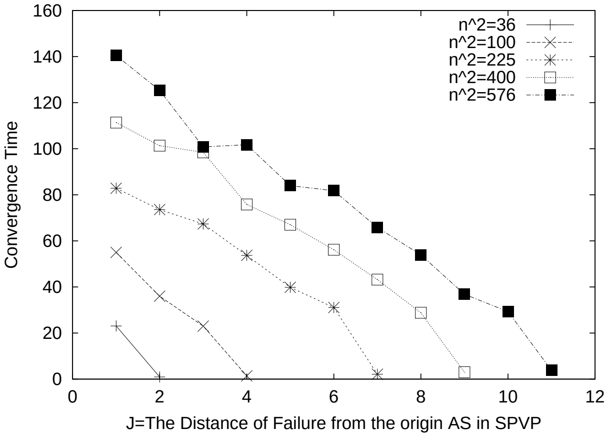


Fig. 11. SPVP T_{long} in Grid(n,4), n and J vary.

Our analysis enables us to provide the first quantitative explanation to the above phenomena. The analytical results of T_{long} delay under Q model is available presented in the Appendix of the extended version of this paper [15]. They are similar to the results under U model (Figure 3) in that the dominant factor in T_{long} convergence time is $nodediameter(G', 0)$ (Figure 10(b)). $nodediameter(G', 0)$, the longest distance to the prefix origin after the failure, is usually a small value in a well connected network, e.g., ≤ 9 in our simulation topologies. Therefore, the room for improvement by any algorithm is far less than that in T_{down} event. 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 3 times, resulting in very long T_{long} convergence time.

Our analysis in Figure 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 simulate SPVP T_{long} in $Grid(n, 4)$ topologies, while varying both $nodediameter(G', 0) = n$ and J. The results in Figure 11 show that the convergence time indeed is proportional to $-J$.

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 $(n-1)\mathcal{M}$ seconds. Further analysis by Labovitz *et al.* in [3] showed that T_{down} convergence delay is upper bounded by $(p \cdot \mathcal{M})$, where p is the length of the longest possible backup path. The above results were obtained using U model, ignoring the routing message queueing delay. Obradovic [9] developed a real-time BGP model which takes into account an edge delay similar to the definition of $\mathcal{D}(G, [v \leftarrow u])$. Based on this real-time model, the author showed that the T_{down} convergence time bound for the shortest-path-first 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 is missing in the above works. Our Q model also provides more accurate and insightful results.

The analysis of both Ghost Flushing [6] and RCN [7] uses the U delay model. Analysis with Q model can provide tighter delay bounds than that provided by these two works. In addition, our general analytical framework allows us to provides T_{long} results for SPVP-GF, which were missing previously. Simulation study using SSFNET by Griffin *et al.* [14] found that for each network topology there is an optimal $\mathcal{M}$ 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).

8 Conclusion

This paper presents a general framework for deriving and analyzing convergence delay bounds in path vector routing protocols, under the assumption of the shortest-path policy and single link failure. To the best of our knowledge, our framework is the first that applies to *all* the existing path vector protocol variants(standard path vector protocol and the convergence improvement algorithms), and take important factors into account, including network connectivity, failure location, and message processing delay. We also believe that our framework can be used to analyze new improvement algorithms of path vector routing protocols, should they occur.

Our framework allowed us to develop analytical bounds that were not previously available, i.e., T_{long} delay bounds 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 $nodediameter(G', 0) - J$, where J is the distance between the failure and the prefix origin and $nodediameter(G', 0)$ is the length of the longest alternate path used to reach the destination after the failure. This value is relatively small in a well-connected network such as today's Internet and this new insight explains why various proposed convergence improvement algorithms bring only modest improvement to the convergence delay of T_{long}. 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.

One simplifying assumption of the SPVP model is that each AS is treated as a single node. For future work we plan to extend the "message processing delay" in our Q model to a multiple border-router AS. We also plan to extend our analysis framework to analyze node failure and other policies. The framework presented in this paper provides a solid basis for understanding convergence delay of path vector algorithms and provides a foundation that enables these future works.

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, N. Nguyen, J. Chen, D. Massey, L. Zhang, BGP-RCN: Improving BGP Convergence Through Root Cause Notification, Tech. Rep. TR-030047, UCLA CSD, http://www.cs.ucla.edu/ peidan/bgp-rcn-tr.pdf (October 2003).

[8] J. Chandrashekar, Z. Duan, Z.-L. Zhang, J. Krasky, Limiting path exploration in path vector protocols, Tech. rep., University of Minnesota (2003).

[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).

[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, http://www.cs.ucla.edu/ peidan/convergence-tr.pdf (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.