Temporal Path Vector Routing Algorithm: TPV

Ken-ichiro Murakami†

Abstract

This paper describes a routing algorithm, called Temporal Path Vector (TPV), for improving the convergence properties of conventional path vector routing protocols. By employing time stamps in routing information, TPV detects obsolete information and purges it. Thus, it suppresses route bouncing and accelerates the routing convergence. Compared with a conventional routing protocol, Border Gateway Protocol (BGP), TPV takes less than one tenth as many steps to converge.

1. Introduction

Temporal Path Vector (TPV) is a routing algorithm developed for accelerating the convergence of conventional path vector routing algorithms used by the Border Gateway Protocol (BGP) [1][*1] and Label Distribution Protocol (LDP) [2].

BGP was originally developed to accelerate the convergence of global Internet routing. A routing node (i.e., a router) advertises its best path to its neighbors. A path consists of a series of node identification numbers (hereafter called node numbers) of the nodes that the path traversed. When a node advertises a path, it appends its own node number to the path. When a node receives a path, it scans the path for its node number. If the node finds that the path includes its own node number, it considers that the path forms a loop and invalidates it. After the inspection, it chooses the best path among the validated paths.

It has been believed that the algorithm suppresses loops and achieves fast convergence. However, a recent study [3] showed that convergence often takes a long time because of route bouncing (discussed in section 2). To cope with this issue, TPV was developed.

† NTT Network Innovation Laboratories
Musashino-shi, 180-8585 Japan
E-mail: murakami@core.ntt.co.jp

TPV employs time stamps in a path. When a node advertises a path, it appends the time stamp to the path as well as its node number. The stamp shows when the path was received at the node. When a node receives a path, it inspects the time stamps as well as the node numbers in the path. If an old time stamp is included, the node considers this path to be obsolete and invalidates it. Thus, TPV purges obsolete paths and accelerates routing convergence. To show the effectiveness of TPV, it was applied to the well-known slow-convergence problem of BGP, described in [1]. The result shows that it takes less than one tenth of the steps that BGP requires.

2. Issues with conventional algorithms

This section discusses the delayed convergence of BGP and explains the cause: route bouncing.

2.1 Delayed convergence

The Path Vector (PV) routing algorithm was developed to address the issues with the conventional Distance Vector (DV)[*2] algorithm. DV selects the best path based solely on the distance to the destination,

*1 BGP is the routing protocol used to exchange routing information across the Internet between autonomous systems (ASs) such as Internet service providers. Each AS has a unique identification number. The routing information carries a sequence of the AS numbers that it has traversed.

making it prone to routing loops and route bouncing. To cope with these problems, PV employs path information. When a path traverses a node, the node appends its node number to the path. Thus, a receiving node can know whether or not the path forms a loop. As the best path, it selects the shortest path among non-looped ones. Therefore, BGP was believed to exclude invalid paths and achieve fast convergence.

However, a recent study [3] showed the nature of BGP convergence, as shown in **Fig. 1**. Although it is believed that convergence typically completes within tens of seconds, the study showed that a failover (a switchover in response to a failure) takes between three to fifteen minutes to converge. The cause is multiple route bouncing. A node selects the best path from the best paths reported by neighboring nodes. The selected best path is reported to the neighbors, triggering best path selection in them. That is, there is no synchronization among nodes involved in calculating the best path to the same destination. Thus, the latest path information may frequently be superseded by obsolete information from other nodes and vice versa. This iteration results in delayed convergence. In addition, it increases the number of update messages for carrying path information and consumes bandwidth as well as CPU and memory resources.

2.2 Example of route bouncing

As an example of route bouncing, consider the simple ring network shown in **Fig. 2**. In this network, we concentrate on paths to node N4. Each node caches received paths in its route cache (RC). The best path is selected from the paths in the RC and placed in the routing table (RT). Even if a path is not selected as the best one, it is kept in RC. When the best path is invalidated, a node can select the second-best path immediately. When a new path appears or an existing path is withdrawn, the node sends an update message indicating the change to its neighbors. A node forwards data packets based on RT. It searches for an entry that matches the destination of the packet and passes the packet along with the path in the entry.

In the steady state, RTs and RCs are as follows.

Node N1
RC: (N4)
RT: (N4)

Node N2
RC: (N1 N4), (N3, N1, N4)
RT: (N1 N4)

Node N3
RC: (N1 N4), (N2 N1 N4)
RT: (N1 N4)

Each pair of parentheses indicates a path and each element in a path indicates the nodes that the path information traversed. The rightmost node is the origin of the path, that is, the destination. As the path traverses a node, its node number is appended to the leftmost position.

Assume that link N4-N1 goes down. A node sends update messages to its neighboring nodes when the best route is withdrawn from RT. First, N1 sends update messages to both N2 and N3 indicating the withdrawal of the path (N1 N4). These messages are marked (1) in Fig. 2. We assume that N3 receives the

*2 Distance Vector (DV): In the DV algorithm, each router maintains a table giving the shortest distance to a destination and the next hop. It advertises the table to its neighbors. When a router receives this information, it compares the distance in its own table with the received value plus the distance to the sender for each destination, selects the shortest route, and advertises the result.

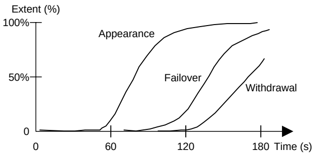

Fig. 1. Slow convergence in BGP.

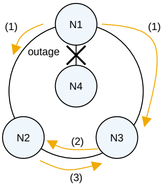

Fig. 2. Route bouncing.

message earlier than N2.

N3 invalidates the path (N1 N4) in RC and RT. Then, it selects an alternative path (N2 N1 N4) in RC as the best path and places it in RT. Since the former best path (N1 N4) has been replaced by the new best path (N2 N1 N4), N3 sends N2 an update message indicating the withdrawal of (N1 N4) and appearance of (N2 N1 N4). This message is marked (2) in Fig. 2.

Node N3
  RC: (N2 N1 N4)
  RT: (N2 N1 N4)

N2 receives the message and processes it. That is, it removes the withdrawn path (N3 N1 N4) from RT and RC. Then, it stores the new path (N3 N2 N1 N4) to RC and inspects it to see whether it is valid. Since its own node number is included in the path, it marks the path as invalid. Next, it looks for an alternative best path in RC. Path (N1 N4) is selected as the best path and stored in RT.

Node N2
  RC: (N3 N2 N1 N4), (N1, N4)
  RT: (N1 N4)

N2 receives an update from N1 indicating the withdrawal of path (N1 N4). It invalidates this path in RT and RC. Although it looks for an alternative best path, there is no valid path in RC. Since path (N1 N4) has been withdrawn from RT and there is no valid path in RC, N2 send an update message to N3 indicating the withdrawal of path (N1 N4). This message finally invalidates path (N2 N1 N4), which resides in N3.

Node N2
  RC: (N3 N2 N1 N4)
  RT: none

Node N3
  RC: none
  RT: none

As described above, N3 initially adopts an invalid path (N2 N1 N4) and then withdraws it. This is an example of route bouncing. Multiple route bouncing delays convergence. An example that requires 48 steps to converge for a small network with four nodes is shown in [3]. In a larger mesh network with a lot of routers connected with long-latency links, the slow convergence problem is worse. To cope with this issue, TPV uses a time stamp to enable nodes to identify and purge obsolete paths.

3. TPV

This section describes the basic concept of virtual time and explains how a received path is processed.

3.1 Time stamp

In TPV, a path has a series of time stamps as a path attribute. Each time stamp is associated with an element, that is, a node number, in the path and indicates when the path traversed the associated node. However, a simple time stamp alone does not solve the slow convergence problem. Time in TPV has the following characteristics.

(1) Virtual time

If real time were employed for time stamps, the resolution would become an issue. For example, link up/down events might occur within a short period and have the same time stamp. In contrast, a fine-grained clock would frequently overflow, making time comparisons difficult. To resolve this problem, TPV employs virtual time (VT). It clocks every event such as link up and down as shown in **Fig. 3**.

(2) Time stamp at the receiving port

A node has ports and a port is connected to a neighboring node's port through a link. When a link failure occurs, adding a time stamp at a sending node causes a problem. That is, the sending node cannot send the path information to the receiving node because of the link failure. To cope with this, TPV stamps a path at the receiving port with the VT associated with the port. This ensures that every path has a complete series of time stamps in any event.

(3) Multiple local clocks

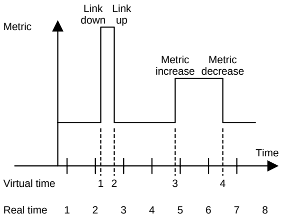

Fig. 3. Virtual time.

Events such as link up and down are local events in a port associated with the link. Thus, a node has multiple local clocks, one for every port.

3.2 Handling a received path

When a node receives a path, it adds its node number and VT to the path. Then, it stores the path in RC. Each node maintains a virtual time table (VTT), which is a collection of the latest time stamps. If a received path has the latest time stamps for a path, the node updates VTT with them. Conversely, the node marks the path as invalid if it has an obsolete time stamp.

VTT consists of entries associated with a link. In the case of a multiple-access link, each entry corresponds to a pair of sending and receiving ports of the link. In the case of a point-to-point link, the receiving port number is enough to identify the link. The received path inspection flow is described in detail below (**Fig. 4**).

(1) Invalidate an obsolete path

A node inspects all the time stamps included in a received path. That is, it compares each time stamp against the corresponding time stamp in VTT. If it finds an obsolete value in the path, the node marks the path as obsolete and makes it invisible to the best path selection process.

(2) Update VTT

In process (1), if the node finds the latest time stamp or a new time stamp in the received path, it updates the corresponding VTT entry with the VT value. If a VTT entry is updated, the node inspects all the paths in RC and RT. If it finds an obsolete path in RC or RT, it marks the path as invalid.

(3) Select the best path

If a new valid path has appeared in RC or the best path in RT has been invalidated because of a withdrawal message or obsolete time stamp, a new best path is selected from among the valid paths in RC. If a new path is selected and placed in RT, it is advertised to neighboring nodes by an update message indicating the withdrawal of the conventional best path (i.e., the best path in the conventional sense) and the appearance of the new best path.

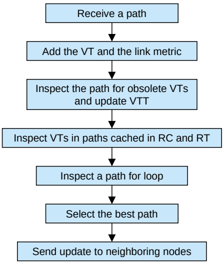
Receive a path
Add the VT and the link metric
Inspect the path for obsolete VTs and update VTT
Inspect VTs in paths cached in RC and RT
Inspect a path for loop
Select the best path
Send update to neighboring nodes

Fig. 4. Path inspection flow.

3.3 Example

Here, we consider the convergence process of TPV. We assume the same network as in Fig. 2 and focus on paths to destination N4. In the steady state, each node has the following paths to N4.

Node N1
RC:
((N4 $VT_{14}0$ $M_{14}0$))
RT:
((N4 $VT_{14}0$ $M_{14}0$))

Node N2
RC:
((N1 $VT_{21}0$ $M_{21}0$) (N4 $VT_{14}0$ $M_{14}0$))
((N3 $VT_{23}0$ $M_{23}0$) (N1 $VT_{31}0$ $M_{31}0$) (N4 $VT_{14}0$ $M_{14}0$))
RT:
((N1 $VT_{21}0$ $M_{21}0$) (N4 $VT_{14}0$ $M_{14}0$))

Node N3
RC:
((N1 $VT_{31}0$ $M_{31}0$) (N4 $VT_{14}0$ $M_{14}0$))
((N2 $VT_{32}0$ $M_{32}0$) (N1 $VT_{21}0$ $M_{21}0$) (N4 $VT_{14}0$ $M_{14}0$))
RT:
((N1 $VT_{31}0$ $M_{31}0$) (N4 $VT_{14}0$ $M_{14}0$))

The outermost pair of parentheses indicates a path. Each pair of parentheses inside indicates an element in the path. The rightmost element indicates the destination, that is, the origin of the path. An element consists of a received node number, the VT value at the received port, and the link metric at the port:

(Node number, VT, link metric).

A node appends an element when it receives a path. Here, we represent VT as $VT_{lm}N$, where VT_{lm} and N indicate VT at node Nl associated with the link from

Nm and the virtual time ID, respectively. The ID is incremented for every event.

The initial VTT entries in each node are follows.

Node N1:
$(VT_{14}0)$

Node N2:
$(VT_{21}0\ VT_{14}0\ VT_{23}0\ VT_{31}0)$

Node N3:
$(VT_{31}0\ VT_{14}0\ VT_{32}0\ VT_{21}0)$

Assume that the link N4-N1 goes down. N1 sends update messages indicating a link failure in link N4-N1. They are marked (1) in Fig. 2. Since the initial value of VT at link N4-N1 is $VT_{14}0$, the value after the failure is $VT_{14}1$. Thus, the message is

withdraw $(N4\ VT_{14}1\ M_{14}1=\infty)$.

This means that the path (N4) is withdrawn because an event occurred at $VT_{14}1$. The metric of the link $M_{14}1$ became ∞ because of the event.

Assume that N3 receives the message earlier than N2. N3 immediately invalidates the conventional path (N1, N4) in RC and RT. Then, it selects the new path. However, it finds that the candidate (N2 N1 N4) includes the obsolete time stamp $VT_{14}0$. Thus, it invalidates this path.

Node N3
RC:
$((N2\ VT_{32}0\ M_{32}0)\ (N1\ VT_{21}0\ M_{21}0)\ (N4\ VT_{14}0\ M_{14}0))$
RT: none

Since no best path is available, it sends an update message indicating the withdrawal of the conventional best path:

withdraw $((N1\ VT_{31}0\ M_{31}0), (N4\ VT_{14}1\ M_{14}1=\infty))$.

The message is sent to N2. When N2 receives the message, it removes the path $((N3\ VT_{23}0\ M_{23}0), (N1\ VT_{31}0\ M_{31}0), (N4\ VT_{14}0\ M_{14}0))$ in RC and updates VTT. The updated VTT is

$(VT_{21}0\ VT_{14}1\ VT_{23}0\ VT_{31}0)$.

In parallel with this, N2 inspects the conventional best path in RT and RC. It finds that it includes the obsolete time stamp $VT_{14}0$. Thus, it is invalidated.

Node N2
RC:
$((N1\ VT_{21}0\ M_{21}0)\ (N4\ VT_{14}0\ M_{14}0))$
RT: none

When it receives an update indicating the withdrawal of $((N1\ VT_{21}0\ M_{21}0)\ (N4\ VT_{14}0\ M_{14}0))$ from N1, the route has already been invalidated.

As described above, TPV enables nodes to identify the time and place of an event, so they can purge obsolete paths and protect the latest paths from being superseded by obsolete ones. Therefore, route bouncing is suppressed and convergence is accelerated.

In addition, suppressed route bouncing inactivates the BGP MinRouteAdver timer [1], which was introduced to suppress excessive updates that cause instability and a computational surge. After a node updates a path, the advertisement of a path associated with the same destination is suspended for thirty seconds by default. Since no MinRouteAdver timer is invoked in TPV, TPV can accelerate the convergence by more than one hundred times.

4. Comparison with BGP

This section presents simulation results for TPV and shows its fast convergence compared with BGP.

4.1 Simulated network

We simulated the same simple four-node network as in [3] with the same scenario. TPV drastically improved the convergence compared with BGP, taking only four steps instead of 48.

The simulated network is shown in **Fig. 5**. Nn denotes a node with node number n. Pm denotes a local port numbered m in a node. In this simulation, we focused on the best paths to N3. Assume that N3

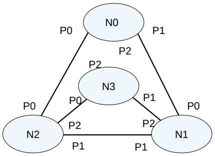

Fig. 5. Simulated network.

Table 1. Initial best paths for N3.

Node	Best path
N0	(N3)
N1	(N3)
N2	(N3)

Table 3. Changes in best paths in TPV.

Node	Step 1	Step 2	Step 3	Step 4
N0	(N1 N3)	(N1 N3)	(N2 N3)	none
N1	(N0 N3)	(N2 N3)	(N2 N3)	none
N2	(N0 N3)	(N1 N3)	none	none

Table 2. Changes in best paths in BGP.

Node	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7-47	Step 48
N0	(N1 N3)	(N1 N3)	(N2 N3)	none	(N1 N2 N3)	(N1 N2 N3)	omitted	none
N1	(N0 N3)	(N2 N3)	(N2 N3)	(N2 N0 N3)	(N2 N0 N3)	none	omitted	none
N2	(N0 N3)	(N1 N3)	(N0 N1 N3)	(N0 N1 N3)	(N0 N1 N3)	(N0 N1 N3)	omitted	none

goes down. After the outage, nodes start exchanging routing information and finally all the paths to N3 disappear in all the nodes.

4.2 Simulation results

Table 1 shows the initial best paths in each node. In the initial state, every node selects the path through the direct links to N3 since these paths are the shortest ones.

Table 2 shows the change in the best paths in BGP. The rightmost element in the parentheses is the destination, that is, the origin of the routing information. For example, (N0 N3) stands for the path to N3 through N0. Note that Table 2 omits the best path changes between steps 7 and 47. In each step, no node adopted a looped path. However, some nodes did adopt obsolete or inconsistent paths, resulting in a loop between nodes. In addition, even if paths to N3 once disappeared in a node, they get reincarnated. Since nodes update their RTs asynchronously, obsolete paths remain in RTs, RCs, and receive buffers. Nodes adopt these obsolete paths with each other.

Table 3 shows the convergence in TPV. TPV ensures that obsolete paths do not supersede new ones by inspecting VTs in paths. In steps 1 and 2, cached paths in RC are adopted and temporary loops, N0<->N1 and N1<->N2, are formed. Once nodes start exchanging update messages, these loops disappear immediately. They are created only when all the available paths are lost simultaneously because of a node failure. However, this is a rare case since usual link failure does not invalidate all the alternative paths. In this case, no loop is formed, as shown in subsection 3.3.

5. Conclusion

The temporal path vector algorithm TPV accelerates the convergence of conventional path vector protocols such as BGP. It employs a time stamp as a path attribute, enabling a node to identify obsolete paths and purge them, protecting new paths from being superseded inadvertently.

TPV uses virtual time for its time stamp. It clocks every event such as link up and down. Thus, unlike a real time clock, event orders can be distinguished. The time is stamped at a receiving port, so even if the sending node or the link goes down, the receiving node can stamp the time on the path.

To show the effectiveness of TPV, it was applied to the well-known slow-convergence problem of BGP, described in [1]. The result showed that it took less than one tenth of the steps that BGP requires.

References

[1] Y. Rekhterand and T. Li, “A Border Gateway Protocol 4 (BGP-4),” RFC1771, 1995.

[2] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, “LDP Specification,” RFC3036, 2001.

[3] C. Labovitz, A. Ahuja, and A. Bose, “Delayed Internet Routing Convergence,” Proceedings ACM SIGCOMM’2000, Computer Communications Review, Stockholm, Sweden, Vol. 30, No. 4, pp. 175-187, Aug. 2000.

Ken-ichiro Murakami

Senior Research Engineer, Supervisor, NTT Network Innovation Laboratories.

He received the B.E and M.E. degrees in computer science and communications engineering from Kyushu University, Fukuoka in 1979 and 1981, respectively. He joined Nippon Telegraph and Telephone Public Corporation (now NTT) in 1981 and engaged in the development of large-scale operating systems and network systems of LISP machines. He received the Ph.D. degree in information science from Waseda University, Tokyo in 2003. He is a member of the Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers, and the Association for Computing Machinery.