

A MORE EFFICIENT DISTANCE VECTOR ROUTING ALGORITHM

Zhengyu Xu

Sa Dai

J. J. Garcia-Luna-Aceves

Computer Engineering Department
School of Engineering
University of California
Santa Cruz, CA 95064
sdai, jj, zxu@cse.ucsc.edu

Abstract

A more efficient distance vector routing algorithm (EDVA) for computer networks is presented. EDVA is based on enhancements to two classes of distance vector algorithms: the path finding algorithms that report complete path information incrementally, and the diffusing update algorithm (DUAL) which is used in Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP) and is based on diffusing computations for internodal synchronization. EDVA operates by specifying a distance, a feasible distance, a predecessor and diffusing hops to each destination. This information is used together with diffusing computations of limited span to avoid routing-table loops. The correctness of EDVA is discussed, and its benefits compared with DUAL and path finding algorithms are illustrated with simple examples.

1. Introduction

The Routing Information Protocol (RIP) and RIP version 2 [7], [8] are based on the Distributed Bellman-Ford algorithm (DBF) [2] for shortest path computation. Although DBF is very simple to implement, it is very inefficient because of the bouncing and counting-to-infinity problems, which cause data packets traverse loops indefinitely. The current Internet routing protocols based on the exchange of distance information that overcomes these problems is the Enhanced Interior Gateway Routing Protocol (EIGRP) [1]. This protocol is based on the diffusing update algorithm (DUAL) [4]. Like DBF, DUAL uses distances to destinations to operate; however, it synchronizes routers that modify their distance information in a way that no long-term or temporary loops are ever created in routing tables.

Other algorithms based on the exchange of distance information are path finding algorithms [3], [6], [10], [11], [12], [14]. According to these algorithms, a router reports the distance and second-to-last hop to each destination; this information allows each router to transmit its entire shortest-path routing tree incrementally to its neighbors. With such information, all path-finding algorithms eliminate long-term routing-table loops; in addition, temporary routing table loops are eliminated by means of interneighbor synchronization mechanisms in the Loop-free Path-Finding algorithm (LPA) [5], [9], [11], which has been shown to be more efficient than DUAL and the topology broadcast algorithms used in OSPF.

In this paper, we propose a new algorithm that combines loop avoidance techniques used in DUAL and path finding algorithms to provide loop-free shortest-

routing more efficiently than those algorithms. We call this algorithm the "Enhanced Distance Vector Algorithm (EDVA)". EDVA relaxes the constraints used in DUAL and LPA for a router to determine when to synchronize the updating of a routing-table entry with its neighbors. In addition, once a router decides to synchronize its routing-table update activity with the updating activity of its neighbor routers, the amount of synchronization used can be adjusted based on network characteristics. For brevity, EDVA is presented as a modification of a path finding algorithm called PFA [10], [12], LPA and DUAL.

Section 2 presents the network model used in EDVA's description. Section 3 presents the information used in EDVA. Section 4 describes EDVA's operation. Section 5 discusses a few examples of EDVA's operation.

2. General Network Model

A computer network is modeled as a finite connected graph in which each link bidirectionally connects two nodes and has a positive cost in each direction. Each node represents a router that has an independent computing unit, storage, I/O, and queues with unlimited capacity. Input and output routing messages are processed within a finite time in FCFS order at each node. We assume the link-level protocol to be correct. Therefore, a router can detect within a finite time the state of any of its adjacent links. A link state includes the cost of the link and the existence of a neighbor. A router executes the routing algorithm correctly. Each router has a unique identifier. The distance between two nodes is the sum of the cost along the shortest path, i.e., a path of minimum cost.

3. Information Exchanged and Maintained in EDVA

Each router maintains a link-cost table, a distance table and a routing table. The link-cost table lists the cost of each link adjacent to the router. The cost of the link from i to n is denoted by L_{ki} and is considered to be infinity when the link fails.

The distance table at each router i is a matrix containing, for each destination j and for each neighbor n of router i , the distance, the feasible distance (defined subsequently), the initial diffusing step number (defined subsequently), the successor and the predecessor reported by router n . As Table 1 states, these variables are denoted by D_{jn} , FD_{jn} , IM_{jn} , S_{jn} and P_{jn} , respectively.

The routing table at router i is a column vector containing, for each destination j , the minimum distance (denoted by D_{ji}), the feasible distance (denoted by FD_{ji}),

the predecessor (denoted by P_{ji}), the successor (denoted by S_{ji}), the initial diffusing step number (denoted by IM_{ji}), diffusing step number (denoted by M_{ji}) and a marker (denoted by TAG_{ji}) used to update the routing table. For destination j , TAG_{ji} specifies whether the entry corresponds to a simple path ($TAG_{ji} = \text{correct}$), a loop ($TAG_{ji} = \text{error}$) or a destination that has not been marked ($TAG_{ji} = \text{null}$).

Table 1. Notation

i :	the node we focus on, which is running EDVA at time t .
j :	the destination node that we are considering.
n, x, y, \dots :	different neighbors of node i .
N_i :	the set of neighbors of i .

D :	the abbreviation of "distance"
D_{ji} :	distance from i to j , known by node i .
D_{jn} :	distance from n to j , known by node i .
FD :	the abbreviation of "feasible distance".
FD_{ji} :	feasible distance from i to j , implicitly known by node i .
FD_{jn} :	feasible distance from n to j , implicitly known by node i .
IM :	the initial value of M .
L_{ni} :	cost of link from i to n known by i .
M :	the abbreviation of "the number of the adjustable diffusion steps".
M_{ji} :	the number of the adjustable diffusion steps required by i for destination j known by node i .
M_{jn} :	the number of the adjustable diffusion steps required by neighbor n for destination j known by node i .
P_{jn} :	the predecessor (second-to last hop) for n to the destination j known by i .
S_{jn} :	the successor of n to j known by i .
C_{jn} :	The Chain (path) from n to j known by i .

If any routing information is changed at a router, it sends this information in an update message to certain neighbors. An update message from router i can consist of one or more entries for one or more destinations, respectively. Each entry specifies an update flag (denoted by U_{ji}), a destination, the predecessor, the distance, and the feasible distance for that destination. The update flag indicates whether the entry is an update ($U_{ji} = 0$), a query ($U_{ji} = 1$) or a reply to a query ($U_{ji} = 2$).

In the specification of EDVA, the successor to destination j for a router is simply referred to as the successor of the router, and the same reference applies to other information maintained by a router. Similarly, updates, queries and replies refer to destination j , unless stated otherwise.

When router i receives an input event regarding its neighbor n (an update message from neighbor n or a change in the cost or status of link (i, n)), it updates its destination table and link-cost table accordingly.

4. Updating The Routing Table

4.1 Principles of Operation

EDVA is based on two basic ideas. First, it uses predecessor information for each destination [5], [10], [11], [12]. Because the second-to-last hop is specified for each destination, each router can infer if its path corresponding to a distance-table or routing-table entry includes the router itself, i.e., if there is a loop in the path offered by a neighbor. This feature eliminates the counting-to-infinity problem present in DBF. Furthermore, a router detects a temporary loop within a finite time that depends on the speed with which correct predecessor information reaches the router, and not on the distance values of the paths offered by its neighbors; therefore, temporary loops are detected much faster than in DBF and its variations [12].

The second main idea in EDVA is blocking temporary (quasi-routing) loops using an inter-nodal synchronization method that is an extension of the synchronization mechanisms of DUAL [4] and LPA [5]. EDVA relaxes the "feasibility condition" first introduced for DUAL and used by a router to decide when to synchronize its activity with its neighbors' activities. Once a router decides to synchronize with its neighbors, the results of its query can impact one or multiple hops. The span of a router's query can be controlled using the number of adjustable "diffusing steps" included in queries.

In EDVA, a router i that decides that a loop might be formed if it changed its successor to destination j is asked to block such a loop by sending its information to certain neighbors in a query containing the associated information to j in the routing table, and by waiting for those neighbors to acknowledge its message with replies, which contain information for the same destination j in their own routing tables. Because of the overhead involved, a router should not send a query every time it has to change its successor to a destination. The router is forced to send a query to block a potential loop only when none of its neighbors satisfies the feasibility condition defined below. How router i decides to update its routing table for a given destination depends on the state it has for that destination. In EDVA, a router can be in one of two states for a given destination: passive or active.

A router is passive if it has a feasible successor, or has determined that no such successor exists and is active if it is searching for a feasible successor. A feasible successor for router i with respect to destination j is a neighbor router that satisfies the feasibility condition defined subsequently. When router i is passive, it reports its current routing information in its updates and replies. However, while router i is active, it sends its modified routing information in its replies and queries. An active router cannot send a new update regarding the destination for which it is active.

4.2 Passive State of a Router

Router i is passive for destination j if there is a neighbor n , called feasible successor, satisfying the revised feasibility condition (RFC) defined by the following two equations:

$$FD_{jn} < FD_{ji} \quad (1)$$

$$D_{jn} + L_{ni} = D_{min} \\ = \min \{ D_{jy} + L_{iy} \mid y \text{ in } N_i \text{ and } i \text{ not in } C_{jy} \} \quad (2)$$

where: C_{jy} is the implicit path from y to j known by i , FD_{jn} is the feasible distance of i 's neighbor n to destination j known by i , and FD_{ji} is feasible distance of node i . FD_{ji} can only be reduced during the passive state, usually by setting $FD_{ji} = \min \{ FD_{ji}, FD_{jn} + L_{ni} \}$. FD_{ji} is initialized with an infinite value at the beginning of each new passive state.

In EDVA as in DUAL, the feasible distance of a node to a destination is a value that establishes an ordering of nodes along loop-free paths. EDVA guarantees that, if a node n can be in the path from node i to destination j , then the feasible distance of node n is strictly smaller than the feasible distance of node i for destination j .

The implicit path C_{jy} is obtained by node i by traversing the routing information obtained from its neighbor y . For example, if $C_{jy} = y-a-i-b-j$, node i infers this path by obtaining the predecessor (a) of y to destination i , the predecessor (i) of y to destination b , and the predecessor (b) of y to destination j . In this example, node i is in the path C_{jy} , and if node i chose y as its successor to j , a loop would be formed. Therefore, node y is excluded as a candidate successor in Eq (2).

When node i needs to update its current successor, it can choose as its new successor, S_{ji} , any router n in its neighbor set N_i , such that RFC is satisfied. If $D_{min} = \text{infinity}$, then $S_{ji} = \text{null}$. After updating its routing table, router i prepares an update to its neighbors if its routing table entry changes. If any entry in its routing-table changes during the passive state, i sends an update message to certain neighbors, which are defined in Section 4.4.

4.3 Active State of a Router

Node i becomes active for destination j once it cannot find a feasible successor. During the active state, node i first follows a strategy that we call Active Changes (AC). There are two independent aspects in AC. First, if i becomes active, i can make z its new temporary successor if:

$$D_{jz} + L_{zi} = \min \{ D_{jy} + L_{yi} \mid y \text{ in } N_i \text{ and } i \text{ not in } C_{jy} \text{ and } FD_{jy} < FD_{ji} \} \quad (3)$$

Since $FD_{jz} < FD_{ji}$, making z the new successor can not form a routing-table loop, but provides the currently available minimum cost during the active state.

In the second part of AC, independently of the first part, node i sends a query for destination j to certain neighbors (see Section 4.4) with the adjustable distance $D_{ji} = D_{jn} + L$ and $FD_{ji} = FD_{jn} + FL$, in which L and FL can independently be set in the range from L_{ni} (as it is done in DUAL) to infinity (as it is done in LPA), where n is a neighbor that satisfies Eq.(2) in RFC. Node i 's query also includes any changed routing information.

The link to the neighbors that should but have not replied to node i (detected by the link-level protocol) after the time out, is treated as infinity. In a finite time, an active

node i receives the replies (or detect the link failure) from the neighbors to whom it sent queries. After that, node i sets FD_{ji} to infinity, becomes passive again and behaves as described in Section 4.2. An example of this operation is shown in Section 5.2.

4.4 Neighbors Needing Updated Routing Information During Passive and Active States

Which neighbors should be sent queries during node i 's active state and be sent updates during the passive state, without increasing any possibility of loop? EDVA uses a strategy that we briefly call the smaller-announces-to-the-larger-rule(SAL). That is: node i updates/queries node n only if node i has a value of FD_{ji} in the last passive state that is smaller than or *equal* to the value FD_{jn} of a neighbor n (i.e., $FD_{ji} \leq FD_{jn}$) in the last update or reply of n and the initial value of the adjustable number of diffusing steps $IM_{jn} \neq 0$ (see Section 4.5).

When $IM_{jn} = 0$, node n will never initiate a query (see Section 4.5), so it always needs the update messages of neighbors, this is excluded by SAL. SAL is used because node i 's neighbor n with a feasible distance $FD_{jn} < FD_{ji}$ cannot be involved in any potential loop due to node i 's active state. Also, if node n wants to increase its feasible distance, it will have to first query node i even if node i has already sent to node n a query or update, which does not affect n 's decision of increasing its feasible distance. An example is shown in Section 5.3.

4.5 Adjustable Diffusing Steps (ADS)

The motivation for this mechanism stems from the following considerations. PFA does not use queries, which produces less overhead but allows temporary loops; LPA sends queries that span only one hop, and DUAL queries all the hops affected by a change. There should exist a way in which the best features of the algorithms can be exploited to obtain better performance than that of the original algorithms. To achieve a better tradeoff of speed versus loop freedom, EDVA uses the following adjustable diffusing step (ADS) mechanism:

With respect to DUAL, LPA and PFA, we add two more diffusing counter numbers: M_{ji} and IM_{ji} ($= 0, 1, 2, \dots, \text{infinity}$) into the query message of node i for destination j . The initial value of M_{ji} , IM_{ji} , is independent on any neighbor n 's query with any M_{jn} . IM_{ji} is decided by node i itself according to node's current conditions. When $IM_{ji} = 0$, i does not query if no other query (i.e., $M_{jn} = 0$, described subsequently) causes it to query; if node i needs to update its routing table, it first sets FD_{ji} to infinity to keep its passive state, instead of checking RFC first. Note that, if all nodes have $IM = 0$, EDVA has the same behavior of PFA. $IM_{ji} = 1$ makes EDVA use single-hop queries similar to LPA for node i to destination j . If queries specify $D = \text{infinity}$ and $IM = 1$ for all nodes, EDVA behaves much like LPA. If all $IM = \text{infinity}$, EDVA operates like DUAL.

When node i is in passive state and receives a query regarding destination j from neighbor n , then it behaves in the following manner:

- (a) If node i finds a feasible successor, it sends a reply to n and sends updates to its other neighbor as needed.
- (b) else, if node i does not find a feasible successor then
 - (b.1) if $M_{jn} = 0$, then i behaves as in (a).
 - (b.2) If $M_{jn} = 1$, then
 - (b2.1) i sends a reply to n ;
 - (b2.2) if $IM_{ji} > 0$, then node i sends a query with $M_{ji} = IM_{ji}$ to its neighbor according to SAL in Section 4.4.
 - (b.3) If $M_{jn} > 1$, then i sends a query with $M_{ji} = M_{jn} - 1$ to its neighbor according to SAL in 4.4.

The larger the value of IM or the larger the value of FD or D in a query are, the smaller the chance to form a temporary or quasi-loop is. In the average case, the average converge time depends on the trade off between M and the distribution of the number of hops along the shortest path between statistically correlated hops. Correlated hops are those in which the link events happen during a given period in the order of converge time. The larger and more connected the network is, the larger IM must be to avoid quasi-loops. There is much research work to be done for choosing the optimized IM , but IM is a parameter that can be used to improve of performance .

4.6 Correctness and Loop Property of EDVA

For the case in which the value of all IM is set to infinity, proving that the revised feasibility condition RFC of EDVA is correct, i.e., that it enforces loop-free paths, can be easily done by simply substituting the value of the feasible distance reported by a node for the value of the true distance reported by the same node in the proof of correctness for DUAL presented in [4].

For the case in which M is smaller than infinity, EDVA still ensures that no counting to infinity exists, because of the predecessor information it uses. This is immediate from the correctness proof of PFA [12]. Temporary loops may exist with some probability, which can be reduced by increasing the values of IM , announced distance, and feasible distance in the query. A simple example is shown in Section 5.4.

5. Examples and Comparison with DUAL and LPA

In this section, we compare EDVA with DUAL and LPA and show the benefits of the new mechanisms introduced in EDVA in Section 4. Our comparison is done by giving a few simple specific examples focused on each mechanism introduced in EDVA.

5.1 Example 1: Benefit of RFC

Fig.1 shows an example illustrating the benefit gained with RFC introduced in Section 4.2. The arrows indicate the current routing choices for destination j . When the cost of link L_{ji} increases from 2 to 9, i 's RFC for destination j is satisfied, because $FD_{ji} = 2 > FD_{jn} = 1$ and $D_{jn} + L_{ni} = 5+1 < D_{ji} = 9$; thus, node i can immediately change its path as $i-n-j$. Accordingly, EDVA acts much like PFA and no query is necessary. In contrast, the

feasibility condition (FC) for node i and destination j in both DUAL [4] and LPA [5] is the following:

$$FD_{ji} > D_{jn} \quad (FC)$$

Accordingly, because $FD_{ji} = 2 < D_{jn} = 5$, FC is not satisfied, and node i must send a query before it can choose a new feasible successor. It is clear that RFC saves queries and consequently reduces the convergence time.

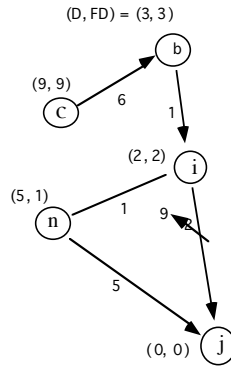


Fig. 1

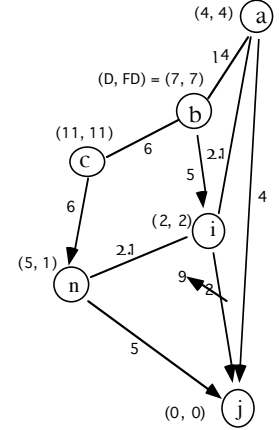


Fig. 2

Fig.1 and Fig.2 include typical routing examples

The larger the positive difference of $FD_{ji} - FD_{jn}$ is, the smaller the possibility of synchronization is. $FD_{jn} < FD_{ji}$ in RFC is a looser constrain than the corresponding $D_{jn} < FD_{ji}$ in FC used in DUAL and LPA. The improvement (i.e., savings) in total number of queries (which has to be sent when active) and replies as well as the converge time for synchronization are the fraction equal to the probability of $\{ D_{jn} \geq FD_{ji} > FD_{jn} \}$ from DUAL/LPA. Of course, the price that must be paid is the need to report and stores feasible distances to and from neighbors.

5.2 Example 2: Benefit of Active Changes in EDVA

One of the differences between DUAL/LPA and EDVA is that FD_{ji} (discussed in Section 4.3) does not have to be equal to D_{ji} (as in DUAL) or to infinity (as in LPA) during the active state, because the announced feasible distance is independent on the node's true distance for the purposes of blocking potential loops.

In the example shown in Figure 2, when node i becomes active after the link L_{ji} changes from 2 to 9, node i can choose n as its temporary successor since $FD_{ji} = 2 > FD_{jn} = 1$ (even if n may be querying i with $FD_{jn} = D_{jn} = 2+2.1$ and i does not reply back to n yet, according to the first part of AC) and $D_{jn} + L_{ni} = 5+2.1 = 7.1 < D_{ji} = 9$. Using RFC and AC instead of FC, i queries with $D_{ji} = D_{ja} + L_{ai} = 4 + 2.1 = 6.1$, instead of $D_{ji} = 7.1$ or even $D_{ji} = 9$. Therefore, while i chooses safely the available minimum cost of 7.1 and queries with the absolutely minimum $D_{ji} = 6.1$ (which is not available currently since the RFC is not satisfied yet), node b saves a query that would be made in

DUAL if i queried with $D_{ji} = 9$ or in LAP if i queried with $D_{ji} = \text{infinite}$.

5.3 Example 3: Benefit of SAL

In Fig. 2, assuming no node has $IM = 0$, then node i does not have to send any of its changed routing information to node n , because $FD_{ji} = 2 > FD_{jn} = 1$ (discussed in Section 4.4). For a similar reason, node b will not send any update or query to node a for node b 's changing ($D_{jb} = 11.1$ due to $D_{ji} = 6.1$ later on), since $FD_{jb} = 7 > FD_{ja} = 4$. These savings in the number of updates are not available in LPA or DUAL, who generally send queries and updates to all the neighbors.

In addition, applying SAL does not cause overhead, while saving update messages. This saving can also reduce the waiting time needed to search for a successor, and consequently reduce the average converge time for synchronization from the average case needed in DUAL/LPA.

5.4 Example 5: Benefit of ADS

Considering the non-choice path or the leaf of a network like the one in Fig. 1: $c - b - i$ for destination j , we can set one parameter of Adjustable Diffusing Steps (ADS) proposed in Section 4.5, $IM = 0$, so that node c and b never send queries, since any query from c or b is a waste. For node i to set $IM > 1$ is obviously a wasteful, since node c will be queried, though it is not always wasteful for the topology in Fig. 2.

Generally, without considering a dynamic topology, DUAL and EDVA with a large M can do more than LPA, PFA and EDVA with a smaller M in making sure that no temporary loop exist, which saves looping packets from consuming.

6. Conclusion

We have proposed EDVA, a new routing algorithm based on distance vectors. EDVA is based on enhancements to two classes of distance vector algorithms, DUAL, which is based on diffusing computations for inter-nodal synchronization, and path finding algorithms that report complete path information incrementally.

The benefits of EDVA's enhancements to DUAL and path finding algorithms have been discussed with a few simple average or specific cases. Briefly, the advantages of EDVA are reducing the routing converge time and the number of packets for routing information time. Our work continues to study and analyze the performance of EDVA with various values of its adjustable reporting parameters.

References:

[1]. R. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle, "EIGRP-A Fast Routing Protocol Based on Distance Vectors," Proceedings Networkworld/Interop 94, Las Vegas, Nevada, May 1994.

[2] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, Inc., Second Edition 1992.

[3] C. Cheng, R. Reley, S. P. R Kumar and J. J. Garcia-Luna-Aceves, "A Loop-Free Extended Bellman-Ford Routing Protocol without Bouncing Effect", ACM Computer Commun. Review, Vol.19, No.4, 1989, pp. 224-236.

[4] J. J. Garcia-Luna-Aceves, "Loop-free Routing using Diffusing Computations", IEEE/ACM Transactions on Networking, Vol. 1, No. 1, Feb, 1993, pp.130-141.

[5] J. J. Garcia-Luna-Aceves and S. Murthy, "A path finding algorithm for loop-free routing," IEEE/ACM Transactions on Networking, Feb. 1997, vol.5, (no.1):148-60.

[6] J. Hagouel, "Issues in Routing for Large and Dynamic Networks," IBM Research Report RC 9942 (No. 44055) Communications, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, April 1983.

[7]. C. Hedrick, "Routing information protocol," RFC 1058, Netw. Inform. Cent., SRI Int., Menlo Park, CA, June 1988.

[8]. G. Malkin, "RIP Version 2 - Carrying Additional Information," RFC-1388, Xylogics, Inc., January 1993.

[9] S. Murthy, "Design and Analysis of Distributed Routing Algorithms", Master's Thesis, University of California, Santa Cruz, 1994.

[10] S. Murthy and J. J. Garcia-Luna-Aceves, "A More Efficient Path-Finding Algorithm," Proc. 28th Asilomar Conference Pacific Grove, CA, October 31-November 2, 1994.

[11] S. Murthy and J. J. Garcia-Luna-Aceves, "Dynamics of a Loop-Free Path-Finding Algorithm," Proc. IEEE Globecom '95 Singapore, Nov. 13-17, 1995.

[12] S. Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," ACM NOMAD Journal, Special issue on Routing in Mobile Communication Networks, 1996.

[13] B. Rajagopalan and M. Faiman, "A Responsive Distributed Shortest-Path Routing Algorithm within Autonomous Systems," Internetworking: Research and Experience, Vol.2, No.1, March 1991, pp. 51-69.

[14] A. U., Shankar, C. Alaettinoglu, K. Dussa-Zieger and I. Mattas, "Transient and steady-state performance of routing protocols: distance-vector versus link-state," Internetworking: Research and Experience, June 1995, vol.6, (no.2):59-87.