International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

LOOP-FREE DISTANCE VECTOR ROUTING TO AVOID COUNT-TO-INFINITY

Santanu Kr. Sen, Professor; Debraj Roy, Assistant Professor; Shirsankar Basu, Assistant Professor; Poojarini Mitra, Assistant Professor

Abstract

The proposed Loop-free Distance Vector Routing (LFDVR) is a distributed dynamic routing, derived from the famous Bellman Ford's Distance Vector Routing Algorithm (BFDVRA) which is widely used in the Internet and private intranets. Several routing protocols based on distance-vector algorithms have been proposed like Routing Information Protocol (RIP), Gateway-to-Gateway Protocol (GGP) and Exterior Gateway Protocol (EGP). Though simple and conceptually elegant, the primary disadvantages of all these algorithms are routing loops, slow convergence and the well-known Count-To-Infinity problem (CTIP). The solutions already proposed to overcome these problems like Split Horizon or Split Horizon with Poisoned Reverse are ad-hoc in nature and often fail. We have, in this paper, proposed an Loop-free Distance Vector Routing (LFDVR) protocol that avoids routing loops, totally solves the count-to-infinity problem and may use any type of metric including link delay. Avoidance of the routing loops and count-to-infinity problem by the LFDVR has been investigated, simulated and illustrated with two examples.

Introduction

The major function of the network layer is routing packets from the source machine to the destination machine through a communication subnet using mostly multiple hops (the term hop, router or node is used interchangeably) choosing the best route either statically (pre-defined or pre-computed) or dynamically (based on current best decision!) where the routing cost is measured in terms of hop-count, distance, bandwidth, mean queuing delay or any other suitable metric [2]. Two fundamental and most common types of adaptive (dynamic) and distributed routing algorithms used are BFDVRA and Link State Routing algorithm (LSRA). Both are widely used in various forms in the Internet, intranets and isolated non-broadcast type networks. Where IGP uses RIP (RIP or RIP2) [5] [6], HELLO and OSPF for Intra-AS routing, EGP mostly uses BGP for Inter-AS routing. Again, RIP, HELLO and BGP are derived from BFDVRA whereas OSPF, IS-IS routing protocols are actually derived from LSRA [8]. Thus, BFDVRA plays a vital role both in IGP as well as in EGP - the two most wildly used routing protocols.

In BFDVRA, a router knows the distance of the shortest path (which may or may not be the best path! Section 2), from each of its neighbours to reach to every network destination and used this information to compute the shortest path and next-hop of the path to each destination. A router sends update messages to all its neighbours, who in turn, process the messages and sends messages of their own if needed. Each update message contains a vector of one or more entries, each of which specifies, as a minimum, the distance to a given destination [7] [8]

Both the BFDVRA and LSRA has the disadvantage of routing loops, may be temporarily, which is a detriment to the overall performance of an internet. The BFDVRA is conceptually elegant because of its simplicity of operation and probably low overhead in terms of memory and processing compared to LSRA. In case of LSRA, each participating router requires to have the complete network topology information to compute the shortest path to each network destination which may constitute excessive storage and communication overhead on account of flooding, in a large, dynamic network. LSRA supercedes BFDVRA in terms of quicker convergence over a link failure or a topological change However; the primary disadvantages of BFDVRA are routing-table loops, slow convergence and the well-known CTIP. In case of an increased link-cost or specifically, failure of a link, the BFDVRA converges slowly and sometimes may not even converge to a stable state at all. This second phenomenon of excessive slow convergence or no convergence is popularly known as CTIP [2] [3] [4] [9].

A number of ad hoc solutions have been made to solve the CTIP by increasing the amount of information exchanged among the nodes, or by making nodes to hold down the updating of their routing tables for some period of time after detecting distance increases or the failure of a link, but none really solves the problem satisfactorily [2] [3] [7] [9].

Keeping in mind, the popularity and elegancy of BFDVRA, we propose in this paper, a solution to overcome the two major problems of BFDVRA viz. routing-loops and CTIP. The proposed LFDVR totally avoids the CTIP and the routing-loops. The proposed protocol is also free to choose any kind of metric rather than very commonly used metric like distance or hop-count. The proposed model thus guarantees to converge within a finite time without creating any loop and thus totally avoiding the well-known CTIP.

II. Best Path vs. Shortest Path

In this new loop free routing protocol, we use the term "best path" in lieu of "shortest path" considering the importance of the term and its practical utility in choosing the best route from source to destination. From the practical point of view, a shortest path may not be the cheapest path and hence may not the best path. The term shortest path basically refers to the path which is geographically shortest in terms of distance, [2][9] whereas, if the cost metric of a link is considered as delay, bandwidth, or something else other that distance, then the term best path seems to be more appropriate and suitable than the term shortest path. It is to be noted that it may happen that the geographical distance between a source and a destination might be same through one route where the cost metric, say delay is more than the other.

In another perspective, if the cost metric of a link between two nodes is computed using more than one factors say queuing delay, distance, hop-count and/or bandwidth, in this case again best path is more fit than the term shortest path. The term "shortest path" used by Djikstra in his famous Shortest Path algorithm did use "distance" as the cost metric and hence the widespread of the term. But now-a-days where the whole world is flooded with networks, routers etc., use of other type of metric is equally important to compute best path and that is the reason of replacing the term shortest path by the term best path. However, we still depend on the technique of Djikstra shortest path algorithm to compute the best path.

III. Network Model and Notation

Throughout this paper, the following notations are used:
G: A connected network of arbitrary topology
E: The set of links in G
N: The set of nodes in G
i: The identifier of destination node i Є N
j: The identifier of current node j Є N
k: The identifier of neighbour node of j; k Є K
K: {set of all neighbours of j}; K Є N
K/: {set of neighbours of j except i} K/ Є N
Ki : {set of neighbours of j that reaches to i via j}
Kj : {set of neighbours of j that reaches to i not via j}
V: {set of all valid neighbours (VN)} = VNL
v: A VN ;v Є N ; v $\notin$ DQNL ;v ≠ jR
Cjk: The current cost from node j to neighbour node k
Cji/k: The cost from node j to node i via neighbour k
jQ: A Requester node of j
jR : A Replier node of j
ERL: {VNL1 U VNL2 U VNL3 U…….VNLn)
VNL: {set of all Valid Neighbours (VN)}
VN: A Valid Neighbour (VN) is a neighbour of j which is not included in the DQNL sent by jQ to j to find an alternative path to reach to a destination node i and through which j can propagate SeekHelp() message to find an alternative path to reach to node i.
RQT : Request Table
DQNL: {set of all disqualified nodes including jR}. A DQNL is a list of nodes none of which can be selected as the next-hop for the current source node j to reach to destination node i. The nodes enlisted in the DQNL are already visited nodes in the current path browsing tree
ERL: is a set union of all valid nodes of j for different requests arrived at j to reach to a particular destination i.
ER: is a neighbouring node of j to which j earlier sent SeekHelp() request message to find an alternative route to reach to i and presently expecting a reply from it.

IV. LFDVR Algorithm

Whenever there is a link failure or increase in link cost, irrespective of the type of metric the routing protocol do uses, takes place between two neighbouring nodes say j (F1) and F2 (i), then to converge the whole network to a stable state, both F1 and F2 runs the LFDVR algorithm as briefly described below. We assume only F1 here for the sake of simplicity.

Step 1: Cij is set to -9 so that if any other node tries to know the cost between i and j, a reply of -9 will make the requested node to assume that j and i are not in stable state and is under repair.

Step 2: Node j tries to get help from all its neighbours K/ (Ki+Kj) except i to reach to i by sending a SeekHelp() request message to all of them ignoring whether a neighbour k is currently reaching to i via j or not, in an intuition that although k currently reaches to i via j depending on the best path algorithm but it doesn't waive out the possibility of an alternative path from k to i not via j with same or higher cost.

Sending SeekHelp() message to all K/ neighbours, node j waits till it gets SeekHelpReply() from all of them.

Each SeekHelp() request message contains a DisQualified Node List (DQNL) and hence a neighbour node of j which is already visited and thereby included in the DQNL, is not required to be scanned again.

Step 3: When a node Receives the SeekHelp() message:

Getting the SeekHelp() message by j from a neighbouring node, the current node j performs the following steps in an objective to search for an alternative path for its requester node jR to reach to the destination node i.

First of all, j checks the entries in the ReQuest Table (RQT) to find if the same requester has already requested for the same destination. If found, j does not forward the request further to its VNs (Valid Neighbour) and just keep silence doing nothing. This "silence concept" is introduced to reduce the redundancy for the same type of requests and thereby saving the network bandwidth.

But if the request is a fresh one, i.e. no such already existing entry is found in the RQT table of j, node j first creates a Valid Neighbour List (VNL).

If length(VNL) = 0, which indicates that there is not a single valid neighbour do exist for node j through which j can take help and therefore, node j responds with a negative reply SeekHelpReply(-1) without keeping all its requesters more in wait.

But If length(VNL) > 0; which indicates that there is at least one VN of j, node j checks if v == i; ∀ v Є V i.e., if any of its VN is equal to destination node i, j immediately terminates its processing and further propagation of SeekHelp() request message through its other VNs and concludes that it has found and alternative route (may not be the best route for the requester but best for j) to reach to desired destination node i.

j computes the cost Cji which is directly obtained from the weight matrix of node j and updates its own Routing Table (RT) by updating both cost to reach to i and the next-hop information.

j now sends SeekHelpReply (Cji) message to all its Requesters giving j's least cost i.e. Cji to reach to i.

On the other hand, if none of the valid neighbours of j is i, j first includes itself in the DQNL and propagates SeekHelp() message to all its VNs seeking help to reach to destination node i and waits till it gets SeekHelpReply() from all of them.

Step 4: When a node receives the SeekHelpReply() message: When a node j receives the SeekHelpReply() message from any of its neighbours, node j then checks if the current Replier (jR) who is of course a neighbour of j and also a member of Expected Replier List (ERL). This check is included to avoid any malicious or unwanted reply from any neighbouring node.

However, if the replier is not found in the ERL, derived from the RQT of j, j does not take any action by simply ignoring the reply.

In case the replier is an ER (Expected Replier) which indicates that the replier is a valid replier, node j first checks the message value. If the message value of SeekHelpReply() message is a negative value i.e. -1, it indicates that there is no path at all through this replier node jR to reach to i. Node j then waits till it receives replies from all the other ERs. But if the message value of SeekHelpReply() message is a non-negative value, node j understands that there is a path to reach to i.

Node j however, comes to a decision only after it receives replies from all its valid repliers.

If all the replies arrived to j each with a negative feedback i.e. -1, it clearly indicates that there is no path at all to reach to destination node i through any neighbour of j and thereby j concludes that it can provide no help to its requesters to reach to i and hence j again replies all its requesters with a negative reply SeekHelpReply (-1).

When a node j receives SeekHelpReply() message from its Expected Repliers, and j has no RQT of its own, it indicates that j is the Original Requester i.e. F1 (in this case). Now, if all the replies received by j (F1) are -1, it indicates with 100% guarantee that there is no route at all to reach to destination node i (F2, in this case) and j concludes that the link between j (F1) and i (F2) is no more and totally unreachable since there is no alternative path to establish link between the two. Node j then updates its own RT and sends this updated message to all its neighbours using general Bellman Ford's BFDVRA to take the whole network to a stable state and totally avoiding CTIP.

After j receives replies from all its ERs with a mixture of message values i.e. -1 and/or positive cost values, j ignores all -1 values and finds out the cheapest one comparing only among the positive values. j then computes the required cost to reach to i by adding the link cost Cjk with the Cki and updates it own RT and then sends the SeekHelpReply(Cji) to all its neighbours telling its cost to reach to destination node i.

It is to be noted that the RQT of a node j is deleted either after all replies have arrived from all ERs or after a certain predefined time (TTL).

Thus, this algorithm not only finds out an alternative path to reach to destination (if any), but also the path found out is the cheapest one. On the other hand, since there is no chance of routing loops, thereby omitting the occurrence of CTIP.

V. Examples

As shown in Fig 1(a) and 1(b), we have considered only two most typical examples out of many. The link failures are shown by cross marks.

In example 1and 2, CTIP takes place using ordinary BFDVRA and hence do not converge at all. The more intelligent routing protocols Split Horizon and Poison Reverse do fail in case of network 2 [Fig. 1(b)] causing Count-To-Infinity. But the proposed LFDVR algorithm does not fail at all and all the networks do converge within a finite time after the shown link failures. The proposed LFDVR algorithm guarantees to converge within a finite time without creating any routing-loops and without creating any CTIP.

Example 1:
In case of network 1(a)], after the link failure between router C and D,

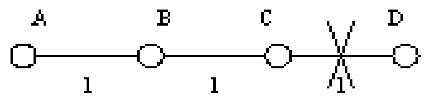

Fig 1(a)

- Router 'C' sends request message SeekHelp() to all itsVNs which is none but 'B' and waits till it gets replies from all its neighbours i.e. 'B' in this case. 'C' does not create any RQT since 'C' is the Original Requester, but creates the DQNL including itself as the first member of the DQNL list.
- Receiving the SeekHelp() message from 'C', node 'B' first of all creates a VNL and a RQT as shown in Table 1.1. 'B' finds that its only VN A is not the destination node 'D' and hence forward the SeekHelp() request message to 'A' including itself as a member in the DQNL.

TABLE 1.1: REQUEST TABLE (RQT) FOR NODE 'B':

Requester	Destination	VNL/ ERL
C	D	A

- Receiving the SeekHelp() message from 'B', node 'A' finds that it has no VN at all and therefore sends SeekHelpReply(-1) with a negative value as shown, to its requester i.e. to node 'B'.
- Getting negative reply from 'A' which is an ER of 'B', node 'B' simply forwards the SeekHelpReply(-1) with a negative feedback to its requester 'C' since 'B' is having one and only one ER and deletes the RQT.
- Similarly, while 'C' gets SeekHelpReply() from all its ERs which is only node 'B', 'C' arrives to a firm decision that there is no alternative path at all to reach to 'D' and accordingly updates its own RT and sends this updated information to all it neighbours using ordinary BFDVRA algorithm.
- Thus all nodes in the network update their corresponding RTs and the whole network ultimately come to a stable state within a finite period of time.

Example 2: Link Failure between node 'C and node 'D':

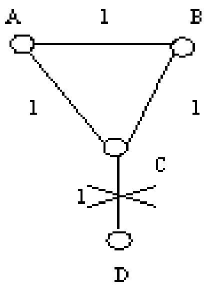

Fig 1(b)

- 'C' sends request message SeekHelp() to all its VNs (A,B) and waits till it gets replies from all of them. 'C' does not create any RQT since 'C' is the original requester, but creates the DQNL including itself as the first member of the DQNL list.
  DQNL={C}
- 'A' receives SeekHelp() from 'C': Node 'A' Receiving the SeekHelp() message from 'C', node 'A' first of all creates a VNL and a RQT as shown Table 2.1. 'A' finds that its only VN 'B' is not the destination node 'D' and hence forward the SeekHelp() request message to 'B' including itself as a member in the DQNL.
  DQNL={C, A}

TABLE2.1: REQUEST TABLE (RQT) FOR NODE 'A':

Entry no	Requester	Destination	VNL/ ERL
1	C	D	B
2	B	D	NULL

- 'B' receives SeekHelp() from 'C': Similarly, at the same instant node 'B' also receives the SeekHelp() message from 'C'. In the same fashion, node 'B' creates its own VNL and a RQT as shown in Table 2.2. 'B' finds that its only VN 'A' is not the destination node 'D' and hence forward the SeekHelp() request message to 'A' including itself as a member in the DQNL.
  DQNL={C, B}

TABLE2.2: REQUEST TABLE (RQT) FOR NODE 'B':

Entry no	Requester	Destination	VNL/ ERL
1	C	D	A
2	A	D	NULL

• 'B' receives SeekHelp() from 'A': Receiving the SeekHelp() message from 'A', node 'B' adds the entry no. 2 in its existing RQT [Table 2.2] and tries to find VNs which is NULL i.e length(VNL)=0. So, 'B' understands that there is no path at all to reach to 'D' through itself and therefore sends a negative SeekHelpReply(-1) to all the ER i.e 'C' and 'A' deleting the RQT.
• 'A' receives SeekHelp() from 'B': Similarly, while the SeekHelp() message arrives to 'A', 'A' adds a new entry entry no. 2 [Table 2.1] and sends back a negative reply SeekHelpReply(-1) to all its ERs 'B' and 'C' in this case.
• 'C' receives SeekHelpReply() from 'A' and 'B': The negative reply thus propagated to 'C' from 'A' and 'B' and thereby 'C' concludes that it has no path at all to reach to destination node 'D' by any means since all its ERs have sent negative replies. 'C' then updates its own RT and sends this updated information to all it neighbours using ordinary BFDVRA algorithm.

Thus all nodes in the network update their corresponding RTs and the whole network ultimately arrives to a stable state within a finite period of time.

VI. Conclusions

Functioning of the LFDVR algorithm has been simulated with randomly generated failures of links in a large number of randomly generated network graphs. Around one hundred cases have been found where link failure between two neighbouring nodes caused Count-To-Infinity using the algorithms like ordinary BFDVRA, Split Horizon and Poison Reverse. Use of RIP2 stopped after 16 iterations concluding CTIP but the proposed routing protocol LFDVR has failed nowhere.

The strength of metric-independency of LFDVR makes it versatile to be used in intra-AS or inter-AS routing and the trick of routing loops avoidance reduces the total number of message flow throughout the whole network to attain a stable state after a link failure. The algorithm not only finds out an alternative path to reach to destination (if any), but also the path found out is the best or cheapest one. On the other hand, since there is no chance of routing loops, thereby omitting the occurrence of CTIP. The justification of using "best path" in lieu of "shortest path" is also tried to be established citing some exciting real-life scenarios.

The somewhat increased processing and communication overhead in the LFDVR, is only a small price to be paid when viewed against the backdrop of the vast performance improvement.

Acknowledgments

The authors are thankful to IJACT Journal for the support to develop this document.

References

[1] D. Bertsekas and R. Gallager, Data Networks, 2nd Ed., PHI-EEE, 1992
[2] A. S. Tenenbaum, Computer Networks, 4th Ed., Pearson Education Asea, LPE, 2003.
[3] J. F. Kurose and K. W. Ross, Computer Networking : A Top-Down Approach Featuring the Internet, 2nd Ed., Pearson Education Asea, LPE, 2003.
[4] L. L. Peterson and B. S. Davie, Computer Networks : A systems Approach, 2nd Ed., Morgan Kaufman, 2000
[5] C. L. Hendrick, "Routing Information Protocol," RFC 1058, June 1988
[6] G. Malkin , "RIP Version 2- Carrying Additional Information," Xylogics, Inc., November 1994 , RFC 1723
[7] Albeto Leon-Garcia and Indra Widjaja, Communication Networks, Tata McGraw Hill, 2000
[8] Radia Perlman, Interconnections: Bridges and Routers, Addison Wesley, 1994.
[9] Tomomasa T, Morikawa A, Sandler RH, "Gastrointestinal Sounds and Migrating Motor Complex in Fasted Humans," *Am J Gastroenterol*, 1999, 94:374–381.
[10] S. K. Ray, S. K. Paira, S. K. Sen, "Modified Distance Vector Routing Avoids Count-To-Infinity Problem", Proc. International Conference CODIS 2004, held in Calcutta during Jan 8-10, 2004, pp. 31-34

Biographies

SANTANU KUMAR SEN, 25th Dec 1971, currently working as Professor and Head in the Department of Computer Science & Engineering in Gurunanak Institute of

Technology. He received his BE(CSE), M.Tech (CSE), MBA (IS) and PhD(Engg.) from REC Silchar and Jadavpur University respectively. He is a Fellow of IET(UK), IE(I), IETE(I) and Sr. Member of IEEE (USA), CSI(I) and life members of ISTE. His research interests mainly focused on Mobile Ad Hoc Networks, Computer Networks, Sensor Networks and Cloud Computing. Dr. Santanu Kumar Sen (Professor Author) may be reached at profsantanu.sen@gmail.com

DEBRAJ ROY, 24th November 1985, currently working as Assistant Professor in the Department of Computer Science & Engineering in Gurunanak Institute of Technology. He received his B.Tech(CSE) from Jalpaiguri Govt. Engg. College(affiliated to West Bengal University of Technology) in 2009 and M.E(CSE) from Jadavpur University in 2011. His research interests mainly focused on Computer Networks, Cloud Computing and Bioinformatics. Debraj Roy may be reached at debraj.roy85@gmail.com

SHIRSANKAR BASU, 25th November'1987, currently working as Assistant Professor in the department of Computer Science and Engineering in Gurunanak Institute of Technology. He received his B.Tech(C.S.E) in the year 2009 from Institute of Technology and Marine Engineering, Joka(affiliated to West Bengal University of Technology) and subsequently M.Tech(C.S.E) from Heritage Institute of Technology in the year 2011(affiliated to West Bengal University of Technology). He also bagged one year of research experience (2011 June – 2012 September) as a junior project assistant in IIT Kharagpur in the department of computer science. His primary research interests focus on Community Detection in Social Network Graphs and wireless sensor networks. Shirsankar Basu may be reached at shirsankarbasu@gmail.com

POOJARINI MITRA, 26th February 1987, currently working as an Assistant Professor in the Department of Computer Science and Engineering in Guru Nanak Institute of Technology. She received her B.Tech (CSE) from Calcutta Institute of Engineering and Management, WBUT in 2010 and M.Tech (CSE) from University College of Science and Technology, University of Calcutta in 2012. Her research interests are mainly focused on Wireless Sensor Network, Security on Computer Networks. Poojarini Mitra may be reached at poojarini.mitra@gmail.com