An Effective Solution to Count-to-Infinity Problem for both Complex and Linear Sub-Networks

Sabrina Hossain[1], Kazi Mushfiquer Rahman[2], Ahmed Omar[3], Anisur Rahman[4]
Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh

***Abstract*—Distance vector routing protocol determines the best route for forwarding information from one node to another node based on distance. For calculating the best route, Distance-vector routing protocols use the Bellman-ford algorithm and the Ford-Fulkerson algorithm. The Bellman-Ford distributed algorithm calculates the shortest path. On the other hand, Routing Information Protocol is commonly used for managing router information management protocol within a Local Area Network or an interconnected Local Area Network group. The main problem with Distance Vector Routing protocols is routing loops. Because the Bellman-Ford Algorithm cannot prevent loops. Moreover, the routing loop triggers a problem with Count to Infinity. This research paper gives an effective solution to the Count to Infinity problem for link down situation and also for the router down situation in both complex and linear sub-network. For the router down situation, when any router goes down, then other nodes will recalculate their routing table with the dependency column. Moreover, the costs are calculated by the shortest path algorithm. If any link is down and all routers are up, then all routers will recalculate their routing table using Dijkstra instead of the Bellman-Ford algorithm. To determine the loops and prevent the loops are the main objectives. This method is mainly based on a routing table algorithm where the Dijkstra algorithm will be used after each iteration and will modify the routing table for each node. Preventing the routing loops will not converge into Count to Infinity Problem.**

Keywords*—Distance vector routing; local area network; routing information protocol

I. Introduction

Nowadays, scientists are trying to reduce the packet loss problem. Everybody needs high-speed internet and, they want everything fast. However, because of packet loss, the service becomes slow, the network connection gets disrupted, and sometimes it loses the whole network connectivity. It creates significant problems in real-time data transfer programs. So, a better network means less packet loss. Distance Vector Routing is one of the dynamic algorithms [1]. Whenever a router goes down, routing loops usually occur in DVR. Then it creates a linear topology. Linear topology generates count to infinity problem. Because of that, a huge number of packets will be lost and, that is an immense issue. If all router gets to know earlier that any router already got down by observing an extra column of the routing table, then the packet loss problem can be minimized and which will improve the network connectivity. The modification of the routing table and alternation of the algorithm into the Dijkstra algorithm might cause less packet loss problem, and then the network connectivity will become well.

II. Routing Algorithm

Routing is a method of determining the routes to reach the destination that data packets will obey. A table of routing table is created in this process which contains information about the routes that data packets follow. Different routing algorithms are used to determine which route an incoming data packet needs to be transmitted efficiently to its destination.

A. Distance Vector Routing Algorithm

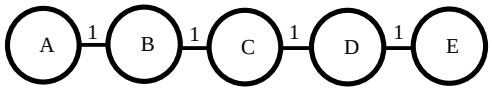
The Distance vector uses the Bellman-Ford algorithm for finding the shortest paths [2]. It can also be calculated by Dijkstra algorithm. Every node calculates the distance from other routers. The shortest path is created based on the metric. The metric is referred to as a count or a distance. In the Distance vector, the process of exchanging information is done iteratively [1]. There is no information exchange between the neighbourhoods until the information received from at least one neighbour directly and the algorithm does not require all the neighbours are asynchronous with each other. In distance vector, each node maintains the distance from it, to its possible destination and sends a periodic routing update. For periodic routing updates, the convergence time is slow. The slow convergence leads to count-to-infinity and routing loops problem [1].

B. Dijkstra Algorithm

Dijkstra algorithm solves the shortest path algorithm, and it is better than the Bellman-Ford algorithm. It works better when multiple paths present in the topology, and it helps to choose the shortest path [3][4]. In the following algorithm, the code u ← vertex in Q with min dist[u], searches for the vertex u in the vertex set Q that has the least dist[u] value. Length (u, v) returns the length of the edge joining (i.e. the distance between) the two neighbour-nodes u and v. The variable alt on line 18 is the length of the path from the root node to the neighbour node v if it were to go through u. If this path is shorter than the current shortest path recorded for v, that current path is replaced with this alt path. The previous array is populated with a pointer to the "next-hop" node on the source graph to get the shortest route to the source [5]. The Dijkstra algorithm is a Dynamic programming approach. A complex problem is divided into sub-problems in the Dynamic programming approach, then combine the solutions of these sub-problems to get an overall solution. So, Dijkstra's algorithm is a greedy approach. So, it does not create routing loops. In the Bellman-Ford algorithm, it cannot prevent loops, but through Dijkstra, it can be prevented. If any link or router down then, it will apply a greedy approach that will prevent the loops and solve the Count to infinity problem.

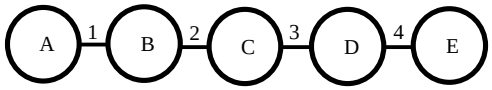
III. Count to Infinity Problem

Count-to-Infinity Problem is one of the most important issues in Distance Vector Routing (DVR) Algorithm. When an interface goes down, routing loops usually occur in DVR. Which actually creates linear subnet. When two routers send an update to each other at the same time, it can also occur [6][7]. Distance Vector Routing reacts rapidly to good news, but leisurely to bad news [8]. In distance vector routing, it uses the Bellman-Ford algorithm to propagate. To see how slow bad news propagates, consider the situation of 1(a) in which all the lines and router are initially up. Router B, C, D, and E have the distance to A of 1, 2, 3, and 4 respectively. Suddenly A goes down, or the line between A and B is cut, which is effectively the same thing from B's point of view in Fig. 1(b). At the first packet exchange, B does not hear anything from A. Fortunately, C says: do not worry; I have a path to A of length 2. As a result, B thinks it can reach A via C, with a path length of 3 and D & E do not update their entries for A on their first exchange.



∞	∞	∞	∞	Initially
1	∞	∞	∞	After 1st exchange
1	2	∞	∞	After 2nd exchanges
1	2	3	∞	After 3rd exchanges
1	2	3	4	After 4th exchanges

(a). Before Count to Infinity Problem.



1	2	3	4	Initially
3	2	3	4	After 1st exchange
3	4	3	4	After 2nd exchanges
5	4	5	4	After 3rd exchanges
5	6	5	6	After 4th exchanges
∞	∞	∞	∞	After infinity exchanges

Fig. 1. (b). After Count to Infinity Problem.

On the second exchange, C notices that each of its neighbour's claims to have a path to A of length 3. It picks one of them at random and makes its new distance to A of length 4, as shown in the third row. Subsequent exchanges produce the history shown in the rest of Fig. 1(a) and 1(b) [8].

Linear Subnet: Whenever a router goes down, routing loops usually occur in DVR. Then it creates a linear topology. Linear topology creates count to infinity problem.

In Fig. 2 a complex network is shown, but every time a router goes down, the routing loops will happen.

If router E from Fig. 2 goes down, then there will be a linear sub-network which is shown in Fig. 3.

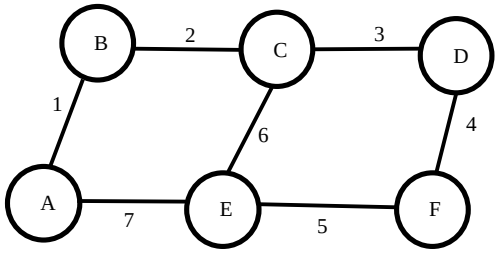


Fig. 2. A Complex Network.

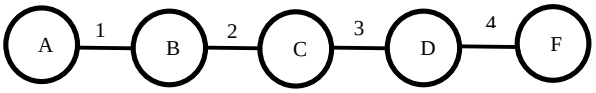


Fig. 3. A Linear Sub-Network.

IV. Related Work

The main difference between link sate and distance vector routing is, link-state uses an algorithm derived from Dijkstra's shortest path algorithm where distance-vector uses a distributed Bellman-Ford algorithm [8]. The distance vector routing algorithm suffers from the count-to-infinity problem. Count-to-infinity problems can be solved by preventing loops [9][8][1]. Researchers tried to prevent them by using various ways. Mr. D. Ganesh solved the count to infinity problem by using RSTP (Rapid Spanning Tree) protocol, Bridge protocol unit, building and maintaining SP (Spanning Tree) tree and changing the topology. They also follow some rules which are If a bridge can no longer reach the root bridge via its root port and does not have an alternate port, it declares itself to the root; A bridge sends out a BPDU immediately after the topology information it is announcing has changed, A designated port becomes the root port if it receives a better BPDO than what the bridge has received before. That is, this BPDU announces a better path to the root than via the current root port. When a bridge loses connectivity to the root bridge via its root port, and it has one or more alternative ports, it as its new root port. By using these rules, they portioned the network. Whenever a network is partitioned, if the partition does not contain the root bridge as a cycle, there exists a race condition that can result in the count-to-infinity behaviour. Count-to-infinity may even occur without a network partition. This new topology information will go around the loop until it reaches an alternate port caching stale, but better information. Again this stale information will choose the new information around the loop in a count to infinity. This will keep going

until the stale topology information reaches its massage [8]. Amit D. Kothari and Dharmendra T. Patel have also solved the count-to-infinity problem by using the test packet. They have some criteria for this test packet. First of all, they give source and destination addresses with the sequence number. They also declare the packet type like a query for any router and answer for the router. They also count the hop which initializes with 0 and increments by each intermediate router. The test packet will travel through source address to lastly. Then the address of the test router via the test packet is to be forwarded. Then it will give status where 1 is positive, and 0 is negative. The value will be a delay for the last neighbour. For error handling it will checksum. To solve the count to infinity, they design this type of test packet [1].

There is various way to solve the Count to Infinity.

1) Routing information protocol uses split horizon. Split horizon is a process where the actual distance to a destination is not reported to a node through which reaches the destination For example if node A has learned a route to node C through B, then A does not send the distance vector of C to node B during a routing update [9].

2) The count to infinity problem can be avoided by using hold-down timers. This is a clock that is set within the node to help ensure network stability.

When a node receives an update from a neighbour indicating that a previously accessible network is not working and is inaccessible, the hold-down timer will start. If a new update arrives from a neighbour with a better metric than the original network entry, the hold-down is removed, and data is passed. Nevertheless, an update is received from a neighbour node before the hold-down timer expires and it has a lower metric than the previous route. Therefore the update is ignored, and the hold-down timer keeps ticking. This allows more time for the network to converge [9].

V. Methodology

A. Procedure

Aiming at the difficulties in the count to infinity problem, this research proposed a method using Dijkstra Algorithm in each iteration and modifying the routing table with additional information. In the routing table, it added cost (shortest path), dependency, and status.

1) Dependency: It denotes the dependency of a router. For example, for calculating the routing table for C if it chooses a path from Router A to C via B and when the node A is down from Fig. 4 then, B is dependency router of C.

2) Status: It denotes the router is in hold situation or not. Which will prevent the routed loop in DVR.

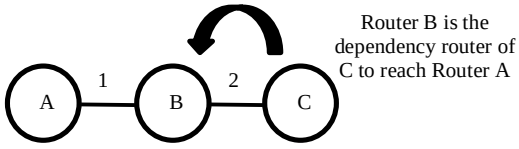


Fig. 4. A Linear Sub-Network of Three Routers.

This research has designed a method for both complex network and linear sub-network. At first, it will check if there is any link down or not. However, it has four different cases.

Case1: If all links are up, then it will go for checking the routers. If any routers are not down, then every node will calculate their routing table.

Case2: If all links are up, but any of the routers is down, then every router will recalculate their routing table. Every router will check the dependency column while recalculating its routing table. From the dependency column, a router can get to know about the present status of each router. Which means a router can easily track if there is any router down or not. If a router finds any router as down and which is its dependency router, then it will also change its status as down. Gradually all router's status will be down, and it will not create count to infinity problem.

Case3: If any link is down and all routers are up, then all routers will also recalculate their routing table using Dijkstra. While recalculating the routing table, a router will call itself as a holding router by updating the status column in the routing table. Therefore, other routers can do their work without any obstacles, and in the meantime, they can transmit their required packets to each other.

Case4: If any link is down and any router is also down then, at first it will follow case number 3 as mentioned in the above. And then it will follow the case of router down.

B. Algorithm of Proposed Method

In the two situations, one is for the topology links, and another is for routers. In four cases, two of the cases occur for the link down situation, and another two cases occur for router down situation.

```
1: Function main(){
Randomly pick up any cases
2: If (case 1){
Run generate routing table (source)
Print routing table
}
3: If (case 2){
 I. Scan which router is down
 II. Switch down the router
 III. Run generate routing table (source)
 IV. Print routing table
}
4: If (case 3){
 I. Scan which link is down
        II. Turn off the link
        III. Run generate routing table (any node of the link)
        IV. Print distance between the link's node
}
5: If (case 4){
 I. Scan the link which is down
        II. Turn off the link
        III. Calculate distance vector
        IV. Print distance vector
}
}
```

As the shortest path algorithm, it will use the Dijkstra Algorithm. Which is,

```
1: Function generate routing table(source){
2: create vertex set Q
3: for each vertex v in Graph:
4: dist [v] ← INFINITY
5: prev[v] ← UNDEFINED
6: add v to Q
7: dist[source] ← 0
8: while Q is not empty:
9: u ← vertex in Q with min dist[u]
10: remove u from Q
11: for each neighbor v of u: // only v that are still in Q
12: alt ← dist[u] + length(u, v)
13: if alt < dist[v]:
14: dist[v] ← alt
15: prev[v] ← u
16: return dist[], prev[]
}
```

For preventing the routing loops if all routers state their dependency with the additional details in the routing table, then all nodes will be slowly informed if there is any router in the down condition which causes routing loops. Here, the routing loops problem will be prevented. Moreover, for link down situation packet loss problem may be arrived. So, if any link is down, then all routers will recalculate their routing table using Dijkstra. While recalculating the routing table, a router will call itself as a holding router by updating the status column in the routing table.

VI. Simulation Result

A. Simulation Result for Complex Network

The proposed method has been simulated using C++ to validate the proposed model. A group of four routers are placed at different costs, see Table I. Fig. 5 of the graph is given below.

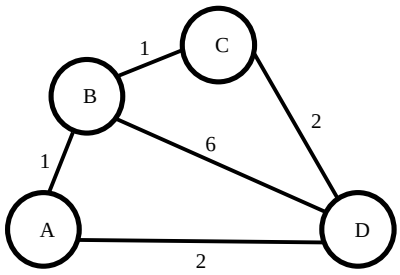


Fig. 5. A Graph with Various Costs.

TABLE I. The Costs of the Paths

Routers	A	B	C	D
A		1		2
B	1		1	6
C		1		2
D	2	6	2	

At first, it will check if there is any link down or not. Then it will choose a case randomly from four cases. If it chooses Case 1 when the source router is A, then the routing Table II is going to appear as mentioned.

TABLE II. Case 1 with Source Router A

Node	Cost	Dependency	Status
A	0	NULL	UP
B	1	A	Up
C	2	B	Up
D	4	C	Up

If it chooses Case 2 when the source router is A, and it goes down, then the routing Table III will be like this

TABLE III. Case 2 with Source Router A

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Up
C	2	B	Up
D	4	C	Up

Then the router B's status will be down as its dependency router A's status is also down in Table IV.

TABLE IV. Router B's Status is Down when its Dependency Router's Status A is also Down

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	2	B	Up
D	4	C	Up

Then the router C's status will be down as its dependency router B's status is also down in Table V.

TABLE V. Router C's Status is Down when its Dependency Router B's Status is also Down

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	2	B	Down
D	4	C	Up

At last the router D's status will be down as its dependency router C's status is also down in Table VI.

TABLE VI. Router D's Status is Down when its Dependency Router C's Status is also Down

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	2	B	Down
D	4	C	Down

So it will not cause the count to infinity problem. This is how this method handles Count to Infinity problem for a complex network.

If it chooses Case 3 when the source is A, and the link between A and B is down, then the graph will be appear as Fig. 6.

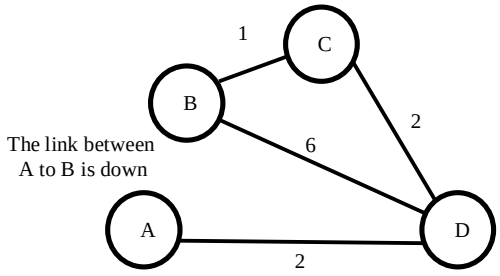


Fig. 6. The Link between A and B is Down.

Then it will recalculate the shortest path with the Dijkstra Algorithm. And then the new path from router A to router B will be **A->D->C->B** which is shown in the Fig. 7.

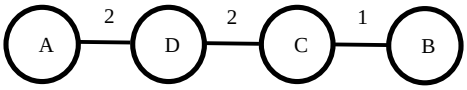


Fig. 7. New Path from Router A to Router B.

If it chooses case 4, then at first, it will follow case number 3 as mentioned in the above. And then it will follow the case of router down.

B. Simulation Result for Linear Sub-Network

Count to Infinity problem occurs due to linear sub-network. In Count to Infinity Problem, the updating of the routing table continues infinite time.

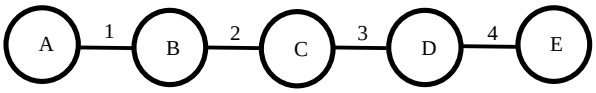


Fig. 8. A Linear Sub-Network of Five Routers.

In the linear sub-network of Fig. 8, when router A goes down, then the first five exchange of information is going to appear as mentioned in Table VII.

After 101 number of exchange of information, the routing Table VIII will be like this.

TABLE VII. FIRST FIVE EXCHANGE OF INFORMATION AFTER ROUTER A IS DOWN

Router	A	B	C	D	E
1st Exchange	NULL	5	3	6	10
2nd Exchange	NULL	5	7	6	10
3rd Exchange	NULL	9	7	10	10
4th Exchange	NULL	9	11	10	14
5th Exchange	NULL	13	11	14	14

TABLE VIII. AFTER 101ST NUMBER OF EXCHANGE OF INFORMATION

Router	A	B	C	D	E
102nd Exchange	NULL	205	207	206	210
103rd Exchange	NULL	209	207	210	210
104th Exchange	NULL	209	211	210	214
105th Exchange	NULL	213	211	214	214

Moreover, the routing Table IX will be updated for an infinite time. So, this is how the count to infinity problem occurs in linear sub-network. Nevertheless, in this proposed method, when router A goes down, then it will gradually let other routers know about its (Router A) condition. So, Count to infinity will not occur. If this proposed method applied for Fig. 8 when router A goes down, then this method will give a solution like this.

TABLE IX. AFTER ROUTER A GOES DOWN

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Up
C	3	B	Up
D	6	C	Up
E	10	D	Up

Then the router B's status will be down as its dependency router A's status is also down in Table X.

TABLE X. ROUTER B'S STATUS IS DOWN AND ALSO IT'S DEPENDENCY ROUTER A'S STATUS IS DOWN

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	3	B	Up
D	6	C	Up
E	10	D	Up

Then the router C's status will be down as its dependency router B's status is also down in Table XI.

Then the router D's status will be down as its dependency router C's status is also down in Table XII.

TABLE XI. ROUTER C'S STATUS IS DOWN AND ALSO IT'S DEPENDENCY ROUTER B'S STATUS IS DOWN

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	3	B	Down
D	6	C	Up
E	10	D	Up

TABLE XII. ROUTER D'S STATUS IS DOWN AND ALSO IT'S DEPENDENCY ROUTER C'S STATUS IS DOWN

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	3	B	Down
D	6	C	Down
E	10	D	Up

At last the router E's status will be down as its dependency router D's status is also down in Table XIII.

TABLE XIII. ROUTER E'S STATUS IS DOWN AND ALSO IT'S DEPENDENCY ROUTER D'S STATUS IS DOWN

Node	Cost	Dependency	Status
A	0	NULL	Down
B	1	A	Down
C	3	B	Down
D	6	C	Down
E	10	D	Down

So, in this method, there will be no routing loops. So, the Count to Infinity Problem will not occur.

C. Simulation of Graphs

Generally, Count to Infinity occurs in linear sub-network. All routers of the topology gradually increase their routing table for an infinite time. A graph of Count to Infinity is given.

From Fig. 9 there is a combined graph of 4 routers of Fig. 8. In Fig. 9, where the blue curve is for Router B. Before router A goes down the cost of B was 1. However, after router A got down, then router B updated its cost with the help of router C. Nevertheless, C's cost was updated with the help of router B before. So, after the 1st exchange of information, the cost of router B was updated from 1 to 5. Moreover, it will gradually update its cost for an infinite time. Furthermore, all routers followed the way of updating the cost as router B followed. So, all curves of the routers converge to infinity. In Fig. 9, the 1st – 5th exchanges, 102nd – 105th exchanges, and 1002nd – 1005th exchanges have shown.

If each Router's costs plot respect to the number of exchanges in this method, then a graph as Fig. 10 will appear.

According to Fig. 8, the cost of router B was 1 before router A goes down. However, in the 1st exchange of information, the cost of the router will not change as there was direct dependency with router A, but now it is down. After 2nd exchange of information, router B's status will also be down as its dependency router A's status is already down. In 3rd exchange of information when the router C wants to update its cost its status will be down as its dependency router B's status is already down. Gradually router D and E will change their status as down while checking the dependency. After 4th exchange of information, all router's status will be down. So the curves of all routers will not converge to infinity.

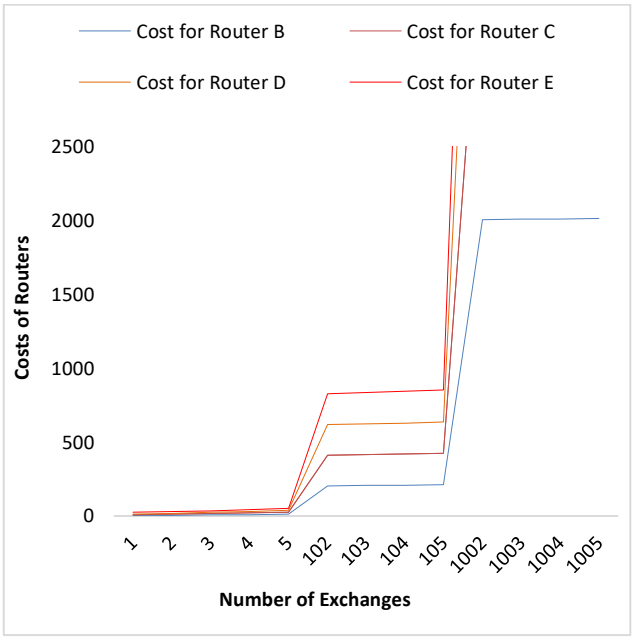


Fig. 9. Number of Exchanges vs each Router's Costs for Count to Infinity.

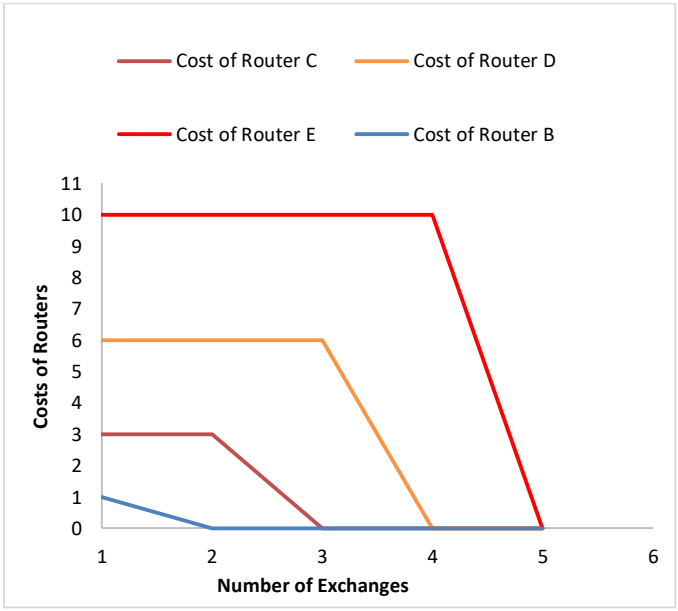


Fig. 10. Number of Exchanges vs each Router's Costs for Proposed Method.

D. Comparison of Complexity

Thus the count to infinity problem uses the Bellman-Ford algorithm, so the time complexity of Count to infinity is $O(V^2)$. However, this method implemented Dijkstra instead of the Bellman-Ford algorithm. Though, Bellman-Ford is simpler than Dijkstra and suites well for distributed systems. Nevertheless, the time complexity of Bellman-Ford is more than Dijkstra. In this method, the Dijkstra Algorithm with minimum priority queue can be reduced the complexity to O(V + E log V).

VII. Conclusion

For avoiding the Count-To-Infinity problem and reducing the packet loss, this method is applying the Dijkstra algorithm instead of the Bellman-Ford algorithm and that solve the Count-To-Infinity problem and reduce the packet loss. Now it does not face any problem with real-time data transfer, and the network connection will be undisrupted. This problem is solved for two types of situations one is router down, and the other one is linked down. For router down, the method handles the situation by giving additional information about the router, which is dependency. When any router goes down, then other nodes will recalculate their routing table with the dependency column. The shortest path algorithm calculates the costs. For link down situation, if any link is down and all routers are up, then all routers will recalculate their routing table using the Dijkstra algorithm instead of the Bellman-Ford algorithm. These are an effective way to solve the Count-To-infinity problem. There are some mechanisms known, such as defining the maximum count, split horizon, poison reverse, triggered update, and hold down timer. However, using this proposed way, an effective result will come out, and the packet loss will become less.

References

[1] D. Kothari and D. T. Patel, "Methodology to Solve the Count-To-Infinity Problem by Accepting and Forwarding Correct and Updated Information Only Using Test Packet," 2009 IEEE Int. Adv. Comput. Conf. IACC 2009, no. April, pp. 26–31, 2009, doi: 10.1109/IADCC.2009.4808974.

[2] Alberto Leon-Garcia and India Widjaja. Communication Networks, Fundamental Concepts and Key Architectures. McGraw´-]Hill Higher Education, Singapore, International Editions 2000. ISBN 0-07-022839-6.

[3] B. F. Zhan, "Three fastest shortest path algorithms on real road networks: Data structures and procedures," J. Geogr. Inf. Decis. Anal., vol. 1, no. 1, pp. 70–82, 1997.

[4] A. Goldberg and R. E. Tarjan, "Expected Performance of Dijkstra's Shortest Path Algorithm," Networks, no. 2 43, pp. 4–10, 1996.

[5] M. J. Bannister and D. Eppstein, "Randomized speedup of the Bellman-Ford algorithm," 9th Meet. Anal. Algorithmics Comb. 2012, ANALCO 2012, pp. 41–47, 2012, doi: 10.1137/1.9781611973020.6.

[6] K. Elmeleegy, A. L. Cox, and T. S. E. Ng, "Understanding and mitigating the effects of count to infinity in ethernet networks," IEEE/ACM Trans. Netw., vol. 17, no. 1, pp. 186–199, 2009, doi: 10.1109/TNET.2008.920874.

[7] K. Elmeleegy, A. L. Cox, and T. S. E. Ng, "On count-to-infinity induced forwarding loops in ethernet networks," Proc. - IEEE INFOCOM, 2006, doi: 10.1109/INFOCOM.2006.229.

[8] V. Rama and P. Vaddella, "An Effective Solution to Reduce Count-to-Infinity Problem in Ethernet," Int. J. Comput. Sci. Issues, vol. 7, no. 4, pp. 44–49, 2010.

[9] R. K. MCA and R. U. MCA, "an Exploration of Count-To-Infinity Problem in Networks," Ijest.Info, vol. 2, no. 12, pp. 7155–7159, 2010, [Online]. Available: http://www.ijest.info/docs/IJEST10-02-12-065.pdf.