Routing Problem Link State Algorithm versus Distance Vector Algorithm Two Approaches to the Complex Problem of Routing

Abdalhamed Alkawash
College of Technical Sciences – Derna
abdalhamed.alkawash@ctsd.edu.ly

Abdelmhsan Elbandac
Higher Institute of Agricultural Techniques – Tarhunah
Elbandaca@gmail.com

Aml M Adris Hashem
University of Derna – Derna
nan_amal@yahoo.com

الملخص: –

التوجيه هو عملية نقل الحزمة من المصدر الى الوجهة حيث يتم تنفيذ هذه العملية بواسطة اجهزة خاصة تسمى أجهزة التوجيه (Router) هذه الاجهزة تحتوي على ذاكرة لحفظ الخوارزميات التي تتحكم في اجهزة التوجيه لنقل الحزم.
لذلك الغرض من أجهزة التوجيه ايجاد مسارات جيدة من المصدر الى الوجهة. في علم الحاسوب توجيه حزم الشبكة من المصدر الي الهدف مشكلة تقليدية حيث تكون هنالك طريقتان مهمتان لحل مشكلة التوجيه وهما Link-State and Distance-Vector. الورقة استخدمت تقنية المحاكاة باستخدام لغة C++ لتوضيح الطرق التي يعمل بها هذين البرتوكولين وايضاً تعقيدات التشغيل المختلفة من حيث اختلاف الوقت وكميات المعلومات التي تمت معالجتها.

الكلمات الدليلية: التوجيه ، الذاكرة ، لغة C ++ ، تعمل البروتوكولات ، معالجة المعلومات

Abstract

Routing is the process of moving a packet from source to destination. This process is implemented by special devices called routers. These routers have memory to save the algorithms that control the routers. Therefore, routing aims to find good paths from source to destination. Network packet routing, a classic computer science problem, has had two significant approaches: link-state and distance-vector. By using

the simulation technique and by using C++language this paper illustrates the ways in which these two protocols operate and demonstrate their different complexities of operation in terms of time variation and quantities of information processed.

Keywords: Routing, memory, C++language, protocols operate, information processed.

1. Introduction.

Like many other problems in the real world, packet routing in networks is one with consequences. The primary nature of the problem is one of fixed (or perhaps dynamic) “distances” between routers, usually expressed as delays based on link speeds, being used as deciding factors for how to send a packet from a source to a destination. The routing problem is not so much one of finding the shortest paths, but how to communicate information amongst nodes so that they can determine
where packets should be sent according to their determinations of shortest
paths. The complexity involved in solving this information-sharing task varies
between the two historical approaches: link-state and distance-vector. The methodology that the paper followed is a simulation technique using the C++ language.

2. Type of Routing Algorithms.

Therefore, the routers work under algorithms that are designed for them, and these algorithms are called routing algorithms. The aim of a routing algorithm finds the optimal path from source to destination. The best path or good path means is one path that has the least cost than other paths that lead to the same destination [1].There are three ways to classify the routing algorithms which are
- First way is according to whether they are global or decentralized
- Second way is according to whether they are static or dynamic
- Third way is according to whether they are load sensitive or load-insensitive [1].

In the first way which is global or decentralized, we have two kinds of information that the router must know them.

In global routing algorithms or known link-state (LS) algorithms must have global knowledge about the network which means the routing algorithm has complete information about connectivity and link costs between nodes. After collecting all information, a global routing algorithm will compute the least-cost path between a source and a destination.

On the contrary, a decentralized routing algorithm, or as known is called a distance-vector (DV) algorithm. In this approach, every node has information or knowledge about its neighbors only. "Node and its neighbors mean there is a link between them." Therefore, a node does not have complete knowledge about all nodes and the costs in the network.

In the second way which is static or dynamic, and from its name, the difference between them is as follows.

“In static routing algorithms, routes change very slowly over time, often as a result of human intervention." On the other hand, dynamic routing algorithms will change the paths due to network traffic loads or topology changes. But a dynamic algorithm is more sensitive to problems such as routing loops and oscillation in routes due to it has a direct response to any cost changes or in the topology of the network.

The third way is load sensitive or load-insensitive. In the load-sensitive algorithm, “link costs vary dynamically to reflect the current level of congestion in the underlying link." On the other hand, routing algorithms are load-insensitive these days as a link’s cost does not explicitly reflect its current level of congestion [1].

3. Complexity Measures

As we know the most common complexity measures are time and space, in addition, to the number of operations. Furthermore, there is some less common measure, such as power consumption which have two types. The first one is direct power consumption which means the power used to run the machine and indirect power which is used to increase the brightness of the screen, for example [2].

In routing algorithms, there are 4 four complexity measures. These complexity measures are the computational rounds, space, the

number of messages, and the amount of time, required by algorithms [3].
1- Computational Rounds: count the number of rounds until the algorithm termination led to measure time in synchronous algorithms. Therefore, the time complexity is the maximum number of rounds until the algorithm has finished. However, in an asynchronous system, the measure of time is less straightforward. We need to propagate waves of events to count the number of rounds.[8]
2- Space: here we must know whether the router algorithm is global or local. If it is global, the whole space will equal the total space used by all computers. If it is local, the space will equal the space needed for each computer.
3- Local Running Time: In a global routing algorithm is very hard to analyze the global running time because computations are carried out across many computers over the network. Therefore, we can analyze the amount of local computing time required for a particular computer.
4- Message Complexity: The message complexity of an algorithm is the maximum of the total number of messages sent between all pairs of nodes during the computation [3].

4. Routing Problem.

As we know computer networks used routers to send packets toward remote destinations, therefore the router must know which neighbor it should forward a packet to so that it reaches its destination. Therefore, the routing problem is actually about communicating and using information from other routers to build a routing table. There are some challenges in routing problems such as sharing necessary connected neighbor information so that optimal next hops can be determined. In some router algorithms, the node is blind or has no knowledge of the whole network except for its immediate neighbors. The question here is, how can you reach a destination when you have no idea where it is?
To solve these problems, this paper will discuss two router algorithms: Link State Routing Algorithm and Distance Vector Routing Algorithm.

5. The solution.

This paper used some graphs to represent the routers and links. Furthermore, a graph is used to formulate routing problems. A graph G = (N, E) where N will represent nodes "routers,”, and E will represent links or edges "physical links" between each node [4].

5.1- Link State Routing Algorithm.

It is known that thelink State represents a global routing algorithm. Therefore, nodes have complete information about connectivity and link costs.

This approach involves gathering link-state packets from all other nodes. Therefore, links state routing algorithm, one assumption is always there that the network is fixed, and a cost is attached to each edge of the network. After that, all the information about connectivity and the cost of an edge is passed as input so that the least-cost route can be selected.

To achieve this link-state broadcast algorithm is used. At the beginning of this process, each router broadcasts its status to all other present routers using a flooding routing algorithm. Each node floods the entire graph with the information that is linked and cost information about all its adjacent edges. After that, each node in the network has identical as well as complete knowledge of the graph or network, which results in the same set of least-cost paths [1]. Then, each node applies Dijkstra's algorithm to its whole graph of the network to identify optimal next hops.

Therefore, the Link State algorithm is also known as the Dijkstra algorithm which is used to compute the least-cost path from one node to all other nodes in the graph. For example, figure 1, the paper used this graph that has 6 nodes and 10 edges to explain the Dijkstra algorithm, and how to compute the shortest path from K to all other nodes.

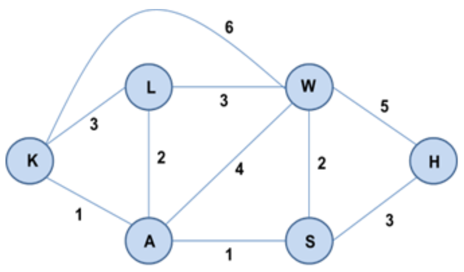


Figure 1: Compute the shortest path from K to all other nodes.

At starting, Dijkstra's Algorithm starts from the source node which is K and assigns the value infinite as the cost between k and other nodes figure 2. Then the K node will compute the paths to all neighbors who are A, L, and W, and the new cost are 1,3,6 respectively figure 3.

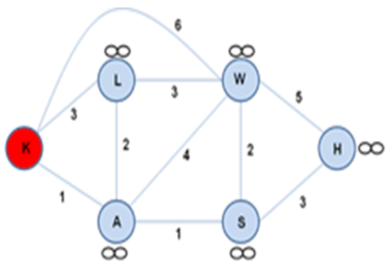


Figure 2: Assigns the value infinite

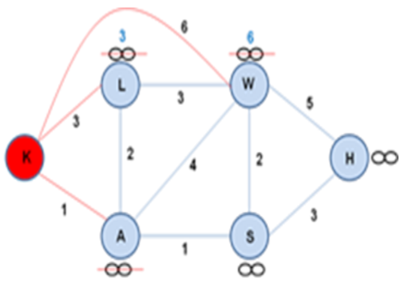


Figure 3: K node Compute the paths to all neighbors

After that, the algorithm will choose the path that has the least cost between A, L, and K to go to the next node. here we will move to A figure 4.

Node A will compute the paths to all neighbors which are L, W, and S, and the
new cost to S and W are 2 and 5 respectively. Because the cost is the same as L there is no change, and because it visited node K the algorithm will not compute the path from A to K although its neighbor. The algorithm will go to the next least cost path which is 2 so we will go to node S figure 5.

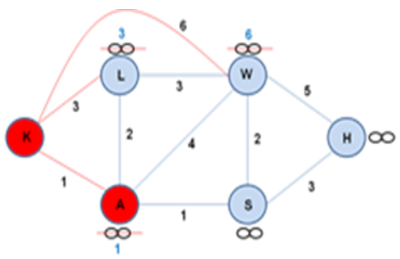


Figure 4: the algorithm goes to node A, due to it has the least cost

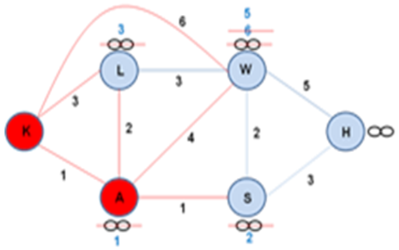


Figure 5: Node A will compute the paths to all neighbors

As well node S will compute the paths to all neighbors which are W, H and the new cost to W and H are 4 and 5 respectively. The last shortest path to W was 5. So, when node S finds another shortest path to W the algorithm will update the value to 4 and will go to the next least cost path which is 3, or to node L figure 6.
Likewise, node L will compute the paths to all neighbors, which is W. Because the path from L to W is 6 which is greater than another shortest path to W. There is no change in the value, and will go to the next least cost path, which is 4 so it will go to node W figure 7.

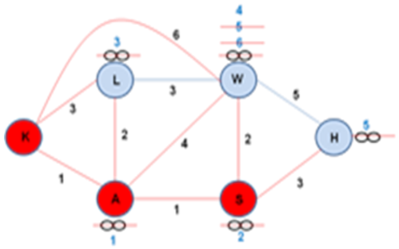


Figure 6: the following least cost path which is node L

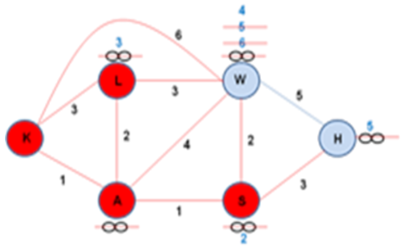


Figure 7: Node L will compute the paths to all neighbors

Repeatedly, Node W will compute the paths to all neighbors, which is H. Because the path from W to H is 9 which is greater than another shortest path to W. There is no change in the value. The algorithm will go to the next least cost path, which is 4 so it will go to node H figure 8, and node H, will not compute the paths to all neighbors because all neighbors already have visited, and there are no other nodes not visited figure 9.

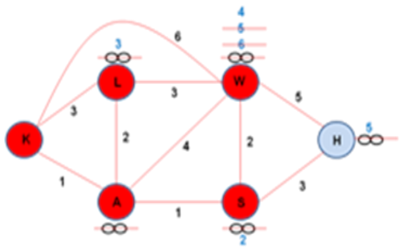


Figure 8: the algorithm goes to node H due to it has the least cost.

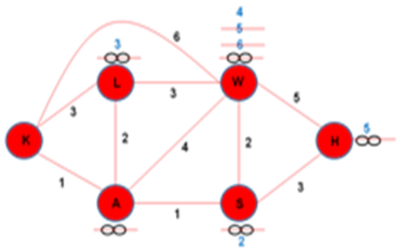


Figure 9: All nodes have visited and saved the shortest path possible to reach each node

Note: all nodes will compute the shortest path to all their neighbors that have not visited. For example, the shortest path from K to W will be from K to A to S to W, which equals 4.

5.1.1 The computational complexity of Link State Algorithm.

How much computation must be done in the worst case to find the least-cost paths from the source to all destinations?
Overall, the total number of nodes we need to search through over all the iterations is n(n + 1)/2, and thus we say that the implementation of the LS algorithm has worst-case complexity of order $O(n^2)$, and each router has O(n) space, and the overall message complexity is O(Ne) where O refer to Big O notation as a mathematical notation used to classify algorithms according to how their run time or space requirements grow as the input size grows.
The Link-State Routing algorithm involves two stages of processing, which are flood and gather and Dijkstra Algorithm. Therefore, the complexity of the first function is the number of nodes and edges, and the complexity of the second is a function of the number of nodes only. Because the algorithm will use merely the shortest path to any node that means will not use all paths in the graph. In addition, the delay in detecting failed nodes may cause a problem known as detection delay. This problem leads to lost packets due to forwarding data packets into a black hole Figure 10 which can be solved by using the Time to live.

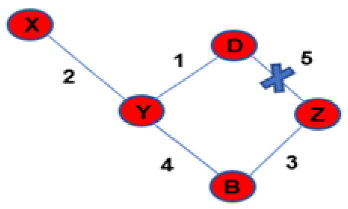


Figure 10: failed link between node D & node Z.

Another issue is Out-of-order packets reaching the destination. These issues lead to an increase in the complexity time [1,3].
Dijkstra algorithm does not accept any negative edge; therefore, the algorithm will return null, or error message. Our idea, if the Dijkstra algorithm accepted a negative edge, thus could lead to fails to accurately compute distances.

5.2 Distance Vector Algorithm.

In decentralized routing algorithms, the calculation is done in a distributive and iterative manner. As the packet arrives each node begins to collect information about its directly connected links. The exchange of information is processed with neighboring nodes and hence least cost route is found. These algorithms are referred to as Distance Vector Algorithm.
Since the Distance Vector Algorithm is the distributive extension of Bellman-Ford algorithm, it is distributed, asynchronous, and iterative. It is distributed because each node in the graph will receive some knowledge from all its neighbors only then compute the path and send it back to its neighbors, and “It is iterative in that this process continues on until no more information is exchanged between neighbors.", and this process does not require all the nodes to work together, hence it is asynchronous.[9]
The word Distance Vector stands for a bucket array as a matrix to store the length of the best-known path from one router to its neighbor router. Now neighbor router behaves as the information center and it passes the information to the next router hence all the routers are tracked with the required information [1].
Distance Vector algorithm involves gathering update packets from neighboring nodes. Basically

1. Each node initially only knows direct delays to all neighboring nodes
2. Each node floods its neighbors with updated packets indicating its own connection information (delays to neighbors). These packets are not forwarded.
3. Based on the updated information, if any, from each neighbor, each node updates its distance to other nodes based on the sum of the distance to its neighbor and the distance from its neighbor to the destination node.[7]
4. Steps 2 and 3 are repeated so long as updates produce changes in tables.

For example, we have the following Figure 11 that has 3 nodes which are A, L, and K, and 3 edges to explain the Distance Vector algorithm and how works.

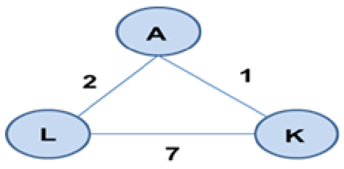
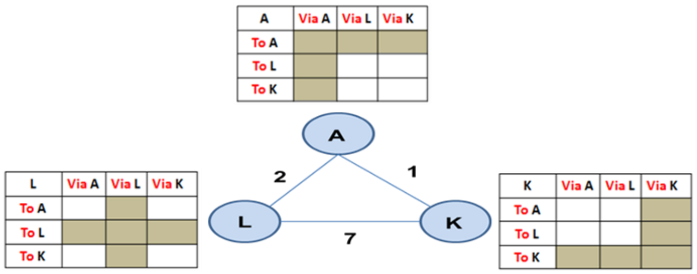


Figure 11: Graph with 3 nodes and 3 edges.

In Distance Vector algorithm every node has a table that will have all information about its neighbors as following Figure 12.
Note that:

🟫	It is not used
⬜	There is a path between these two nodes but not a short path
🟥	Short path.
🟦	Became not the short path.



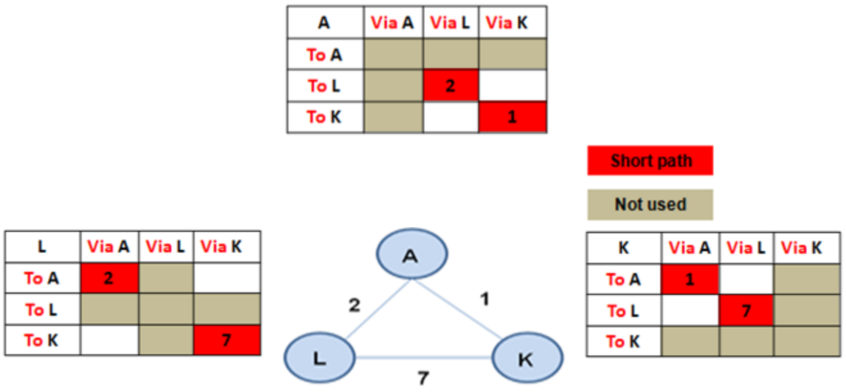
A	Via A	Via L	Via K
To A	▨	▨	▨
To L	▨		
To K	▨		

L	Via A	Via L	Via K
To A		▨	
To L	▨	▨	▨
To K		▨	

K	Via A	Via L	Via K
To A			▨
To L			▨
To K	▨	▨	▨

Figure 12: Each node has a table containing neighbors' information

After that, every node collects data for the cost of the path to its neighbors only. Take notice, if there is any node that is linked to L, node A and K will not have any information about this node at this time figure 13.



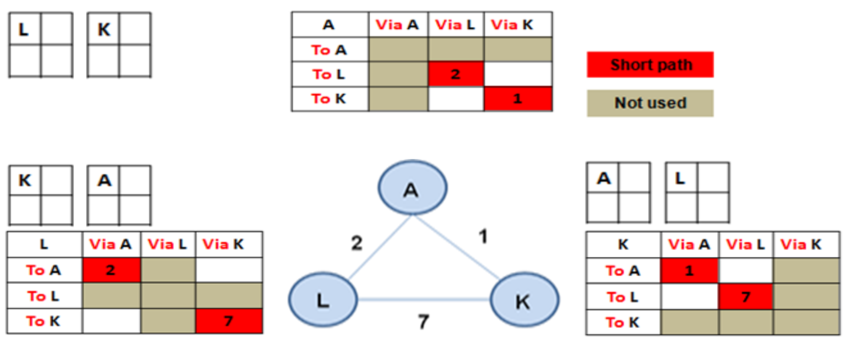
A	Via A	Via L	Via K
To A	▨	▨	▨
To L	▨	2	
To K	▨		1

L	Via A	Via L	Via K
To A	2	▨	
To L	▨	▨	▨
To K		▨	7

K	Via A	Via L	Via K
To A	1		▨
To L		7	▨
To K	▨	▨	▨

Figure 13: Nodes A, L, and K know the cost of the path between them

Next step, every node will receive data from its neighbors. For example, Node A will receive two tables or data from node L and node K figure 14.



A	Via A	Via L	Via K
To A			
To L		2	
To K			1

L	Via A	Via L	Via K
To A	2		
To L			
To K			7

K	Via A	Via L	Via K
To A	1		
To L		7	
To K			

Figure 14: Nodes A, L, and K receive complete data about each other.

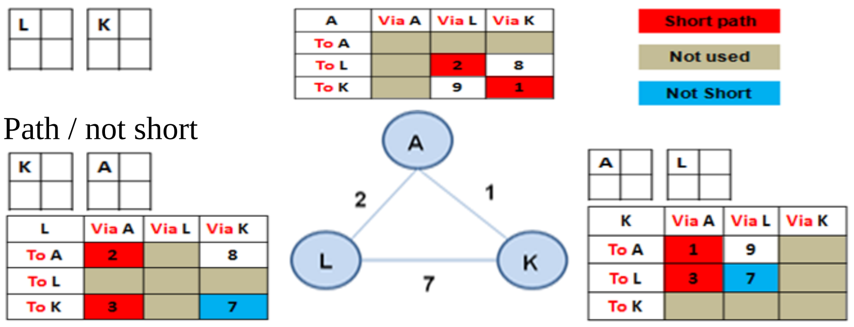
Figure 14: Nodes A, L, and K receive complete data about each other.

After nodes receive these data, they will update their data, and depending on this information all nodes will find a new path to the same destinations, and these paths could be the shortest path or not figure 15.

Finally, the Distance Vector algorithm has the shortest path between all nodes. Note that, if the cost has changed between any two nodes, all nodes will update their data to compute the shortest path again.

5.2.1 The computational complexity of Distance Vector Algorithm.

Speed of convergence of the Distance Vector algorithm can converge slowly and can have routing loops while the algorithm is converging, also.



A	Via A	Via L	Via K
To A			
To L		2	8
To K		9	1

L	Via A	Via L	Via K
To A	2		8
To L			
To K	3		7

K	Via A	Via L	Via K
To A	1	9	
To L	3	7	
To K			

Figure 15: All nodes have the shortest path

Distance Vector suffers from the count-to-infinity problem. The Distance Vector algorithm requires message exchanges between directly connected neighbors at each iteration. Each node needs O(N e) storage where N is the number of nodes in the network and m is the number of link, and in the worst case, the total message complexity is $O(n^2m)$ [1,3,5].

6. Comparing the performance of algorithms.

In order to compare the protocols a C++ program simulating them was developed for this project. Table 1 shows the execution time for Link State and Distance Vector, in the experiment used several inputs starting from 50 nodes to 300 nodes and calculate execution time. Figure 16 is a graph showing the difference between the two algorithms, and we note that the gap between Link State and Vector State increases with the number of nodes, due to the different approaches between them**.**

Let us conclude our study of Link State and Distance Vector algorithms with a comparison of the complexities of operation table 2.

In message complexity, Link State requires each node to know the cost of each path in the network, therefore it needs O(NE) messages to be sent. Also, whenever a path cost changes, the new path cost must be sent to all nodes. On the other hand, The Distance Vector algorithm requires message exchanges between directly connected neighbors. Also, whenever a path cost changes, the Distance Vector algorithm will propagate the results of the path cost only
if the new path cost results in a changed least-cost path for one of the nodes attached to that path.

Table 1. Show execution time

Number of nodes	Execution Time (second)	
	Link State	**Distance Vector**
50	0.011	0.016
100	0.02	0.071
150	0.063	0.3
200	0.114	0.491
250	0.213	1.077
300	0.34	1.447

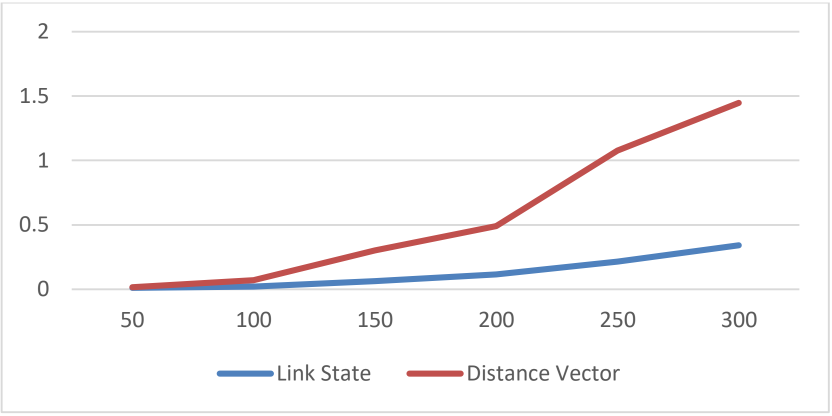


Figure 16: Execution time vs number of node

Speed of Convergence. the Link State is an O(n2) algorithm requiring O(nE) messages and potentially suffering from oscillations. The Distance Vector algorithm can converge slowly and maybe has routing loops while the algorithm is converging. Furthermore, Distance Vector suffers from the count-to-infinity problem.

Table 2. Complexities of Operation

Complexities of Operation	**Link State Routing Algorithm**	**Distance Vector Algorithm**
Message complexity	O(N E)	The exchange between neighbors only
Speed of Convergence	O(n^2) - Relatively fast	-It can converge slowly convergence time varies - Maybe have routing - Count-to-infinity problem

7. Conclusion.

In order to transfer packets from a source to the destination, the network layer must determine the path that the packets are to follow. Therefore, at the heart of any routing protocol is the routing algorithm that determines the path for a packet. The aim of a routing algorithm is to find a short path from source to destination, but there are some concerns that come into play to complicate the work of algorithms. In this paper, we presented the implementation of the link-state and distance-vector routing protocol. It presented an overview and explanation of the ways to operate and demonstrate the different complexities of operation in terms of time variation. Our results indicate that the Link State overall message complexity is O(Ne) while the total message complexity of the Distance Vector is O(n^2m). Both algorithms work correctly only when all nodes maintain paths to all destinations. finally, neither algorithm is the best than the other, and both of them are still used in the internet.

References

[1] James Kurose, Keith Ross, "Computer networking A Top-Down Approach." Seventh edition, Pearson Education 2017.

[2] Elaine Rich, "automata, computability and complexity Theory and Application" second Impression. Dorling Kindersley India 2012.

[3] Distributed Algorithms, http://www.personal.kent.edu/~rmuhamma/. 11 October 2019.

[4] Herbert Edelsbrunner ,Design and Analysis of Algorithmshttps://www.cs.duke.edu/courses/fall08/cps230/Book.pdf. 25September2019.

[5] Abhishek Rai, Kapil kumar, Performance Comparison of Link State and DISTANCE VECTOR Routing Protocols using NS" http://www-public.it-sudparis.eu/~gauthier/Courses/NS-2/FichiersAnnexe/files/routing.pdf. 20November 2021.

[6] Abreu, C., Ricardo, M ; Mendes, P. M, “Energy-aware routing for biomedical wireless sensor networks”. Journal Of Network & Computer Applications, 40270-278. doi:10.1016/j.jnca.2013.09.015, vol 40, pp 270,278 April 2014.

[7] L. Devroye, On the probabilistic worst-case time of FIND in Routing 31 (2001) 291–303.
[8] Tao Chen; Zheng Yang; Yunhao Liu; Deke Guo; Xueshan Luo, "Localization-Oriented Network Adjustment in Routing Wireless Ad Hoc and Sensor problem in Networks," Parallel and Distributed Systems, IEEE Transactions on , vol.25, no.1, pp.146,155, Jan. 2014
[9] Abreu, C., Ricardo, M ; Mendes, P. M, “Energy-aware routing for biomedical wireless sensor networks”. Journal Of Network & Computer Applications, 40270-278.