# Pathfinding Alternatives in OSPF Routing Protocol for Determining Routes on Routers

Aidil Rezjki Suljztan Syawaludin 13517070
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha No. 10 Bandung 40132, Indonesia*
*riskisultan@yahoo.com*

***Abstract*—The world is currently entering a new era. Internet (interconnected network) now has become one of the most important things in our lives. The internet itself is a collection of interconnected networks, that is, networks that are connected to one another. However, to connect networks to other networks, routers are needed. Routers need to know the routes for data to be sent through that guarantees the data to be delivered to the correct destination. This is where the Open Shortest Path First (OSPF) Routing Protocol comes into play. In this paper is going to be discussed several pathfinding methods alternatives that can be implemented based on Open Shortest Path First protocol.**

***Keywords*—Networks, Router, Internet, Path**

## I. Introduction [1]-[3]

The world is currently entering a new era. Internet (interconnected network) now has become one of the most important things in our lives. In March 2019, the number of internet users in the world has reached 4.3 billion users, while the world population itself is around 7.7 billion, thus 56.3% of the whole world population already uses internet. The number of internet users has increased rapidly from around 360 million users back in 2000. This rapid growth is mainly caused by technological advances, especially in information technology. Personal computers were made available for most people with affordable prices, causing the first major growth of internet usage in the early 1990s. Furthermore, the emergence of smartphones in the 2000s and 2010s boosted the growth even more. With affordable means of accessing the internet, the internet users grow exponentially. Another factor for such growth is because the internet offers various things that were not possible before. The internet allows almost limitless information access and exchange. For example, the internet allows for easier communication, without any geographical concerns. Various social medias emerge thanks to the internet. The internet helps in making our lives more convenient and it is natural for it to grow even more. With the internet playing a major part in civilization, it is at most importance to ensure that the internet is functioning properly.

The internet itself is a collection of interconnected networks, that is, networks that are connected to one another. A network consists of two or more devices linked to one another through several connectors, such as cables, radio waves, and others, that are sharing exchanging information and data or sharing resources. The internet is built of these networks. However, to connect networks to other networks, routers are needed. Routers serve a purpose as a gateway, that is, a gate that forwards data to other connected networks and receives data that are directed to the network represented by the router.

As stated before, internet is a collection of interconnected network, and networks are connected using routers. This means that for data to be forwarded to the correct destination, or to be received by the correct receiver, a protocol is needed. Routers need to know the routes for data to be sent through that guarantees the data to be delivered to the correct destination. This is where the Open Shortest Path First (OSPF) Routing Protocol comes into play.

In this paper is going to be discussed several pathfinding methods alternatives that can be implemented based on Open Shortest Path First protocol.

## II. Open Shortest Path First Protocol [4], [5]

Open Shortest Path First (OSPF) protocol is a TCP/IP routing protocol that is an Internal Gateway Protocol, that means OSPF works in a single Autonomous System. Autonomous System is a collection of routers sharing routing informations using the same routing protocol.

In OSPF protocol, a router has a link state database. This link state database contains information of the Autonomous System topology, meaning it contains list of connected devices and networks. In this protocol, every routers need to have exactly same link state databases. To ensure such requirements met, this protocol makes routers do link state advertisements. Link state advertisement is advertising or telling other routers the informations contained in a link state database. Each router will do a link state advertisement, until all routers in the Autonomous System has the same information in their corresponding link state database.

Each router then will build a tree based on the information contained in the link state database. This tree has the router itself as the root and contains the topology of the Autonomous System. Thus, a graph containing devices and networks connected in the Autonomous System is created, and the graph represents the Autonomous System topology.

Consider the following scenario of routers connected to each other inside an Autonomous System:

- Router A is connected directly to Router B, Router C, and Router H. The time it takes for a packet from Router A to reach Router B is 150ms, from Router A to reach Router C is 65ms, and from Router A to reach Router H is 80ms.
- Router B is connected directly to Router A and Router C. The time for a packet from Router B to reach Router A is 150ms and Router C is 60ms.
- Router C is connected directly to Router B, Router E, and Router G. A packet originating from Router E needs 60ms to reach Router B, 80ms to reach Router E, and 140ms to reach Router G.
- Router D is connected directly to Router A, Router E, and Router F. A packet from Router D takes 65ms to reach Router A, 90ms to reach Router E, and 70ms to reach Router F.
- Router E is connected directly to Router C, Router D, and Router G. Any packet that originates from Router E takes 80ms to reach Router C, 90ms to reach Router D, and 70ms to reach Router G.
- Router G is connected directly to Router C and Router E. The time needed to route packets from Router G to Router C is 140ms and to Router E is 70ms.
- Router F is only connected directly to Router D, and it needs 70ms to send a packet from Router F to Router D.

In this scenario, a link state database is built by the routers through mechanisms such as link state advertisements. After that, the following graph is built. This graph represents the current topology on the Autonomous System related to the routers.

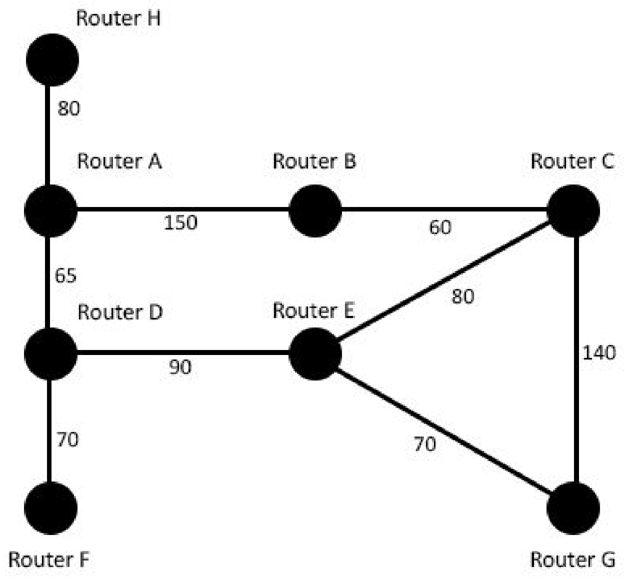

**Fig. 1 A sample graph of an Autonomous System topology built using the link state database ot the routers**

Notice that Router A is actually only connected physically to Router B and Router D. However, Router A still actually can send packets to any other routers currently connected in the Autonomous System. This is because router works as a packet director. When a router receives a packet from another router, it checks whether that packet is sent into any network known by the router. If it is, then the router will receive the packet and send it to the corresponding network through a route that it thinks is the best route. If it is not, then the packet will either be sent to the default gateway of the router (if set), or will be dumped. This mechanism ensure that as long as the routers are connected, be it directly or indirectly, any router is able to send packets to any other routers in the network topology.

The graph representing Autonomous System link state is a weighted graph. The weight on the edge of the graph represents the cost (usually in time needed to the selected router). In OSPF protocol, data will be sent to the destination through the shortest route. Thus, it is important for router to know which route is the fastest to reach the destination.

There are several methods to determine the shortest path to nodes in a weighted graph. In this paper, is going to be discussed some alternatives for determining the shortest path to nodes in a weighted graph, consisting of Greedy Best-First Search and Dijkstra Algorithm. The example scenario aforementioned, Fig. 1, will be used as an example throughout this paper.

## III. Greedy Best First [6]

Greedy Best-First search is one alternative for finding route to a node in a weighted graph. This search algorithm has a simple greedy strategy, that is, to choose a node that looks like it may be the best node to go through to reach the destination. This strategy needs a heuristic evaluation function (f(n)). The heuristic evaluation function will be used in this strategy to determine which node is the most promising to expand to. The following is a basic Greedy Best-First algorithm.

1. Start from the start node as the current node.
2. Check whether the current node is the target node or not. If it is, stop here.
3. Add the current node to visited nodes list.
4. Check all possible path to take from the current node. Choose a path that has the minimal (or maximal, depends on the algorithm goal) cost, in respect to the heuristic evaluation function f(n).
5. Return to step 2.

To further explain how Greedy Best-First search strategy works, consider the following example. A packet from Router A is going to be sent to Router F, in network topology presented in Fig. 1. The time cost will be used as the heuristic evaluation function, and also as the real cost. Then, the strategy will work as follows.

1. Start from Router A as the current node.
2. Check whether the current node is the target node. It is not, so the search continues.
3. There are three possible routes to choose from the current node, that is, through Router B with the time cost of 150ms, through Router D with the time cost of 65ms, and through Router H with the time cost of 80ms. It is perfectly possible to send the packet to Router H, since every router in this topology knows how to reach all the other routers. However, this would create a loop and cause the packet to not reach the destination. This loop can be avoided using another protocol, the Spanning Tree Protocol, but it will not be discussed in this paper.
4. From the three possible routes, this Greedy Best-First strategy will choose the one that looks like the best route. Thus, it will choose the route going to Router D, because it has the minimal cost. Now, the current node is Router D.
5. Check whether the current node is the target node. It is not, so the search continues.
6. From the current node, there are two possible routes to choose, going to Router F and going to Router E. Note that Router A is no longer a possible route, because it has already been visited before.
7. From the two possible routes, the minimal one will be chosen. Thus, Router F will be chosen.
8. Check whether current node is the target node. It is the target node, thus the search ends here.

A path from Router A to Router F is found, which is Router A - Router D - Router F, with the cost of 135ms. Thus, the packet will be sent to the destination through the corresponding routes.

However, this Greedy Best-First strategy does not always guarantee the best route. That is because in this strategy, the current node chosen may choose the fake promising node. It may choose a node that looks like the best route to go through, but is actually not. This may cause a trap in local minimal. For example, consider the following scenario. A packet from Router C is going to be sent to Router D, in the network topology presented in Fig. 1. Then, the strategy will work as follows.

1. Start from Router C as the current node.
2. Check whether the current node is the target node. It is not, so the search continues.
3. There are three possible routes to choose from, through Router B, Router E, and Router G, with the cost of 60ms, 80ms, 140ms respectively. The algorithm will choose Router B, because it considers the path through Router B more promising than the others. Now, the current node is Router B.
4. Check whether the current node is the target node. It is not, so the search continues.
5. There is only one possible route to choose from, which is going through Router A. Thus, the current node now is Router A.
6. Check whether the current node is the target node. It is not, so the search continues.
7. There are two possible routes, going through Router D and through Router H, with the cost of 65ms and 80ms. In this step, the path going through Router D is chosen. Thus, the current node is now Router D.
8. Check whether the current node is the target node. It is, thus the search ends here.

Going through the algorithm shows that the path found is through Router C - Router B - Router A - Router D, with the cost of 275ms. However, by looking at the graph in Fig. 1, we can see that there is actually another route that has a lower cost than the route gotten from the Greedy Best-First search algorithm. That route is Router C - Router E - Router D, with the cost of 170 ms. This shows that Greedy Best-First search strategy does not always guarantee the best path.

## IV. Dijkstra Algorithm [7], [8]

Dijkstra Algorithm is a pathfinding algorithm that is classified as a *single source shortest path* algorithm in graph theory, which means, this algorithm solves for all paths from a single source or node to all the other nodes in a graph. This algorithm works on both directed and undirected graphs. However, Dijkstra Algorithm requires the graph to be connected, that is, to have all nodes connected without any isolated subgraphs or nodes. This algorithm also requires the weight of the graph to be nonnegative.

Dijkstra Algorithm works by having a list of visited nodes, list of current shortest path cost from source node to other nodes, and the previous path to a node list. The visited nodes are used to ensure that the algorithm does not check the same node more than once. The list of current shortest path cost from source node to other nodes is used to keep track of the costs from the source node to other nodes and to determine which node to check next. While the previous path to a node list is used to rebuild the path that was found when the algorithm runs.

| Visited Nodes |
|---|
| {Node A, Node B, Node C, … } |

**Table 1 Example of the visited nodes list**

| Nodes | Node A | Node B | Node C | ... |
|---|---|---|---|---|
| Current Cost | 0 | 3 | 5 | ... |
| Previous Nodes | - | A | B | ... |

**Table 2 Example of the costs and previous nodes table**

The following steps are the basics of Dijkstra Algorithm.

1. Set all the current cost in the table to infinity, except the cost to the source node, set all the previous nodes to none, and set the visited nodes as an empty set.
2. Start from the source node as the current node.
3. For all the unvisited nodes N connected to the current node, find the cost to node N, by adding the cost from source node to the current node with the cost from the current node to node N. Check the result cost with the cost that is currently in the cost table. If the result cost is less than the cost currently in the cost table, update the table with the result cost and update the previous node with the current node.
4. Add the current node to the list of visited nodes.
5. Select a new current node, that is, by selecting from the unvisited nodes that has the lowest current cost in the table. If none exists, then the algorithm stops here.
6. Return to step 2.

To further understand how Dijkstra Algorithm works, consider the following example. A packet is going to be sent from Router C to Router D, in the network topology presented in Fig. 1. The Dijkstra Algorithm works as follows.

Set all the current cost in the table to infinity, except the cost to the source node, in this case is Router A, then set all the previous nodes to none, and set the visited nodes as an empty set.

| Visited Nodes |
|---|
| {} |

**Table 3 Initial visited nodes list**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
| Previous Nodes | - | - | - | - | - | - | - | - |

**Table 4 Initial costs and previous nodes table**

Start from Router C as the current node. Router C is connected directly to three unvisited nodes, which are Router B, Router E, and Router G. Each connected node will be examined. The cost from the source node, which is Router C, to the current node, which is also Router C, is 0. Examining Router B, the cost from the current node to Router B is 60ms. Comparing the cost from source to Router B with the current cost to Router B in the costs table, it costs 60ms, comes from 60ms + 0ms, which is less than infinity, thus the table is updated. Onto the next, Router E costs 80ms from the current node, and costs 80ms from the source node, which is less than infinity, thus the table is updated. Now, Router G costs 140ms from the current node, and costs 140ms from the source node, thus the table is updated. After all the nodes connected directly to the current node has been examined, the current node is added to the list of visited nodes. The tables are updated as follows.

| Visited Nodes |
|---|
| {Router C} |

**Table 3 Visited nodes list after checking on Router C as the current node**

| Nodes | Router | Router | Router | Router | Router | Router | Router | Router |
|---|---|---|---|---|---|---|---|---|

| | C | A | B | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | ~ | 60 | ~ | 80 | ~ | 140 | ~ |
| Previous Nodes | - | - | C | - | C | - | C | - |

**Table 4 Costs and previous nodes table after checking on Router C as the current node**

Now, another node needs to be checked as the current node. The next current node needs to be an unvisited node with the least known cost. The possible nodes are Router B, Router E, and Router G. Router B is selected as the next current node, as it have the minimal known cost and is unvisited.

The current node, which is Router B, is connected to only unvisited node Router A. Router C is ignored because it is already visited before. The cost from the source node to the current node is 60ms. Examining Router A, the cost from the current node to Router A is 150ms, which means that it adds up to 210ms as the cost from source node to Router A. Comparing the result to the current known cost from source node to Router A, which is infinity, 210ms is less. Thus the table is updated. All unvisited nodes that are connected to the current node has already been examined, now the current node is added to the visited nodes list. The updated tables are as follows.

| Visited Nodes |
|---|
| {Router C, Router B} |

**Table 5 Visited nodes list after checking on Router B as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | ~ | 80 | ~ | 140 | ~ |
| Previous Nodes | - | B | C | - | C | - | C | - |

**Table 6 Costs and previous nodes table after checking on Router B as the current node**

Router E is the selected next current node, because it has the least known cost and is unvisited.

The current node, which is Router E, has connections to two unvisited nodes, which are Router D and Router G. The cost from the source node to the current node is 80ms. From the current node to Router D costs 90ms, and as total adds up to 170ms from the source node to Router D. Comparing that to the current known cost, which is infinite, 170ms costs less, and the table is updated. From the current node to Router G costs 70ms, and as total adds up to 150ms from the source node to Router G. Comparing that to the current known cost, which is 140ms, it is not better. And the table is not updated for Router G. All the unvisited connected nodes have been examined, thus the current node is added to the visited list node. The table after this iteration is as follows.

| Visited Nodes |
|---|
| {Router C, Router B, Router E} |

**Table 7 Visited nodes list after checking on Router E as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | 170 | 80 | ~ | 140 | ~ |
| Previous Nodes | - | B | C | E | C | - | C | - |

**Table 8 Costs and previous nodes table after checking on Router E as the current node**

Router G is selected as the next current node. However, the current node, which is Router G, has no unvisited nodes connected directly. Thus, the table is not updated, but Router G is still added to the visited nodes list.

| Visited Nodes |
|---|
| {Router C, Router B, Router E, Router G} |

**Table 9 Visited nodes list after checking on Router G as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | 170 | 80 | ~ | 140 | ~ |
| Previous Nodes | - | B | C | E | C | - | C | - |

**Table 10 Costs and previous nodes table after checking on Router G as the current node**

Now, Router D is selected as the next current node.

Router D as the current node neighbors Router A and Router F directly. The cost from the source node to the current node is 170ms. Checking on Router A as a neighbor, it costs 65ms from the current node, and adds up to 235ms as total from the source node. It is no less than 210ms, the current known cost, so the table is not updated for Router A. Checking on Router F as a neighbor, it costs 70ms from the current node, and adds up to 240ms from the source node. It is less than infinity, the current known cost, so the table is updated for Router F. All the unvisited neighbors of the current node is already checked and the current node is added to the visited nodes list. The table becomes as follows.

| Visited Nodes |
|---|
| {Router C, Router B, Router E, Router G, Router D} |

**Table 11 Visited nodes list after checking on Router D as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | 170 | 80 | 240 | 140 | ~ |
| Previous Nodes | - | B | C | E | C | D | C | - |

**Table 12 Costs and previous nodes table after checking on Router A as the current node**

Now, Router A is selected as the next current node.

Router A, the current node, has only Router H as unvisited neighbors. From the source node to the current node costs 210ms. Checking on Router H as a connected neighbor, it costs 80ms from the current node. It totals to 290ms from the source node to Router H. Comparing that result to the current known cost, which is infinite, it is certainly less, thus the table is updated. All the unvisited neighbors of the current node is already checked, thus the current node is added to the visited node list. Now, the tables look as follows.

| Visited Nodes |
|---|
| {Router C, Router B, Router E, Router G, Router D, Router A} |

**Table 13 Visited nodes list after checking on Router D as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | 170 | 80 | 240 | 140 | 290 |
| Previous Nodes | - | B | C | E | C | D | C | A |

**Table 14 Costs and previous nodes table after checking on Router A as the current node**

Now, Router F becomes the current node. Router F, only neighbors Router D, and that node is already visited, thus the neighbor check is skipped and the costs and previous nodes table is unchanged. However, Router F is still added to the visited nodes lists. The tables becomes as follows.

| Visited Nodes |
|---|
| {Router C, Router B, Router E, Router G, Router D, Router A, Router F} |

**Table 15 Visited nodes list after checking on Router F as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | 170 | 80 | 240 | 140 | 290 |
| Previous Nodes | - | B | C | E | C | D | C | A |

**Table 16 Costs and previous nodes table after checking on Router F as the current node**

The only unvisited node now is Router H, and now becomes the next current node. However, the current node, Router H, only neighbors Router A, and it is already visited before, thus there are no checkings in this iteration, but Router H is still added to the visited nodes list.

| Visited Nodes |
|---|
| {Router C, Router B, Router E, Router G, Router D, Router A, Router F, Router H} |

**Table 17 Visited nodes list after checking on Router H as the current node**

| Nodes | Router C | Router A | Router B | Router D | Router E | Router F | Router G | Router H |
|---|---|---|---|---|---|---|---|---|
| Current Cost | 0 | 210 | 60 | 170 | 80 | 240 | 140 | 290 |
| Previous Nodes | - | B | C | E | C | D | C | A |

**Table 18 Costs and previous nodes table after checking on Router H as the current node**

Now, all the nodes in the graph has already been visited, and the algorithm stops here. From the table built by running through the Dijkstra Algorithm, the shortest path from a single source, Router C, to any other nodes in the graph is found, with the respective cost.

Back to the example problem before, the shortest path for a packet originating from Router C to Router D is now found, with the cost of 170ms. However, the information here is only about the cost of the path and the previous nodes to reach a node. Thus, a path needs to be rebuilt from the information.

In rebuilding the path, a stack data structure may be used. The path is rebuilt by traversing backward, that is, traversing from the goal node, Router D, back to the source node, Router C. First, push the node Router C to an empty stack. Then, check the previous node, which is Router E. Push that Router E node to the stack, and check the previous node again, which is Router C. Push that Router C node to the stack. Now, because node Router C is the source node, the path is found. The stack looks like as follows.

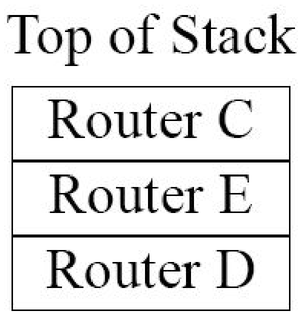

**Fig. 2 Stack for the path rebuilt from Router D to Router C**

The stack now represents the path that should be taken to send a packet originating from Router C to Router D. The path can be taken by popping the path stack, until the stack is empty.

With the Dijkstra Algorithm, the shortest path is guaranteed to be found. This algorithm does not only find a path from a single source to a single goal node, but also finds all the path from that single source to any other nodes in the graph. Thus, now Router C knows exactly to what route to send any packets to.

## V. Comparison and Conclusion

The Greedy Best-First search strategy does not need so much computations to run. However, this strategy does not have any guarantee that the result will be the optimal result. As seen in this paper, this strategy may not produce the most optimal result, as it can get trapped into a local minimal.

The Dijkstra Algorithm on the other hand needs more computation, since it loops through the nodes of a graph and the neighbors of each. However, this algorithm ensures that the result produced is already the optimal result. This algorithm also produces all the optimal routes from a single source node to any other nodes in the graph just by running it once. Though, if the graph changes, it needs to be run again, but the re-run may not be as heavy as the first run if the algorithm is modified.

Pathfinding in routers needs to be as accurate as it can, while having moderate computations is not necessarily a problem. This means that even though Greedy Best-First strategy may be used in routers pathfinding, it is best to avoid because the result is not always optimal. However, the Dijkstra Algorithm that ensures optimal result is suited to the task, and to compensate the computation process, this algorithm is only needed to be run when the network topology is changed. Thus, no constant computation is needed. Also, Dijkstra Algorithm finds all routes, instead of one routes, from a single source. This is suited for the nature of routers routes, where all routes are needed to be known by routers. Thus, the Dijkstra Algorithm is well-suited for pathfinding in routers.

## VI. Acknowledgement

The author of this paper thanks the lecturer of the authors class of Algorithm Strategy, Dr. Nur Ulva Maulidevi ST, M.Sc, for the knowledge shared and taught in the class and also the guidance that allows the author to finish this paper. The author also thanks all the authors of the references, also to others that may have helped the making of this paper directly and indirectly.

## References

[1] World Internet Users and 2019 Population Stats. Retrieved on 24 April 2019 from https://www.internetworldstats.com/stats.htm

[2] Internet Growth in 2000 to 2005 in the World. Retrieved on 24 April 2019 from https://www.internetworldstats.com/emarketing.htm

[3] Internet Growth Statistics 1995 to 2019. Retrieved on 24 April 2019 from https://www.internetworldstats.com/emarketing.htm

[4] What is a Network?. Retrieved on 25 April 2019 from https://fcit.usf.edu/network/chap1/chap1.htm

[5] RFC2328 - OSPF Version 2. Retrieved on 25 April 2019 from https://tools.ietf.org/html/rfc2328#section-1

[6] Munir, R. (2019). Route/Path Planning. Retrieved on 25 April 2019 from http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf

[7] Munir, R. (2019). Algoritma Greedy. Retrieved on 25 April 2019 from http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf

[8] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". *Introduction to*

*Algorithms* (Second ed.). MIT Press and McGraw–Hill. pp. 595–601. ISBN 0-262-03293-7.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2019

[illegible]

Aidil Rejzki Suljztan Syawaludin
13517070