

A Multi-path Nonoverlapping Routing Algorithm for Deterministic Networking

Jiawang Shao ^a, Gengyu Wei ^b, Shenghua Dong ^c

School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China

^a sjww1997@bupt.edu.cn, ^b weigengyu@bupt.edu.cn, ^c shenghuadong@bupt.edu.cn

Abstract. Deterministic Networking (DetNet) is a new network architecture proposed by IETF DetNet working group for industrial Internet of Things and other application scenarios. Its communication service with high reliability and low delay needs multi-path nonoverlapping routing algorithm. Because it is different from existing routing algorithms on the Internet, it is an important subject to study routing algorithms that meet requirements of DetNet. Based on routing characteristics of DetNet, we studied a multi-routing algorithm - Bhandari algorithm. Aiming at the problem that the algorithm cannot calculate nonoverlapping routes in some cases, we improve the algorithm, then we also make the improved algorithm more efficient by introducing Tarjan algorithm. The improved algorithm is verified and tested in the simulation environment and results show the effectiveness of the improved algorithm.

Keywords: Deterministic Networking, Multi-path nonoverlapping routing algorithm, Bhandari algorithm, Tarjan algorithm.

1. Introduction

In order to transmit 4K or 8K high-definition video in Ethernet environment, IEEE started the standardization work of 802.1 TSN (Time Sensitive Network), which is applied to transmit delay-sensitive real-time audio or video services in enterprises or families and can also be applied to high-quality multimedia transmission, industrial control systems and other applications. Application scenarios of the network are defined in enterprises or households, which are mainly used in real-time audio or video applications with time delay sensitivity, as well as multimedia transmission, industrial control systems and other general industrial and vehicle applications. The network technology of TSN is limited to Ethernet or other IEEE 802 series networks to provide end-to-end communication services with high reliability and low delay.

IETF DetNet (Deterministic Networking) working group put forward DetNet, which is a deterministic network based on Internet technology, as a new communication architecture of WAN (Wide Area Network). DetNet mainly guarantees end-to-end high reliability and low delay through three core technologies [5]: Resource Allocation (RA) [2], Service Protection (SP) [3] and Explicit Routing (ER) [4]. The purpose of resource allocation is to provide a high-reliability and low-latency transmission path for DetNet flow by allocating resources (such as buffers and link bandwidth) on the path. With resources allocations, DetNet flow can basically eliminate packet loss caused by output packet contention. Once the path is lost due to random media errors or equipment failures, service protection and explicit routing will produce corresponding capabilities. Service protection capability guarantees packet coding and orderly transmission, and addresses errors in the path through redundancy mechanism of packet replication and elimination. On the one hand, explicit routing selects the route of packet transmission, especially the selected route supports the transmission of multiple copies of data packets, thus requiring multiple nonoverlapping paths, which is at least two routes. The purpose of choosing two or more nonoverlapping routes is to ensure that packets are not lost due to characteristics of a certain path. Therefore, DetNet chooses multi-path nonoverlapping routing algorithm, which is different from existing routing algorithms in Internet.

There is a simple method to find two nonoverlapping paths from source to destination in a network, but the simple method has obvious problems. This simple method is divided into two steps: finding a shortest path between two vertexes; delete all links on this path from the graph to get a modified

graph, and find a shortest path on the modified graph. The algorithm has at least two defects. The first defect is that even if there are two nonoverlapping paths, they may not be found. The second one is the second path may be too long. In view of the above defects, Suurballe[6] proposed an algorithm to find two nonoverlapping paths, and the total weight of the paths is the smallest. Furthermore, Bhandari [7] modified the original Dijkstra algorithm on the basis of Suurballe algorithm to allow the negative weight on the shortest path, avoiding the cost transformation and allowing the operation of directly setting all links of the first shortest path as negative weights. This improvement made Bhandari algorithm have the advantages of completeness and optimality in Suurballe algorithm, and also increase the simplicity of the algorithm.

However, through the research and analysis of Bhandari algorithm, we pointed out that this algorithm cannot be directly applied to DetNet. Because in some cases, Bhandari algorithm cannot calculate nonoverlapping paths even if they are. To solve this problem, we propose an improvement to Bhandari algorithm. In the initial algorithm improvement work, RS (Remove-Search) algorithm is used to judge the necessary link. Although the improved Bhandari algorithm meets requirements of DetNet, the efficiency of the improved algorithm is low. We further introduce Tarjan algorithm [8], which obviously improves the efficiency of the improved algorithm. Simulation and experimental results show that our proposed algorithm is an effective algorithm for DetNet routing.

The structure of our subsequent chapters is as follows. In the second chapter, we firstly introduce Bhandari algorithm and point out the existing problems of this algorithm in DetNet scene. Then, we describe the improvement of Bhandari algorithm based on RS algorithm. Finally, we describe the efficiency improvement of the improved algorithm by introducing Tarjan algorithm. The third chapter is simulation and experiment. By simulating Bhandari algorithm and the improved algorithm, success rate and response time are evaluated on typical network topology, which shows the effectiveness of the improved algorithm. The fourth chapter summarizes the research approaches and achievements of our work.

2. Research content

In this chapter, we will firstly introduce Bhandari algorithm and problems it will encounter in DetNet scene. Then, we will improve it and introduce Tarjan algorithm to make the improved algorithm more efficient.

2.1 Bhandari algorithm

The main steps of the algorithm are as follows:

- 1) Calculate a first shortest path on the original topological graph by using the shortest path algorithm to obtain a first shortest path.
- 2) On the basis of the original topological graph, delete the link on the first shortest path, and add the link which is opposite to it and whose weight is the inverse of the original weight to generate a modified graph.
- 3) Calculate the shortest path on the modified graph to obtain a second shortest path.
- 4) Integrate two paths, delete reverse links in two paths, and get two non-overlapping paths that have no overlapping link.

The steps of Bhandari algorithm are very simple and direct, which does not involve complex cost transformation. If there are nonoverlapping paths, the corresponding result can be calculated, and the result is the lowest total cost among all solutions. It is an excellent nonoverlapping path algorithm. However, this algorithm has not been applied to DetNet research, and the problem of how Bhandari algorithm calculates the routing path in DetNet needs further exploration.

2.2 Defects and improvement of Bhandari algorithm in DetNet scene

Bhandari algorithm is an excellent nonoverlapping path algorithm, but it still has defects in DetNet scene. As shown in Fig. 1, taking the nonoverlapping path from node A to node F as an example.

Subgraph (a) is the initial topological graph, and the delay of the link is marked on the link. Subgraph (b) is the first shortest path from node A to node F obtained by successful calculation. However, after deleting links on the first shortest path based on the original topological graph and adding links with negative reverse weights, the second shortest path cannot be calculated so that Bhandari algorithm cannot calculate two nonoverlapping paths. [9]

As shown in subgraph (a), there is no strictly nonoverlapping path, which means each of link of two paths is different, between two nodes A and F. So it is impossible to calculate the result according to the strictly nonoverlapping principle. Subgraph (a) does not have two strictly nonoverlapping paths from node A to node F, because there is a necessary link (E, F). That is to say, all paths starting from node A must pass through this link to reach node F. Therefore, subgraph (a) does not have strictly nonoverlapping paths between nodes A and node F. However, in DetNet environment, it is too strict to require paths to be strictly nonoverlapping.

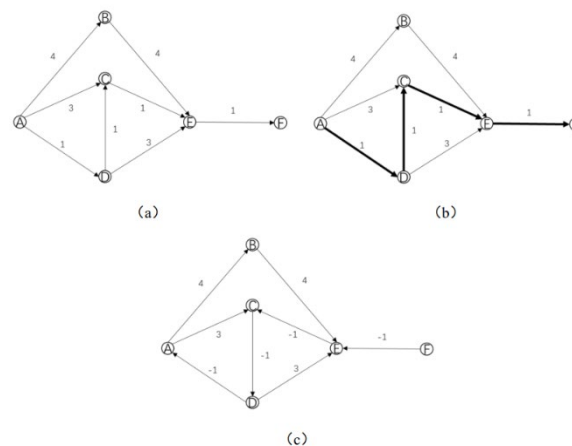


Figure 1. Bhandari algorithm fails to calculate the second shortest path

The core idea of DetNet service guarantee is to transmit replicas through multiple paths to avoid the degradation of service quality caused by irresistible link damage or node failure. However, the RFC standard document of DetNet does not stipulate that the nonoverlapping paths in DetNet must be strictly nonoverlapping. Therefore, we assume that nonoverlapping paths in DetNet should be as nonoverlapping as possible. If there is a necessary link between nodes, the overlapping of this link can be tolerated. So it is totally unnecessary for paths pursued in Bhandari algorithm to be strictly nonoverlapping in DetNet.

According to the above analysis, we improve Bhandari algorithm, and the steps of the improved Bhandari algorithm are as follows:

- 1) Calculate the shortest path in the original topological graph by using the shortest path algorithm to obtain a first shortest path.
- 2) Sequentially judging whether each link in the first shortest path is a necessary link or not. If not, deleting the link on the basis of the original topological graph, adding links with opposite directions and weights of the original weights, and generating a modified graph.
- 3) Calculate the shortest path on the modified graph to obtain a second shortest path.
- 4) Integrate two paths, and delete reverse links in two paths so that there is no overlapping link except the necessary link, that is, the nonoverlapping paths are obtained.

2.3 Improve the efficiency of the improved Bhandari algorithm

The improved Bhandari algorithm involves judging whether a link is a necessary link, and there is a simple method to judge whether a link is a necessary link between two nodes, which consists of two steps:

- 1) Delete the link on the basis of the original topological graph and generate a modified graph.

2) On the modified graph, starting from the source vertex, search in depth-first search mode to detect whether the source vertex can reach the end vertex. If so, the link is not a necessary link. If not, it is a necessary link.

In this paper, this method is called Remove-Search algorithm or RS algorithm for short. And we look up the relevant material and find that there is a more efficient method for judging necessary links. We use Tarjan algorithm here.

Two auxiliary arrays are needed in Tarjan algorithm: dfn and low. The dfn array is used to store the search order number of each vertex, which can be understood as timestamp. The low array is used to store the earliest sequence number that each vertex can trace back to, which is still in the recursive stack. Tarjan algorithm mainly consists of the following steps:

- 1) Make $dfn[u] = low[u] = \text{current timestamp}$ when vertex u is searched for the first time.
- 2) Every time a vertex is searched, push it into the top of the stack.
- 3) When vertex u and vertex v is connected:
 - a) If v is not in the stack, firstly traverse v with depth first, then make $low[u] = \min \{low[u], dfn[v]\}$.
 - b) $low[u] = \min \{low[u], dfn[v]\}$, if v is in the stack.
- 4) When $dfn[u]$ equals $low[u]$, the vertex u and the elements above it are popped out of the stack.
- 5) Continue searching until all vertexes in the graph have been searched.
- 6) Traverse all links, for a link (v, w) , if $dfn[v] < low[w]$, the link is a necessary link.

Tarjan algorithm only needs one depth-first search to find out all necessary links in the graph. Then it only needs one query to judge whether a certain link is necessary. So, we can reduce the time complexity from $O(n*(n+m))$ of RS algorithm to $O(n+m)$, in which n represents the number of nodes and m represents the number of links. This further improves the efficiency of the improved Bhandari algorithm. Then the steps of the improved Bhandari algorithm after introducing Tarjan algorithm are as follows:

- 1) Use Tarjan algorithm to calculate all necessary links in the original topological graph.
- 2) Calculate the shortest path in the original topological graph by using the shortest path algorithm to obtain a first shortest path.
- 3) Sequentially judging whether each link in the first shortest path is a necessary link or not. If not, deleting the link on the basis of the original topological graph, and adding a link with opposite directions and weight to the original weight to generate a modified graph.
- 4) Calculate the shortest path on the correction graph to obtain a second shortest path.
- 5) Integrate two paths, and delete reverse links in two paths, so that there is no overlapping link except the necessary link, that is, the nonoverlapping paths are obtained.

The improved Bhandari algorithm not only improves defects of Bhandari algorithm in DetNet scene, but also does not increase the overall time complexity of the algorithm because the time complexity of Tarjan algorithm is lower than that of the shortest path algorithm.

3. Experiment

In order to test the performance of the improved Bhandari algorithm in the network, experiments are carried out in different scale networks aiming at the success rate and average response time of these algorithms. Because the transmission process in the network is unpredictable, the network has certain randomness. Therefore, in order to solve this problem, different scale networks with paths between the source vertex and the end vertex are randomly generated in the experiment. Network scales are (30, 70), (30,80), (30,90), (60,140), (60,150). The former parameter represents the number of nodes in the network, and the latter parameter is the number of links in the network. And the link delay, as the link weight in the network topological graph, is evenly distributed between 1 and 5 milliseconds.

In this chapter, the nonoverlapping path between the source vertex and the end vertex is calculated in the above randomly generated network. The specific experimental environment is as follows:

Processor: Intel (R) Core (TM) i5-8500 CPU @ 3.00 GHz.
Operating system: Windows10 Home Edition.
Memory: 8.00GB DDR4 2666MHz.
Development language: C++.
Integrated development environment: Visual Studio Community 2017.

3.1 Effectiveness

In order to verify the improvement of the improved Bhandari algorithm in calculating nonoverlapping paths, the experiment was conducted under the above experimental condition, and the success rate was recorded.

Assuming that N networks are randomly generated for the same scale in an experiment, the number of times that nonoverlapping paths can be successfully calculated is K, and the recording success rate is P, there is a formula:

$$P = (K / N) \times 100\% \quad (1)$$

For each network of different scales, 10,000 networks with paths between source and end vertex were randomly generated in the experiment. The nonoverlapping paths were calculated by Bhandari algorithm and improved Bhandari algorithm respectively, and their success rates were compared.

The experimental result is shown in Table I, Bhandari algorithm has obvious defects in random networks, and overlapping paths cannot be calculated in all scale networks in the experiment. It can be seen from the table that when the number of the nodes is constant, the smaller the number of links, the lower the success rate. When the ratio of the number

Table 1. Comparison of the success rate of algorithms before and after improvement in computing overlapping paths in random networks of different scales (percentage).

Network Scale \ Before and After	Before	After
(30, 70)	93.36	100
(30, 80)	96.81	100
(30, 90)	98.53	100
(60, 140)	91.88	100
(60, 160)	96.19	100
(60, 180)	97.64	100
(90, 210)	92.12	100
(90, 240)	94.64	100
(90, 270)	97.31	100
(120, 280)	90.96	100
(120, 320)	94.58	100
(120, 360)	96.57	100

of links is constant, the higher the number of nodes, the lower the success rate. This is because when the number of nodes in the network is the same, the less the number of links, the sparser the network is. The larger the number of nodes, the larger the scale of the network and the more links that can exist, which is proportional to the square of the number of nodes. Therefore, when the ratio of the number of nodes to the number of links is constant, the more nodes, the sparser the network is. This shows that the sparser the network is, the more likely there are necessary links between the source vertex and the end vertex, and the more likely Bhandari algorithm is to be unable to calculate nonoverlapping paths. Experiments also prove that the improved Bhandari algorithm solves defects of Bhandari algorithm in DetNet environment, is not affected by the necessary link between the source vertex and the end vertex, and can ensure the calculation of nonoverlapping paths. In the experiment, nonoverlapping paths can be successfully calculated in random networks of all sizes, and the

calculation success rate is 100%. This experiment shows that the success rate of the improved Bhandari algorithm is higher than that of the Bhandari algorithm in DetNet environment, and the improved Bhandari algorithm has higher effectiveness, which can ensure that the nonoverlapping path meets the requirements.

3.2 Average response time

In the previous chapter, it is analyzed theoretically that the time complexity of the improved Bhandari algorithm is the same as that before the improvement, and this section verifies it from the experimental point of view.

Let N be the number of randomly generated networks in an experiment, and it takes $T_1, T_2, T_3, \dots, T_N$, the average response time is T_{avg} , then there is a formula:

$$T_{avg} = \frac{\sum_{i=1}^N T_i}{N} \quad (2)$$

Table 2. Comparison of algorithms before and after improvement in calculating average response time of nonoverlapping paths in random networks of different scales (unit: microsecond).

Network Scale	Before and After	Before	After
(30, 70)		7.48609	7.84872
(30, 80)		7.20981	7.35776
(30, 90)		7.46963	7.54295
(60, 140)		13.3223	13.9507
(60, 160)		13.7511	14.0528
(60, 180)		14.1694	14.3712
(90, 210)		22.8389	23.8415
(90, 240)		22.3225	22.9468
(90, 270)		22.3782	22.3782
(120, 280)		31.2497	32.5643
(120, 320)		32.9112	33.739
(120, 360)		34.0408	34.5968

For each network of different scales, the experiment randomly generated 10,000 networks with paths at source and end vertex, calculated nonoverlapping paths by Bhandari algorithm and improved Bhandari algorithm respectively, and compared their average response time.

The experimental result is shown in Table II. The average response time of the improved Bhandari algorithm is slightly longer than that of Bhandari algorithm in all scale networks, but the difference is very small, no more than 1 microsecond, and with the expansion of network scale, the difference is getting smaller and smaller. This is because even though the time complexity of two algorithms is the same, but the improved Bhandari algorithm needs to add some extra computation to improve the defect. Although the computation has been reduced by introducing Tarjan algorithm, it will still increase the computation and thus increase the time compared with that before the improvement. On the other hand, with the expansion of the network scale, the increased amount of computation is getting smaller and smaller compared with the total amount of computation, and the gap between the two is getting smaller and smaller. This experiment shows that in DetNet environment, there is a microsecond difference between two algorithms, and the gap narrows with the expansion of the network scale. Therefore, the efficiency of the improved Bhandari algorithm is still excellent.

4. Summary

In view of the lack of routing algorithms that meet characteristics of DetNet, we studied Bhandari algorithm, and pointed out that this algorithm cannot guarantee the calculation of nonoverlapping

paths in some cases. By analyzing Bhandari algorithm, we pointed out that it is looking for strictly nonoverlapping paths, which is unnecessary in DetNet. By improving Bhandari algorithm, it turned to looking for nonstrictly nonoverlapping paths, thus improving this defect. Then Tarjan algorithm was introduced so that all necessary links in the graph can be found only in linear time. And the operation of judging whether a certain path is a necessary link would not become the bottleneck of the improved algorithm so that the time complexity of the improved Bhandari algorithm is consistent with that before the improvement.

References

- [1] Eveleens J. Ethernet AVB overview and status[A]. // SMPTE 2014 Annual Technical Conference & Exhibition [C]. SMPTE, 2014: 1-11.
- [2] Shubha Rao V and M Dakshayini. Priority based optimal resource reservation mechanism in constrained Networks for IOT applications[A]. // 2016 International Conference on Wireless Communications, Signal Processing and Networking (Wi SPNET) [C]. Chennai, 2016:1228-1233.
- [3] Y Zhao, B Zhang, L Wang, F Xie and C Li, Selective Redundant Transmissions for Real-Time Video Streaming Over Multi-Interface Wireless Terminals[A]. // 2016 IEEE Global Communications Conference (GLOBECOM) [C]. Washington, DC, 2016: 1-6.
- [4] H Owens and A Duresi. Explicit Routing in Software-Defined Networking (ERSDN): Addressing Controller Scalability[A]. //2014 17th International Conference on Network-Based Information Systems[C]. Salerno, 2014: 128-134.
- [5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press. Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019.
- [6] Suurballe, J. W. ""Disjoint paths in a network. " Networks: An International Journal 4.2 (1974): 125-145.
- [7] Bhandari R. Optimal diverse routing in telecommunication fiber networks [A]//Proceedings of INFOCOM'94 Conference on Computer Communications [C]. IEEE, 1994: 1498-1508 .
- [8] R. Tarjan, "Depth-first search and linear graph algorithms," 12th Annual Symposium on Switching and Automata Theory (swat 1971), 1971, pp. 114-121, doi: 10.1109/ SWAT.1971.10.
- [9] Shenghua Dong. Research on Delay Stability Mechanism in DetNet Based on NS3 [D]. Beijing University of Posts and Telecommunications, 2021.