One-Shot Multiple Disjoint Path Discovery Protocol (1S-MDP)

Diego Lopez-Pajares, Joaquin Alvarez-Horcajo, Elisa Rojas, Juan A. Carral and Isaias Martinez-Yelmo

***Abstract*—Multipath routing over disjoint paths is a classic solution to allow better resource allocation, resilience, and security. Current proposals rely on centralised computation or iterative distributed algorithms and exhibit large convergence times. We propose 1S-MDP, a distributed mechanism based on a single network exploration with concurrent path selection to discover multiple available paths among the target node and the remaining nodes in the network. The paper evaluates 1S-MDP in two different scenarios against previous solutions. We show how it reduces the convergence time by several orders of magnitude with a small decrease in the number of disjoint paths discovered.**

***Index Terms*—Multiple disjoint paths, Distributed environments, Ethernet networks, Network exploration, Concurrence.**

I. Introduction

THE calculation of multiple paths is a classic problem from computer science. The challenge is even tougher if we require these paths to be both the shortest and disjoint, features that hold many benefits, such as allowing better resource distribution or boosting resiliency to failures in the specific case of computer networks.

Current algorithms and protocols to obtain multiple paths in wired networks are mainly based on centralized computation techniques over a previously acquired network topology graph. They exhibit a high computational cost both in terms of time and computing resources and depend on other protocols or applications to discover the topological information to build the network graph. To decrease the global computational cost and minimize the convergence time, they can be distributed among the network devices, so that, each network node only computes local information collected by an active packet exchange with its neighbors. Moreover, a distributed approach may increase resilience by deleting single failure points while enhancing security by decentralizing data storage. However, we must be very careful not to impose strong processing requirements on networks nodes that, in a worst case scenario, may further aggravate the convergence time.

Our motivation is to propose an alternative solution that leverages the benefits of distributed processing, while avoiding the above mentioned drawbacks. 1S-MDP is a fully distributed protocol designed to discover multiple disjoint paths by applying an exploration technique to discover the network topology followed by a concurrent path selection phase to set up the paths. The exploration phase guarantees an updated knowledge of the network topology while the concurrent phase path selection drastically reduces the convergence time. Furthermore, the protocol can work in two different modes to provide node-disjoint or link-disjoint paths as desired.

Diego Lopez-Pajares, Joaquin Alvarez-Horcajo, Elisa Rojas, Juan A. Carral and Isaias Martinez-Yelmo are with Dpto. de Automatica, University of Alcala, 28805, Spain. Corresponding author: elisa.rojas@uah.es

This work was funded by grants from Comunidad de Madrid through Project TAPIR-CM (S2018/TCS-4496) and Project IRIS (CM/JIN/2019-039), and from University of Alcalá through ”Programa de Formación del Profesorado Universitario (FPU)”.

The paper is structured as follows: In Section II, we review the current state of the art regarding the multipath and the disjoint path problems. In Section III we define the 1S-MDP protocol and its features. Finally, we evaluate the protocol in Section IV and provide the main conclusions in Section V.

II. Related work

Solutions to finding multiple paths in wired networks can be grouped in two big categories depending on whether they rely on the network topology knowledge (usually depicted as a graph of nodes and links) to operate.

Within the first category (those depending on a graph to later compute the paths), Suurballe and Tarjan [1] were the first to study the disjoint multipath problem. Both developed an algorithm based on the Dijkstra's algorithm [2] that finds two link-disjoint paths in a directed graph, keeping the computation time at the same bounds than Dijkstra's algorithm, while adding the disjointness. Their work was later enhanced with new features such as reducing the algorithm complexity, disjointness or convergence time [3]–[5].

The second category (those independent of topology knowledge) groups solutions that work in a distributed manner to obtain the paths, usually via packet exchanges that convey the topological information needed to discover the available paths. Itai and Rodeh [6] proposed a method to transform centralized algorithms into distributed ones. They also provided an algorithm that finds *k-rooted* spanning trees with disjoint paths from every node in the tree towards the root. The proposal is optimal for *k=2* but does not provide shortest paths, which are key to network efficiency. There are other distributed proposals based on shortest-path algorithms to build the disjoint paths, or that directly provide the shortest path in the solution, such as [7], [8], [9]. The first one only provides two minimum-cost disjoint paths, while the second provides multiple minimum-cost node-disjoint paths (but not link-disjoint) and improves previous results. Finally, the third obtains two node-disjoint paths using the Breadth First Search (BFS) algorithm. Tanaka [10] takes advantage of a previously known shortest path (obtained by another protocol or application) to discover a second, minimum-latency disjoint path (for backup) by using network exploration. Moreover, its performance is only limited by the network latency itself. A similar exploration principle is followed by Iterative Multipath Proposal (IMP) [11] to discover multiple disjoint low-latency paths between a source-destination node pair. The procedure performs several exploration rounds by iteration and discovers a new disjoint path in each step.

Finally, there are other proposals that simply obtain multiple paths without the requisite of disjointness, e.g. [12], [13] (in the group of protocols requiring the topology knowledge), or [14], [15] (in the group of topology independent protocols). An extensive study of the related work can be found in [16].

The main novelty of 1S-MDP is the distributed discovery of multiple low-latency (link- or node-) disjoint paths among any source node and the rest of the network, which does not require any previous topology knowledge. Additionally, and differently from other distributed solutions, this discovery is performed in a single iteration of the protocol, independently of the number of paths and number of destinations required in the calculation, which drastically reduces convergence time in comparison to any other solution. To the best of our knowledge, no previous work merges all these features.

III. One-Shot Multiple Disjoint Path Protocol

A. Protocol description

1S-MDP is a distributed protocol designed to discover multiple disjoint paths among a given node and the remaining nodes in the network. 1S-MDP is based on network exploration techniques, but instead of starting a new exploration for each path, it requires a single exploration for all. The protocol works in two phases: a network exploration phase, issued from the source node, that conveys the topological information to all the network nodes (see section III-B), followed by a path selection phase (see section III-C), started from the destination nodes, that is able to select multiple link- or node-disjoint paths for a set of destination nodes concurrently. Since finding multiple paths for any network node might not be necessary, only the nodes previously marked as ”edge” would participate of the path selection phase as destination nodes (note that all nodes could be marked as edge if needed).

B. First phase: Network exploration

This phase is triggered by an external or internal signal/event (control message, timer, ...) at any edge node. At this stage, the node becomes a source node that searches for multiple disjoint paths towards the remaining edge nodes by flooding an exploration packet to all its neighbors. This exploration packet travels across the whole network and conveys topological information to all traversed nodes. Figure 1 summarizes the whole exploration process, in a wired network composed of 7 nodes, where only 3 of them are considered edge nodes: *S*, *D1* and *D2*; for simplicity, the remaining nodes are core nodes. This figure could represent a traditional full-duplex wired layer-two network, e.g. nodes would be Ethernet switches. In those scenarios, finding multiple disjoint paths is particularly relevant for nodes located at the network edge, e.g. nodes serving end systems, interconnecting nodes, or nodes selected by the network administrator. The aim of 1S-MDP is to search multiple disjoint paths between the source node *S* and the other (destination) edge nodes, that is, multiple disjoint paths between *S-D1* and *S-D2*. The network exploration phase continues as follows: First, *S* initiates the process by flooding an exploration packet towards its neighbors. The exploration packet arrives at these neighbors, which associate the input port where the packet arrives with the source node Id (*SrcId*). This information is stored and ordered according to the packet arrival time, being the highest priority entries the first ones arriving, as depicted in Fig. 1 (this priority will be required in the second phase to select the paths). To continue the exploration process and avoid loops, each node receiving an exploration packet only floods the first packet copy towards all its neighbors, while late copies are discarded to prevent packet storms. By the end of the process each node should have received a copy of the exploration packet at each port.

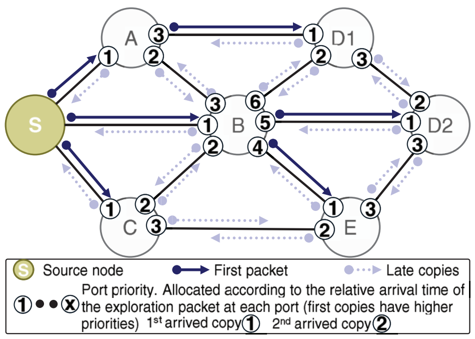
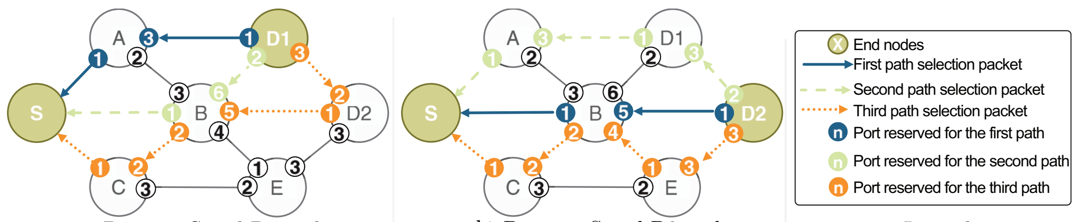

S
A
B
C
D1
D2
E
Source node
First packet
Late copies
Port priority. Allocated according to the relative arrival time of the exploration packet at each port (first copies have higher priorities) 1st arrived copy ① 2nd arrived copy ②

Fig. 1. Exploration process in 1S-MDP

C. Second phase: Concurrent path selection

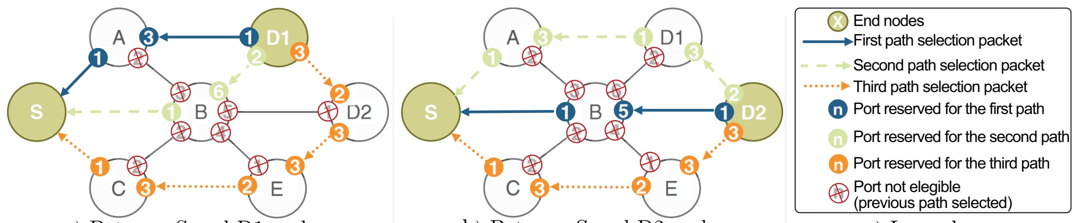
This phase provides a concurrent path selection mechanism from several edge nodes based on the information collected during the exploration phase that reduces the multiple disjoint path search time by avoiding waiting times and iterative processes. To simplify the description, this section explains the process from one destination edge node, nevertheless, the process happens concurrently at every destination edge node.

When an edge node receives an exploration packet from a source node, it becomes a destination node and initiates the path selection process to select the multiple disjoint paths towards the source node. First of all, this destination node generates a path selection packet with source and destination node IDs (*SrcId* and *DstId*) together with a disjoint path identification (*pathId*). This *pathId* is unique for each disjoint path between a pair of source-destination nodes. Afterward, the destination node sends this packet back to the source node through the port associated to the arrival of the exploration packet. This process is repeated for each copy of the exploration packet received at the destination node to guarantee that the procedure discovers all possible disjoint paths between the source and the destination. Upon reception of a path selection packet, intermediate nodes select the higher priority available port for that path. Then, they mark the port as unavailable for the *SrcId-DstId* pair, to guarantee that new disjoint paths for that pair are unique and do not overlap. And finally, the path


End nodes
First path selection packet
Second path selection packet
Third path selection packet
Port reserved for the first path
Port reserved for the second path
Port reserved for the third path
S
A
B
C
D1
D2
E

a) Between S and D1 nodes b) Between S and D2 nodes c) Legend

Fig. 2. Path selection process in 1S-MDP (link mode)


End nodes
First path selection packet
Second path selection packet
Third path selection packet
Port reserved for the first path
Port reserved for the second path
Port reserved for the third path
Port not elegible
(previous path selected)
S
A
B
C
D1
D2
E

a) Between S and D1 nodes b) Between S and D2 nodes c) Legend

Fig. 3. Path selection process in 1S-MDP (node mode)

selection packet is forwarded via the selected port to continue the path selection phase. If an intermediate node finds a dead-end (there is no available output port to forward the packet), the path selection packet hops back to the previous node in the path to seek an alternative. The path is completed when the path selection packet reaches the source node. Disjoint paths discovered for a given pair of nodes are ordered by latency due to the port prioritisation set by the exploration phase.

1S-MDP is capable of creating link-disjoint or node-disjoint paths. An example of path selection using link-disjoint mode is shown in Fig. 2, which is split into Figs. 2a and 2b, although the path selection phase from *D1* and *D2* happens simultaneously. In Fig. 2a the edge node *D1* starts the path selection phase towards node *S* when it receives the exploration packets from such node. The first exploration packet arrives at the port labeled as 1, and the path selection packet response is sent through this port, traversing *A* and setting up the first disjoint path. The same procedure happens with the second and the third path, whose selection phase starts from ports labeled as 2 and 3, respectively. As previously mentioned, intermediate nodes choose the highest priority output port adopting the link-disjoint path restrictions; e.g. *B* selects port 1 for the second path, and port 2 for the third path because port 1 is used by the second path. Figure 2b shows similar steps for node *D2*. The node-disjoint mode follows a similar –but simpler– philosophy because, to satisfy the node-disjoint restriction, each node must only have one active path. Thus, in the disjoint-node mode, each intermediate node only selects the first available port, and later requests are discarded. Figure 3 shows the node-disjoint mode under the same constraints as the link-disjoint mode.

IV. Evaluation

This section aims to evaluate and verify the behavior of 1S-MDP in terms of the number of disjoint paths discovered and the time needed to obtain them (convergence time). We leverage the *ns-3 network simulator* [17] with the *Ofswitch13* library to deploy 1S-MDP because it provides a real software switch model. For comparison, we choose from the related work a multiple disjoint path centralized solution and a distributed one, and launch the same set of experiments in all platforms. Within the centralized approach, we select the primitive (naive) implementation of the Dijkstra algorithm as a well-proven solution, whereas IMP [11] was selected as the distributed solution because it is the only one that finds multiple (link- and node-) disjoint paths. To obtain the disjoint paths between a pair of nodes in a centralized way, we run the Dijkstra algorithm [2] on a weighted graph. After the run, we remove the obtained path to ensure the disjointness and repeat until no more paths are discovered. The process is repeated for each pair of nodes in the topology. Moreover, we launch six concurrent instances in an Intel(R) Core(TM) i7-8700K CPU with 32 GB RAM to reduce Dijkstra's convergence time.

The simulation scenario uses two well-connected types of topologies to stress the number of existing disjoint paths: an extended version (3 stages, 10 nodes) of the topology shown in Section III, and several 2-dimension (2D) square mesh networks ranging from 4 to 36 nodes (2x2 to 6x6). Each test is repeated 10 times to compute 95% confidence intervals. Furthermore, we randomly set the propagation time per link in each run with an upper bound of 1.5 ms (the queuing time of 125 packets of 1500 bytes), to emulate different network load conditions. Besides, to perform a fair comparison between the different approaches, these propagation times are used as the link cost for the Dijkstra algorithm. Finally, the link speed is set at 1Gbps and all the nodes are marked as edge nodes to obtain the disjoint paths between all pairs of nodes.

TABLE I
Extended example topology results

			Dijkstra	IMP	1S-MDP
Node-disjoint	Convergence time (s)	Mean	0.2299	13.97	**0.0058**
		95% c.i	0.0875	1.27	0.0009
	Number of paths discovered	Mean	**232.60**	**232.60**	230.20
		95% c.i	9.69	9.69	7.51
Link-disjoint	Convergence time (s)	Mean	0.1087	18.52	**0.0105**
		95% c.i	0.040	0.0022	0.0018
	Number of paths discovered	Mean	**281.8**	**281.8**	266.40
		95% c.i	1.32	1.32	7.52

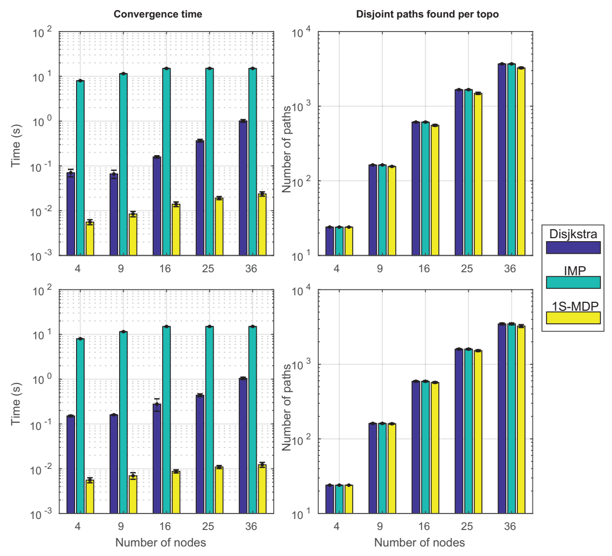


Fig. 4. Convergence time and number of paths discovered: link-disjoint mode (top), node-disjoint mode (bottom)

A. Results

This section shows the comparison of 1S-MDP, against the Dijkstra algorithm and IMP in terms of convergence time and number of disjoint paths discovered. Firstly, the proposals are evaluated in the example extended topology. The results are summarized in Table I and show the average values of 10 runs and the corresponding 95% confidence interval (c.i). It can be seen that 1S-MDP is, by far, the fastest protocol (lower convergence time), whereas Dijkstra and IMP are the proposals that discover more paths. Nevertheles, 1S-MDP discovers 99.0 % of the number of paths discovered by Dijkstra and IMP in node-disjoint mode and 94.5 % in link-disjoint mode. This decrease is a side effect of the single exploration per node in 1S-MDP (both Dijkstra and IMP perform a new computation/exploration for each path), but it is negligible in link-disjoint mode. While it is clear that Dijkstra and IMP obtain more disjoint paths, the advantage of 1S-MDP is that it reduces the convergence time of the Dijkstra algorithm by one or two magnitude orders (and more for IMP), which over-compensates the disjoint paths not discovered. Furthermore, in the centralized solution results, we have omitted the time spent to discover and build the underlying topology, which would further increase the advantage of 1S-MDP in terms of convergence time. Fig. 4 shows the results obtained in square mesh topologies of increasing sizes (from 4 to 36 nodes). The left side displays the convergence time, and the right side the number of disjoint paths discovered in each topology. As in the example topology, the number of discovered paths by 1S-MDP is slightly lower and decreases with the size of the topology but is still above 88% for 36 nodes in link-disjoint mode and above 93% in node-disjoint mode. However, it drastically reduces the convergence time by two to three magnitude orders.

V. Conclusion and future work

In this paper we have presented 1S-MDP, a distributed protocol that looks for multiple disjoint paths among multiple nodes with a single network exploration and concurrent path selection. The exploration applied guarantees low-latency disjoint paths ordered by increasing latency, while the concurrent path selection drastically reduces the convergence time.

1S-MDP has been tested in a simulation environment using a real software switch model to bring it over real scenarios, and has been compared with existing centralized and distributed solutions. The results obtained by 1S-MDP are promising, as it decreases by two to three magnitude orders the convergence time while it keeps the number of disjoint paths discovered close to its competitors (whether centralized and distributed approaches), independently of the underlying topology.

As future work, we will evaluate the algorithmic complexity of 1S-MDP to delve into its behavior and performance. Furthermore, we will study the quality of its disjoint paths in terms of load balancing or failure resilience.

References

[1] J. W. Suurballe and R. Tarjan, *A Quick Method for Finding Shortest Pairs of Disjoint Paths*. Wiley Online Library, jun 1984, vol. 14.
[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[3] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks," *Proceedings of INFOCOM '94 Conference on Computer Communications*, no. 908, pp. 1498–1508, 1994.
[4] A. Prakash, R. Kannan, and G. BAFNA, "Disjoint path computation systems and methods in optical networks," Jun. 19 2018, uS Patent App. 10/003,867.
[5] B. Martín *et al.*, "Solving the edge-disjoint paths problem using a two-stage method," *International Transactions in Operational Research*, vol. 27, no. 1, pp. 435–457, 2020.
[6] A. Itai and M. Rodeh, "The Multi-Tree Approach To Reliability In Distributed Networks," *25th Annual Symposium onFoundations of Computer Science, 1984.*, pp. 137–147, 1984.
[7] R. Ogier and N. Shacham, "A distributed algorithm for finding shortest pairs of disjoint paths," in *IEEE INFOCOM '89, Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, 1989, pp. 173–182 vol.1.
[8] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," *Conference on Communications architecture & protocols (SIGCOMM '91)*, pp. 43–51, 1991.
[9] R. Hadid, M. H. Karaata, and V. Villain, "A Stabilizing Algorithm for Finding Two Node-Disjoint Paths in Arbitrary Networks," *International Journal of Foundations of Computer Science*, vol. 28, no. 4, pp. 411–435, 2017.
[10] K. Jun Tanaka, "Path Generating method, relay device, and computer product," 2014, uS Patent App. 8/665,754 B2.
[11] D. Lopez-Pajares *et al.*, "Iterative discovery of multiple disjoint paths in switched networks with multicast frames," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, Oct 2018, pp. 409–412.
[12] W. Jiawei, Q. Xiuquan, and N. Guoshun, "Dynamic and adaptive multipath routing algorithm based on software-defined network," *International Journal of Distributed Sensor Networks*, vol. 14, no. 10, p. 155014771880568, oct 2018.
[13] M. F. Ramdhani, S. N. Hertiana, and B. Dirgantara, "Multipath routing with load balancing and admission control in Software-Defined Networking (SDN)," in *2016 4th International Conference on Information and Communication Technology (ICoICT)*. IEEE, may 2016, pp. 1–6.
[14] H. Geng *et al.*, "A hop-by-hop dynamic distributed multipath routing mechanism for link state network," *Computer Communications*, vol. 116, pp. 225 – 239, 2018.
[15] C. Minkenberg and M. R. Gusat, "Multipath discovery in switched ethernet networks," jan 2012, uS Patent App. 8/107,482 B2.
[16] J. Domżał *et al.*, "A survey on methods to provide multipath transmission in wired packet networks," *Computer Networks*, vol. 77, pp. 18–41, 2015.
[17] "ns-3 simulator," https://www.nsnam.org/, Accessed January 2020.