Path Computation in Multi-switching Multi-layer Networks

Zihe Yi and Lin Chen
School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
yizh8@mail2.sysu.edu.cn, chenlin69@mail.sysu.edu.cn

***Abstract*—In this paper, we study the problem of path computation in multi-layer multi-switching networks. Compared to the standard shortest path problem, path computation in this context needs to take into account the heterogeneous switching capabilities of nodes. We develop a routing algorithm by adapting the Floyd-Warshall algorithm to take into account the switching technology conversion. Our algorithm solves the all-pair min-cost continuous path problem by building routing tables for each node to allows hop-to-hop routing. We then extend our efforts by developing a distributed routing algorithm to construct the routing tables based on local information and interactions with direct neighbors. We complete our algorithmic analysis with extensive simulations to demonstrate the effectiveness of the developed routing algorithms.**

***Index Terms*—Path computation, multi-switching multi-layer networks, distributed algorithms.**

I. Introduction

Pushed by the ever-increasing traffic demand, today's carrier networks are constantly integrating emerging network components running over heterogeneous switching technologies. For example, Generalized Multiprotocol Label Switching (GMPLS) [1] has specified 5 switching capabilities ranging from Packet Switching Capable (PSC), Layer 2 Switching Capable (L2SC), TDM Capable (TDM), to Lambda Switching Capable (LSC) and Fiber Switching Capable (FSC). These heterogeneous switching technologies span over multiple layers, forming a multi-layer network, where the term layer is defined in GMPLS as a bandwidth granularity level within a switching technology [2].

The multi-switching multi-layer architecture brings non-trivial challenges to path computation, as we need to ensure smooth transition among different switching technologies along paths. Figure 1 illustrates an example of multi-layer multi-switching network. Node s and t can only support only one switching technology each, TDM and L2SC, respectively. Node u can support both TDM and L2SC, but cannot adapt between them. Node v is a hybrid node that not only support both TDM and L2SC but also adapt between them. In this example, the only feasible path between s and t is $s-u-v-u-t$ that relies on v to converse TDM to L2SC.

The above example demonstrates two fundamental technical challenges hinging behind the path computation in multi-layer networks. First, a topologically connected path may not be feasible, e.g., the path $s-u-t$ is not feasible in the above example, because u cannot transform TDM into L2SC. Second, a shortest feasible path in multi-layer networks may contain cycles, e.g., the path $s-u-v-u-t$ is an optimal feasible path but contains a cycle. These challenges make classical routing algorithms fail to find the optimal feasible path in the multi-layer context.

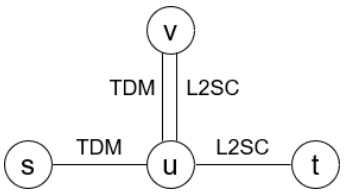


Fig. 1: A multi-layer multi-switching network.

Motivated by the above technical challenges, in this paper we embark to solve the path computation problem in multi-layer networks. Specifically, we design both centralized and distributed routing algorithms that can compute the min-cost paths between each pair of nodes by taking into account the nodes' switching capability constraints. Technically, we formulate the path computation problem in mutli-layer networks as the min-cost continuous path problem, where we formally define a continuous path as a feasible path along which a packet can successfully travel from the source to the destination with the nodes in the path taking appropriate switching actions subject to their switching capacity constraints. We then develop a routing algorithm by adapting the Floyd-Warshall algorithm to take into account the switching technology conversion. Our algorithm solves the all-pair min-cost continuous path problem by building routing tables for each node to allows hop-to-hop routing. We further extend our efforts by developing a distributed routing algorithm to construct the routing tables based on local information and interactions with direct neighbors. We complete our algorithmic analysis with extensive simulations to demonstrate the effectiveness of the developed routing algorithms.

Roadmap. The remainder of the paper is organized as follows. Section II presents the network model and problem formulation. In Section III we develop a centralized routing algorithm. In section IV we further develop a distributed routing algorithm. Section V presents our simulation results. In Section VI we provide an overview of related work on this topic. Section VII concludes the paper.

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

II. Network Model and Problem Formulation

We consider a multi-layer network whose topology is modelled by a directed graph $G \triangleq (V, E)$, with V and E denoting the sets of vertices and edges, representing the sets of switches and communication links, respectively. Each node v has a set of interfaces, each connected to an edge. Each interface can support a set of switching technologies. For our modelling convenience, if a pair of nodes (u, v) are connected by several links each supporting a specific switching technology, we model them as a single logical link supporting the union of the technologies. Therefore, G can be modelled logically as a simple graph.

In the above multi-switching multi-layer networks, to forward a packet, the corresponding node needs to decide either to forward the packet with the original switching technology or switch to another switching technology subject to its switching capability constraint. We refer the former operation as simple forward or more concisely forward and the latter operation as switching technology conversion or more concisely conversion. The switching capacity of each node v is characterized by a matrix M_v such that if $M_v(a, b) = 1$ then v can convert technology a to technology b. For technical convenience, we define a void technology denoted by ϵ and set $M_v(\epsilon, \lambda) = 1$ for each technology λ under which v can send packets. This operation can model the action that v sends packets as the source node. Symmetrically, we set $M_v(\lambda, \epsilon) = 1$ for each technology λ under which v can receive packets. This operation can model the action that v receives packets as the destination node.

In multi-layer networks, the standard topological connectivity is no longer sufficient as it does not take into account the heterogeneous switching capabilities among nodes. Therefore, we define continuous paths to capture the particularity in our problem.

Definition 1 (Continuous Path)**.** *A continuous path is a sequence of nodes and switching technologies* $U \triangleq \{u_1\lambda_1 u_2\lambda_2 \cdots u_{|U|-1}\lambda_{|U|-1}u_{|U|}\}$ *satisfying the following properties*

- U *is a walk in* G*;*
- λ_i *is supported by the communication link* $\overrightarrow{u_i u_{i+1}}$ *in the sense that* u_i *can send packets over the link using technology* λ_i *and* u_{i+1} *can receive over the same link using* λ_i*;*
- $M_{u_{i+1}}(\lambda_i, \lambda_{i+1}) = 1$.

A non-continuous path is called a broken path.

In Definition 1, λ_i denotes the technology used by the packet to traverse the edge $\overrightarrow{u_i, u_{i+1}}$. The last condition states that the mapping from each pair of successive switching technologies λ_i to λ_{i+1} should follow the switching capacity of the corresponding node u_{i+1}.

Let w_e denote the cost of edge e. We seek to find a min-cost continuous path between each pair of nodes, which maps to the canonical all-pair shortest path problem, but we need to take into account the switching technology conversion in our context. We term our problem as the **min-cost continuous path problem**. In the following sections, we develop routing algorithms to find the min-cost continuous paths by building routing tables for each node to allow hop-to-hop routing.

III. Centralized Routing Algorithm Design

In this section, we develop a centralized routing algorithm. Technically, our design rationale is based on the structural property that two continuous paths, one from s to v, the other from v to t, can be concatenated to form a continuous path from s to t if they satisfy certain condition. We can then adapt the Floyd-Warshall algorithm to iteratively compute the min-cost continuous path between each pair of nodes.

Given a pair of continuous paths, P_1 from s to v, P_2 from v to t. Let λ_1 denote the terminating technology of P_1, i.e., the switching technology of the edge incident to v in P_1 is λ_1. Similarly, let λ_2 denote the starting switching technology of P_2. We can concatenate P_1 and P_2 to form a continuous path P if and only if $M_v(\lambda_1, \lambda_2) = 1$. Specifically, we can further distinguish two cases.

- Case 1: $\lambda_1 = \lambda_2$. In this case, in order to form P, it suffices for v to perform forward operation;
- Case 2: $\lambda_1 \neq \lambda_2$. In this case, in order to form P, v needs to perform conversion from λ_1 to λ_2.

In both cases, we denote the concatenation by $P = P_1 + P_2$.

We next describe our path computation algorithm. Each entry of the routing table constructed by our algorithm is a quintuple, denoted by $(\lambda, t, w, \lambda', r)$, where λ denotes the switching technology of the inbound edge, t denotes the destination node, w denotes the cost to reach t, λ' denotes the switching technology of the outbound edge, r denotes the next hop. In other words, if the node receives a packet operating on the switching technology λ destinated to t, it should send to r using the switching technology λ'. The local operation to perform is forward, if $\lambda = \lambda'$, or conversion, if $\lambda \neq \lambda'$. The routing table is indexed by the couple (λ, t).

Algorithm 1 gives the pseudo-code of constructing the routing table. Our algorithm starts with the initialization phase where the entries corresponding to the neighbors are added to the routing table of each node. Our algorithm then enters the construction phase to gradually build the routing tables by adapting the Floyd-Warshall algorithm. More specifically, the two entries in the algorithm, i.e., $(\lambda_1, t_1, w_1, \lambda_1', r_1)$ and $(\lambda_2, t_2, w_2, \lambda_2', r_2)$ in the routing tables of v and t_1, respectively, maps to a pair of continuous paths from v to t_1 and t_1 to t_2. We concatenate them to form another continuous path from v to t_2 if there is no entry corresponding to the path in the routing table of v or there is an entry but with higher cost. In both cases, we update the routing table of v. The whole construction process terminates once there is no further modification of the routing tables.

Algorithm 1 Centralized routing algorithm

1: **Input**: G, $\{M_v\}_{v \in V}$
2: **Output**: routing tables for each node u, denoted by R_u

Initialization
$R_v \leftarrow \emptyset$ for each $v \in V$
for each $e \triangleq \overrightarrow{uv} \in E$ **do**
  for each λ_1, λ_2 such that $M_u(\lambda_1, \lambda_2) = 1$ **and** $\lambda_2 \in \Gamma_e$ **do**
    add $(\lambda_1, v, w_e, \lambda_2, v)$ to R_u
  end for
end for
Construction
repeat
  for each $v \in V$ **do**
    for each entry $(\lambda_1, t_1, w_1, \lambda_1', r_1) \in R_v$ **do**
      for each entry $(\lambda_2, t_2, w_2, \lambda_2', r_2) \in R_{t_1}$ **do**
        if $M_{t_1}(\lambda_1', \lambda_2) = 1$ **then**
          if there is no entry in R_{t_1} indexed by (λ_1, t_2) **then**
            add $(\lambda_1, t_2, w_1 + w_2, \lambda_1', r_1)$ in R_v
          end if
          if there is an entry in R_{t_1} indexed by (λ_1, t_2) but with higher cost **then**
            replace the entry by $(\lambda_1, t_2, w_1 + w_2, \lambda_1', r_1)$
          end if
        end if
      end for
    end for
  end for
until no routing table is modified

The complexity of our algorithm is $O(|V|^3 L)$ with L being the number of hops of the min-cost continuous path between any pair of nodes in the network. In the following lemma, we give the upper-bound of L.

Lemma 1. *Let a denote the maximum number of switching technologies supported by a node. It holds that L is upper-bounded by $a^2|V|^2$.*

Proof. Let P denote a min-cost continuous path. We write P as a sequence $u_1, \lambda_1, u_2, \lambda_2, \cdots, u_{|P|-1}, \lambda_{|P|-1}, u_{|P|}$, where λ_i denote the switching technology employed between u_i and u_{i+1}. We can show that there does not exist i and j such that the following equations hold simultaneously.

$$u_i = u_j,\ u_{i+1} = u_{j+1},\ \lambda_i = \lambda_j,\ \lambda_{i+1} = \lambda_{j+1}.$$

The proof is rather obvious because if the above equations hold, we can construct another continuous path with lower cost by by-passing the sub-path between u_{i+1} and u_j, thus contradicting the optimality of P. Therefore, it then follows that P cannot traverse more than $a^2|V|^2$ nodes. The lemma is thus proved. □

It follows from Lemma 1 that the worst-case complexity of our algorithm is $O(a^2|V|^5)$. However, in practice, the number of hops of the min-cost continuous path is much smaller than the worst-case upper-bound. The practical complexity we observe oscillates between $O(|V|^3)$ and $O(|V|^4)$.

IV. Distributed Routing Algorithm Design

In the cases where the topology information is not available at each node or there does not exist a central controller to compute the routes for all the nodes in the network, we need to design distributed routing algorithms that can build the routing table of each node locally by exchanging information only with its neighbors. In this section, we design a distributed routing algorithm. Technically, we extend our idea of forming a longer continuous path by concatenating two continuous paths P_1 and P_2. However, to adapt to the distributed environment, we restrict to the case where P_1 is a single edge, say $P_1 = (u, v)$, so that the node u can obtain the information on P_2 from its neighbor v to perform the concatenation operation. The pseudo-code of our distributed routing algorithm is depicted below. We assume that the control channel to exchange routing tables between neighbors are bi-directional.

Algorithm 2 Distributed routing algorithm: routing table initialization at node u

$R_u \leftarrow \emptyset$
for each λ such that $M_u(\lambda, \epsilon) = 1$ **do**
  add $(\lambda, u, 0, \epsilon, u)$ to R_u
end for
send R_u to all the neighbors of u

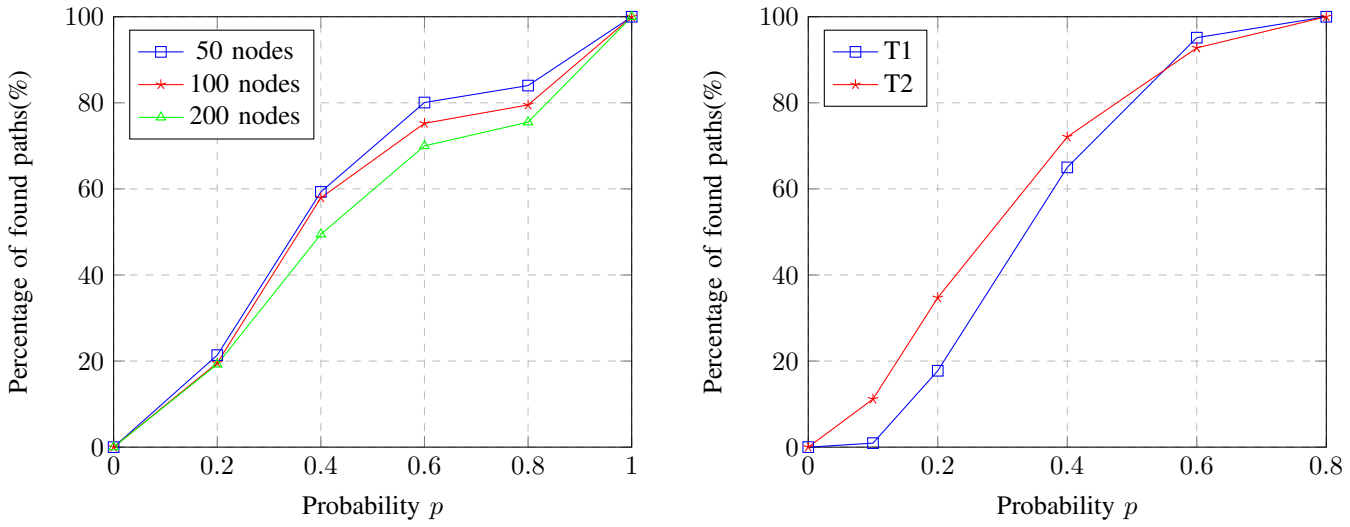
Algorithm 3 Distributed routing algorithm: updating routing table, invoked at node u upon receiving the routing table R_v from a neighbor v

for each entry $(\lambda, t, w, \lambda', r) \in R_v$ **do**
  for each λ_u such that $M_u(\lambda_u, \lambda) = 1$ **do**
    if there is no entry in R_u indexed by (λ_u, t) **then**
      add $(\lambda_u, t, w + w_{(u,v)}, \lambda', v)$ in R_u
    end if
    if there is an entry in R_u indexed by (λ_u, t) but with higher cost than $w + w_{(u,v)}$ **then**
      replace the entry by $(\lambda_u, t, w + w_{(u,v)}, \lambda', v)$
    end if
  end for
end for
send R_u to all the neighbors of u

By performing similar analysis as in the previous section, we can show that, if time is divided in rounds and each node update its routing table and sends out the updated table to its neighbors once per round, all the routing tables converge to the optimal routes after $O(a^2|V|^2)$ rounds.

V. Simulation Results

In this section, we conduct simulations to evaluate the efficiency of our algorithms. We generate scale-free topologies using the Barabàsi-Albert model, where each new node connects to 3 existing nodes, forming edges in pairs. Besides the synthetic random network topologies, we also perform simulations on two real-word topologies described in [3]. The first one is sampled from the EBONE network of Europe,



(a) Synthetic topologies

(b) Real-word topologies

Fig. 2: Percentage of found paths in synthetic and real-word topologies under different p.

called T1, which has 89 nodes and 246 links. The second, T2, corresponds to the Abovenet network in the US, with 54 nodes and 417 links.

All of our networks incorporate two types of protocols, with each node u maintaining a two-dimensional matrix M_u. Within this matrix, each element represents a switching capacity, which can be either forward or conversion and is available with probability p. We implemented centralized and distributed routing algorithms using Python 3.9 and compared their performance. In the distributed algorithm, each node is represented as a thread, and each edge, denoted as $\overrightarrow{uv}$, is managed as a queue, where u is responsible for writing and v for reading. Notably, in our networks, feasible paths between nodes do not always exist due to the switching capability constraint. Therefore, our analysis mainly focuses on tracking running time and the probability of feasible path existence. All results in the subsequent figures are averaged over 100 simulation runs.

Figure 2 shows the percentage of found paths as a function of the probability p. Obviously, we can see that increasing probability p helps find more feasible paths both in synthetic and real-word topologies. For instance, for 100 nodes of synthetic topologies, only roughly 20% pairs of nodes are linked by feasible paths when $p = 0.2$, while the percentage increases to approximately 75% for $p = 0.6$. Comparing Figure 2a and Figure 2b, it seems that the real-word topologies are more influenced by the probability p. On both T1 and T2, the percentage of found paths reaches 80% when the probability is around 0.5, and is nearly 100% when $p = 0.6$.

Figure 3 illustrates the impact of probability p on the running time when using the centralized algorithm. The result indicates that the running time grows as p increases. Also, we find that the more nodes in the network, the more running time is affected by p. For example, for $p = 0.4$, our algorithm completes its task in 1.58s when there are 50 nodes, whereas it requires 12.1s for 100 nodes. Thus, our observation is that the performance of the centralized algorithm may pose a computation bottleneck for large networks.

Figure 4 shows the convergence time of the distributed algorithm under different p. From the result, we observe that when p is less than 0.2, the convergence time increases with p. However, as p surpasses 0.4, it gradually tends to a stable value. For example, in the interval $p \in [0.4, 0.8]$, the convergence time hovers around 50s for 100 nodes and approximately 58s for 200 nodes. We also observe that the convergence time of T2 is higher than T1, even though T2 has fewer nodes.The result suggests that the convergence time may be intricately more related to the number of network links.

VI. Related Work

The problem of path computation in multi-layer switched networks was initiated by Jannari *et al.* [4]. They classified the constraints related to path computation into prunable and non-prunable constraints. The switching capability constraint belongs to the latter. To compute the end-to-end paths, the authors proposed a channel graph to model this constraint. A variant of the classical Yen's algorithm [5] was applied to compute the paths. Gong and Jabbari later extended the work in [6] and developed a polynomial-time algorithm to compute the optimal end-to-end path between a given pair of source and destination nodes. A comprehensive performance evaluation of different path computation algorithms in multi-layer networks was further conducted in [7]. Compared to these solutions in the literature, our contributions are two-fold. First, our routing algorithms developed in this paper can solve the all-pair min-cost path problem and hence compute the min-cost path for each pair of nodes. Second, we also develop a distributed routing algorithm without the knowledge of the

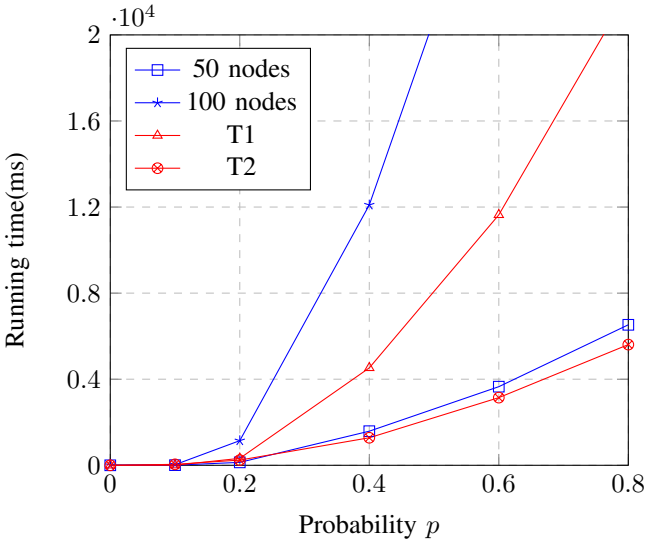


Fig. 3: Running time under different p.

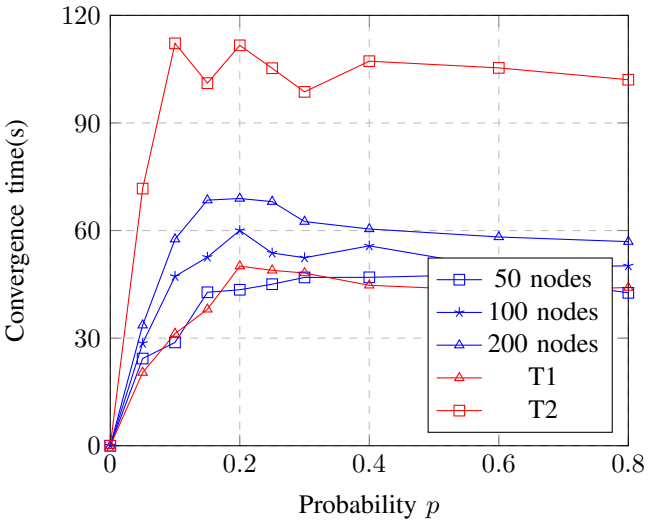


Fig. 4: Convergence time under different p.

entire topology. Moreover, our algorithms can support hop-to-hop routing.

Recently, there is another thrust of research focusing on path computation in multi-layer networks, where, besides the standard conversion from one switching technology to another, nested conversion by encapsulating one protocol or technology inside another is also considered, thus making the path computation problem even more challenging. We refer readers to [8]–[14] for representative works on this topic. The path computation problem in this context is not fully understood yet, leaving a number of open territories to be explored.

VII. Conclusion and Perspective

We have formulated the problem of min-cost continuous path computation in multi-layer networks in this paper. We have developed a routing algorithm by adapting the Floyd-Warshall algorithm to take into account the switching technology conversion. We have further developed a distributed routing algorithm to construct the routing tables based on local information and interactions with direct neighbors. In our future work, we plan to explore the more challenging case of nested adaptation and design efficient routing algorithms in this context.

References

[1] E. Mannie, "Generalized multi-protocol label switching (GMPLS) architecture," RFC 3945, Oct 2004.

[2] I. Bryskin and A. Farrel, "Lexicography for the in- terpretation of generalized multiprotocol label switching (GMPLS) terminology within the context of the itu-t's automatically switched optical network (ASON) architecture," RFC 4397, Feb 2006.

[3] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 231–236.

[4] B. Jabbari, S. Gong, and E. Oki, "On constraints for path computation in multi-layer switched networks," *IEICE Transactions on Communications*, vol. E90B, 08 2007.

[5] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, 1971.

[6] S. Gong and B. Jabbari, "Optimal and efficient end-to-end path computation in multi-layer networks," in *Proc. ICC*, 2008, pp. 5767–5771.

[7] X. Yang, T. Lehman, K. Ogaki, and T. Otani, "A study on cross-layer multi-constraint path computation for ip-over-optical networks," in *Proc. ICC*, 2009, pp. 1–6.

[8] F. Dijkstra, B. Andree, K. Koymans, J. van der Ham, P. Grosso, and C. de Laat, "A multi-layer network model based on itu-t g. 805," *Computer Networks*, vol. 52, no. 10, pp. 1927–1937, 2008.

[9] F. Kuipers and F. Dijkstra, "Path selection in multi-layer networks," *Computer Communications*, vol. 32, no. 1, pp. 78–85, 2009.

[10] M. L. Lamali, H. Pouyllau, and D. Barth, "Path computation in multi-layer multi-domain networks," in *Proc. Networking*, 2012, pp. 421–433.

[11] ——, "Path computation in multi-layer multi-domain networks: A language theoretic approach," *Computer Communications*, vol. 36, no. 5, pp. 589–599, 2013.

[12] F. Iqbal, J. van der Ham, and F. Kuipers, "Technology-aware multi-domain multi-layer routing," *Computer Communications*, vol. 62, pp. 85–96, 2015.

[13] M. L. Lamali, N. Fergani, and J. Cohen, "Algorithmic and complexity aspects of path computation in multi-layer networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2787–2800, 2018.

[14] M. L. Lamali, S. Lassourreuille, S. Kunne, and J. Cohen, "A stack-vector routing protocol for automatic tunneling," in *Proc. INFOCOM*. IEEE, 2019, pp. 1675–1683.