Virtual Network Construction Technique, Treating All VPNs Simultaneously

Srecko Krile[1, *], Martin Medvecky[2]
[1]*University of Dubrovnik, Electrical Engineering and Computing Department, Cira Carica 4, 20000 Dubrovnik, Croatia*
[2]*Slovak University of Technology, Faculty of Electrical Engineering and Information Technology, Institute of Multimedia Information and Communication Technologies, Ilkovicova 3, 812 19 Bratislava, Slovakia*
srecko.krile@unidu.hr

***Abstract*—In the paper, a new construction technique for virtual network (e.g., Virtual Private Network (VPN)) based on flow permutation algorithm is proposed. In existing methods for creating virtual networks, whereby virtual networks are constructed one by one in time and the new virtual network can use only the remaining resources, it could be non-optimal. Our approach treats all traffic flows simultaneously and is capable of balancing the network much better than other existing techniques. As we show, the proposed new construction technique work well, even in the condition of hard loaded networks operating on the edge of capacity, i.e., in situations when traditional techniques could cause unbalanced network and significant congestion problems. For huge number of traffic flows, heuristic algorithm, whose complexity rises linearly, is evaluated.**

***Index Terms*—Computer networks; Network topology; Virtual private networks; Routing protocols; Algorithms.**

I. INTRODUCTION

This paper is concentrated on virtual networks (sub-networks) constructed over a physical network. It means that we use a part of physical network resources (BW-bandwidth) to enable service as Virtual Private Network (VPN) broadly offered from the service provider to the users. Link capacity between ending points (network nodes) has to be sufficient to enable services to the users, but the capacity utilization of the physical network has to be optimal.

In the condition of the hard loaded network working on the edge of capacity, the existing techniques based on pure Shortest Path First (SPF) algorithms are not sufficient. In general, this problem is common with traffic routing. As it is an offline calculation, we can perform much more complicated computing techniques than for online traffic routing. We can say that each virtual sub-network (e.g., VPN) consists of a number of traffic flows connecting pairs of endpoints (nodes) in the customer network as we can see from the example shown in Fig. 1. The demanding throughput (BW) can be equal in the whole sub-network, but it is not necessary. Normally, on some links, the throughput has to be higher to satisfy aggregated traffic consisted of many flows passing to the different locations. The most important fact for multiple virtual networks functioning together in the same physical network, is that new VPN cannot be constructed independently. Also, the Quality of Service (QoS) for data transmission over each virtual network (sub-network) has to be maintained carefully.

Manuscript received 18 March, 2019; accepted 30 January, 2020.

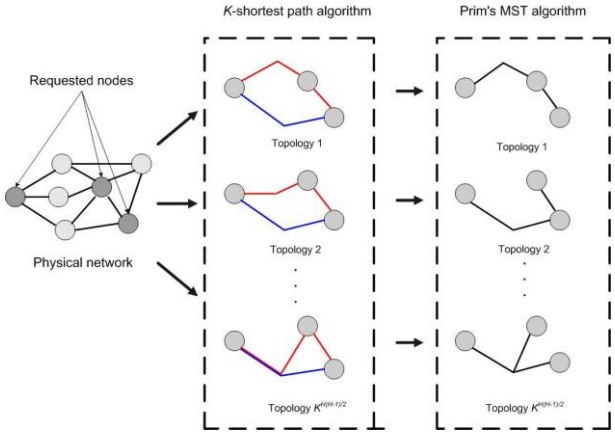


Fig. 1. An example how we can construct the new virtual network using *K*-shortest path algorithm and Prim's Minimum spanning tree algorithm. Source: authors.

Now, we are focused on the physical network with existing virtual networks to enable the construction of a new virtual network. Here, we talk about the resources (link capacity) that are unused as *remaining resources*. The set of nodes can be represented with $V = \{v_1; \ldots; v_i; \ldots; v_N\}$. A link between pair of nodes v_i and v_j is represented with e_{ij} and the set of all links in the network is represented with E. The remaining resources of each link e_{ij} is w^e_{ij} and the set of those resources is represented with W^E. In the process of the network virtualization, we can represent the physical network with $G = (V; E; W^E)$.

The user generates to a service provider a new traffic demand for the construction of the new virtual network (Fig. 2). The request consists of information related to the nodes $\{v^*_1; \ldots; v^*_H\}$ that should be the part of the new virtual network and the amount l of the network resources (link capacity). The service provider has to find out the optimal configuration of the new virtual network to satisfy users' demands.

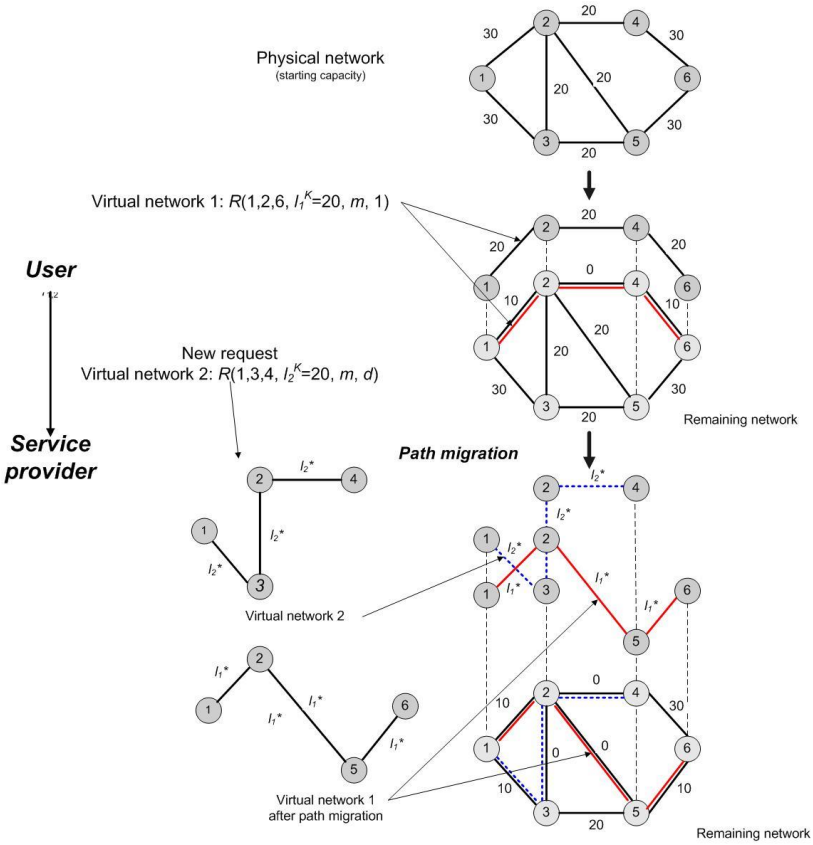


Fig. 2. An example of path migration as the crucial element of classical approach in VPN construction. Source: authors.

In Section II, we compare traffic routing technique with virtual network construction, as both problems are very similar. In Section III, we explain different methods used by other authors mostly treated as the classical approach with some improvements. In Section IV, we propose a new approach with an algorithm based on flow permutations. After that, in Section V, we validate it on some simple test-examples. In Section VI, we explain a heuristic algorithm appropriate for solving huge problems with many traffic flows. The calculation of heuristic is divided into stages. In Section VII, we talk about the network criticality/robustness as an important measure to decide, which traffic solution is better than another. Finally, in Section VIII, conclusions are presented.

II. Relation to the Traffic Routing Problem

Traffic routing problems in computing networks are widespread in virtual network construction. Both are solved mostly with algorithms based on the Shortest Path First (SPF) approach. In traffic routing, it has to be done with online algorithms that could be more demanding related to the speed of calculation. Similar techniques are used in distributed routing (e.g., Multi Protocol Label Switching – MPLS) same as in centralized routing, e.g., in Software Designed Networking (SDN). It means that virtual paths for each traffic flow are defined only once, at the beginning of service invocation, and stay unchanged until the service ends. Of course, link state algorithms, e.g., Open Shortest Path First (OSPF), Intermediate System to Intermediate System (ISIS), take care of network capacity dynamically, using the remaining capacity to construct (calculate) new traffic path optimally. It functions well only if capacity on the path is sufficient; that is the case in over-provisioned networks. Also, the path splitting of existing flows can help, but it is not allowed for specific services. Therefore, in the condition of hard loaded networks working on the edge of capacity, VPN construction could cause unbalanced network and significant congestion problems [15]. It means that we still have no sufficient routing technique for such hard traffic load [23]. In that case, new traffic demand could be rejected, no matter if existed traffic flows could be much better routed.

In the paper [2], the authors proposed another routing technique instead, one that takes care of all existing traffic flows at the same time. It is based on classical SPF algorithm, but in combination with permutations of all existing traffic flows crossing the network simultaneously. In that research, it is shown that such an approach can position traffic paths more efficiently and decrease congestion problems in a given network. However, such complicated calculation is time-consuming, so we still cannot perform such a technology online. Notably, it is hard to introduce it in the networks with distributed routing (e.g., MPLS), where many online algorithms work simultaneously in different edge routers. Nowadays, the SPF technique applied to the remaining network capacity is the only acceptable solution. Of course, the introduction of artificial intelligence based on traffic statistics can help a lot, but generally, we cannot optimize the whole traffic jet. In centralized oriented routing (e.g., Software Defined Networking (SDN)) with main controller (e.g., Python-based software defined networking (POX)), as we have in Open Flow networks, there is some

perspective to introduce such capable routing technique. In that way, we could find out the optimal utilization of network capacity for all existing traffic flows crossing the network simultaneously [21], [22].

The main goal of this research is to show that the application of such traffic routing algorithm can be a good solution for the offline VPN construction taking care of all existing traffic flows at the moment of introduction of the new one traffic.

III. PREVIOUS WORK

In the paper [1], we have a survey of methods we can use to design a new virtual network as an overview of some papers [3]–[12]. Also, in papers [1] and [13] the new solution of VPN sub-network construction is proposed. It is based on the *K*-shortest path algorithm and Prim's Minimum spanning tree algorithm with some modification consisting of adding techniques: the path splitting and the path migration. It is the improvement of classical approach by Kou-Markowsky-Berman (KMB) algorithm explained in the paper [20]. The problem of KMB algorithm is that the method does not consider the possibility of changes in the virtual network that has been constructed. It means that it is not possible to construct many virtual networks, one after another or simultaneously. Fig. 1. shows how multiple network topologies could be designed. For the first step, we have to check all paths capable of satisfying the traffic flow between requested nodes using Traffic Engineering (TE) technique (e.g., bandwidth, number of hops). Typically, the *K*-shortest path algorithm is used to find out alternative paths without loops. As we can see from Fig. 1, a number of possible paths does exist. The *K*-shortest path algorithm can be seen as a variation of Dijkstra's algorithm [24].

As shown in Fig. 1, for the requested nodes $\{v_1; ...; v_H\}$, $H = 3$, a number of graphs are constructed by using the K-shortest path algorithm. That algorithm is based on pure shortest path first (SPF) algorithm, e.g., Dijkstra algorithm. With such a method, it is possible to find out the shortest paths between the requested nodes and to construct the virtual network in the starting physical network. Then Prim's Minimum spanning tree algorithm is applied. For the number H of the requested nodes, we have $KH(H-1)/2$ possible paths and the number of different graph topologies is $KH(H-1)/2$.

The service provider has to check if one of the designated topologies is acceptable. It means that it has to satisfy the traffic demands connecting designated nodes with sufficient link capacity. It is mostly related to the robustness of the network [17], which is explained in chapter VII. As we can see from Fig. 1 and Fig. 2, all proposed topologies have to be without loops and multiple edges, so it means we have Minimum Spanning Tree (MST) topology. In the case of tree topology, we can measure how well a graph is connected [19]. It is mostly related to the robustness of the network [17], which is explained in Section VII. If all offered topologies cannot satisfy the traffic demands, the service provider has to perform the path splitting and path migration that are explained in chapters 3.3 and 3.4 of the paper [1]. As it is said in chapter 3.3, "The path splitting can decrease the loss probability of user's request by allowing another topology. On the other hand, the path splitting may trigger packet reordering in each virtual network because multiple routes can be used between the source node and the destination node. Therefore, users select whether the path splitting is allowed or not." In chapter 3.4, it is elaborated that the path migration of the existing VPNs can solve the problem efficiently, but it may decrease Quality of Service (QoS) for the particular virtual network, especially if such a technique of topology re-design, is time-consuming. In addition, the path migration technique is not defined clearly as we do not know what paths have to be changed. Therefore, that can be a very demanding task, especially for the networks running on the edge of capacity. An example of the path migration can be seen in Fig. 2. We can see that traffic flow of VPN1 has to migrate to enable sufficient remaining capacity on the link 2-4, so that VPN2 can be efficiently constructed.

As we said before, the papers [1] and [13] explained the improvement of well-known KMB algorithm. Such a technique is capable of constructing many virtual networks taking care of the robustness of the physical network [17], [18]). We have to notice that the introduction of the *K*-shortest path algorithm (pure SPF technique) ensures more virtual designed topologies, but it lacks in relation to the existing virtual sub-networks sharing the same physical network. That means that it suffers from inefficient capacity utilization. It could be the main reason that no acceptable topologies are found. As the final result of that research, it is concluded that the loss probability of such virtual network construction significantly decreases if K is set to 2 or 3. However, the processing time significantly rises. As we said before, the situation is much worse in over-provisioned networks. In the case the network capacity is close to the edge of capacity, that technique suffers very much. Then, the utilization of the path splitting and the path migration techniques starts to be the factor of process degradation. significantly increasing calculating time. Some extensions and applications of such an approach are explained in [14] and [16].

IV. A NEW APPROACH BASED ON SPF BASED ALGORITHM WITH TRAFFIC FLOW PERMUTATION

As we said before, the virtual network construction is very common with traffic routing problem. In the condition of hard traffic, working on the edge of capacity, introduction of a new traffic flow could cause unbalanced network and significant congestion problems. The existing routing methods based only on SPF algorithms cannot solve the routing problem generally. We need a much more capable technique to balance the physical network and utilize the resources efficiently. Well-known SPF-based routing issues are explained in paper [2]. Positioning of huge (elephant) traffic flows and relatively much smaller flows in time order (one after another) could cause the problem of optimal exploitation of the network. Network balancing can be done additionally by path migration, but such a technique is demanding and time-consuming, so we try to avoid it with a different approach.

We have similar problem in virtual network construction, looking for the remaining physical network (remaining capacities only), but it can be processed offline. The proposed algorithm is based on SPF algorithm, but in combination with permutations of all traffic flows crossing the network simultaneously. The network can be balanced much better and we can find out near-optimal solution, if it exists. Of course, it is possible only if remaining resources are sufficient, or in opposite, we can apply path splitting just to balance the traffic. After the splitting of the problematic flow, we have a similar approach, but routing has to be done with one traffic flow more. So, we can formulate our task: for simultaneous traffic flows (M) with a variety of ingress/egress node pairs in the network of N nodes, we have to examine each permutation of incoming flows. An acceptable solution has to satisfy all traffic demands. For the first step, we have to apply the shortest path technique for each flow entering the empty network G. As we said before, we can use Dijkstra algorithm, but for huge problems (a large number of traffic flows), Floyd's algorithm is more efficient [26]. The complexity is $O(N^3)$. For a configuration with a small number of links L (e.g., backbone or fat tree shape), the complexity is closer to $O(N^2)$.

The pseudocode for such algorithm can be represented as follows:

1. We have to calculate the shortest path for each traffic flow M in an empty network. After serving that traffic flow (eliminating the bandwidth on the path), we will get a new network G' with remaining capacity (for each flow we have another G').
2. After one flow is served, the algorithm starts with a similar procedure 1., to find out the shortest path for the next traffic flow (in network G').

This procedure ends when all traffic flows (traffic load) are satisfied. The branching tree of traffic flow permutations is growing very fast, respectively how many flows still we have to satisfy.

On the first step, we have M branches. After that, we have M-1, M-2, M-3, etc. For each branching level, we have to calculate a number of the shortest paths for a number of traffic flows. It is clear that we have a large number of SPF calculations in total. The final routing solution (permutation of traffic flows) is acceptable only if all traffic flows are accommodated successfully. We can say that the routing solution is in firm correlation with the order of flows entering the network.

The complexity of our algorithm (calculating flow permutations for each traffic flow as starting one) depends of the number of SPF calculations and of the complexity of Floyd's algorithm. Instead of $M!$ for total number of permutations, we have the significant reduction of the total number of SPF calculations when using Floyd's algorithm:

$$1+\sum_{k=1}^{k=M-1}\frac{M!}{(M-k)!}. \quad (1)$$

Generally, there are more available flow permutations capable to satisfy all traffic demands. Sometimes that number is huge, sometimes it is very small or a unique solution exists. The result is in firm correlation to the networks robustness and it can be calculated [17]. It is opposite to the network criticality (see Section VII). So, we can compare network criticality (robustness) to decide what available permutation is the best (the most perspective). In over-provisioned networks, where the network criticality is very low, the number of available permutations can be huge. However, for networks working on the edge of the capacity, there is a tiny number of available (acceptable) solutions. If no available solution exists, the only way is to apply path splitting and to try again. Therefore, in that case, we have one more flow M + 1, and the procedure starts again. Such a method is capable to find out the optimal construction of VPNs with no need for extra modifications, especially not by the technique of the path migration that is explained on example in Fig. 2. We can see similar result in Fig. 3.

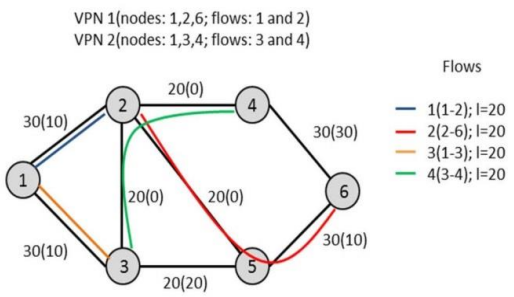


Fig. 3. An example from previous figure solved with algorithm based on traffic flow permutations. Source: authors.

Today, in practice, service providers usually calculate new VPN from the remaining network capacity, which could be non-optimal or the superficial approach can degrade services of the existing customers. In our approach, we propose a new offline algorithm that is capable of recalculating all existing VPNs (virtual sub-networks) in relation to new VPN introduction. It leads to efficient network exploitation eliminating points of congestion much better than with the technique proposed in [1] or [13].

V. VERIFICATION OF NEW APPROACH TO TEST-EXAMPLES

An example in Fig. 2 is taken from [1] and efficiently solved with a new approach. We just calculate the traffic routing path solution for all traffic flows that serve the demanding traffic (all VPNs). For virtual private network 1 (VPN1) defined with R (1, 2, 6; l_1^3 = 20, m = 1, d = 1), we have two flows connecting nodes 1, 2, and 6: flow1 (connecting nodes 1 and 2) and flow2 (connecting nodes 2 and 6) with demanding traffic between nodes l = 20. In a new traffic demand (VPN2) defined with R (1, 3, 4; l_2^3 = 20, m = 1, d = 1), we have two adding flows connecting nodes 1, 3, and 4: flow 3 (1-3) and flow 4 (3-4), also with demanding traffic l = 20.

So, we are calculating paths for all flows crossing the network simultaneously. As we can see from the final result on Fig. 3, the solution is the same as a result in Fig. 2. However, path migration is not necessary.

At first, the flow2 (red color: links 2-5 and 5-6) should be solved over links 2-4 and 4-6. But in that case, an acceptable solution for flow4 (connecting nodes 3 and 4) does not exist.

Instead, an acceptable solution is shown on Fig. 3 and Fig. 4. So, if we examine all possible flow permutations, the optimal solution can be discovered.

The remaining network is shown on the bottom of Fig. 2 and in Fig. 3 (values in brackets).

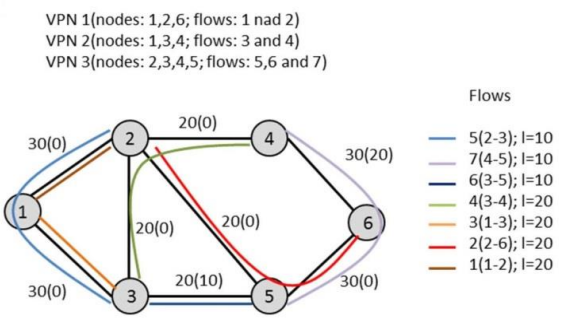


Fig. 4. Four VPNs with respective traffic flows. In this routing solution, all traffic demands between node pairs (7 traffic flows) are satisfied. Source: authors.

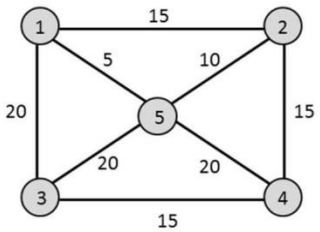
For the next test-example, we can introduce another virtual network - VPN3. Connecting nodes: 2, 3, 4, and 5. VPN3 can be defined with R (2, 3, 4, 5; l_3^4 = 10, m = 1, d = 1), where m = 1 means that path splitting is allowed and d = 1 means that path migration is allowed, too. If not, these values are set to zero.

Traffic is realized with flows 5, 6, and 7. Demanding traffic is the same between all neighbor nodes l = 10 forming the VPN3. That means that each node of VPN generates different traffic to all others (among couples), but the overview of traffic amounts is not discussed in this paper.

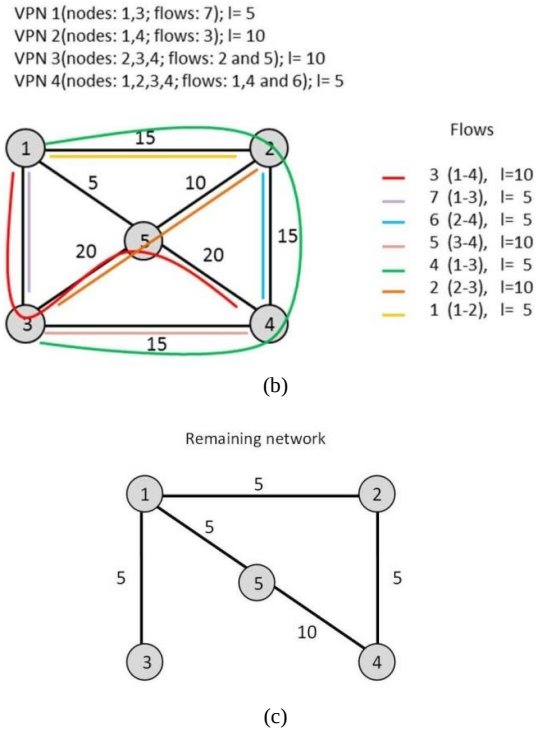
We suppose the shape of the sub-network serving the VPN3 can be different, but traffic demands should be satisfied with three traffic flows: flow5 (connecting nodes 2 and 3), flow6 (connecting nodes 3 and 5), and flow7 (connecting nodes 4 and 5). Now, we are calculating all 7 flows simultaneously (all together). An acceptable solution (flow permutation: 5, 7, 6, 4, 3, 2, 1) is shown in Fig. 4. In this case, path splitting is not necessary.

In the examples above, it is obvious that more available solutions (flow permutations) do exist. However, we can see that no one acceptable solution exists if flow2 enters the network first. It is caused by metric definition we use for SPF calculations. Here, the link capacity (bandwidth-BW) is the only criterion for routing decision. More explanation about the proposed routing technique we can see from the examples in Fig. 5 and Fig. 6.

Here, we have a network with 5 nodes and 4 VPNs plus new one crossing the network simultaneously. At the beginning, we calculate 7 traffic flows (4 VPNs for the first step) and after that, we add another VPNs (4 flows more), totally 11 flows.



(a)



(b)

(c)

Fig. 5. An example with the remaining network after utilization of four VPNs: (a) new test network with five nodes and capacities between them; (b) four VPNs are accomodated succesfully. All flows are marked with different color; (c) an example from (b) and remaining network after utilization of four VPNs. Source: authors.

An acceptable routing solution is ensured by flow permutation: 3-7-6-5-4-2-1-9-11-10-8 (Fig. 6). For the case shown in Fig. 5, many flow permutations are available starting with flows 1(274), 2(0), 3(322), 4(212), 5(188), 6(192), and 7(212). For the example in Fig. 6, only eight (8) flow permutations are available, all starting with flow3.

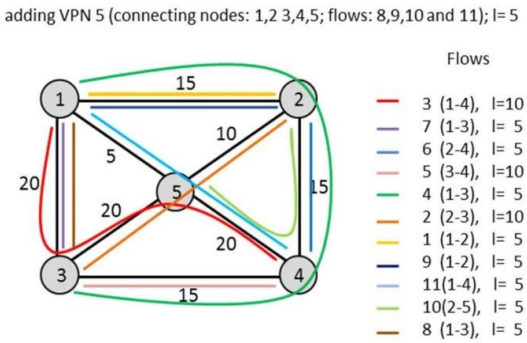


Fig. 6. Traffic situation after utilization of five VPNs, no free capacity exists (remaining network is zero). Such solution we got by proposed algorithm calculating all 11 flows simultaneously. In this case, the path migration and path splitting are not necessary. Source: authors.

We can see that the proposed technique is very effective and very clear to understand. Of course, too many flows are the crucial limitation for the algorithm based on the exact approach. One good thing is that network complexity (number of nodes and branches) is not so critical. For the case shown in Fig. 6, many flow permutations are available. Looking statistically in most of the cases, the flow3 is served first, because it is the huge one. If the flow2 is the first one, we have no available solution. From examples above, we can notice that flow3 (1-4) is crossing the network always by the similar path. Thus, we expect that probability should be

better if we accommodate flow3 always on that way. Possible we can calculate that problem without flow3, but we have to reduce the capacity of the remaining network for each link on the path for amount of flow3. So, we can develop the heuristic using artificial intelligence to reduce the calculation effort [25]. As it is shown in Fig. 7, the problem with a large number of flows can be solved with a special heuristic algorithm.

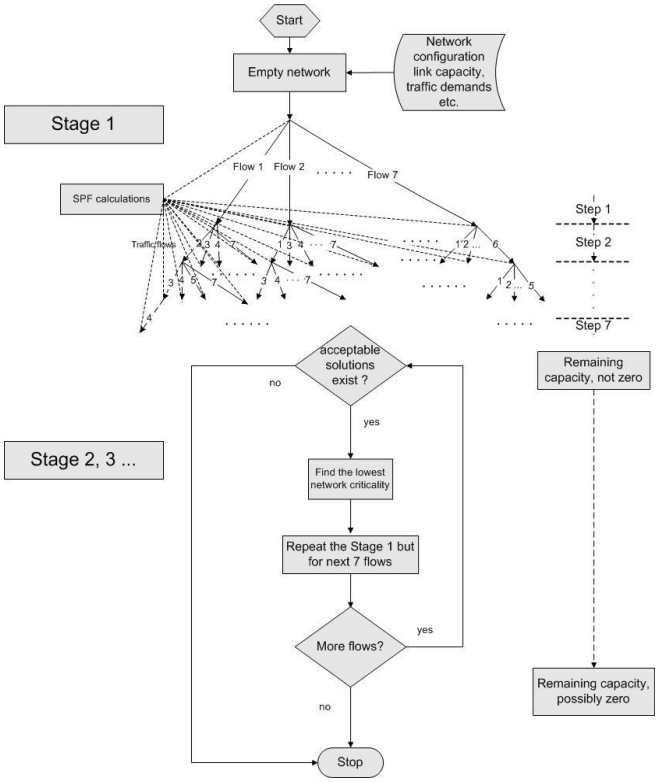


Fig. 7. A heuristic approach for huge problem with many traffic flows. The calculation is divided into more stages. Each acceptable solution generates a new remaining network as the starting network for the next stage. So, we can choose the one with the lowest network criticality. Source: authors.

VI. HEURISTIC APPROACH

For the problem from Fig. 5, the number of potential permutations is $M! = 7! = 5040$ and an appropriate number of SPF calculations is huge (as a measure of complexity). Because of specific configuration ($L = 8$ links), that number falls to 3569 SPF calculations (1). For the example in Fig. 6, we have a number of permutations $M! = 11!$, that is extremely high for average computing power.

As we said before, the crucial limitation of the proposed algorithm is the number of traffic flows. Now, we can explore in detail what happens if the number of flows increases. The complexity rises proportionally with M, but it is firmly correlated to the number of links L too. For more demanding problems (many traffic flows), we can apply heuristic from Fig. 7. In that approach, the calculation is divided into two or more stages. Also, it is recommended to sort flows by amount. Huge (elephant) flows should be processed first, before the small (mouse) flows.

For the next test-example (Fig. 8), we will use the same network structure as in Fig. 6, but we will double the link capacity. For example, the link 1-2 has capacity 15 in both directions. In addition, we added more VPNs and now we have 14 flows in total. For example, traffic flow 3 (1-4) is doubled. So, if we have $M = 14$, we can divide calculation into stages. On first stage, we calculate for 7 flows (flow1–flow7). As a result, we will get a number of available routing solutions (flow permutations). For each of them, we have a new remaining network G', so we can start the new procedure again - the second stage (for 7 flows more: flow8–flow14). In the research, we got many different final solutions. For solution from Fig. 8, we had approximately 12 000 SPF calculations (1st stage 4189 + 2nd stage 7556).

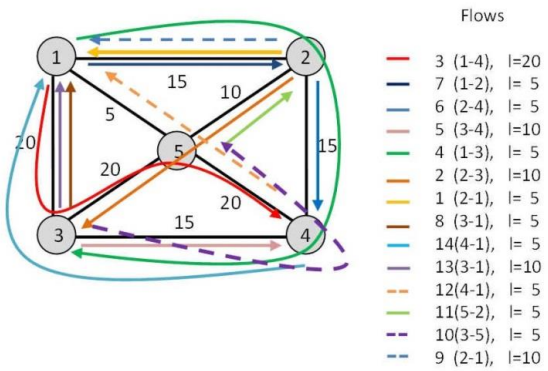


Fig. 8. A number of VPNs consisted of 14 flows are successfully accommodated. An available solution represented with flow permutations is 3-7-6-5-4-2-1-8-14-13-12-11-10-9. Source: authors.

As it said before, after the first stages (for each flow), we will get the new remaining network for each available solution (acceptable flow permutation). So, we can compare the criticality of each remaining network (3). It is recommended to choose the new starting network with the highest network robustness (the smallest criticality). It is the key element of heuristic to increase the chances of an acceptable final solution. We can see one of them in Fig. 8.

For further testing, we can change the number of bidirectional links L from 8 to 10 to see the influence on the calculation effort. In the network from Fig. 8, we can add links 1-4 and 2-3 with capacity increments (adding bandwidth, e.g., 5, 10, 15, 20). From Fig. 9, we can see that the number of SPF calculations rises as we extend the capacity, but the increase is not strongly dependent of L.

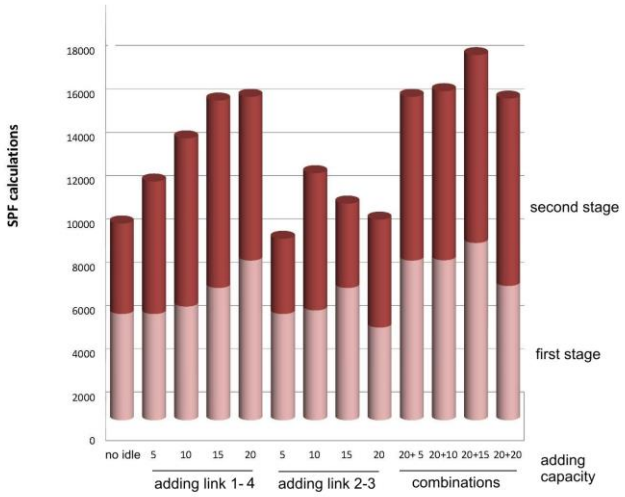


Fig. 9. Algorithm complexity, for example from Fig. 8, in relation to adding capacity on the bidirectional links 1-4 and 2-3. The heuristic algorithm is divided into two stages significantly reducing the complexity. Source: authors.

It is obvious that the best efficiency (the smallest calculation effort) appears if the network works close to the capacity limit. We can find more information about similar testing in [2].

The algorithm complexity is related to the number of SPF calculations increasing the computational time. However, that number grows linearly with L (number of links) and M (number of flows). If some adding flow overlaps with the existing flows (the same ingress and egress node), we can aggregate them in one, so we can significantly reduce the complexity.

VII. Calculating the Network Robustness/Criticality

Generally, we have more feasible (acceptable) routing solutions, that means there are more than one acceptable flow permutation for a given traffic load. Better to say: more path solutions can satisfy a given network load. Sometimes, it is critical to decide what solution is better than another. That measure is the robustness of the remaining network (remaining capacity) and it can be calculated [17]. The network criticality is opposite to the network robustness. If the network capacity decreases, the robustness of the network grows and the criticality falls down.

For the test network, we have graph G shown in Fig. 10.

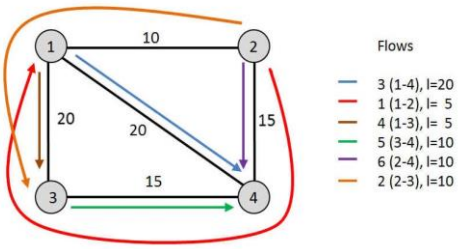


Fig. 10. Starting network G with six traffic flows. Optimal routing solution (flow permutation) is 3-1-4-5-6-2. Source: authors.

Table I represents the weighted adjacency matrix with the link capacity of the remaining network and it is denoted with G' (Table I). Starting capacity is shown in brackets. The weighted adjacency matrix is denoted as W', where weights are related to the capacity shortage. With D', we denoted the diagonal matrix, which is consisted from the sum of each column of the matrix W'.

TABLE I. THE REMAINING NETWORK.

Node	1	2	3	4
1	0	10(10)	5(20)	0(20)
2	5(10)	0	0	0(15)
3	15(20)	0	0	5(15)
4	20(20)	15(15)	10(15)	0

With L', we denoted a new Laplacian matrix

$$L' = D' - W'. \quad (2)$$

The network criticality τ' is calculated as follows

$$\tau' = 2N' \times \mathrm{Tr}(L'+). \quad (3)$$

As it is said in [17], "N' denotes the number of nodes in G', $\mathrm{Tr}(L'+)$ means a trace of matrix $L'+$, and $L'+$ is the pseudoinverse matrix of L'."

Here, we have a symmetrical test network with the same capacity in both directions. Starting criticality is $\tau' = 0.5358$.

In Fig. 10, we have a solution for six traffic flows and it means that traffic flow permutation 3-1-4-5-6-2 (order of flows) entering the network is an acceptable solution. All flows are successfully served and free capacity still exists. However, it is obvious that network criticality should be much higher than for an empty network. Now, the value by (3) is $\tau' = 1.111$. Therefore, the network robustness is much weaker (lower) than before.

VIII. Conclusions

For virtual network construction, we need a capable tool if we want to utilize the network capacity optimally. Existing techniques use pure SPF-based algorithms looking for an optimal path from the remaining network. If we observe some congestion caused by existing VPNs (their flows), the path splitting and path migration techniques are necessary. However, such techniques are very complicated and time-consuming with no guaranty that optimal solution will be discovered. That happens very often if the networks work on the edge of capacity. The proposed approach of virtual network construction combines SPF routing algorithm with traffic flow permutations, so path migration technique is not needed. The effectiveness of the proposed method has been tested on some examples. Generally, we can say that the proposed algorithm is very robust for large networks (huge number of nodes), but it has limitation for a large number of traffic flows crossing the network simultaneously. In that case, we can use the heuristic algorithm, which complexity rises linearly.

The calculation can be divided into more stages, each stage for a limited number of flows supported by network robustness calculation. Statistically, we can choose the most important flows to be processed first, which could be very important for good results of such a heuristic approach. In addition, the calculation of the network robustness enables searching for the best solution (traffic flow permutation), as appropriate starting network for the next stage. The new technique based on traffic flow permutation shows excellent performances for virtual network construction as we perform it offline.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] Y. Urayama and T. Tachibana, "Virtual network construction for robust physical networks", *International Journal of Communication Systems*, vol. 30, no.1, 2015.

[2] S. Krile, M. Rakús, and F. Schindler, "Centralized routing algorithm based on flow permutations", in *Proc. of 39th International Conference on Telecommunications and Signal Processing (TSP'16)*, Vienna, 2016, pp. 68–73. DOI: 10.1109/TSP.2016.7760831.

[3] A. Nakao, "Network virtualization as the foundation for enabling new network architectures and applications", *IEICE Transactions on Communications*, vol. E93.B, no. 3, pp. 454–457, 2010. DOI:

10.1587/transcom.E93.B.454.

[4] J. Turner and D. Taylor, "Diversifying the internet", in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM '05)*, 2005. DOI: 10.1109/GLOCOM.2005.1577741.

[5] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet", *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 59–64, 2003. DOI: 10.1145/774763.774772.

[6] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm", *International Journal of Communication Systems*, vol. 26, no. 8, pp. 1054–1073, 2013. DOI: 10.1002/dac.1399.

[7] W. Deng, F. Liu, H. Jin, X. Liao, and H. Liu, "Reliability-aware server consolidation for balancing energy-lifetime tradeoff in virtualized cloud datacenters", *International Journal of Communication Systems*, vol. 27, no. 4, pp. 623–642, 2014. DOI: 10.1002/dac.2687.

[8] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components", in *Proc. of 25th IEEE International Conference on Computer Communications (IEEE INFOCOM 2006)*, 2006, pp. 1–12. DOI: 10.1109/INFOCOM.2006.322.

[9] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration", *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008. DOI: 10.1145/1355734.1355737.

[10] J. He, Z. S. Rui, Y. Li, C. Y. Lee, J. Rexford, and M. Chiang, "DaVinci: Dynamically adaptive virtual networks for a customized internet", in *Proc. of the 2008 ACM Conference on Emerging Network Experiment and Technology (CoNext 2008)*, 2008. DOI: 10.1145/1544012.1544027.

[11] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection" in *Proc. of the 1st ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA 2009)*, 2009. DOI: 10.1145/1592648.1592662.

[12] M. Mori, T. Tachibana, K. Hirata, and K. Sugimoto, "A proposed topology design and admission control approach for improved network robustness in network virtualization", in *Proc. of 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011. DOI: 10.1109/GLOCOM.2011.6134260.

[13] Y. Urayama and T. Tachibana, "Virtual network construction with k-shortest path algorithm and prim's MST algorithm for robust physical network", in *Proc. of the International MultiConference of Engineers and Computer Scientists (IMECS 2014)*, 2014.

[14] Y. Urayama, H. Tsubota, and T. Tachibana, "Virtual network construction scheduling based on network criticality for robust physical networks", in *Proc. of 2015 IEEE International Conference on Consumer Electronics*, Taiwan, 2015. DOI: 10.1109/ICCE-TW.2015.7217007.

[15] J. Duan, Z. Guo, and Y. Yang, "Cost efficient and performance guaranteed virtual network embedding in multicast fat-tree DCNs", in *Proc. of 2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, vol. 2015, pp. 136–144. DOI: 10.1109/INFOCOM.2015.7218376.

[16] Y. Urayama and T. Tachibana, "Rapid topology design based on shortest path betweenness for virtual network construction", *IERI Procedia*, vol. 10, pp. 105–111, 2014. DOI: 10.1016/j.ieri.2014.09.098.

[17] A. Tizghadam and A. Leon-Garcia, "Autonomic traffic engineering for network robustness", *IEEE Journal on Selected Areas in Communication*, vol. 28, no. 1, pp. 1–12, 2010. DOI: 10.1109/JSAC.2010.100105.

[18] A. H. Dekker and B. D. Colbert, "Network robustness and graph topology", in *Proc. of 27th Australasian Computer Science Conference (ACSC 2004)*, 2004, pp. 359–368.

[19] M. Filder, "Algebraic connectivity of graphs", *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[20] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees", *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981. DOI: 10.1007/BF00288961.

[21] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks", *IEEE Network Magazine*, vol. 14, no. 2, pp. 11–19, 2000. DOI: 10.1109/65.826367.

[22] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008. DOI: 10.1145/1355734.1355746.

[23] A. Meddeb, "Building cost-effective lower layer VPNs: The ILEC/CLEC dilemma", *International Journal of Communication Systems*, vol. 23, no. 11, pp. 1405–1430, 2009. DOI: 10.1002/dac.1114.

[24] J. Y. Yen, "Finding the k-shortest loopless paths in a network", *Management Science*, vol. 17, pp. 712–716, 1971. DOI: 10.1287/mnsc.17.11.712.

[25] Z. Li and R. Wang, "Multipath routing algorithm based on traffic prediction in wireless mesh network", *Communications and Networks*, vol. 1, no. 2, pp. 82–90, 2009. DOI: 10.4236/cn.2009.12013.

[26] Case Study: "Shortest-Path Algorithms". [Online]. Available: http://www.mcs.anl.gov/~itf/dbpp/text/node35.html#algdij1