A Performance Study of Network Migration to SDN-enabled Traffic Engineering

Marcel Caria†, Admela Jukan†, and Marco Hoffmann*
Technische Universität Carolo-Wilhelmina zu Braunschweig†, Nokia Siemens Networks Germany*
Email: {caria, jukan}@ida.ing.tu-bs.de†, marco.hoffmann@nsn.com*

***Abstract*—In this paper, we analyze the question of network migration to Software Defined Networking (SDN) from the perspective of Traffic Engineering (TE). For a given network topology and migration planning horizon, we ask the question of which routers in the IP network should migrate to SDN-enabled operation to reduce the need for network capacity upgrades over a given time horizon. We propose an algorithm to determine the optimum schedule for node substitution within a network, which shows that already a few SDN routers, when strategically located, provide a stably large number of path alternatives to be used in TE, thus substantially reducing the need for large network capacity upgrades.**

I. Introduction

Software Defined Networking (SDN) has gained a lot of attention during recent years and most network equipment vendors have announced the intention to build SDN enabled devices, or have already released SDN-capable products [1]. SDN decouples the control plane from the data plane along with the centralization of all network control functions. By using a standardized interface between the control and data plane (e.g., OpenFlow protocol), SDN allows the network to become vendor agnostic, so that an operator can combine networking equipment from different suppliers. In addition, the separation of the control and data plane allows innovation in each of them independently. Finally, the centralized software based network control allows for great flexibility and programmability of the network and enables the operator to customize the network operations. Even though the centralized SDN concepts have been criticized in the past for the lack of scalability, recent work has shown that SDN has no inherent scalability issues [2].

A natural question for every network operator is the issue of migration from IP routers to SDN-enabled equipment, e.g., OpenFlow switches. On the one hand, not all routers in the network domain can migrate to SDN at once, due to operational and economic constraints. On the other hand, since SDN allows for more sophisticated traffic engineering in the packet layer compared to the common IP routing protocols with destination-based forwarding and least-cost routing, the immediate benefits in traffic engineering are obvious. However, if we assume that a typical medium to large scale ISP will not migrate to SDN at once, but in multi-period (years) planning cycle, the network operator might want to know which network nodes should be migrated first, and which subsequently, etc. To this end, it is important to understand how a network can make best use of SDN-enabled traffic engineering during all stages of a migration process. Especially those stages are of interest which assume that the native IP routing, such as OSPF, co-exists with SDN-enabled traffic engineered routing. Please note that our focus here is on IP layer, and not on network technologies with built-in traffic engineering capabilities, e.g., MPLS-TE. The reason behind that is, as we believe, SDN routers will substitute legacy IP routers, whereas other technologies work in the layers below and do not allow gradual migration; for instance, one cannot deploy a few Carrier Ethernet switches in combination with legacy SONET equipment.

In this paper, we propose a novel two-stage algorithm that optimizes the sequence of nodes for SDN migration, referred to as *migration scheduling*. In the first stage, the algorithm analyzes the network topology in order to find all path candidates for traffic engineering, and to identify routers (nodes) that have to be SDN migrated to allow the usage of the said paths. The second stage is the optimization of the scheduling with the objective to maximize the total number of path alternatives over the whole migration planning horizon. Our initial results show that an optimized migration scheduling can increase the number of alternative paths notably, especially in the early migration stages. This implies that just a few properly located SDN routers can provide a significant performance improvement. To evaluate how this result translates into capacity savings through traffic engineering, we also develop a corresponding performance benchmark. To this end, we first simulate traffic, including the growth of traffic over the migration planning horizon. After that, we optimize the routing in each migration period – according to the available paths through the nodes already migrated in that period – with the objective to minimize the installed link capacities. The results obtained show that our approach is effective and can result in significant capacity savings up to 16% already in earliest stages of the migration time horizon, as compared to the 9% of an SDN migration with randomly chosen nodes for migration.

The rest of the paper is organized as follows: Section II discusses the related work. Section III presents the network architecture. The optimized migration scheduling algorithm is presented in Section IV. Sec-

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

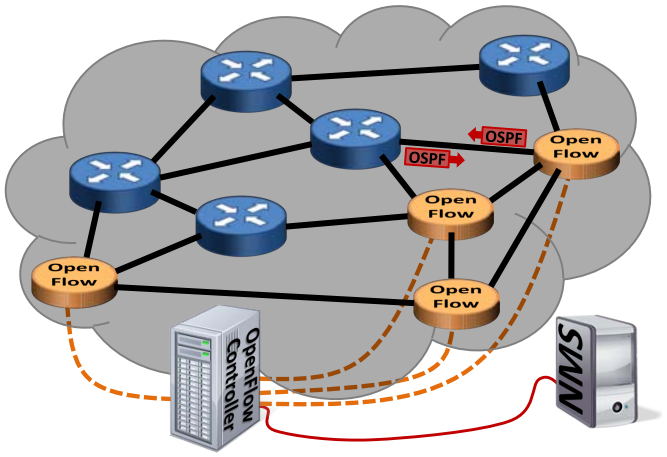


Fig. 1: During the migration from (legacy) OSPF routers to (new) SDN routers (e.g., OpenFlow switches), the SDN routers must be OSPF enabled.

tion V presents the performance study and Section VI the conclusions.

II. Related work

Since our main goal is to find an optimal sequence of nodes to migrate to SDN, the capacity planning problem is out of scope. We therefore refer to the comprehensive work on planning techniques for optical core networks [3], where one of the most important findings is that the network operator's choice of the prediction interval for the demand and cost forecast is important. By choosing a short prediction interval, it may be not possible to take the future evolutions sufficiently into account, while choosing a long prediction interval may suffer from uncertain predictions. An efficient approach using an ant colony meta heuristic for the multi-layer multi-period migration problem formulated as a path finding problem is proposed in [4]. Stochastic programming approaches like [5] have shown to better handle the uncertainties in multi-period network planning, when the future can be classified into only a few different scenarios.

As we model our migration scenario as a multi-period planning problem, the decision on how long the planning horizon should be (i.e., after how many years should the network be fully SDN-enabled), is important for the network operator. To this end, we refer to [6], where the timing issues of migration to a new technology were studied, and it was shown that demand growth, migration cost, and cost savings from the new technology have to be taken into consideration. In [7] the number and location of SDN controllers is studied, which is an important related aspect, but is outside the scope of this paper. As far as other work is concerned, to the best of our knowledge there is no other related work focusing on gradual network migration to SDN-enabled routing.

III. Reference Migration Scenario

The reference migration scenario is illustrated in Figure 1, showing a network with five OSPF routers and four SDN routers. As it can be seen, after the initial migration, the conventional routers with a legacy routing protocol (i.e., OSPF) are deployed together with SDN-enabled routers. As it can also be seen, all SDN routers are controlled by a centralized controller, which in turn is managed by the Network Management System (NMS). Please note that SDN alone does not provide TE capabilities in the sense of network optimization itself. TE applications still remain implemented in the NMS, since a typical TE application also requires input from the traffic monitoring system, network policies from the operator, etc. However, what SDN does provide is a thorough configurability of the routing of all flows in the network, and thus a much greater solution space for network optimization, which in turn leads to an overall better performance of TE.

To inter-operate the OSPF and SDN-enabled routers, we also assume that the SDN routers are also OSPF-enabled. In other words, SDN routers[1] have to be capable to exchange OSPF packets with the legacy routers, or else the routing service cannot be properly configured, since the OSPF routers would not be able to find paths via SDN nodes. This feature can either be implemented in SDN routers, or by letting the SDN routers forward all OSPF packets to the SDN controller, which then has to process the protocol and send according "response" packets back. The latter solution is shown to be feasible, see [8].

Regarding the actual traffic engineering enabled actions through SDN, our assumption is that the SDN controller has access to all necessary information including network topology, traffic monitoring, and routing. Without loss of generality, we assume that traffic engineering is performed only once at the beginning of every migration period. Therefore, a reasonable capacity headroom (e.g., maximum link utilization set to 70%) assures that despite the expected traffic growth, all links are loaded fairly below the threshold until the end of the current migration period. Before the migration starts, routing is based on the OSPF protocol only, i.e., traffic is always forwarded on the shortest path based on the destination IP address[2]. Finally, traffic engineering through OSPF link cost (weight) changes is not considered in this paper, which we justify with the known fact that network operators do not commonly deploy OSPF cost changes for the reasons of routing stability [9]. For instance, Cisco recommends to just set the link cost inversely proportional to the capacity of the link, without taking its utilization into account [10].

[1]Please note that even though we refer to OpenFlow switches interchangeably as SDN-enabled routers, where former is capable of forwarding based on layer 2 header information, the scope of this paper is limited to the IP layer.

[2]Load balancing schemes (e.g., OSPF's Equal-Cost Multi-Path) are not taken into account here. This is a valid assumption, since whenever in use, the split ratio is statically rather than dynamically assigned, which in turn means that load balancing can not be considered as traffic engineering. We show later that this restriction is not limiting the generality of our numerical analysis.

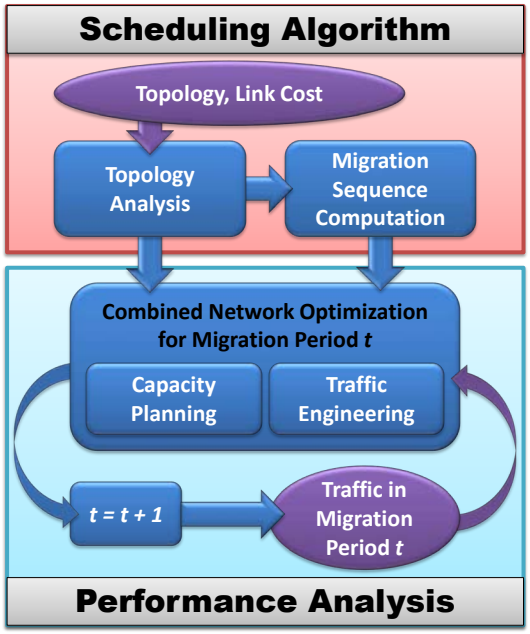


Fig. 2: The proposed algorithm (upper box) and the performance benchmarking (lower box).

As we consider a multi-period network migration (e.g., duration of five years, with ten 6-month migration periods), we take into account the growth of traffic load over the entire planning horizon. Without migration, an expectation is that link capacities have to be upgraded as the traffic grows; our goal here is to study whether advanced traffic engineering capabilities of SDN-enabled routers may be equally sufficient to reduce the amount of capacity upgrades. For instance, if there is 12 Gb/s traffic load on a certain link in the network, an upgrade from 10Gb/s to 40 Gb/s maybe necessary, which would require installation of 40 Gbit/s ports on both ends, while it might be possible to re-route a few traffic flows on the said link in case SDN routers are already deployed along that path.

IV. Migration Scheduling

The migration strategy that we propose here is based on a two stage heuristic algorithm to calculate an order of network nodes advantageous for the network operator to be used as migration sequence. An illustrative flow chart of our algorithm is shown in Figure 2: the ovals depict the input data and the rounded rectangles show the functions used. The two functions (i.e., stages) of the algorithm are topology analysis and migration sequence computation. The performance analysis ("benchmark") is illustrated in the lower part of Figure 2. This benchmark simulates for the given topology information and node sequence the traffic growth over the migration planning horizon and optimizes the network by means of combined capacity planning and traffic engineering for each migration period. As such, the performance benchmark is used to evaluate how the proposed migration strategy actually translates into capacity savings in the network. It can be seen that, for the benchmark, the available paths and the information on already migrated nodes are input from the scheduling algorithm (upper part). The benchmark starts in migration period 0, in which there is no router migrated yet, which means that all flows are routed via their least-cost paths. In this way, the network optimization is restricted to just choose the smallest possible link capacities according to the flows they carry. Then, the scenario changes to the next migration period (depicted by the "t=t+1" box) and the traffic load on all flows is increased, so that a combined network optimization can be performed with the new set of parameters. This analysis is iterated over all migration periods, so that the final result of our migration scheduling can be evaluated by comparing it to a random migration sequence. The two migration sequences (optimized vs. random) are compared based on how much the capacity upgrades (which are necessary due to the growing traffic) can be reduced over time.

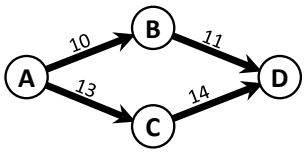
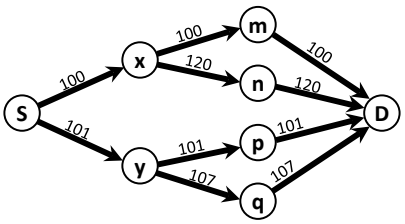


Fig. 3: The maximum difference between link costs is assumed to be constrained by the longest path.

A. Topology Analysis

The first step of our algorithm is to compute *all* shortest paths (in number of hops) in the topology. To reduce the problem complexity, we consider only paths of the same length (i.e., same hop count) compared to the least-cost routes as path alternatives for traffic engineering. While SDN-capable routers could actually route flows on longer paths, we justify this restriction with the facts that 1) longer paths (i.e., more hops) imply a reduced Quality of Service for the users due to an increased packet delay, and 2) flows routed over more links than necessary waste more capacity installed. However, please note that there are possible cases in which re-routing flows to longer paths could save capacity, e.g., when re-routing a flow from a poorly utilized 40 Gbit/s link to a longer path with enough spare capacity, so that a 10 Gbit/s link could be used instead. To assure shortest path routing, we assume that the difference between any two link costs is below the inverse of the global maximum path length relative to the minimum link cost in the topology. In other words, this constraint assures that no path with n hops between any node pair can have a higher path cost than any path with more than n hops between the same node pair. In fact, the constraint provokes that only shortest paths can be used for routing, as it assures that more hops must result in a higher link cost in any case. Please note that leaving out this assumption would only result in slightly improved SDN traffic engineering capabilities. Figure 3 illustrates the



Path	(S→x→m→D)	(S→x→n→D)	(S→y→p→D)	(S→y→q→D)
Key Nodes	{}	{x}	{S}	{S, y}

Fig. 4: Key nodes: There are four paths from S to D, and the route via m is the least cost-route; all other paths require some of the traversed nodes to be SDN, so we denote them as key nodes on these paths.

idea: The maximum path length is $L_{\max} = 2$ (i.e., $A \to B \to D$ and $A \to C \to D$), and the minimum link cost is $W_{\min} = W(A \to B) = 10$. Hence, the maximum allowed difference between any two link costs is $\Delta_{\max} < W_{\min}/L_{\max} = 10/2 = 5$. It can be seen in Figure 3 that the shown link costs meet the condition: $\Delta_{\max} = 14 - 10 = 4 < 5$.

The second step of the topology analysis is to identify the so-called *key nodes* on all paths. We define a key node n on path p as a node which has to be SDN migrated in order to allow the usage of path p for traffic engineering. The fact that certain paths can not be used for traffic engineering without SDN capabilities is due to the nature of destination-based forwarding and least-cost routing. Figure 4 illustrates the context: There are four different paths from S to D, via node m, n, p, or q. Without any SDN migrated node, only the m-path can be used, because it is the least-cost. That means node S will forward any traffic for D towards node x without taking into account what x will do with that traffic. The reason for that is that S learned through the routing protocol from x that it can reach D via x with an accumulated cost of 300, which is minimum for S. In case node x is already migrated to SDN, the path via node n can also be assigned for the traffic between S and D by the traffic engineering algorithm, because x can be configured by the central SDN controller to forward traffic with source S and destination D to node n. This example shows why x (and only x) is called the key node on the path from S to D via n, and how SDN eliminates the least-cost routing and destination based forwarding constraints. Accordingly, the p-path has node S, and the q-path has nodes S and y as key nodes.

However, ordering the nodes in the migration sequence according to the number of paths in which they appear as a key node is not necessarily leading to a good solution (i.e., as many paths as possible as early as possible), as it can be seen from the following example: Assume five arbitrary paths, here identified only by the set of their key nodes: $\{a\}$, $\{a, b, x\}$, $\{b, c, x\}$, $\{c, d, x\}$, and $\{d\}$. Obviously, x appears in the most paths as key node, but in all four optimal migration sequences $adxbc$, $adxcb$, $daxbc$, and $daxcb$, the node x appears as third one. Of course, the results can worsen further in more complex scenarios, which is why we developed the mathematical model described in the following subsection.

Parameter	Meaning
$0 \le t \le \mathrm{T}$	Migration period t
$p \in \mathrm{P}$	Path p, set of all pre-calculated paths P
$n \in \mathrm{N}$	Network node n
φ_p	Priority of path p
α_p	Number of key nodes on path p
β_p^n	Boolean routing parameter, true if node n is a key node on path p
Variable	**Meaning**
μ_t^n	Boolean to determine if node n is already migrated in period t
π_t^p	Boolean to determine if path p is already available in period t

TABLE I: Summary of Notation

B. Migration Sequence Computation

We now present the used Integer Linear Programming (ILP) model we used to determine the schedule (i.e., the sequence of nodes) in a multi-period migration scenario. A summary of the used notation is given in Table I, explaining all necessary input in the *Parameter* part, followed by a description of the two necessary variables. We model all migration periods including $t = 0$, which poses the situation before migration, i.e., with zero nodes migrated. In the final period $t = T$, all nodes have been migrated. This means that independently of the migration sequence, all pre-calculated paths $p \in P$ (which is the input coming from the topology analysis) are available for traffic engineering. In the previous subsection, we defined a key node n of a path p as a node that must be migrated in order to allow the usage of p for traffic engineering. This means that p is in fact not the least cost route. Therefore, the availability π_t^p of a path p in period t depends on whether all its key nodes have already migrated in that period. The following constraint models this feature:

$$\forall p, t: \qquad \pi_t^p \cdot \alpha_p \le \sum_n \beta_p^n \cdot \mu_t^n$$

The number of nodes that can migrate per migration period has to be limited to the number of nodes divided by the total number of migration periods. We model this with the following constraint:

$$\forall t: \qquad \sum_n \mu_t^n \le \left\lceil \frac{|N|}{T} \right\rceil$$

In case $|N|$ can't be divided by T without a remainder, we simply round up, which leads to a final migration period with a lower number of nodes that migrate. Finally, we also have to constrain the trivial fact that we disallow any node to "migrate backwards":

$$\forall t: \qquad \mu_t^n \le \mu_{t+1}^n$$

The objective function is to maximize the number of paths used for traffic engineering, summarized over the whole planning horizon, i.e.,

$$\text{Maximize} \sum_t \sum_p \pi_t^p \cdot \varphi_p$$

We leave the question how a path is used for traffic engineering (i.e., what values are assigned to all φ_p) to be answered individually by the user of our model, as we do not provide a general answer. We assume that an optimal migration scheduling can not be found by only determining φ_p, because for an optimal migration scheduling one would definitely have to take the process of traffic engineering during all migration periods into consideration. For illustration, we define three different strategies to determine φ_p, i.e., three different heuristics to find migration schedules:

– Number of paths: In this strategy, the value φ_p of each path p is set to 1, so that all paths are assumed to be identically beneficial for traffic engineering. This leads to a migration schedule where nodes are migrated first, which provide the highest total number of path alternatives for traffic engineering.

– Traffic: This strategy sets the value φ_p of each path p to the initial traffic demand between its end nodes. This way, nodes are migrated first, which can provide the most path alternatives for high traffic flows.

– Path diversity: This strategy is a modification of the first one, having the same initial assignment of $\varphi_p = 1$ for all paths. Then, for each s-d node pair, one path p (that is not the least cost path) is randomly chosen to be set to a higher φ_p. This strategy leads to a migration schedule where already in the early migration at least one path alternative for as many existing least-cost routes as possible are available for TE.

V. Performance Evaluation

For our performance analysis we used the TA2 network from the SNDlib topology library [11], which has 65 nodes, 108 links, and 4160 traffic flows. The migration planning horizon was set to ten migration periods, in which routers had to be migrated. This time is partitioned into nine periods each with seven routers to migrate and a final period with only two routers. We have assigned the link weights randomly (with the constraints explained in Subsection IV-A) and averaged the results of the performance benchmark over ten different random assignments of link weights. The migration sequence optimization was computed with the GUROBI optimizer on an Intel Core i7-3930K CPU (6 x 3.2 GHz) in less than ten seconds, while the performance benchmark's traffic engineering part took up to 30 minutes (with an allowed MIP gap of 1%). For the performance benchmark we assumed that links are available in the following granularity: 1 Gbit/s, 5 Gbit/s, 10 Gbit/s, 40 Gbit/s, 100 Gbit/s, 400 Gbit/s, 1 Tbit/s. The traffic matrix was also randomly generated and the values are uniform distributed between 0 and 400 Mbit/s per traffic flow. For the traffic growth during the migration planning horizon we set the growth factor of each flow in each period to a random value between 1.05 and 1.35 so that the mean growth is 20% in every period. For every one of the ten sets of random link weights, we also generated an individual traffic scenario.

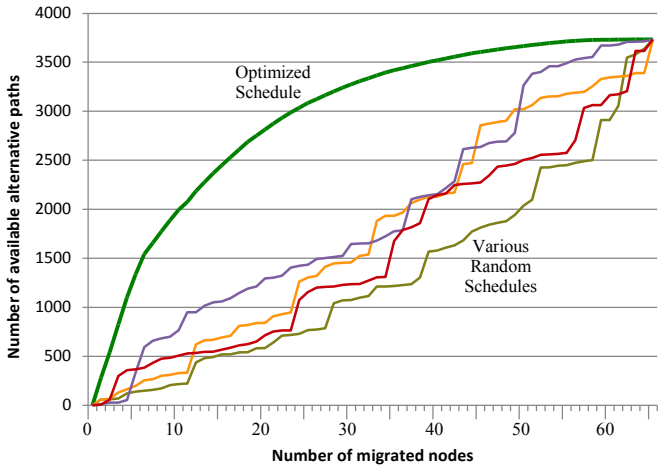


Fig. 5: Performance comparison of optimized migration schedule and random schedule.

Figure 5 shows our initial result, which is the number of path alternatives, i.e., the paths that the TE algorithm can use to minimize the network capacity, depending on the number of migrated nodes. The green (monotonic) graph shows the result for an optimized migration schedule, while the other (zigzag) graphs show results for various random sequences of node migration. As it can be seen from the figure, all graphs start at the same point and end at the same point, which is due to the following fact. Before the migration process, none of the routers have migrated, so that there are also zero alternative paths available. All graphs also end at the same point, because independently of the migration schedule, all alternative paths are available after all nodes migrate to SDN. The most important insight from this initial result is that an optimized migration sequence shows a drastically increased number of alternative paths through SDN nodes, especially during the early migration periods. The more nodes are migrated, the less is the difference compared to any random migration sequence.

In order to evaluate how beneficial this increased number of alternative paths actually is for the network operator, we used the performance benchmark, as explained in Section IV. Figure 6 shows the saved network capacity through traffic engineering in each migration period. The green graph is the result for an optimized migration order using the first heuristic that maximizes the total number of alternative paths (explained at the end of Subsection IV-B). While with any migration sequence, SDN allows to save considerable amounts of network capacity through the alternative paths, it can also be seen from the figure that especially in the first third of the migration planning horizon (i.e., periods 1, 2, and 3), an optimized migration

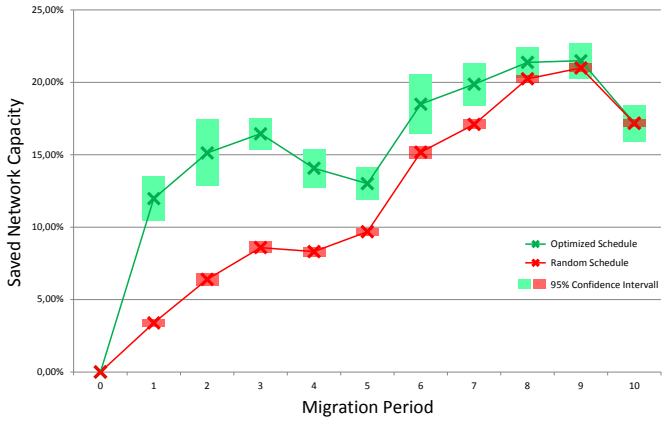


Fig. 6: Performance comparison of optimized migration schedule and random schedule.

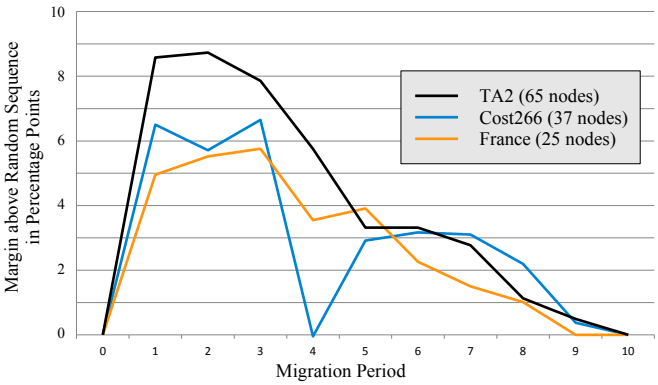


Fig. 7: Optimized vs. random migration sequence for different network topologies

sequence clearly outperforms random sequences. The advantage of the optimal sequence is almost constantly at 8 percentage points over the random sequences during this first third. An eye-catching point of the result is the performance drop between the fourth and the fifth period, and again at the tenth migration period. An analysis of the used port sizes in the result data showed that starting with the fourth period, multiple 400 Gbit/s links had to be used, which are likely to be underutilized in that early stage. Something similar happened in the final migration period, where for the first time multiple 1 Tbit/s links had to be used. In another study (not shown here), we also generated results for the other two migration sequence heuristics (i.e., the strategy that prefers alternative paths for high bandwidth flows, and the strategy that prefers alternative paths for flows which have no such alternatives yet). However, all three strategies exhibited the same performance (with differences within the confidence interval), and are therefore not shown here.

Finally, we evaluated the behavior of our scheme with two sample topologies, i.e., the Cost266 network (37 nodes, 57 links, 1332 traffic flows) as a medium size topology and the France network (25 nodes, 45 links, 600 traffic flows) as a small size topology. This allowed us to compare the outcome with the TA2 network (65 nodes, 108 links, 4160 traffic flows), already used in the previous subsection. In all three networks we set the number of migration periods to ten and the average traffic increase to 1.2 per migration period. However, in order to make the results comparable, we had to scale up the initial traffic for France and Cost266 networks, so that they show the same performance drop in the middle and at the end of the planning horizon, like seen in the TA2 network in Figure 6. The Cost266 network was set to 0–1.2 Gbit/s per flow and the France network was set to 0–2.5 Gbit/s per flow. Figure 7 shows only the margin between the optimized and the random migration sequence in percentage points. As it can be seen from the result, all topologies show comparable graphs, with a high advantage of the optimum over the random sequence in the first third of the migration planning horizon.

VI. Conclusions

In this paper, we analyzed network migration to Software Defined Networking (SDN) from the perspective of Traffic Engineering (TE). Under the assumption that a network operator will not migrate all routers at once, but rather in an extended period of time, we showed that the order in which the network nodes are substituted is of high importance for the performance. While SDN already provides an excellent capacity saving opportunity regardless of a migration sequence, the migration strategy proposed in this paper has been shown to further significantly increases these benefits.

Acknowledgment. This work has been supported by the German Federal Ministry of Education and Research (BMBF) under code 01BP12300A; EUREKA-Project SASER.

References

[1] Open Networking Foundation, members list, www.opennetworking.org/membership/members
[2] S.H. Yeganeh, A. Tootoonchian, Y. Ganjali, *"On Scalability of Software-Defined Networking,"* IEEE Communications Magazine, vol.51, no.2, pp.136,141, February 2013
[3] S. Verbrugge, *"Strategic Planning of Optical Telecommunication Networks in a Dynamic and Uncertain Environment,"* Ph.D dissertation, Universiteit Gent, 2007
[4] S. Türk, R. Radeke, R. Lehnert, *"Network Migration Using Ant Colony Optimization,"* CTTE, June 2010
[5] C. Kronberger, T. Schöndienst, D.A. Schupke, *"Impact and Handling of Demand Uncertainty in Multiperiod Planned Networks,"* IEEE ICC, Kyoto, Japan, June 5-9, 2011.
[6] S. Rajagopalan, *"Adoption timing of new equipment with another innovation anticipated,"* IEEE Transactions on Engineering Management, volume 46. issue 1, pp. 14-25, Feb. 1999
[7] B.Heller, R. Sherwood, N. McKeown, *"The controller placement problem,"* HotSDN, Helsinki, Finland, 2012
[8] The RouteFlow Project, https://sites.google.com/site/routeflow/
[9] S. Das, G. Parulkar, N. McKeown, *"Why OpenFlow/SDN Can Succeed Where GMPLS Failed,"* ECOC, Amsterdam, June 2012
[10] Cisco Support Community, *"How to configure OSPF cost,"* https://supportforums.cisco.com/docs/DOC-5349
[11] SNDlib library, http://sndlib.zib.de