HOP-BY-HOP FORWARDING ROUTING SURVEY OF NETWORK VISUALIZATION

DASARI TANUJA 1*, S.SHANAWAZ BASHA 2*

1. II.M.Tech, AVR & SVR Engineering College, KURNOOL.

2. ASSISTANT PROFESSOR, AVR & SVR Engineering College.

Abstract: *The process of planning a virtual topology for a Wavelength Devision Multiplexing (WDM) network is called Virtual Topology Design (VTD). The goal of VTD is to find a virtual topology that supports forwarding the expected traffic without congestion. In networks with fluctuating, high traffic demands, it can happen that no single topology fits all changing traffic demands occurring over a longer time. Thus, during operation, the virtual topology has to be reconfigured. Since modern networks tend to be large, VTD algorithms have to scale well with increasing network size, requiring distributed algorithms. Existing distributed VTD algorithms, however, react too slowly on congestion for the real-time reconfiguration of large networks.*

We propose Selfish Virtual Topology Reconfiguration (SVTR) as a new algorithm for distributed VTD. It combines reconfiguring the virtual topology and routing through a Software Defined Network (SDN). SVTR is used for online, on-the-fly network reconfiguration. Its integrated routing and WDM reconfiguration keeps connection disruption due to network reconfiguration to a minimum and is able to react very quickly to traffic pattern changes. SVTR works by iteratively adapting the virtual topology to the observed traffic patterns without global traffic information and without future traffic estimations. We evaluated SVTR by simulation and found that it significantly lowers congestion in realistic networks and high load scenarios..

1. INTRODUCTION

A reconfigurable optical Wavelength Division Multiplexing (WDM) network consists of a set of Reconfigurable Optical Add/Drop Multiplexers (ROADMs) interconnected by optical fibers. Based on this physical topology, a virtual topology (VT) is built using what are called light paths. A light path is a path through the physical network along with a wavelength for each fiber; no two light paths must use the same wavelength on any shared edge. The light paths constitute the edges of the virtual topology. Two nodes are able to directly communicate with each other only if they are connected by a light path. In an IP-over-WDM network, IP traffic is routed over the VT of the WDM network.

Planning the virtual topology of an optical WDM network is a multilayer optimization problem called Virtual Topology Design (VTD). Depending on the traffic demands of the network, an appropriate virtual topology has to be found that is compatible with the physical topology of the network and allows routing of the traffic without congestion. VTD is an NP-complete problem commonly solved by using heuristics [1].

These heuristics usually work offline using full knowledge about both the expected traffic patterns and the physical topology. The outcome of such a heuristic is a virtual lightpath topology and a routing of the traffic on that topology.

**** DASARI TANUJA***
II.M.Tech, AVR & SVR Engineering College, KURNOOL.

When using centralized algorithms to perform VTD, the global network state has to be gathered in one node. Since the network state contains the traffic matrix, it grows quadratic ally with the size of the network and fluctuates quickly. This makes it hard to acquire up-to-date information in real time. With increasing network size, centralized VTD algorithms cannot react to traffic changes quickly enough for fast network reconfiguration. Since this leads to worse resource usage, more network resources (i.e., ROADMs and fibers) are required, leading to both higher OPEX and CAPEX.

We present a distributed Virtual Topology Design algorithm for IP-over-WDM networks that can be used for iterative network reconfiguration with no need to determine global traffic demands. We call our approach Selfish Virtual Topology Reconfiguration (SVTR). In SVTR, switches selfishly manipulate the existing virtual topology of the WDM network based on their observed traffic demands. Through extensive simulation with traffic demands collected from a real network, we show that SVTR reacts quickly to traffic pattern changes and that the achieved throughput is superior to both a well studied centralized VTD algorithm [2] and a distributed VTD algorithm [3].

Our algorithm differs from existing distributed VTD algorithms in the following ways: a) we have no need to gather global traffic information, b) we need neither synchronization nor election processes making the distributed application scalable, c) our approach does not require every node to have information about optical resource availability and d) we include backplane capacity limits of switches in our reconfiguration decisions.

The rest of this paper is structured as follows: Section II discuses related work in Virtual Topology Planning. In Section III our assumed network model consisting of WDM equipment and OpenFlow enabled switches is presented. In Section IV the Selfish Virtual Topology Reconfiguration paradigm is presented which is evaluated in Section V. Section VI concludes this paper.

II. RELATED WORK

Existing approaches for planning virtual topologies for IP-over-WDM networks can be classified through several properties. These include using knowledge about future traffic demands, being distributed or centralized, the reconfiguration triggering method, and by proactively or reactively reconfiguring the network [4]. In addition, there are approaches planning a topology from scratch and approaches which iteratively reconfigure a given topology into a target topology. Reference [2] presents three fundamental, centralized VTD algorithms that assume future traffic to be known. These algorithms build a topology from scratch and work by ranking each pair of nodes according to different metrics. The used metrics are all based on either the total traffic demands or the residual traffic demands of the individual nodes. Based on the ranking, lightpaths are created between pairs of nodes. Inspired by these algorithms, we adapted the use of residual demands for our distributed algorithm.

Distributed VTD algorithms spread the logic used to compute reconfigurations among the participants of the network (i.e.,the switches) and promise faster decisions and better scalability than centralized algorithms. The distributed algorithm designed by Shiomoto et al. [3] works by distributing traffic information to all nodes. Based on this information each node computes a new virtual topology and reconfigures the WDM network appropriately. A major drawback when using this algorithm in practice is that it requires all nodes to have the exact same traffic information, which is far from being trivial to achieve in real time. Nevertheless, among the few available distributed VTD algorithms, it is the one with most realistic requirements and from our point of view the only one which could be applied in practice. Unfortunately, the network assumptions used for the simulation in its original paper were too strong (uniform traffic distribution). We simulated the algorithm with our realistic network model and found out that the algorithm yields poor results under these assumptions.

Reference [5] introduces a distributed algorithm for connection rerouting at sub-wavelength granularity for WDM networks. Depending on the routing, lightpaths are added/removed from the virtual topology. Unfortunately, the rerouting calculations require all nodes to have global information about the routing of all sub-wavelength granularity flows. Obviously, this approach is not applicable even to medium-scale networks.

The distributed algorithm introduced in [6] aims at grooming SONET circuits to optical connections and could be adapted to IP-over-WDM networks. In contrast to [3], not every node requires traffic information from other nodes. Nevertheless, all the nodes require information about the status of all WDM-related hardware and information about all established lightpaths. Additionally, for each reconfiguration of the network, all nodes have to participate in a decision process. Since this does not scale with increasing network size, it prevents the algorithm from reconfiguring networks quickly.

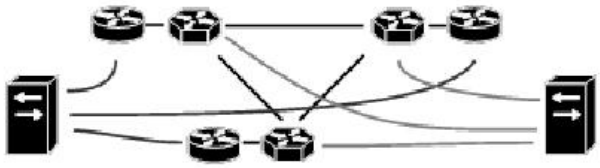

Fig. 1.OpenFlow (OF) over WDM Network. Solid lines represent optical fibres. ROADMs are controlled by a WDM controller and OpenFlow switches are controlled by a OpenFlow controller.

III. SOFTWARE DEFINED IP-OVER-WDM NETWORKS

A. Network Architecture

Software Defined Networking (SDN) has recently gained much attention in the networking community and is handled as one of the top candidates for future internet architectures. SDN enables switches to be remotely controlled by a centralized network controller that has global information about the status of the network. This global view can be used for deciding switching and routing. Today, network operators use a global view on their networks to perform traffic engineering. This global view is established by different protocols and traffic engineering is done mostly by Multiprotocol Label Switching (MPLS). Since all this (and much more) can also be done with SDN, we think that SDN will most probably be used in backbone networks in the near future. First deployments in large backbone networks have shown a dramatic increase in performance due to the use of SDN [7]. OpenFlow [8] is the first widely accepted protocol for SDNs that is supported by an increasing number of commodity hardware vendors.

We hence assume an “OpenFlow over WDM network” as the future backbone network architecture; OpenFlow is used to route IP traffic over the virtual topology of a WDM network (This means there is no additional IP routing happening on top of the OpenFlow routing). Fig. 1 shows a sample OpenFlow over WDM network consisting of 3 ROADMs and 3 OpenFlow-enabled switches as well as a WDM controller and an OpenFlow controller. The WDM controller decides wavelength conversion and lightpath switching at the ROADMs. Each ROADM has an OpenFlow-enabled switch attached. These switches are controlled by an OpenFlow controller. Based on the gathered information on the network (i.e., connected subnets), the OpenFlow controller takes over IP routing and controls the

switches appropriately. Depending on the lightpath configuration of the network, the switches see different network topologies while being attached to a static optical network. Thus, by thoughtful manipulation of the WDM lightpath configuration, the virtual topology can be adapted to meet traffic demands without changing any physical fiber.

B. WDM / SDN Assumptions

For the WDM network, we assume that every optical fiber is able to carry W wavelengths and that ROADMs have both heterogenous wavelength conversion and light switching capabilities. We do not impose the wavelength continuity constraint.

Switches are OpenFlow-enabled; each offers a fixed number of physical interfaces. Due to hardware limitations, each switch can switch traffic only up to a fixed data rate. In consequence, not all physical interfaces of a switch can be operated at full data rate in parallel. In addition, there are strict limitations on the total number of flow entries (Flow entries identify different traffic flows and specify where each flow is forwarded to). Due to this limitation a switch is able to handle only a fixed number of different flows at a time. The more flows it should handle, the more expensive it is.

C. Routing and Rerouting on IP level

Routing should maximize throughput while minimizing congestion. In OpenFlow, whenever a switch receives a packet without having a matching flow entry, a matching route is calculated and pushed to all switches along this route. For route calculation we use a Constrained Shortest Path First (CSPF) routing algorithm [9] that takes the residual bandwidth capacities of each link and the residual switching capacities of the switches into account. This is possible since in addition to global information about the virtual topology of the network,the OpenFlow controller collects information about each single switch and about the data rates on each link. To protect against traffic oscillations due to CSPF, the route of a flow does not adapt when residual link or backplane capacities change.

IP packets are routed over the VT of the WDM network; edges of the VT correspond to active lightpaths. Since these lightpaths can be removed, edges in the VT are deleted and all flows passing these edges are disrupted. To circumvent this, before a lightpath is removed, all affected flows are rerouted over other lightpaths. To implement such a rerouting feature, the OpenFlow controller has to keep track of all active flows. This is simple to do since the OpenFlow controller is informed whenever a new flow is created and when an old flow expires. When a lightpath is to be removed, first for each flow routed over this lightpath a new route is calculated by CSPF. Then, the network is updated by pushing the newly calculated flow entries to the switches. By doing so, the affected flows are seamlessly migrated to other routes. Since after updating the network, the lightpath to be deleted does not carry any more flows, it can be deleted without affecting the IP traffic.

To implement rerouting, the OpenFlow controller needs an interface for receiving rerouting requests. Note that rerouting without introducing congestion need not always be possible. This is why the interface can also be used to test whether or not it is possible to reroute flows without creating congestion. With the interface, the influence of deleting a lightpath on congestion can be determined before the actual network reconfiguration is triggered.

D. Network Reconfiguration in WDM Networks

Whenever a new lightpath is to be created, wavelengths on a path through the physical topology have to be allocated. For this task we use CR Dijkstra [10]. CR Dijkstra is a preemptive RWA algorithm that uses feedback for preemption decisions. On receiving a lightpath request that cannot be fulfilled due to resource limitations, multiple preemption candidates are calculated by CR Dijkstra. Such a candidate is defined as a minimum set of established lightpaths1 such that after removal of all lightpaths in the set, the newly requested lightpath can be created. Once CR Dijkstra calculated the sets for a lightpath request, these sets are passed to the requester of the new lightpath who can now choose which of it to preempt. It is also possible to cancel the new lightpath request in case no preemption seems advantageous.

Network reconfigurations are triggered by the OpenFlow-enabled switches by querying the WDM controller for a new edge in the VT. The WDM controller then runs the CR Dijkstra algorithm to create a corresponding lightpath. If during this process another lightpath l has to be preempted, first all traffic routed over l has to be rerouted over other lightpaths.

IV. SELFISH VIRTUAL TOPOLOGY RECONFIGURATION

A. Overview

In SVTR, each switch tries to selfishly reconfigure the network to meet its own traffic demands. To this end, each switch has a residual demand vector, indicating the amount of traffic originating in this switch that is destined for another switch but not routed so far.

Let Di,j be the rate at which switch i wants to send data to switch j at time t. Let Tit ,j be the rate at which switch i
actually sends data to switch j at time t. The residual traffic demand Rt for time t is defined as Rt = Dt −T t . Based on this matrix Rt , the switches request new lightpaths. The individual switches know only their own residual demand vector and not the whole matrix.

B. Rules

Selfish Virtual Topology Reconfiguration is distributed over the OpenFlow-enabled switches. It is based on three simple rules: Residual Demand Direct Link Establishment, Backplane Capacity

Load Relaxation and Idle Lightpath Removal. All three rules are executed periodically by each switch.

Residual Demand Direct Link Establishment tries to lower the residual demands by establishing lightpaths to all switches with positive residual demands. Therefore, if at time t there are two switches u and v such that Rt u,v > 0, then u requests a new lightpath from u to v by querying the WDM controller. After a lightpath is created, the corresponding traffic from u to v is routed and Ru+v1 , is lowered. In case existing lightpaths need to be preempted to create the new one, first all traffic passing these lightpaths is to be rerouted on other paths.

Backplane Capacity Load Relaxation (BCLR) is used to take load off the switches. Since switches have restricted backplane capacities, a selfish switch is interested in not being overloaded. To this end, each switch tries to minimize traffic it has to forward. This is done by creating lightpaths connecting two adjacent switches. Let nei(u) be the set of neighboring switches of u. If u is overloaded, it selects two switches v, w ∈ nei(u) such that the amount of traffic u forwards from v to w is maximized. Then, u requests a lightpath from v to w from the WDM controller. If lightpath creation was successful, the OpenFlow controller is queried to reroute all traffic passing the (sub-)path (u, v, w) on the newly created link (u, w). After that, the load on the backplane of v is lowered and the hop count for all rerouted flows is decremented by one.

Idle Lightpath Removal is used to free unused resources. We say a lightpath is underutilized if it carries data at less than a constant fraction TL of its bottleneck capacity. Whenever a switch detects that a lightpath is underutilized, the switch queries the OpenFlow controller to reroute all corresponding flows over other lightpaths. If rerouting succeeds, the lightpath is removed.

V. SIMULATION

A. Network

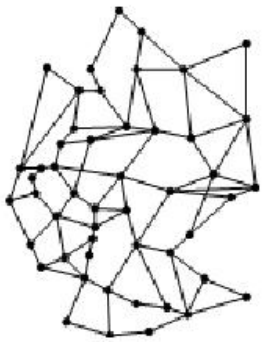

Fig. 2. Germany50 network with 50 nodes and 88 links.

For our simulations we use the germany50 network, depicted in Fig. 2. The network is considered to have a structure comparable to the German National Research and Education Network (DFN) [11]. We assume that every fiber provides 16 wavelengths, each operating at 10 Gbps, and that WDM equipment has full wavelength conversion capabilities and is able to switch optical signals from every input port to every output port. We do not use the real DFN network topology because the actual topology and traffic information in the DFN network are confidential. Thus, we use traffic information measured in the DFN network that were mapped to the germany50 network [11].

The core IP routers present at most locations of the DFN network are Cisco 7600 Series routers which we assume to switch 360 Gbps. The locations at Hannover (HAN) and Frankfurt (FRA) are equipped with Cisco CRS-1 systems offering 1200 Gbps switching capacity. Since we use traffic measured in the DFN network for our simulations, we adopt these values for the germany50 network.2

B. Traffic Model

We use IP traffic demands measured between the years 2004 and 2005 in the DFN network. From these measurements, the average traffic demand over a 24-hour period was calculated and afterwards mapped to the germany50 network. The traffic resolution is 5 minutes. To create different load levels we multiply the traffic demand matrices with a scalar ϱ.

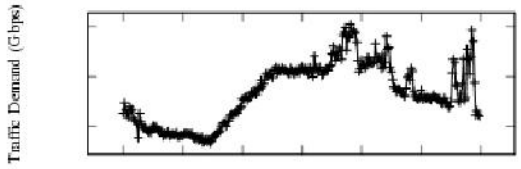

Fig. 3. Total traffic demand over a 24-hour period.

Amongst others, the DFN network connects various German universities with the internet. Since Frankfurt is DFN's main peering point with the rest of the internet, it is the bottleneck when scaling traffic. We do not have any information about the further destination of the traffic once it reaches Frankfurt. Thus, we cannot distinguish between traffic that is destined to a host in Frankfurt and traffic that is routed off the network. To get traffic demands for intra-network traffic only, we decided not to consider the traffic from and to Frankfurt in our simulation. The corresponding cumulative intra-network traffic is depicted in Fig. 3.

Our routing is flow-based but since we only have traffic demand matrices there is no information about individual traffic flows. To divide traffic into independent flows we assume each flow is on average of equal size (γ) and that the lifetime of a flow is Pareto-distributed with α = 0.95 [12].

C. Method

We compare SVTR to both the Residual Demand Hop-Count

Product (RDHP) heuristic [2] and the distributed algorithm by Shiomoto et. al [3]. RDHP is a well-studied VTD algorithm that creates lightpath topologies from scratch. The algorithm works by first creating lightpaths forming a spanning tree over the switches. Then, the algorithm iterates through the following three steps: a) route as much traffic as possible over the network without creating congestion; b) compute the residual demands for each pair of switches and multiply it with their hop count on IP level; c) in descending order of the residual demand-hop count product, try to establish lightpaths between switches; once a lightpath is established, continue at a; d) after no more lightpaths can be established, spend the free optical resources by establishing lightpaths randomly.

We also compared SVTR to the distributed algorithm designed by Shiomoto et al. [3]. Unfortunately, the algorithm yields poorly designed VTs under the assumptions used in this paper; see Fig. 4. Due to this shortcoming, the algorithm is left out in the further evaluation.

We measure the achieved throughput as the ratio of demanded traffic to routed traffic and create different load levels by multiplying the original measured traffic by a scalar $\varrho \in \{100, ..., 800\}$.

To obtain the best possible results, for each traffic matrix, the rule sets of SVTR have to be executed until no additional flows can be established. A discussion on the influence of the number of iterations to the achieved quality is given in the following section. For the Idle Lightpath Removal rule, we set TL to 0.01 in our simulations, only removing lightpaths that are very close to idle.

D. Simulation Results

Fig. 4 shows the achieved throughput over load levels. When scaling the original traffic by $\varrho = 800$, SVTR creates about 50% less congestion than RDHP. With increasing load the throughput gap between both algorithms increases. As in all scenarios SVTR generates a higher throughput than RDHP, it can be concluded that SVTR is able to react to traffic pattern changes and reconfigures the network appropriately. This is a very important result since it clearly shows that the idea of Selfish Virtual Topology Reconfiguration works in a realistic scenario.

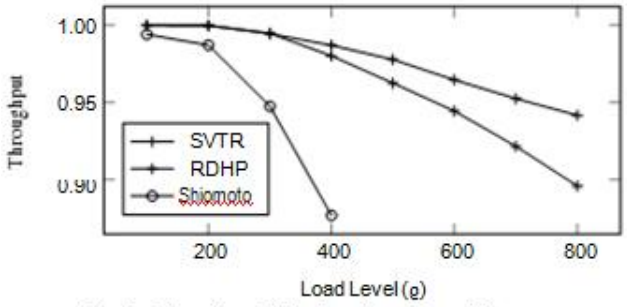

Fig. 4. Throughput with a flow size of $\gamma = 1$ Mbps.

Fig. 5 illustrates the physical interface usage of individual switches for both VTD strategies. It can be observed that in a low load scenario, RDHP needs fewer interfaces: for $\varrho = 100$ it yields a virtual topology with a 34% smaller maximum degree. But with increasing ϱ, the maximum number of physical interfaces in RDHP increases and for $\varrho \geq 200$ the maximum number of allocated physical interfaces is over 64 for RDHP. For SVTR, the maximum degree of the virtual topology seems to be independent of the load as it is between 38 and 48 in all considered load levels.

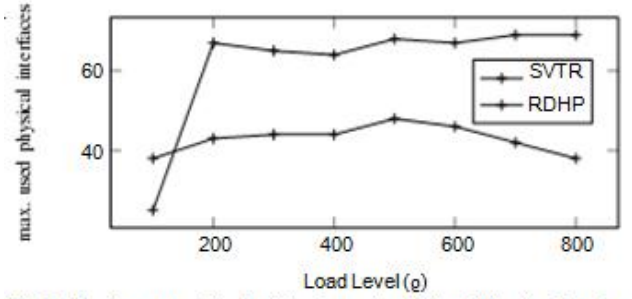

Fig. 5. Maximum used physical interfaces at a single switch with a flow size of $\gamma = 1$ Mbps.

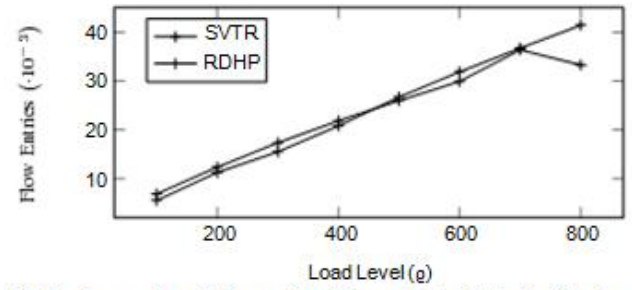

Fig. 6. Average FlowEntries per OpenFlow-enabled switch with a flow size of $\gamma = 1$ Mbps.

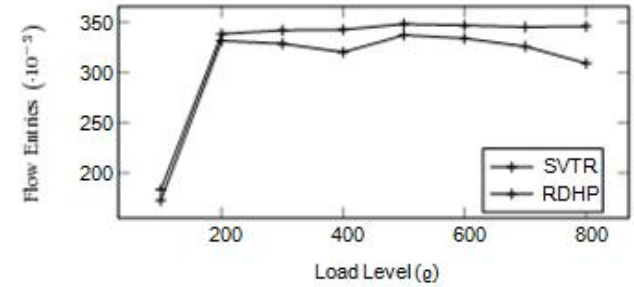

Fig. 7. Maximum FlowEntries used at a single OpenFlow-enabled switch with a flow size of $\gamma = 1$ Mbps.

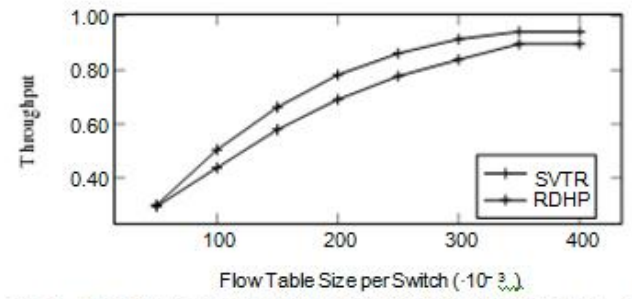

Fig. 8. Throughput over different Flow Size limits with a flow size of $\gamma = 1$ Mbps and $\varrho = 800$.

entries used to decide how to forward a flow. If the amount of concurrent flows routed through a switch exceeds the maximum number of flow entries, the performance is massively degraded because the switch has no possibility to hold the controller's decisions in its cache. Fig. 8 shows what happens if the number

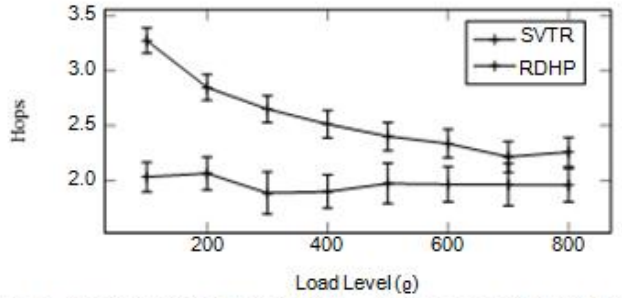

Fig. 9. Hop Count with a flow size of $\gamma = 1$ Mbps. Bars illustrate the confidence interval for a confidence coefficient of 95%.

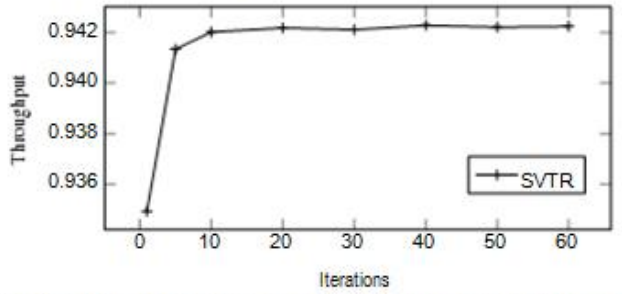

Fig. 10. Achieved Throughput over Iterations with a flow size of $\gamma = 1$ Mbps and $\varrho = 800$.

of flow table entries is exceeded. For both VTD algorithms, the routing algorithm (CSPF) used on IP level included flow table limitations as a constraint. The achieved throughput is plotted over flow table size. Obviously, the fewer flow entries are allowed, the worse the overall throughput is. SVTR is still superior to RDHP when limiting the maximum possible flows per switch.

Fig. 9 shows the average hop count over different load levels where the hop count is the number of lightpaths a flow is routed over. The average hop count in a virtual topology designed by SVTR is—independent of the considered load level—very close to 2 hops, while for RDHP with increasing load the hop count decreases from 3.3 hops at $\varrho = 100$ to 2.2 hops at $\varrho = 700$. Thus, average end-to-end delay in SVTR is comparable or even better than end-to-end delay in a virtual topology built by RDHP. The shape of RDHP's curve is due to the way RDHP constructs its virtual topology. Since RDHP starts with a spanning tree and creates edges between nodes with high traffic demands, nodes with low traffic demands are potentially connected by only a few edges. This leads to a high network diameter and to a high hop count.

The applicability of an iterative algorithm in practice strongly depends on the number of iterations necessary for the algorithm to converge. We fixed $\varrho = 800$, set $\gamma = 1$ Mbps and varied the number of iterations the algorithm executes in a single 5-minute period. Fig. 10 depicts the outcome of this experiment. We can conclude that SVTR requires only a very small number of iterations to reconfigure the network to meet the experienced traffic requirements. By increasing the number of iterations from 5 to 10 in a 5-minute period, the achieved throughput increases only by less than 0.1%. Even more iterations have no noteworthy impact on achieved throughput. We can conclude that SVTR is very applicable in practice where only a small number of iterations seems feasible.

We could show that under our realistic network model SVTR clearly outperforms both other algorithms in throughput while offering at least the same performance in any other metric. To the best of our knowledge, the distributed VTD algorithm stated in this paper is the only one that is able to reconfigure the VT of large backbone networks quickly in an iterative manner.

VI. CONCLUSION

We proposed a simple, rule-based reconfiguration system for a backbone network employing OpenFlow for an IP-over-WDM network as a possible future internet architecture. The combination of SDN together with a reconfigurable optical network makes the network highly flexible. This flexibility creates new opportunities to support a very large set of different traffic patterns without any human interaction, leading to a high-speed, self-adaptive network. A network equipped with this technology is able to transparently reconfigure very quickly from one topology to another without noticeable performance degradation during the transformation. It has no need for multilayer optimization processes and promises a much higher utilization than present algorithms. Additionally, this work showed how to incorporate manipulation of Wavelength Division Multiplexing optical networks into Software Defined Networking.

REFERENCES

[1] R. Dutta and G. Rouskas, "A survey of virtual topology design algorithms for wavelength routed optical networks," Optical Networks Magazine,vol. 1, 2000.
[2] K. H. Liu, C. Liu, J. L. Pastor, A. Roy, and J. Y. Wei, "Performance and Testbed Study of Topology Reconfiguration in IP Over Optical Networks," in Proceedings of the IEEE Transactions on Communications, 2002.
[3] K. Shiomoto, E. Oki, W. Imajuku, S. Okamoto, and N. Yamanaka, "Distributed virtual network topology control mechanism in gmplsbased multiregion networks," IEEE Journal on Selected Areas in Communications, 2003.
[4] J. Wu, "A survey of wdm network reconfiguration: Strategies and triggering methods." Computer Networks, vol. 55, no. 11, 2011.
[5] Q.-D. Ho and M.-S. Lee, "Connection level active rerouting in wdm mesh networks with traffic grooming capability," in Proceedings of the IEEE International Conference on Communications (ICC), 2006.

[6] I. Widjaja, I. Saniee, L. Qian, A. Elwalid, J. Ellson, and L.Cheng, "A new approach for automatic grooming of SONET circuits to optical express links," in Proceedings of the IEEE International Conference on Communications (ICC), 2003.
[7] U. Hoelzle, "Openflow@Google," Keynote at OpenNetSummit; Youtube video, URL: http://youtube.com/watch?v=JMkvCBOMhno, 2012.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," SIGCOMM, vol. 38, 2008.
[9] M. KB, "Constrained Shortest Path First," IETF, Internet Draft <draftmanayya-constrained-shortest-path-first-02.txt>, Feb. 2010.
[10] P. Wette and H. Karl, "Introducing Feedback to preemptive Routing and Wavelength Assignment algorithms for dynamic Traffic scenarios," University of Paderborn, Tech. Rep., 2012.
[11] "SNDlib." [Online]. Available: http://sndlib.zib.de/
[12] J. Charzinski, "Internet client traffic measurement and characterization results," in Proceedings of the 13th International Symposium on Services and Local Access, 2000.