QoS-Aware Virtual SDN Network Planning

Junchao Wang
University of Amsterdam, NDSC
Email: j.wang2@uva.nl

Cees de Laat
University of Amsterdam
Email: delaat@uva.nl

Zhiming Zhao
University of Amsterdam
Email: z.zhao@uva.nl

***Abstract*—Software Defined Networking (SDN) technologies provide applications opportunities to manipulate underlying network flows and topologies via network controllers during runtime. In cloud environments, networked virtual machines can be enhanced by SDN by providing applications with controllable infrastructures to meet system-level quality requirements; however, customizing a suitable network topology with optimally placed controller(s) for given quality requirements and workload characteristics is often not an easy task. We call such problem *virtual SDN network planning problem*. In this paper, a Topology-Controller planner (TCPlanner) is proposed for customizing the network topology and placing the controllers. Experiments with different scales of network show that our approach can effectively plan virtual SDN networks to meet the various QoS requirements and reduce costs.**

I. Introduction

By decoupling the control plane from the data plane, Software-Defined Networking (SDN) technologies allow administrators or applications to manipulate the underlying network behavior via open interfaces [1][2]. SDN-based standards, e.g. Network Service Interface (NSI) [3] and Openflow [4], have shown great impact on dynamic provisioning and reconfiguration in lightpaths and data center networks in physical infrastructures [1]. In cloud environments, SDN-based virtual switches [5] can be used together with networked virtual machines (VMs) to allow applications to dynamically adjust network flows via open interface and to maintain the system-level performance [6].

At an abstract level, designing a topology of virtual network devices, and placing suitable number of controllers are two key issues of designing a SDN network in a Cloud environment. When an application is distributed and with a high quality requirement such as communication latency, designing a suitable SDN network can be very difficult. Mapping application-level quality constraints onto network-level properties, e.g., topology, is not straightforward, in particular when the application has different requirements needing to be considered. The design of the virtual network should also take non-functional requirements, such as cost and reliability into account. To make the virtual network software definable, one or more controllers are needed, and they can reside in the same VM with the virtual network devices or on a separate VM. Unnecessarily high numbers of controllers can not only make the resource cost high, but also increase the control complexity. Moreover, the placement of the controllers can also influence the control latency between controller and devices; when the application has critical time requirements, limiting such latency can also be crucial.

In this paper, we formulate this problem as the *virtual SDN network planning problem*, and propose a Topology-Controller planner (TCPlanner) to solve the problem. We will first review the related work, and then give a formal problem description, after which we present our proposed solution for network topology customization and controller placement, and discuss experimental results.

II. Related works

In recent years, network topology optimization and SDN controller placement have attracted lots of research attention.

The problem of network topology customization is often studied using optimization approaches. In [7], Gódor et al. proposed a heuristic algorithm which combines clustering and local optimisation operators to optimise the cost of hierarchical network planning. The costs of the network are aggregated together from level to level with the degree constraints. In [8], Eric tries to design a network topology with the minimum number of links under the constraints of network diameter, degree and survivability. Eric conducted theoretical analysis on the problem and proposed a method with a mathematical model. However, detailed information about the algorithm is not given. In this paper we consider similar Quality-of-Service (QoS) requirements as [8] and propose a meta-heuristic approach. In [9], Kamiyama et al. targets at designing a network topology which can guarantee the connectivity and total link length. As the search space enumerating all the possibilities of links is too large, they applied binary partitioning and introduce extra constraints to reduce the search space. In [10] the maximum entropy method (MEM) is applied to solve the problem of network design with the objective to minimise the cost under the constraint of link capacity and latency. In [11], to solve the problem of network topology design with the objective of fault tolerance and capacity of traffic and delays, a genetic algorithm is applied. In summary, only Eric [8] considered the QoS of network reliability and network diameter constraints and presented theoretical analysis when customizing network topology. However, a detailed description of the solution is not given.

The controller placement problem was first addressed in [12]. The placement metrics average-case latency and worst case latency are still widely used in current studies [13], [14], [15]. Pareto-based Optimal COntroller placement(POCO) [13] [14] is a framework for Pareto optimal

978-3-901882-89-0 @2017 IFIP

controller placement in terms of different performance metrics. The controller-to-switch and controller-to-controller latency are considered to measure the network resilience. In [15], Cheng et al. proposed three heuristic algorithms to solve the problem of QoS-guaranteed controller placement. The algorithms are more concerned with how to partition the network from the controller viewpoint. However, these existing works assume the number of controllers is assumed to be given. In the SDN network planning problem, this number is not known before. So we propose a solution that can determine the placement of controllers and the number of controllers needed.

From the existing work, we can see most of the topology customization work study physical networks, without considering the SDN aspects; and the SDN placement studies mainly focus on the pre-defined physical networks. In cloud environments, combing these two perspectives are clearly needed.

III. Virtual SDN network planning problem

The virtual SDN network planning problem is to customise a network topology and place the controllers that can meet the given QoS requirements.

As discussed above, the VMs provided by the cloud can act as the switches. Network topology customisation determines how these virtual switches are connected. We assume that the users specify the number of virtual network devices (routers or switches) as N and QoS requirements (network diameter and reliability). The network diameter d is the communication cost of the longest path between all the pairs in the graph. It can reflect the worst end-to-end latency in the network [8]. Therefore, we consider the network diameter specified by the user as the one of the QoS requirements. Due to the dynamics of cloud, the virtual links between VMs can fail or degrade occassionally [16]. Therefore, the reliability in the virtual network topology customisation is another important issue which should be considered. In this paper we use single arc survivability to represent the reliability of the network. Single arc survivability means that when a single link in the network topology fails, the network is still connected. There are limited number of ports in network devices even though it is virtual instead of physical. The cloud provider may also limit the number of links that a VM is able to connect due to the limitations of physical infrastructures. More specifically, Δ is the maximum number of links from any given VM. The network topology customisation problem is to define a network topology that has single-arc survival with network diameter no greater than d and node degree no larger than Δ. The overall objective of the network topology customisation is to design a network topology with the minimum number of network links within the constraints described above.

The controller placement problem is to determine the number of controllers and places where controllers should be deployed. We assume that the controllers can manage the same number of virtual switches and the controllers can be placed in the same place as the virtual switch. The controller-to-controller and controller-to-switch communication are also enabled through the virtual network links in the network topology planning phase so that no extra links need to be re-planned. The controller-to-controller latency and controller-to-switch latency are two typical QoS requirements when placing controllers [17]. In this paper we use $\pi_C^{maxlatency}$ and $\pi_S^{maxlatency}$ to represent the maximum permitted controller-to-controller latency and controller-to-switch latency. We $\pi_S^{avglatency}$ to represent the average controller-to-switch latency. $\pi_S^{avglatency}$ is quite crucial to SDN because the controller needs to communicate frequently with the switches [4]. Thus, in this paper we try to minimise the number of controllers and $\pi_S^{avglatency}$ within the constraints of $\pi_C^{maxlatency}$ and $\pi_S^{maxlatency}$.

IV. Proposed approach: Topology-Controller planner

An approach called Topology-Controller planner (TCPlanner) is proposed to solve the virtual SDN network planning problem. The TCPlanner first customizes the network topology to meet the high level requirements, which can be given by the network developer or applications, and then places the optimal number of controllers within the planned topology.

A. From application QoS to network topology

TCPlanner plans a topology to connect virtual network devices based on network diameter and reliability. We use d' to represent the maximum end-to-end latency tolerable for users. Thus, to guarantee the d' and $\pi_C^{maxlatency}$, we set $d = max\{d', \pi_C^{maxlatency}\}$.

Such a problem can be viewed as a transformation of the Minimum-Cardinality-Bounded-Diameter(MCBD) Edge Addition Problem and has already been proved to be NP-hard[18][19]. Theoretically, there exists a brute-force algorithm that solves the problem by iterating through all feasible solutions. In cloud, the virtual links can be planned between any pair in the virtual network. Thus, there exists $N \times (N-1)/2$ links. Then the scale of searching space can be $2^{((N\times(N-1)))}$. This is possible for small-scale graphs, but it becomes computationally very expensive or even impossible when N is very large. A meta-heuristic approach based on evolutionary algorithms is adopted in TCPlanner, because evolutionary algorithms have been demonstrated as a feasible solution for several similar problems[20].

We model the network connectivity using a communication matrix and assume the links between nodes are not directed; the communication matrix is thus symmetric. We encode a solution to a chromosome with length of $\frac{N \times (N-1)}{2}$. Each element in the chromosome is 1 or 0, which indicates whether a link exists or not between vertices. Correspondingly we also design a decoding algorithm to decode the chromosome as a graph. The initial population can be seen as the "seed" of the initial state which can have great effect on the performance of the GA. Usually the initial population can be heuristically crafted or randomly generated. It is difficult to follow certain

heuristics to create the initial population, so all the individuals in initial population are randomly generated. The assignment of each position has equal possibilities.

Due to the complex various situations in which the constraints described above can be violated, we add a penalty factor for each violation and aggregate them with the number of links into the fitness function. As the objective is to minimise the number of links, we use the reciprocal of the sum of the link number and penalties as the fitness function which is calculated as:

$$1.0/(LinkNum + \sum_{i=1}^{k} x_i \times penalty_i)$$

$$x_i = \begin{cases} 0 & \text{The i-th constraint is not violated} \\ 1 & \text{The i-th constraint is violated} \end{cases}$$

$LinkNum$ represents the number of links in the planned network topology. $penalty_i$ represents the extent of the violation of the i-th constraint. The penalties above are to avoid unfeasible solutions like unconnected graphs or graphs that violate the specified constraints. In our scenario, there are four possible situations where the constraints can be violated: diameter violation, connectivity violation, degree violation and survivability violation.

We use genetic operators crossover and mutation to produce new generations of individuals and introduce diversity. We set the probability of crossover between two parents as a static value p_1 for each generation so that in each iteration new chromosomes will be produced by intersecting the parent's chromosomes with a certain probability p_2. After the offsprings are generated, $p_3 \times PopSize$ individuals are mutated by switching certain places in their chromosomes from 1 to 0 or 0 to 1. $PopSize$ refers to the population size. p_3 represents the probability of mutation of individuals. Then the next generation is selected with the individuals of the best fitness value and the population remains the same size as last generation.

B. SDN controller placement

After planning the topology, TCPlanner will determine the number and placement location of the SDN controllers. The objective is to minimise the number of controllers and average controller-to-switch latency under the constraints of the capacity of controllers and maximum latency of controller-to-switch[17].

In TCPlanner, we sort the degree of the nodes in the planned network topology in descending order and first choose the vertex v with the maximum degree as the center of the first cluster. The higher degree a vertice has, the more chances the average latency can be reduced when looking at all its neighbours one step further away. At each level neighbours of the center node, we first choose the node with the minimum degree so that it minimises the interference on other clusters. The v tries to "absorb" its neighbours in this way until the controller capacity or the maximum controller-to-switch latency is violated. Such process will continue until all the nodes are assigned to a cluster. In each cluster, its center node is the place where controllers should be placed. The number of controllers is equal to the number of clusters.

V. Performance characteristics

To test the effectiveness of TCPlanner, we compared it with a K-Medoids-based solution as the baseline [21]. The K-Medoids algorithm is intended to classify a data set into several clusters based on the node distance. The basic process of K-Medoids is to randomly initialise K centers of clusters and add nodes to the clusters based on the distance between the center node and non-clustered node. Then the algorithm will try to calculate some centers to reduce the inter-cluster and intra-cluster distance. The algorithm will converge when no better centers can be found.

As the K-Medoids algorithm need to specify the number of clusters before the execution of the algorithm, we use the square root of the number of nodes as the initial number of clusters. When a cluster in the solution given by K-Medoids algorithm exceeds the capacity of the controller or the maximum controller-to-switch latency is violated, we increase the number of clusters by 1.

In this paper we conduct simulated experiments on different scales of networks to test the effectiveness of the proposed solution. Our solution is implemented in Python and depends on NetworkX and DEAP (Distributed Evolutionary Algorithms in Python) [22].

A. Planning network topology

We set the number of network devices N ranging from 6 to 25. The diameter of the network is set as $\lceil\sqrt{N}\rceil$. The maximum degree of the network devices is $d+1$. d represents the diameter of the network. We set the maximum generation number of the genetic algorithm to be 250 to ensure that a feasible solution can be found.

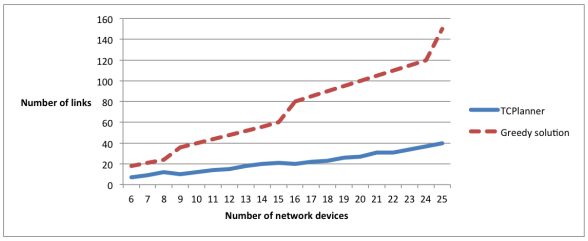


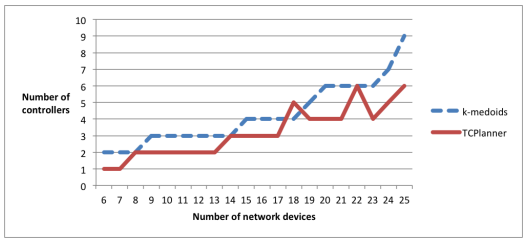
Fig. 1: Number of links for a network topology with certain QoS requirements

From Fig. 1 we can see that the number of links needed to guarantee the QoS increases with the number of network devices which is roughly linearly. In order to evaluate the performance of TCPlanner, we design a greedy algorithm which utilises all the degrees of each port. Therefore, the number of links that can meet the QoS requirements reaches $N \times \Delta$. From the result we can see that TCPlanner outperforms the greedy solution.

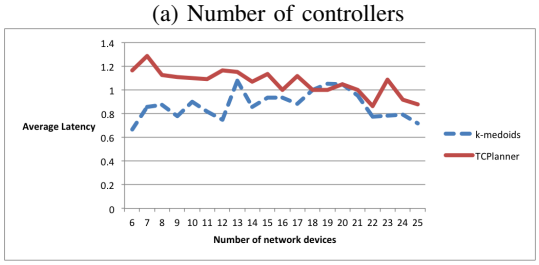
B. Controller placement

We take the network topology generated from the data plane planning phase and compare the results of K-Medoids and TCPlanner from the number of controllers and average

controller-to-switch latency. The results are shown in Fig. 2. From the result we can see that the K-Medoids-based solution needs more controllers than TCPlanner but can reduce the average latency. The result is reasonable because K-Medoids-based solution tries to cluster the graph into clusters to minimise the intra-class distance and inter-class distance. With the increase of the number of network devices, the number of needed controllers can be reduced dramatically. When the scale of the topology reaches 25, TCPlanner can have 3 less controllers than the K-Medoids-based solution.



(a) Number of controllers



(b) Average controller-to-switch latency

Fig. 2: Results of K-Medoids and TCPlanner

VI. Conclusion and Future Works

In this paper we study the problem of virtual SDN network planning and present a TCPlanner to solve the problem. We first design the network topology given the QoS requirements on network performance and reliability. Then we place the controllers that can meet the controller-to-controller and controller-to-switch latency.

We only consider the latency constraints in the current prototype; other QoS attributes such as bandwidth can also have great impact on the performance of the network. One of our future work is thus to include more network QoS constraints in the planning process. Moreover, the characteristics of application traffic patterns and the dynamic QoS control of SDN network will also be investigated in the planning algorithm.

Acknowledgements

This research has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements 643963 (SWITCH project), 654182 (ENVRIPLUS project) and 676247 (VRE4EIC project). The research is also partially funded by the COMMIT project.

References

[1] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] F. Ongaro, E. Cerqueira, L. Foschini, A. Corradi, and M. Gerla, "Enhancing the quality level support for real-time multimedia applications in software-defined networks," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pp. 505–509, IEEE, 2015.

[3] G. Roberts, T. Kudoh, I. Monga, J. Sobieski, and J. Vollbrecht, "Network services framework v1.0," Tech. Rep. GFD 173, 2010.

[4] O. S. Specification-Version, "1.4.0," 2013.

[5] "Openvswitch." Accessed: 2016-3-30.

[6] K. Jeong and R. Figueiredo, "Self-configuring software-defined overlay bypass for seamless inter-and intra-cloud virtual networking," in *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, pp. 153–164, ACM, 2016.

[7] I. Gódor and G. Magyar, "Cost-optimal topology planning of hierarchical access networks," *Computers & operations research*, vol. 32, no. 1, pp. 59–86, 2005.

[8] E. Rosenberg, "Hierarchical topological network design," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 6, pp. 1402–1409, 2005.

[9] N. Kamiyama, "Efficiently constructing candidate set for network topology design," in *Communications, 2009. ICC'09. IEEE International Conference on*, pp. 1–6, IEEE, 2009.

[10] M. Tuba, "An algorithm for the network design problem based on the maximum entropy method," in *Proceedings of the American Conference on Applied Mathematics, Cambridge, USA*, pp. 206–211, 2010.

[11] T. Fencl, P. Burget, and J. Bilek, "Network topology design," *Control Engineering Practice*, vol. 19, no. 11, pp. 1287–1296, 2011.

[12] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 7–12, ACM, 2012.

[13] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 4–17, 2015.

[14] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, "Poco-framework for pareto-optimal resilient controller placement in sdn-based core networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pp. 1–2, IEEE, 2014.

[15] T. Y. Cheng, M. Wang, and X. Jia, "Qos-guaranteed controller placement in sdn," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2015.

[16] K. Hwang, X. Bai, Y. Shi, M. Li, W.-G. Chen, and Y. Wu, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 130–143, 2016.

[17] S. A. Shah, J. Faiz, M. Farooq, A. Shafi, and S. A. Mehdi, "An architectural evaluation of sdn controllers," in *Communications (ICC), 2013 IEEE International Conference on*, pp. 3504–3508, IEEE, 2013.

[18] C.-L. Li, S. T. Mccormick, and D. Simchi-Levi, "On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems," *Operations Research Letters*, vol. 11, no. 5, pp. 303–308, 1992.

[19] M. Abd-El-Barr, "Topological network design: A survey," *Journal of Network and Computer Applications*, vol. 32, no. 3, pp. 501–509, 2009.

[20] C.-W. Tsai and J. J. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *Systems Journal, IEEE*, vol. 8, no. 1, pp. 279–291, 2014.

[21] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[22] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.