PREPRINT

Flexible distributed control plane deployment

Shaoteng Liu
shaoteng.liu@ri.se
RISE SICS

Rebecca Steinert
rebecca.steinert@ri.se
RISE SICS

Dejan Kostić
dejan.kostic@ri.se
RISE SICS/KTH Royal Institute of Technology

Abstract— For large-scale programmable networks, flexible deployment of distributed control planes is essential for service availability and performance. However, existing approaches only focus on placing controllers whereas the consequent control traffic is often ignored. In this paper, we propose a black-box optimization framework offering the additional steps for quantifying the effect of the consequent control traffic when deploying a distributed control plane. Evaluating different implementations of the framework over real-world topologies shows that close to optimal solutions can be achieved. Moreover, experiments indicate that running a method for controller placement without considering the control traffic, cause excessive bandwidth usage (worst cases varying between 20.1%-50.1% more) and congestion, compared to our approach.

I. INTRODUCTION

The early definition of the control plane in a software-defined network (SDN) setting assumes that one controller handles flow requests over a set of associated switches. For the purpose of improving reliability and scalability, more recent solutions assume a distributed control plane, which consists of multiple physically distributed but logically centralized control instances. Deploying multiple control instances can help to decrease the control latency, prevent a single controller from overloading; and, tolerate controller failures.

However, as the number of deployed control instances increases, there is a significant risk that the consequent inter-controller traffic grows into an unacceptable overhead. Regardless of the consistency level (strong vs. eventual), updating shared state at one of the n controllers intuitively requires a one to many style communication to update the $n - 1$ remaining instances. Observations and investigations in [1], [2], [3], [4] confirm this dramatic increase in the communication overhead for maintaining shared state among controllers.

In existing controller placement approaches, the importance of considering the control plane traffic as part of the solution is usually overlooked. Dealing with the traffic associated with a certain control plane deployment is typically ignored, although control plane traffic flows have to be forwarded timely and reliably through a network infrastructure with varying link capacities, availability, and other networking properties. Control traffic congestion, for example, is especially destructive since it may degrade control service performance, or worse, cause availability issues - the latter cannot be tolerated in e.g., services critical to human safety.

In this paper, we advance the current state of the art by: 1) proposing a novel formalization of the problem of distributed control plane optimization, enabling 2) tuning of reliability and bandwidth requirements (i.e., routability). Essentially, by

analyzing the challenges and complexity of the controller placement and traffic routability problem, we introduce a generic black-box optimization process formulated as a feasibility problem. We specify each step of the process along with guiding implementation examples. Unlike existing approaches, our optimization process adds the extra steps needed for quantifying the consequences of deploying a control plane solution that fulfills the specified reliability and bandwidth requirements (Sections V-F). As a powerful prediction tool, the approach can be used by service providers and operators to fine-tune control plane deployment policies. In combination with the generic design of the black-box optimization process, virtually any existing method can be plugged in and used for control plane optimization and management.

II. RELATED WORK

Existing works on control plane scalability generally focus on aspects such as switch-to-controller or controller-to-controller delay reduction [5], [6], [7]; controller capacity and utilization optimization [8]; flow setup time and communication overhead minimization [9], etc. Compared to delay or load, reliability or failure resilience is harder to compute, often requiring sophisticated modeling. Examples of prior work with reliability models or measurements include [10], [11], [12], [13], which instead use intuitive objective functions to obtain a placement solution. However, they do not provide an estimate of the achieved reliability. In contrast, the authors in [14], [15] designed a way to estimate the network reliability in polynomial time and provide a lower bound of the actual reliability. Additionally, the authors have proposed a heuristic algorithm to decide the number of controllers and their locations, in order to guarantee a certain reliability requirement.

None of the aforementioned prior approaches address control traffic routability. A scalability optimization model for placement which have constraints on traffic demands and link bandwidths has been proposed in [16], albeit in a simplified context by assuming 1) there is exactly one dedicated one-hop link between each switch and each controller; 2) the control traffic between a switch and a controller is always routed on that dedicated link; and, 3) no inter-control traffic. In comparison, our approach can be applied to any network topology and can deal with any control traffic pattern, while quantifying reliability and bandwidth requirements associated with a certain control plane solution.

III. BACKGROUND AND MOTIVATION

Figure 1a illustrates two typical cases of the distributed control plane of a programmable network. The term *aggregator* here represents either an OpenFlow switch in the context of

(a) Schemes of a distributed control plane
 Fig. 1. The schemes of a distributed control plane (a), and the general steps in the approach (b).

software-defined networks (SDN), or a radio access point in a software-defined radio access network (SoftRAN) context. In any case, the aggregator acts as a data forwarding device. The term *controller* refers to a distributed control instance. The out-of-band control setting is shown on the left of Figure 1a. In this case, all the controllers are connected with dedicated networks and running in dedicated nodes, e.g., running all the controllers in a remote cloud infrastructure. The in-band control case, where both control and data traffic share the same network, is illustrated on the right of Figure 1a. A control instance can in this case be co-located with an aggregator in one node. In both cases, the control of the aggregators is distributed over two controllers, c_1 and c_2 (Figure 1a).

Coordinating distributed controllers, appearing as one single logical control entity, requires that system events and network state information are shared between the controllers with a certain level of consistency. In general, the behavior of such inter-controller traffic depends on the control application and varies with the size and intensity of information transactions, number of controllers, as well as the consistency model. Hence, inter-controller traffic can become potentially expensive in terms of communication overhead in addition to control messages. For example, as observed in the evaluations of [1], a single update of shared information can generate $4n$ transactions in the control plane, where n is the number of the controllers. This finding confirms our intuition behind the required amount of communication: 1) that it increases with the number of controller instances, and 2) it is a source of considerable overhead.

IV. OVERVIEW OF THE APPROACH

A. Problem formulation and challenges

Control plane management here refers to the planning of the controller placement and associated control traffic in the distributed control plane. There are two major challenges: First, the control instances must be placed in a way to satisfy the given constraints related to e.g. reliability and scalability. This includes decisions on how many control instances to use, where to place them and how to define their control regions. The controller placement problem in general is NP-hard [5]. To solve the problem, existing work [5], [6], [7], [17], [18],

[19], [20], [12] in general resort to heuristics to reduce the search space.

Second, we must verify that the control traffic introduced by a placement solution can be scheduled on the underlying network without overloading any link. Such verification can be modelled as a multi-commodity flow problem [21]. Depending on the underlying routing mechanisms of the infrastructure, if flows can be splittable [22], then the problem can be formulated as a Linear Programming (LP) problem; otherwise [23], it is a Mixed Integer Linear Programming (MILP) problem, which is known to be NP-hard. Moreover, the number of decision variables inside the problem increases exponentially with the topology size. Thus, even if it is a LP, it is still challenging to solve it in polynomial time [24].

B. The proposed approach

Our approach addressing the aforementioned challenges is centered around an optimization process aimed at providing a feasible solution by the execution of four steps, as Figure 1b suggests. Next, we outline each step of the process:

a) The **mapping** step places controllers on a given network topology. The input here contains (but is not limited to) network topology including link bandwidths, as well as the constraints on the placement, such as reliability. The output is a controller location map. b) The **association** step associates aggregators to controllers. The input is the controller location map. The output is an aggregator-controller association plan. c) The **traffic estimation** step outputs the demand of each control flow according to the input aggregator-controller association plan, assuming a known traffic model (see an example in Section V-D). d) The **routability check** step outputs a decision variable which indicates whether all the control flows can be scheduled or not. The input is the network topology properties and the control flows.

The process of finding a feasible solution satisfying all conditions includes iteration over the four steps until the end condition is met, which means the limit of iterations is reached or the solution is found. A user-defined cost function is used for specifying the behavior of the optimization process when executing the mapping or association step (see Section V-B and Section V-C).

Note, that the process is generic and can be extended to include other (single or multiple) requirements related e.g.

to load balancing and response delays, by adding proper constraints in the mapping and association step along with a cost function and end conditions. In principle, each step can be viewed as a black-box implementation matching the input and output at each step as previously defined. In the following section, we will exemplify how each step can be implemented to address aforementioned challenges and solve a control plane management problem. We also propose heuristic algorithms for mapping and association.

V. CONTROL PLANE OPTIMIZATION

According to [25], the reliability of a system is defined as the probability that the system operates without failure in the interval $[0, t]$, given that the system was performing correctly at time 0. In this section, we describe our optimization approach that targets at *service reliability*. The service reliability refers to the minimum reliability among all the aggregators (noted as $Rmin$). Here, the reliability of an aggregator corresponds to the probability that an operational aggregator is connected to at least one operational controller during the interval.

In the following parts of this section, we first formulate the control plane management problem as a feasibility problem. Then, we show how a solver for the feasibility problem can be implemented in line with the proposed optimization process. In Section V-F we demonstrate how to optimize different objective functions by using the solver. We will show that we can maximize the $Rmin$, given fixed link bandwidth. Also, we can minimize the link bandwidth, given $Rmin$ should be guaranteed and higher than a certain threshold, which is referred as the *reliability threshold* and noted as β .

A. Feasibility problem formulation

Let $G(V = N \cup M, E)$ be a graph representing a network topology, where V and E denote nodes and links, respectively. Moreover, let N denote the set of nodes holding aggregators and M a candidate set of nodes eligible for hosting controller instances. Further, each aggregator $n \in N$ and each control instance have a given probability of being operational, denoted by p_n and p_c , respectively. Analogously, links $(u, v) \in E$ are operational with probability $p_{u,v}$. We assume different i.i.d. operational probabilities for links, nodes, and controllers. Note, that this probability can be set based on expert knowledge or inferred by learning about the system performance.

We use binary variables y_i , where $y_i = 1$ if node $i \in M$ hosts a controller, and $y_i = 0$ otherwise. Let us define $C = \{i | y_i = 1, i \in M\}$ denotes the set of deployed controllers. Let binary variable $a_{ij} = 1$ if aggregator $j \in N$ is controlled by the controller in $i \in C$, otherwise $a_{ij} = 0$. Although each aggregator j can only be controlled by one controller at a time, it can have multiple backup controllers (e.g., with Openflow V1.2 protocol [26]). The reliability of node j is represented as $R(G, j, C)$ (among $|C|$ controllers), capturing the probability of node j connecting with at least one of the operational controllers. Solutions satisfying the constraints

given topological conditions and reliability threshold β are found by $Rmin = \min(R(G, j, C), \forall j \in N) > \beta$.

For the traffic routability problem in programmable networks, we can formulate it as a multi-commodity flow problem [27] by taking flow splitting into account [22]. Let u_e be the reserved bandwidth capacity on each link $e \in E$ for control plane traffic. Suppose (s_f, t_f) being the (*source, sink*) of control traffic flow f . Let d_f denotes the demand (throughput) of f . Let $F = \{f = (s_f, t_f, d_f)\}$ be the set of all the control traffic. Let $F_c \subset F$ be the inter-controller traffic that $F_c = \{f = (s_f, t_f, d_f) | s_f \in C, t_f \in C\}$. Let κ_f denote all the possible non-loop paths for $f \in F$, and let $\kappa = \cup_f \kappa_f$. Let variable $X(K)$ denote the amount of flow sent along path $K, \forall K \in \kappa$. Then, the reliable control plane management problem can be formulated as follows:

$$\begin{aligned} & \text{maximize} && 0 \\ & \text{s.t. :} && \sum_{i \in C} a_{ij} = 1, \forall j \in N \end{aligned} \quad (1)$$

$$\sum_{i \in M} y_i \geq 1 \quad (2)$$

$$R(G, j, C) \geq \beta, \forall j \in N \quad (3)$$

$$\sum_{K \in \kappa_f} X(K) \geq d_f, \forall f \in F \quad (4)$$

$$\sum_{K: e \ni K} X(K) \leq u_e, \forall e \in E \quad (5)$$

$$y_i, a_{ij} \in \{0, 1\} \quad (6)$$

$$X(K) \geq 0, \forall K \in \kappa \quad (7)$$

Note that the above formulation of the control plane management problem is general, and covers both in-band and out-of-band control cases. For example, $M \subseteq N$, corresponds to an in-band control plane problem formulation, whereas $N \cap M = \phi$, corresponds to the out-of-band case¹. Although the out-of-band case may additionally require that the paths for the inter-controller traffic F_c should be limited within the control network, this limitation has already been implicitly included in the definition of the set κ_f . The κ_f is defined as the set of all the possible paths for a certain flow f . A possible path for flow $f \in F_c$ in the out-of-band case can only be selected among links belonging to the control network.

The main difference between this formulation and the traditional reliable controller placement problem [15] is that we model the control plane management as a feasibility problem without an optimization objective. The feasibility problem formulation takes into account the constraints on control traffic which, to our knowledge, has not been addressed previously.

This problem is hard in terms of computational complexity for the following reasons. First, constraints (1), (2), (3), (6) constitute a fault tolerant facility location problem. Second, constraints (4), (5), (7) form a multi-commodity flow problem. Third, the computation of the reliability $R(G, j, C)$ can be an NP-hard problem by itself [15]. Fourth, the number of

¹In other situations, it may denote the mixture of in-band and out-of-band control scheme, which is not so common in practice.

variables $X(K)$ can be exponential in the number of nodes and edges.

B. Mapping

Recall, the generality of the optimization process allows for a black-box implementation, here exemplified by simulated annealing for mapping (SAM). In short, the SAM algorithm in general follows the standard simulated annealing template [28], [29], except that it generates a new solution and decreases the temperature when receiving the *redoMapping* signal. The function for generating a new solution is designed as randomly adding or removing a controller based on the current mapping solution. The cost (energy) function for evaluating a mapping is defined as

$$cost = -\min(0, \log_{10} \frac{1 - R_{min}}{1 - \beta}, \lambda - 1) \quad (8)$$

The λ is calculated in the routability checking step. It is an indicator on whether control traffic is routable ($\lambda \geq 1$) or not ($\lambda < 1$). When both routability and reliability constraints are satisfied, the cost function reaches its minimum value 0.

Since directly computing the reliability $R(G, j, C)$ is NP-hard [15], the approximation method proposed in [15] is applied for computing a lower bound $\hat{R}(G, j, C)$ instead. The approximation method first computes the set of the disjoint paths from a node j to all the controllers C , noted as κ^j . Given the i.i.d operational probability of links/nodes on each disjoint path, we can calculate the failure probability of each path, noted as $F_k, k \in \kappa^j$. Then, compute the $\hat{R}(G, j, C) = 1 - \prod_{k \in \kappa^j} F_k$. See [15] for more details.

C. Association

The algorithm implements simulated annealing for association (SAA) and is similar to SAM. The two main differences relate to the cost function $cost = \min(0, \lambda - 1)$ and the generating new solution function, which is implemented by random association of aggregators and controllers towards obtaining a satisfying solution.

D. Traffic estimation

The demands of aggregator-controller and controller-controller flows have to be estimated. Let (s_f, t_f, d_f) represents the source, sink and demand of a flow f respectively. The objective of this step is to estimate each d_f while s_f and t_f are known from the mapping and association steps.

In principle, since the optimization process treats the model of control traffic as an input variable, any traffic model can be applied for estimating each d_f . For example, we can model either average or worst case demands, with either simple linear modelling method or advanced machine learning techniques.

However, as the scope of this paper concerns the generic optimization process, we just employ a simple traffic estimation model, assuming that the message sizes of aggregator request and corresponding controller response are $T_{req} = 128$ and $T_{res} = 128$ bytes, respectively. Furthermore, after dealing with a request, the controller instance sends messages of size $T_{state} = 500$ bytes to each of the other $|C| - 1$ control

instances notifying about the network state changes. Note that, this traffic model is essentially in line with the *ONOS* traffic model as described in [4]. The message sizes are here set according to [3], [4], [30], but can be set arbitrarily. With these parameter settings and given the request rate $r_j, j \in N$ of each aggregator, we simply estimate the traffic between aggregator j and its associated controller is $r_j T_{req}$ and $r_j T_{res}$, for aggregator-controller direction and controller-aggregator direction, respectively. We also use a simple linear model to estimate the outgoing inter-control traffic from controller i to another controller, which is described as $T_{res} \sum_{j \in N} a_{ij} r_j$.

E. Routability check

Solving the routability problem means dealing with an undesired exponential number of variables, as indicated by the constraints (4), (5), (7). This issue can be circumvented by formulating a maximum concurrent flow problem [31] (as (9), (10), (11), (12) suggest), which is easier to solve and equivalent to the multi-commodity flow problem.

The fundamental idea of the maximum concurrent flow problem is to keep the capacities of the link fixed while scaling the injected traffic so that all flows fit in the network. The optimization objective λ reflects the ratio of the traffic can be routed over the current injected traffic. If the we get $\lambda \geq 1$, the current traffic is routable and all link utilizations are less than one. The interpretation is that more traffic variation can be tolerated with larger λ .

$$\text{maximize} \quad \lambda \quad (9)$$

$$\text{s.t. :} \quad \sum_{K: e \ni K} X(K) \leq u_e, \forall e \in E \quad (10)$$

$$\sum_{K \in \kappa_f} X(K) \geq \lambda d_f, \forall f \in F \quad (11)$$

$$X(K) \geq 0, \forall K \quad (12)$$

The dual [32] of the above maximum concurrent flow problem has a linear number of variables and an exponential number of constraints. This elegantly allows for solving the problem to a desired level of accuracy using a primal-dual algorithm. We can apply the primal-dual algorithm designed by Karakostas [24] based on fully polynomial time approximation schemes (FPTAS). With this algorithm, we can get the near-optimal λ , which is guaranteed within the $(1 + \epsilon)$ factor of the optimal, within the time complexity of $O(\epsilon^{-2} |E|^2 \log^{O(1)} |E|)$. For details, please refer to [24], [31].

To accelerate the step further, we have in our implementation also introduced lower and upper bounds of λ , which allows for skipping the execution of Karakostas' algorithm under certain conditions. Moreover, considering that all the control flows are related to controllers (either originating from a controller or sink at a controller), we may further reduce the running time of the algorithm. We omit the details here as it is out of scope of this paper.

F. Usages

In this section, we list two usages of the feasibility solver as an example. Given constraints in the format $f > k, k \in R$

in the feasibility problem, the outlined process can in general be used to optimize k [33]. Intuitively, with a binary search method to adjust the value of k and re-run the feasibility solver each time, a maximum value of k that still guarantees a feasible solution can be found. This method allows for estimating, for example, the maximum R_{min} (defined in Section V-A), satisfying constraints u_e on the bandwidth, or similarly, the minimum bandwidth needed for guaranteeing a given reliability threshold constraint β . Note however, that this approach is only applicable for optimizing single objectives. In the case of multi-objective optimization, hierarchical optimization or trade-off methods [34] can be applied.

We outline the two optimization cases further. The topology for the two cases is taken from the Internet topology Zoo (ITZ) [35], called "Internetmci". For simplicity, we assume in-band control scheme that $M = N$. We assume that each node holds an aggregator with a request rate of 500 requests/s [36] and that the operational probability of each node, link and controller is 0.9999 [37]. The first case exemplifies minimization of the bandwidth u given a reliability threshold $\beta = 0.99999$ as a constraint on the minimum $Rmin$. In this case, assuming equal bandwidth consumption such that $u_e = u, \forall e \in E$, a minimum bandwidth of 35.112 Mbits/s is needed to ensure that $Rmin > \beta = 0.99999$. An example of the deployment solution is shown in Figure 2. The second case exemplifies maximization of $Rmin$ using the aforementioned binary search method, with the bandwidth constraint u_e of each link set as 24 Mbits/s. The maximum $Rmin$ achieved under these particular conditions using the proposed method is 0.99989 (with SAM and SAA for mapping and association) and produces a similar deployment plan as in Figure 2 but with only two controller instances.

Fig. 2. The corresponding deployment plan of controller instances (red), when the minimum required reserved bandwidth is 35.112 MBits/s per link, given the reliability threshold $\beta = 0.99999$ and requirement $R_{min} > \beta$.

VI. EVALUATION

Next, we evaluate the performance of different implementations of the optimization process, followed by a scaling test on the bandwidth and reliability. In the end, we compare the resulting bandwidth utilization of running [15] stand-alone, with integrated [15] into the optimization process.

The parameters used in the experiments are set based on the context of a simple distributed control service which only manages flow-tables in OpenFlow switches. The aggregator

Fig. 3. In (a) the failure probability ratio; (b) the optimization time ratio for various topologies and implementations.

request rate varies randomly within $[250\text{req/s}, 750\text{req/s}]$ by the use of a truncated normal distribution where $\mu = 500, \sigma = 500$ (in line with OpenFlow traffic characteristics [36]). The operational probability of links and nodes is randomly drawn from a Weibull-distribution, with $\lambda = 0.9999$ and $k = 40000$, considering the long tails in the downtime distribution of WAN links with four nines of mean reliability [15], [37]. To effectively display the minimum service reliability R_{min} (defined in Section V) in the figures, we plot the *failure probability* (i.e., $1 - R_{min}$) instead, since it is more suitable for plotting in log-scale.

The main purpose of the evaluation is to illustrate the capabilities and shortcomings of our proposed optimization process. However, our optimization process is in general applicable to other more complicated control services with different traffic parameters.

A. Evaluation of implementations

Implementation comparisons of the mapping and association steps (while holding remaining steps fixed) encompass the following combinations referred to as EE, AA and FS, respectively: 1) exhaustive search mapping (ESM) - exhaustive association (ESA); 2) simulated annealing mapping (SAM) - simulated annealing association (SAA); and, 3) heuristic FTCP mapping [15] - closest aggregator-controller association (CAA).

In the context of the $Rmin$ maximization similar to the use case outlined in Section V-F, we compare the performance in terms of achieved $Rmin$ and the optimization time. Three small, three medium and five large topologies [35] are used as test cases. In all cases the link bandwidth $u_e = u$ varies within $[\mu/2, 3\mu/2]$, randomly drawn using truncated normal

distribution with $\mu, \sigma = 4$. We set μ as 8 Mbits/s, 24 Mbits/s and 48 Mbits/s for small, medium and large topologies, respectively, sufficient for satisfying at least 3-nine reliability, but not for yielding trivial solutions. All results are based on 100 repetitions.

In Figure 3a, the performance of AA and FS for the small topologies is shown as a ratio relative to the baseline implementation EE in terms of the failure probability ($1 - R_{min}$) achieved. For the medium and large topologies, we only plot the performance ratio between FS and AA, as EE is too slow for getting any result. In Figure 3b the optimization time is shown as a ratio over each other, as the x-axis suggests.

Overall, the results in Figure 3a-3b demonstrate that the outlined optimization process in combination with suitable implementations of the mapping and association steps can provide a tunable control plane management solution close to optimal. The choice of methods is a trade-off between the ability to produce close to optimal solutions for different topology sizes and optimization time. For example, AA offers better performance (in terms of lower failure probability with the same link bandwidths), while FS provides faster running speed. As the comparisons in the context of bandwidth optimization (Section V-F) lead to similar results, we omit them.

B. Link bandwidth scaling test

We systematically quantify the influence on the achieved maximum R_{min} relative to an increasing link bandwidth constraint. In general, when scaling up the link bandwidths in a topology, the failure probability decreases, and hence R_{min} increases. Figure 4 illustrates this effect for the "Internetmc1" topology. Note however, that the failure probability decrease does not scale linearly with the bandwidth. By analyzing Figure 4, we are able to quantify the reliability gain relative to a certain bandwidth limit, at around 40Mbit/s. Beyond this point, increasing the bandwidth will only lead to a insignificant increase in reliability.

Fig. 4. Failure probability versus link bandwidth - the graph can be used to determine the optimal trade-off between required reliability and associated bandwidth demands.

The experiment demonstrates that the proposed optimization process can be used by service providers and operators as a practical tool for quantifying the trade-off between bandwidth and reliability gains, enabling development of flexible and fine-tuned controller deployment policies.

C. A comparative study

To the best of our knowledge, we are the first addressing the control plane management problem by the outlined process. Hence, it is hard to find existing work suitable for quantitative comprehensive comparisons.

TABLE I
RESULTS OF AL BASED ON 100 RUNS WITH $\beta = 0.99999$ AND LARGE TOPOLOGY RENATOR2010

Reserved BW.	No. Congestions	BUR (Median, Worst case)
50 Mbits	100	(0.0%, 0.0%)
100 Mbits	31	(7.6%, 20.1%)
150 Mbits	0	(36.6%, 50.1%)

Instead, we evaluate two cases using the method in [15] for mapping, when: 1) integrated into the optimization process as in the FS implementation in Section VI-A; and, 2) combined with only an association step (CAA) to enable it to work in a practical scenario, here referred to as AL. Given a minimum reliability constraint β , our optimization process offers the capability to estimate the control traffic and decide the bandwidth required to avoid congestion, as opposed to the AL approach which is limited to only providing a placement satisfying β . This limitation immediately leads to the following dilemma: manually reserving too little bandwidth will likely cause congestion, while reserving too much is a waste. We exemplify this dilemma in Table I, showing the bandwidth utilization ratio (BUR) for fixed bandwidth reservations relative to the estimates produced by FS. The average running time of FS is around 8 times longer than AL in the topology used (Table I). The dilemma regarding the trade-off between congestion risk and bandwidth reservation is apparent by the second case in Table I. Further, we observe that for the third case in particular, the bandwidth consumption when applying the outlined optimization process can be heavily reduced by at least 36.6% in half of the observed cases without the risk of introducing any congestion.

VII. CONCLUSION

We have proposed an optimization approach for flexible deployment of distributed control plane. The approach can automatically decide the number of controllers, their locations and control regions, and is guaranteed to find a non-congestion plan fulfilling requirements on reliability and bandwidth reserved for control traffic. This feature is specifically relevant in the context of future distributed control service applications, where the inter-control traffic required for shared information consistency could potentially become very large with the number of controller instances. Evaluation results indicate that the approach allows for finding close to optimal solutions under varying conditions and in comparison with relevant state-of-the-art. Moreover, the approach can be used as a practical tool for quantifying and predicting the trade-off between bandwidth and reliability, suitable for service providers and operators to develop control plane deployment policies. The code of our approach presented in the paper is available at <https://github.com/nigsics/dcpmtool>.

ACKNOWLEDGMENT

This work was funded in part by the Swedish Foundation for Strategic Research (reference number RIT15-0075) and by the Commission of the European Union in terms of the 5G-PPP COHERENT project (Grant Agreement No. 671639).

REFERENCES

- [1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, “Onix: A distributed control platform for large-scale production networks,” in *Proc. OSDI*, vol. 10, 2010, pp. 1–6.
- [2] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow *et al.*, “Onos: towards an open, distributed sdn os,” in *Proc. workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.
- [3] A. S. Muqaddas, A. Bianco, and P. Giaccone. (2016) Inter-controller traffic in onos clusters for sdn networks. [Online]. Available: http://onos-cord-eu.create-net.org/wp-content/uploads/2016/09/01-ONOS_CORD_Workshop16-InterClusterTraffic_ONOS-Abridged.pdf
- [4] A. S. Muqaddas, A. Bianco, P. Giaccone, and G. Maier, “Inter-controller traffic in onos clusters for sdn networks,” in *Proc. ICC*. IEEE, 2016, pp. 1–6.
- [5] B. Heller, R. Sherwood, and N. McKeown, “The controller placement problem,” in *Proc. HotSDN*. ACM, 2012, pp. 7–12.
- [6] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, “On the controller placement for designing a distributed sdn control layer,” in *Proc. Networking Conference*. IEEE, 2014, pp. 1–9.
- [7] T. Zhang, A. Bianco, and P. Giaccone, “The role of inter-controller traffic in sdn controllers placement,” in *Proc. NFV-SDN*, Nov 2016, pp. 87–92.
- [8] G. Yao, J. Bi, Y. Li, and L. Guo, “On the capacitated controller placement problem in software defined networks,” *Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.
- [9] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, “Dynamic controller provisioning in software defined networks,” in *Proc. CNSM*. IEEE, 2013, pp. 18–25.
- [10] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, “Pareto-optimal resilient controller placement in sdn-based core networks,” in *Proc. ITC*. IEEE, 2013, pp. 1–9.
- [11] 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 *Proc. NOMS*. IEEE, 2014, pp. 1–2.
- [12] 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,” *Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.
- [13] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, “Survivor: an enhanced controller placement strategy for improving sdn survivability,” in *Proc. GLOBECOM*. IEEE, 2014, pp. 1909–1915.
- [14] F. J. Ros and P. M. Ruiz, “Five nines of southbound reliability in software-defined networks,” in *Proc. workshop on Hot topics in software defined networking*. ACM, 2014, pp. 31–36.
- [15] —, “On reliable controller placements in software-defined networks,” *Computer Communications*, vol. 77, pp. 41–51, 2016.
- [16] A. Sallahi and M. St-Hilaire, “Optimal model for the controller placement problem in software defined networks,” *Communications Letters*, vol. 19, no. 1, pp. 30–33, 2015.
- [17] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, “Reliability-aware controller placement for software-defined networks,” in *Proc. International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 672–675.
- [18] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, “On reliability-optimized controller placement for software-defined networks,” *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.
- [19] Y. Zhang, N. Beheshti, and M. Tatipamula, “On resilience of split-architecture networks,” in *Proc. GLOBECOM 2011*. IEEE, 2011, pp. 1–6.
- [20] Q. Zhong, Y. Wang, W. Li, and X. Qiu, “A min-cover based controller placement approach to build reliable control network in sdn,” in *Proc. NOMS*. IEEE, 2016, pp. 481–487.
- [21] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, “Network flows: theory, algorithms, and applications,” pp. 649–686, 1993.
- [22] S. Kandula, D. Katabi, S. Sinha, and A. Berger, “Dynamic load balancing without packet reordering,” *SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 51–62, 2007.
- [23] M. Zhang, C. Yi, B. Liu, and B. Zhang, “Greente: Power-aware traffic engineering,” in *Proc. ICNP*. IEEE, 2010, pp. 21–30.
- [24] G. Karakostas, “Faster approximation schemes for fractional multicommodity flow problems,” *Transactions on Algorithms (TALG)*, vol. 4, no. 1, p. 13, 2008.
- [25] J. W. Rupe, “Reliability of computer systems and networks fault tolerance, analysis, and design,” *IIE Transactions*, vol. 35, no. 6, pp. 586–587, 2003.
- [26] (2011) Openflow switch specification v1.2. [Online]. Available: <https://www.opennetworking.org/>
- [27] S. Agarwal, M. Kodialam, and T. Lakshman, “Traffic engineering in software defined networks,” in *In Proc. INFOCOM*. IEEE, 2013, pp. 2211–2219.
- [28] A. Corana, M. Marchesi, C. Martini, and S. Ridella, “Minimizing multimodal functions of continuous variables with the simulated annealing algorithm corrigenda for this article is available here,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 13, no. 3, pp. 262–280, 1987.
- [29] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi *et al.*, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [30] A. Bianco, P. Giaccone, A. Mahmood, M. Ullio, and V. Vercellone, “Evaluating the sdn control traffic in large isp networks,” in *Proc. ICC*. IEEE, 2015, pp. 5248–5253.
- [31] N. Garg and J. Koenemann, “Faster and simpler algorithms for multi-commodity flow and other fractional packing problems,” *SIAM Journal on Computing*, vol. 37, no. 2, pp. 630–652, 2007.
- [32] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.
- [33] R. Impagliazzo, S. Lovett, R. Paturi, and S. Schneider, “0-1 integer linear programming with a linear number of constraints,” *arXiv preprint arXiv:1401.5512*, 2014.
- [34] L. S. d. Oliveira and S. F. Saramago, “Multiobjective optimization techniques applied to engineering problems,” *Journal of the brazilian society of mechanical sciences and engineering*, vol. 32, no. 1, pp. 94–105, 2010.
- [35] (2011) The internet topology zoo. [Online]. Available: <http://www.topology-zoo.org/>
- [36] D. Levin, A. Wundsam, A. Feldmann, S. Seetharaman, M. Kobayashi, and G. Parulkar, “A first look at openflow control plane behavior from a test deployment,” Technical Report, No. 2011/13 2011. [Online]. Available: <http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/2011>
- [37] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, “California fault lines: understanding the causes and impact of network failures,” in *SIGCOMM Computer Communication Review*, vol. 40, no. 4. ACM, 2010, pp. 315–326.