The SDN Control Plane Challenge for Minimum Control traffic: Distributed or Centralized?

Kostas Choumas, Dimitris Giatsios, Paris Flegkas and Thanasis Korakis

Dept. of ECE, University of Thessaly, Volos, Greece
Email: kohoumas, gidimitr, pflegkas, korakis@uth.gr

***Abstract*—Software Defined Networking (SDN) decouples the control and data planes and moves the control logic to the SDN controllers. The traffic in the control plane is either related to the interconnection of the controllers or to the connectivity between the SDN switches and the controllers. The controller placement affects the total control traffic. Multiple distributed controllers increase the inter-controller traffic, while few concentrated controllers increase the controller to switch traffic. We model the problem of determining the optimal controller placement for minimum control traffic. We further discuss the complexity of the optimization problem, and devise a heuristic algorithm for its solution. Our simulations and testbed experimentation reveal close to optimal performance of the presented heuristic algorithm.**

***Index terms*—** Software Defined Networking, Controller Placement, Testbed Experimentation

I. Introduction

Software Defined Networking (SDN) employs a novel networking architecture, in which control and data planes are decoupled, and the control logic is transfered from the network switches to special purpose servers, named SDN controllers. The initial SDN design features a single controller resulting in poor network scalability, while advanced new approaches leverage on multiple interconnected controllers. Due to the presence of multiple controllers, the switches are not connected to a single point of failure and the controller-to-switch (*Ctr-Sw*) traffic is shared among them. On the other hand, extra controller-to-controller (*Ctr-Ctr*) traffic is required in order for all controllers to be synchronised, sharing a common network view. In the first work examining the controller placement problem [1], under the assumption that each controller is directly attached to a switch, the approach is to decouple it into two subproblems. One is to find the number of controllers that should be used in a given SDN network, and the other is the selection of the switches hosting these controllers.

Although the existing literature proposes controller placements that mainly consider the time delays between the controllers and the switches, this work focuses on the analysis of the controller placement effect on the volume of control traffic, and how it could be minimized. The takehome message of this paper is that as the number of controllers increases and the control plane is more distributed, the controllers are closer to the switches and the total bandwidth required for the *Ctr-Sw* traffic decreases. On the other hand, if the controllers are fewer and closer, shaping a centralized control plane,

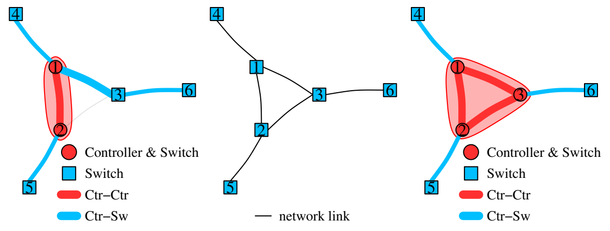


Fig. 1: Toy example. The left controller placement increases the *Ctr-Sw* traffic, while the right controller placement increases the *Ctr-Ctr* traffic. The link weights in both placements are proportional to the bandwidth required by these links.

then the total bandwidth for the *Ctr-Ctr* traffic decreases. The balance between these two types of traffic is the goal of the proposed solutions. To exemplify, Figure 1 shows in the middle a network with six switches. The left controller placement is more centralized, using only two controllers located at switches 1 and 2, while the right one is more distributed, using three controllers located at switches $1, 2$ and 3. The *Ctr-Ctr* traffic is less in the centralized placement, while the *Ctr-Sw* traffic is less in the distributed one.

The minimization of the total bandwidth required for the control traffic is the highest priority objective in networks with low capacity control plane, such as low speed wireless networks with “in-band” control. If the great majority of the SDN flows are proactively configured, then the time delays between the controllers and the switches are negligible for the whole performance. Especially for a network with limited-energy sensor nodes, the reduction of the control and data traffic is the most important objective, since the volume of the network traffic affects the total energy consumption [2]. These three aforementioned conditions are typically encountered in IoT networks exploited by massive machine-type communications (mMTC) in 5G. The application of SDN in these networks, especially regarding the controller placement models [3], is an open issue with a lot of ongoing research.

In this paper, i) we model the controller placement problem using Integer Quadratic Programming (IQP) with the objective to minimize the required bandwidth of the control traffic. The complexity of this problem does not scale polynomially with the size of the network, thus ii) we propose and evaluate a heuristic algorithm that expedites its solution. iii) We compare the performance of the optimal and heuristic controller placement, using network topologies given by the Internet Topology Zoo collection [4]. Finally, iv) we provide testbed measurements for estimating the magnitude of the bandwidth requirements of both control traffic types. In our model, the *Ctr-Sw* and *Ctr-Ctr* protocols are OpenFlow and Raft [5], [6] respectively, which are used by the state-of-the-art SDN controllers [7], such as OpenDaylight and ONOS.

The remainder of this paper is organized as follows. Section II introduces related work. We present the system model and problem statement in Section III, followed by three methodologies for the problem solution in Section IV. In particular, the first methodology is an analytical solution for a special network topology, which sheds some light on the control traffic dependencies, while the other two methodologies are the optimal and heuristic solutions. Section V presents our experimentation results in the NITOS testbed. The final Section VI concludes the paper.

II. Related Work

The controller placement problem was first introduced in [1]. The authors of this paper narrow their focus to two questions; given a network topology of SDN switches, how many controllers are needed and where in the topology should they go? They present multiple optimization goals for answering these questions, such as minimizing the average propagation latency by solving the minimum k-median problem, or minimizing the worst propagation latency by solving the minimum k-center problem. However, these optimization problems are NP-hard. The potential of heuristics, such as the k-medoids algorithm, is explored in [8] for solving the controller placement problem. The optimized criteria are related to the minimization of the average *Ctr-Sw* latency and the controller load imbalance. In [9], the capacitated controller placement problem is proposed, which incorporates a constraint on the controllers capacity and is formulated as a Mixed Integer Linear Programming (MILP) problem.

In [7], a new dimension on the controller placement problem is introduced, since the *Ctr-Ctr* traffic is also considered, besides the *Ctr-Sw* one. However, the optimization criterion is related to the reaction time perceived at the switches, that is dependent on the *Ctr-Sw* and *Ctr-Ctr* communication delays. In [10], a joint study of the *Ctr-Sw* and *Ctr-Ctr* traffic overhead is presented, which are considered as two of the four costs that are included in the objective's minimized weighted sum. The problem is formulated as an Integer Linear Programming (ILP) problem and two heuristics are introduced for its solution. In [8], [11], [12], a similar approach is followed, performing multi-objective optimization, where one of the objectives is the minimization of the control traffic. To the best of our knowledge, this is the first study of the controller placement focusing exclusively on the minimization of the total required bandwidth for the control traffic and showcasing the effects of the contradictory tendencies of either centralizing or distributing the control plane. Furthermore, this work is one of the few works that presents experimentation results in a realistic environment offered by a SDN testbed.

III. System Model and Problem Statement

We consider an undirected connected graph $\mathcal{G} = (\mathcal{S}, \mathcal{L})$, representing the control plane of an SDN network, where $\mathcal{S}$ represents the set of SDN switches and $\mathcal{L}$ represents the set of network links. Let $S = |\mathcal{S}|$ and $L = |\mathcal{L}|$ be the number of the switches and the number of the links respectively. Without loss of generality, we assume that the routing algorithm used for the control traffic is shortest path routing, the path connecting the couple of switches $s_1, s_2 \in \mathcal{S}$ is $p(s_1, s_2)$ and the number of links included in this path is $|p(s_1, s_2)|$. Finally, $\mathcal{C} \subseteq \mathcal{S}$ is the subset of switches where $C = |\mathcal{C}|$ controllers are placed. From now on, we may refer to $c \in \mathcal{C}$ as a controller or the switch hosting it, interchangeably. Let $c_s \in \mathcal{C}$ denote the controller that switch $s \in \mathcal{S}$ is assigned to. Vector $\mathbf{c} = (c_s \in \mathcal{C} : s \in \mathcal{S})$ describes a controller placement and switch assignment, where each vector's coordinate maps to a switch $s \in \mathcal{S}$ and the vector's value indicates the corresponding controller $c_s \in \mathcal{C}$.

On the one hand, we aim at minimizing the required bandwidth for the *Ctr-Sw* traffic from all network links, noted as

$$B^S \triangleq \sum_{s \in \mathcal{S}} \sum_{l \in p(s, c_s)} b^s = \sum_{s \in \mathcal{S}} w^s b^s, \tag{1}$$

where b^s is the bandwidth required for the *Ctr-Sw* traffic exchanged between switch s and controller c_s, while $w^s \triangleq |p(s, c_s)|$ is the length of the path connecting s to c_s. *The decrease of B^S depends on the decrease of the weights w^s, which happens when there are multiple distributed controllers close to the switches.*

On the other hand, we aim at minimizing the bandwidth requirements for the *Ctr-Ctr* traffic from all network links, noted as

$$\begin{aligned} B^C &\triangleq \sum_{c_1 \in \mathcal{C}} \sum_{c_2 \in \mathcal{C} - \{c_1\}} \sum_{l \in p(c_1, c_2)} b^{(c_1, c_2)} \\ &= \sum_{(c_1, c_2) \in \mathcal{C}^2} w^{(c_1, c_2)} b^{(c_1, c_2)}, \end{aligned} \tag{2}$$

where $b^{(c_1, c_2)}$ is the bandwidth required for the *Ctr-Ctr* traffic initiated from controller c_1 and exchanged with controller c_2, while $w^{(c_1, c_2)} \triangleq |p(c_1, c_2)|$ is the length of the path connecting the two controllers c_1 and c_2[1]. *The decrease of B^C depends on the decrease of the weights $w^{(c_1, c_2)}$, which happens when fewer and more centralized controllers are placed, one close to the other.*

In this work, we study what is the optimal placement $\mathcal{C}^*$ of $C^* = |\mathcal{C}^*|$ controllers and the optimal assignment $\mathbf{c}^* = (c_s^* \in \mathcal{C}^* : s \in \mathcal{S})$ of the switches, that minimizes the total bandwidth required for the control traffic, that is

$$\begin{aligned} \mathbf{c}^* \triangleq \arg\min_{\mathbf{c}} (B^S + B^C) = \\ \arg\min_{\mathbf{c}} \Big(\sum_{s \in \mathcal{S}} w^s b^s + \sum_{(c_1, c_2) \in \mathcal{C}^2} w^{(c_1, c_2)} b^{(c_1, c_2)} \Big). \end{aligned} \tag{3}$$

[1]If $c_1 = c_2$, then $w^{(c_1, c_2)} = 0$ and $b^{(c_1, c_2)} = 0$

At this point, we make the following Remarks, which are validated by our experimentation with SDN controllers, exploiting the OpenFlow protocol for their southbound interface and the Raft Consensus algorithm for their synchronization. More details about our experimentation results will be given later in Section V.

REMARK 1: *The required bandwidth for the Ctr-Sw traffic exchanged between a switch and its controller is proportional to the number of flows existing in this switch.*

Let β^s denote the required bandwidth for each flow. If f^s is the number of flows existing in $s \in \mathcal{S}$, then $b^s = f^s \beta^s$, $\forall s \in \mathcal{S}$. Let also $f \triangleq \sum_{s \in \mathcal{S}} f^s / S$ denote the average number of flows per switch.

REMARK 2: *The required bandwidth for the Ctr-Ctr traffic exchanged between two controllers and initiated from one of these two is proportional to the number of switches assigned to this controller.*

Let β^c denote the required bandwidth for each switch. If $y^c \triangleq \sum_{s \in \mathcal{S}: c_s = c} 1$ is the number of switches assigned to controller $c \in \mathcal{C}$, then $b^{(c_1,c_2)} = y^{c_1}\beta^c$, $\forall (c_1,c_2) \in \mathcal{C}^2 : c_1 \neq c_2$.

Based on these Remarks, the problem of Equation 3 changes to

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \Big(\sum_{s \in \mathcal{S}} w^s f^s \beta^s + \sum_{(c_1,c_2) \in \mathcal{C}^2} w^{(c_1,c_2)} y^{c_1} \beta^c \Big). \tag{4}$$

IV. PROBLEM SOLUTION

A. Closed form solution for Mesh control plane

Let's consider a mesh control plane where all switches have the same number f of flows and there is a link between each pair of switches. We search for the optimal controller placement and switch assignment that is the solution to the problem of Equation 4. Because of the symmetry of the mesh network, only the number of controllers affects the efficiency of a solution and not their placement. The only challenge is to find the optimal size C^* and the location of the controllers is chosen randomly. Then, the switches are equally assigned to the controllers, having also in mind that a switch hosting a controller is assigned to this controller.

From Equation 1 and Remark 1, we have

$$B^S_{\text{mesh}} = \sum_{s \in \mathcal{S}-\mathcal{C}} f^s \beta^s = \sum_{s \in \mathcal{S}-\mathcal{C}} f \beta^s = (S-C) f \beta^s. \tag{5}$$

Moreover, each controller c is one-hop away from the other controllers, and it is responsible for $\sum_{s:c_s=c} 1$ switches[2]. Thus, $\sum_c \sum_{s:c_s=c} 1 = S$, since all controllers are responsible for all switches. As follows, from Equation 2 and Remark 2, we have

$$B^C_{\text{mesh}} = \sum_{(c_1,c_2):c_1 \neq c_2} \sum_{s:c_s=c_1} \beta^c = \sum_{c_1} \sum_{s:c_s=c_1} \sum_{c_1 \neq c_2} \beta^c = \sum_{c_1} \sum_{s:c_s=c_1} (C-1)\beta^c = S(C-1)\beta^c. \tag{6}$$

The number C^* minimizing the sum $B^S_{\text{mesh}} + B^C_{\text{mesh}} = (S - C) f \beta^s + S(C-1)\beta^c = S(f\beta^s - \beta^c) + C(S\beta^c - f\beta^s)$ is equal to

$$C^* = \begin{cases} 1, & \text{if } S\beta^c - f\beta^s \geq 0 \Rightarrow f\beta^s/\beta^c \leq S \\ S, & \text{if } S\beta^c - f\beta^s < 0 \Rightarrow f\beta^s/\beta^c > S \end{cases}. \tag{7}$$

This is a toy example that clearly presents an outcome of this study, that is the dependency of C^* on the fraction $f\beta^s/\beta^c$ and the network size S. Later, we show how mathematical programming is exploited for solving the same problem for all control plane topologies.

B. Integer Quadratic Programming (IQP)

Now, let's examine the case of a general control plane topology. The optimization problem of Equation 4 is equivalent to the following IQP problem

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad \sum_{i=1}^{S}\sum_{j=1}^{S} x_{ij} w_{ij} f^s \beta^s + \sum_{i=1}^{S}\sum_{j=1}^{S} w_{ij} y_i z_j \beta^c \tag{8}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j=1,\ldots,S} x_{ij} = 1, \quad \forall i = 1,\ldots,S, \\ & \sum_{i=1,\ldots,S} x_{ij} = y_j, \quad \forall j = 1,\ldots,S, \\ & z_i \geq y_i / S, \quad \forall i = 1,\ldots,S. \end{aligned} \tag{9}$$

This problem has S^2 binary variables x_{ij}, S integer variables y_i and S binary variables z_j. Indexes i and j take integer values from 1 to S, where each value corresponds to a switch $s \in \mathcal{S}$. According to the problem solution, if binary $x_{ij} = 1$, then switch s_i (the switch corresponding to index i) has to be assigned to controller c_j (the controller collocated with switch s_j). Integer y_i is the number y^{c_i} of switches assigned to controller c_i, if $y_i > 0$. If $y_i = 0$, then there is no controller placed at switch s_i. Binary $z_j = 1$ if and only if a controller is placed at switch s_j. Finally, w_{ij} is the length of the path connecting switch s_i (or controller c_i) and controller c_j.

In Equation 8, the first and second sum terms are the B^S and B^C respectively. The first term is the sum of the lengths of the paths connecting all switches to their controllers, multiplied by the bandwidth $f^s\beta^s$. The second term is the sum of the products between the length of each path connecting a couple of controllers and the number of switches connected at each controller, multiplied by the bandwidth β^c. In 9, the first constraint restricts each switch to be assigned to only one controller, while the second and third constraint guarantee that y_i and z_j have the meaning we mentioned before. Especially for the third constraint, binary variable z_j has to be minimized, thus $z_j = 0$, if $y_j = 0$, otherwise $z_j = 1$, since $0 \leq y_j \leq S$.

[2] $\sum_s$, $\sum_c$ and $\sum_{(c_1,c_2)}$ are equivalent to $\sum_{s \in \mathcal{S}}$, $\sum_{c \in \mathcal{C}}$ and $\sum_{(c_1,c_2) \in \mathcal{C}^2}$ respectively.

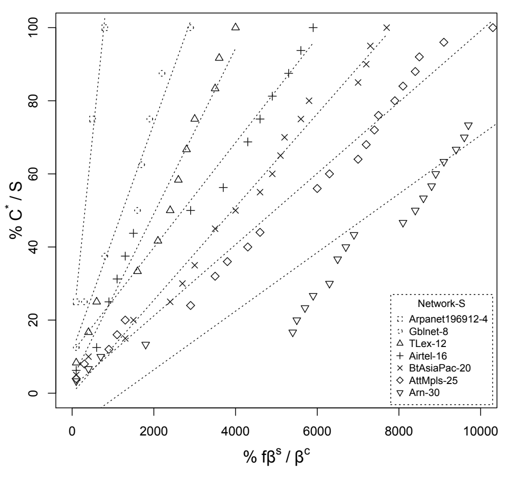


Fig. 2: Linear relationship between C^*/S and $f\beta^s/\beta^c$ for various network topologies. The legends indicates the name and the size S of the corresponding network topology, as it is given by the Internet Topology Zoo collection.

The optimal controller placement given by IQP is $\mathcal{C}^* = (s_j \in \mathcal{S} : z_j = 1)$ and the switch assignment is $\mathbf{c}^* = (c^*_{s_i} = c_j : x_{ij} = 1)$. We use R [13] and CPLEX [14] for solving this problem for multiple network topologies, provided by the Internet Topology Zoo collection [4]. In our measurements, we assume that each of the network topologies represents the control plane $\mathcal{G}$. Given the symmetric matrix $\mathbf{w} = (w_{ij} : i, j = 1, \ldots, S)$ induced by $\mathcal{G}$, the objective function of IQP is $\mathbf{v}^T\mathbf{Q}\mathbf{v}/2 + \mathbf{q}^T\mathbf{v}$, where

$$\mathbf{Q} = \begin{bmatrix} \mathbf{0}^{S^2\times S^2} & \mathbf{0}^{S^2\times S} & \mathbf{0}^{S^2\times S} \\ \mathbf{0}^{S\times S^2} & \mathbf{0}^{S\times S} & \mathbf{w}^{S\times S} \\ \mathbf{0}^{S\times S^2} & \mathbf{w}^{S\times S} & \mathbf{0}^{S\times S} \end{bmatrix}, \quad (10)$$

$$\mathbf{q} = \left(\tilde{\mathbf{w}}^{1\times S^2}\mathbf{0}^{1\times S}\mathbf{0}^{1\times S}\right) \;\&\; \mathbf{v} = \left(\tilde{\mathbf{x}}^{1\times S^2}\mathbf{y}^{1\times S}\mathbf{z}^{1\times S}\right). \quad (11)$$

The vectors $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{x}}$ are composed of all the rows of the matrixes $\mathbf{w}$ and $\mathbf{x}$ respectively. Moreover, the matrix $\mathbf{Q}$ is symmetric but not positive definite, which means that this problem is not a Convex Programming (CP) one. Actually, it is a Quadratic Assignment Problem (QAP), that is NP-hard and cannot be resolved in reasonable time for size greater than 30 [15]. This is also verified through our extensive IQP solving for multiple networks. Next, we present how this problem could be heuristically solved in reasonable time.

C. Heuristic Algorithm

Let $\mathcal{C}^h$ denote the controller placement according to the proposed heuristic algorithm. We will show that $\mathcal{C}^h$ depends only on the fraction $f\beta^s/\beta^c$, the network size S and an appropriate centrality metric. We first search for the number $C^h = |\mathcal{C}^h|$ of controllers that should be placed in the network. Our testing using IQP clearly shows a linear dependency between the fractions C^*/S and $f\beta^s/\beta^c$. Figure 2 depicts this linear dependency for 7 networks with various network sizes. The full set of our simulations includes 135 networks with size $S \leq 30$ and confirms this linear dependency. The analysis of the results shows that:

- For low $f\beta^s/\beta^c$, B^C is more weighted than B^S, controllers are placed more centrally and C^*/S decreases.
- For high $f\beta^s/\beta^c$, the weight of B^S is amplified, more distributed controllers are placed and C^*/S increases.

We model the linear relationship between C^*/S and $f\beta^s/\beta^c$, that is $(C^*/S) \simeq a(f\beta^s/\beta^c) + b$, and we use our simulation results to extract slope a and y-intercept b. The slope a decreases as the network size S increases. Their exponential relationship is illustrated by the dashed line in Figure 3(a). The corresponding dashed line in Figure 3(b) depicts the linear relationship between y-intercept b and network size S. Using these models, C^h is extracted in constant time ($\mathcal{O}(1)$ complexity) applying the following formulas:

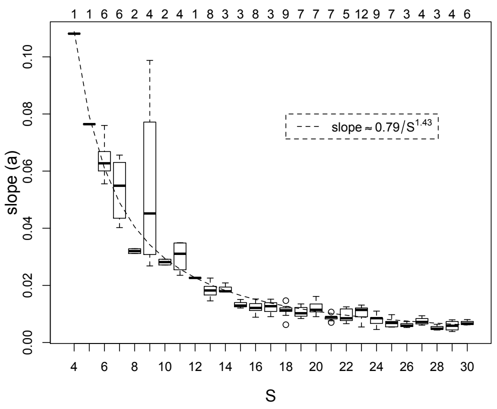
$$\begin{aligned} a = 0.79/S^{1.43} \;\&\; b = -0.3S + 9.61, \\ C^H \triangleq \big(a(f\beta^s/\beta^c) + b\big)S. \end{aligned} \quad (12)$$

Finally, the $\mathcal{C}^h$ controllers are placed at the C^h most "central" switches. The centrality of each switch is evaluated with the use of the betweenness [16] metric, which is based on shortest paths and gives higher value to nodes with more control over the network. The black boxplots of Figure 4 show the percent increase of $B^S + B^C$ for various networks, if the controller placement is $\mathcal{C}^h$ instead of $\mathcal{C}^*$. The average increase over all cases is 2.62% and the highest is 19.54%. The median increase is very low for small networks and extends up to approximately 10% for large networks.

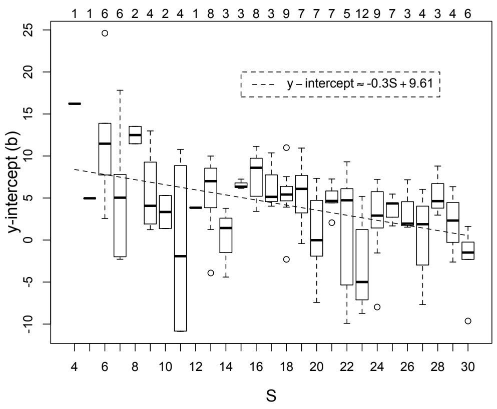
The red boxplots of Figure 4 show the percent increase of $B^S + B^C$, when we place the controllers at the C^* most outer switches with the lowest betweenness metric (let $\mathcal{C}^w$ denote this placement), depicting how much worst could be the bandwidth increase if the controller placement was random, without use of centrality metrics. The average worst increase over all network sizes is equal to 48.04% and the highest is 74.16%, which records are much higher than the previous ones for the $\mathcal{C}^h$ placement. The time complexity of estimating the betweenness metric for all switches is quasilinear ($\mathcal{O}(SL + S^2 log(S))$) using the Brandes algorithm [17]. This process is not repeated when the fraction $f\beta^s/\beta^c$ changes. In our simulations, the time needed for estimating $\mathcal{C}^h$ is significantly less than this of IQP solving.

V. Testbed Experimentation

The evaluation of the given solutions has been done in the SDN testbed of NITOS [18], which consists of 50 powerful Linux machines, the NITOS nodes. Open vSwitch (OvS) is installed on the NITOS nodes, enabling their exploitation as SDN/OpenFlow switches. The Kandoo controller [19] is deployed on each NITOS node. One Kandoo controller is the root controller and the others are the local controllers, however, this two-layer hierarchy does not make any difference



(a) slope (a) for various S.



(b) y-intercept (b) for various S.

Fig. 3: Slope (a) and y-intercept (b) for various networks of size $S = 4, 5, \ldots, 30$. The top axis gives the number of samples for each S. The dashed lines model the relationship between the boxplot medians and S.

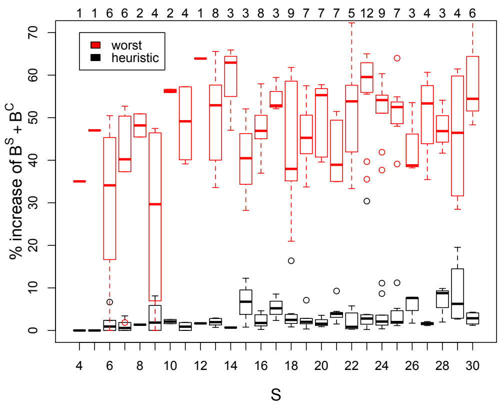


Fig. 4: % increase of $B^S + B^C$ for various networks of size $S = 4, 5, \ldots, 30$. The top axis gives the number of samples for each S. The black and red boxplots refer to the increase produced when $\mathcal{C}^h$ and $\mathcal{C}^w$ respectively are used instead of $\mathcal{C}^*$.

TABLE I: The *Ctr-Sw* bandwidth requirements in Kbps for a Kandoo controller. Columns *sw→ctr* and *ctr→sw* refer to the traffic sent from the sw(itch) to the c(on)tr(oller) and the opposite.

# sw	# flows	*sw→ctr*	*ctr→sw*	total	increase
1	0	1.06	2.50	3.56	-
	2	3.81	2.50	6.31	2·1.38
	3	5.19	2.50	7.69	1.38
	10	14.80	2.50	17.30	7·1.37
	20	29.40	2.50	31.10	10·1.38
10	0	10.60	25.0	35.60	-
	10	148.0	25.0	173.6	10·10·1.38
	20	294.0	25.0	311.0	10·10·1.37

in their inter-connections and their connections to the switches. This hierarchy comes mainly from the process of introducing each controller to the others, since all controllers first build a connection with the root controller and then they are also connected to the other local controllers.

A. Experimental Confirmation of Remarks 1 and 2

Tables I and II present the experimentation results that confirm the validity of Remarks 1 and 2. More specifically, Table I shows the bandwidth requirements for the *Ctr-Sw* traffic between a Kandoo controller and 1 or 10 switches. The bandwidth has been measured with the use of *iftop*. The first and second column are the number of switches connected to the controller and the number f of flows existing at each switch. The next three columns show the bandwidth required for the *Ctr-Sw* traffic for each direction (controller to switch and the opposite), as well as their sum. The last column shows the increase of bandwidth requirements when extra flows are added at the switches, compared to the previous state. It is easily extracted, that the traffic initiated by each switch is independent of the other switches assigned to the same controller and linear dependent on the number of flows existing at that switch. Thus, Remark 1 holds and $\beta^s \approx 1.38$Kbps.

Table II shows the bandwidth requirements for the *Ctr-Ctr* traffic between two or three Kandoo controllers, when they are responsible for various numbers of switches. The left subtable of Table II shows the bandwidth requirements from the path connecting a couple of Kandoo controllers, the root (or r) and the local (or l) controller, when various number of switches are connected to these controllers. The first column shows how many empty switches (without flow entries) are assigned to each controller. The next two columns show the bandwidth requirements of the traffic sent from one controller to the other, for both directions (r→l and l→r), while the last two columns show the increase of bandwidth requirements when extra switches are added to the controllers, compared to the previous state. For example, the 4th column of the row starting with (1,1) shows the 83Kbps increase in bandwidth requirements for both directions, when one switch is added to each controller. The 4th and 5th columns of the row starting with (1,0) indicate 50Kbps and 15Kbps increases for each

TABLE II: The *Ctr-Ctr* bandwidth requirements in Kbps for two or three Kandoo controllers. In the left table, there are two controllers, the r(oot) and the l(ocal). In the right table, there are three controllers, which are named as r(oot), l(ocal)$_1$ and l(ocal)$_2$.

# sw	bandwidth		increase	
at r, l	$r\to l$	$l\to r$	$r\to l$	$l\to r$
0,0	20	20	-	-
1,1	103	103	83	83
2,2	151	151	48	48
3,3	197	197	46	46
4,4	239	239	42	42
5,5	284	284	45	45
10,10	490	490	5·41	5·41
1,0	70	35	50	15
2,0	112	42	42	7
3,0	152	47	40	5
4,0	192	50	40	3
5,0	230	53	38	3
10,0	422	67	5·38	5·3
0,1	54	88	34	68
0,2	60	130	6	42
0,3	65	168	5	38
0,4	68	210	3	42
0,5	72	247	4	37
0,10	87	437	5·3	5·38

# sw	bandwidth						increase					
at r, l_1, l_2	$r\to l_1$	$l_1\to r$	$r\to l_2$	$l_2\to r$	$l_1\to l_2$	$l_2\to l_1$	$r\to l_1$	$l_1\to r$	$r\to l_2$	$l_2\to r$	$l_1\to l_2$	$l_2\to l_1$
0,0,0	20	20	20	20	0	0	-	-	-	-	-	-
1,1,1	105	105	105	105	105	105	85	85	85	85	105	105
2,2,2	151	151	151	151	151	151	46	46	46	46	46	46
10,10,10	490	490	490	490	490	490	8·42	8·42	8·42	8·42	8·42	8·42
20,20,20	913	913	913	913	913	913	10·42	10·42	10·42	10·42	10·42	10·42
1,0,0	70	35	70	35	0	0	50	15	50	15	0	0
2,0,0	112	42	112	42	0	0	42	7	42	7	0	0
10,0,0	422	69	422	69	0	0	8·39	8·3	8·39	8·3	0	0
0,1,0	54	88	20	20	68	34	34	68	0	0	68	34
0,2,0	60	130	20	20	110	40	6	42	0	0	42	6
0,10,0	87	438	20	20	422	67	8·3	8·39	0	0	8·39	8·3
1,1,0	103	103	70	35	70	35	83	83	50	15	70	35
2,2,0	152	151	111	41	111	41	49	48	41	6	41	6
10,10,0	489	488	423	68	422	68	8·42	8·42	8·39	8·3	8·39	8·3
0,1,1	54	88	54	88	101	101	34	68	34	68	101	101
0,2,2	60	129	60	129	150	150	6	41	6	41	49	49
0,10,10	87	439	87	439	486	485	8·3	8·39	8·3	8·39	8·42	8·42

direction $r\to l$ and $l\to r$ respectively, when one switch is added solely to r. Finally, the row starting with (0,1) shows the corresponding 34Kbps and 68Kbps increases, when one switch is added solely to l.

As more and more switches are assigned to a controller, the increase of the traffic sent from this controller to the other one due to an extra added switch converges to 38Kbps, while the increase of the traffic sent in the opposite direction converges to 3Kbps. Moreover, the increase when x switches are added at each controller is approximately the sum of the individual increases when x switches are added exclusively at either r or l, for both directions. Based on these two observations, it is safe enough to assume that Remark 2 holds and each extra switch adds an overhead on the bandwidth requirements equal to $\beta^c \approx 38 + 3 = 41$Kbps.

The right subtable of Table II shows the corresponding increases of the traffic forwarded through each of the three paths connecting the three controllers, called root (or r), local$_1$ (or l$_1$) and local$_2$ (or l$_2$). Similarly, the effect on the bandwidth requirements for each extra switch assigned to a controller is independent of the other already assigned switches and converges to an increase of the traffic exchanged through each path connecting this controller to another one, equal to $\beta^c \approx 42$Kbps. Thus, Remark 2 is also confirmed by the right part of Table II.

Finally, we observed that the increase of the *Ctr-Ctr* traffic, when flows are added to the switches, is minor and negligible. Due to space limitations, we present a fraction of our experimentation results, although we have been experimenting with more than three Kandoo controllers obtaining the same qualitative conclusions.

B. The Heuristic Algorithm in the "Abilene" topology

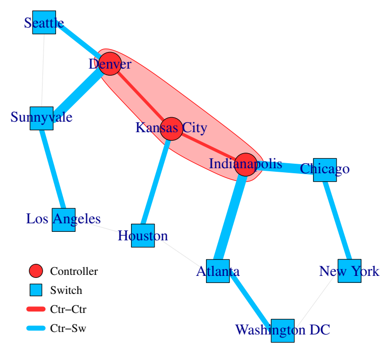
We use NITOS for deploying the network topology of "Abilene" from the Internet Topology Zoo Collection. We configure 11 nodes of the NITOS testbed to behave as OpenFlow switches, leveraging on OvS. The hosts are virtually created at each node with the use of Mininet [20]. The flows of each switch are proactively configured and enable the hosts to ping each other. The total number of flows at each switch is $f = h(h-1)$, where h is the number of hosts connected at each switch.

In this experimentation, we configure each switch having either 15, 21 or 28 hosts, meaning that each switch has $f = 210$, 420 or 756 flows respectively. As follows, $f\beta^s \approx 290$Kbps, 580Kbps or 1043Kbps and $\beta^c \approx 42$Kbps. Figure 5 presents the optimal and heuristic controller placements, as they are given by IQP and the heuristic algorithm respectively. For $f\beta^s = 290$Kbps, the optimal (Figure 5(a)) and heuristic (Figure 5(d)) solutions use the same number of controllers, equal to 3. Two of the three controllers are located at the same place ("Kansas City" and "Indianapolis"), while the third controller of is at "Denver" and "Houston" for the optimal and heuristic solutions respectively. The difference occurs since "Houston" features the third highest betweenness centrality after "Indianapolis" and "Kansas", and not "Denver". Due to this difference, the heuristic solution requires 6% more bandwidth than the optimal one.

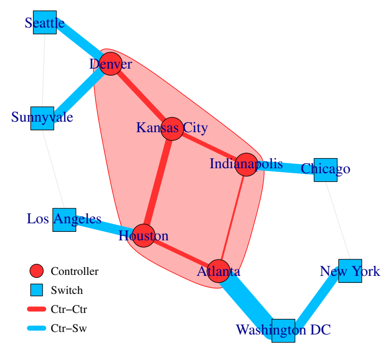
For $f\beta^s = 580$Kbps, the two solutions are identical, as it is depicted in Figures 5(b)-5(e). Finally, for $f\beta^s = 1043$Kbps, the heuristic solution (Figure 5(f)) exploits a more extended set of controllers, compared to the optimal solution (Figure 5(c)). The result is an approximate 4% increase on the total required bandwidth for the control traffic.

VI. Conclusion & Future Work

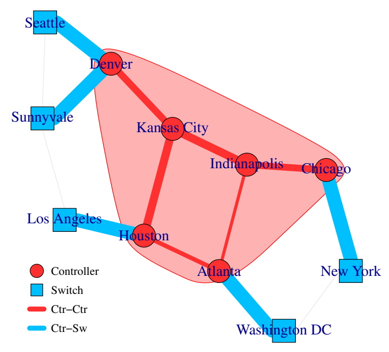
In this work, we investigate the optimal controller placement for minimum control traffic, using IQP. We also present a heuristic algorithm that expedites the controller placement procedure, while incurring negligible traffic increases with respect to the optimal solution. We exploit the NITOS SDN testbed and measure the control traffic for various network topologies and controller placements. The heuristic placement requires 2.62% more bandwidth than the optimal one. As part of future work, we plan to extend our research by using other centrality metrics, apart from betweenness, in our heuristic algorithm. Finally, we plan to measure the bandwidth requirements for the control traffic produced by other OpenFlow controllers, such as the OpenDaylight controller.



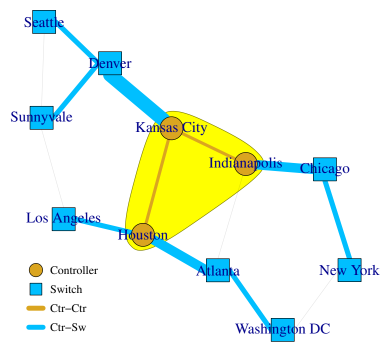
(a) Optimal plac/nt ($f\beta^s = 290$Kbps).



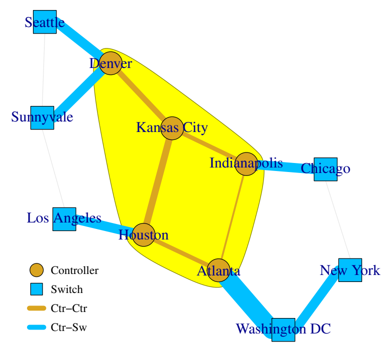
(b) Optimal plac/nt ($f\beta^s = 580$Kbps).



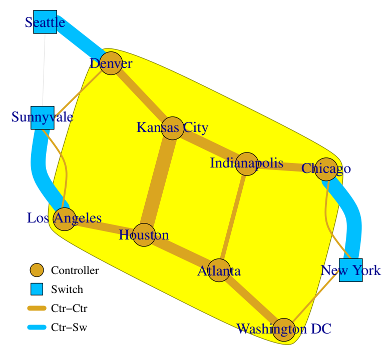
(c) Optimal plac/nt ($f\beta^s = 1043$Kbps).



(d) Heuristic plac/nt ($f\beta^s = 290$Kbps).



(e) Heuristic plac/nt ($f\beta^s = 580$Kbps).



(f) Heuristic plac/nt ($f\beta^s = 1043$Kbps).

Fig. 5: Visualization of the optimal and heuristic controller placements on the "Abilene" network topology for $\beta^c = 42$Kbps and various $f\beta^s = 290$, 580 or 1043Kbps. The colour and the weight of each link indicates whether it is used by the *Ctr-Ctr* and/or *Ctr-Sw* traffic, as well as the bandwidth requirements of this traffic.

VII. Acknowledgments

This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Program "Human Resources Development, Education and Lifelong Learning 2014-2020" in the context of the project "Dynamically Reconfigurable Backhaul/Fronthaul in Software Defined 5G Networks" (MIS 5005859).

References

[1] B. Heller, R. Sherwood, and N. McKeown. The Controller Placement Problem. In *Proc. HotSDN*, 2012.

[2] S. Keranidis, G. Kazdaridis, V. Passas, T. Korakis, I. Koutsopoulos, and L. Tassiulas. Online Energy Consumption Monitoring of Wireless Testbed Infrastructure Through the NITOS EMF Framework. In *Proc. ACM WiNTECH*, 2013.

[3] T. M. C. Nguyen, D. B. Hoang, and Z. Chaczko. Can SDN Technology Be Transported to Software-Defined WSN/IoT? In *iThings-GreenCom-CPSCom-SmartData*, 2016.

[4] The Internet Topology Zoo. http://www.topology-zoo.org/dataset.html.

[5] D. Ongaro and J. Ousterhout. In Search of an Understandable Consensus Algorithm. In *Proc. USENIX ATC*, 2014.

[6] The Raft Consensus Algorithm. https://raft.github.io/.

[7] T. Zhang, A. Bianco, and P. Giaccone. The role of inter-controller traffic in SDN controllers placement. In *Proc. IEEE Conference on NFV and SDN (NFV-SDN)*, 2016.

[8] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, and P. Tran-Gia. Specialized Heuristics for the Controller Placement Problem in Large Scale SDN Networks. In *Proc. International Teletraffic Congress (ITC)*, 2015.

[9] B. P. R. Killi and S. V. Rao. Capacitated Next Controller Placement in Software Defined Networks. *IEEE Trans. on Network and Service Management*, 14(3):514–527, Sept 2017.

[10] 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. International Conference on Network and Service Management (CNSM)*, 2013.

[11] Z. Su and M. Hamdi. Online Energy Consumption Monitoring of Wireless Testbed Infrastructure Through the NITOS EMF Framework. In *Proc. IEEE ICPADS*, 2015.

[12] Q. Qin, K. Poularakis, G. Iosifidis, and L. Tassiulas. SDN Controller Placement at the Edge: Optimizing Delay and Overheads. In *Proc. IEEE INFOCOM*, 2018.

[13] The R project for Statistical Computing. http://www.r-project.org.

[14] IBM ILOG CPLEX for Faculty. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

[15] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *Elsevier, European Juornal of Operational Research*, 176(2):657–690, 2007.

[16] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[17] S. Bhardwaj, R. Niyogi, and A. Milani. Performance Analysis of an Algorithm for Computation of Betweenness Centrality. In *Proc. ICCSA*, 2011.

[18] Network Implementation Testbed using Open Source platforms (NITOS). https://nitlab.inf.uth.gr/NITlab/nitos.

[19] S. Hassas Yeganeh and Y. Ganjali. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In *Proc. HotSDN*, 2012.

[20] Mininet: An Instant Virtual Network on your Laptop. http://mininet.org.