Multi-criteria based Optimization of Placement for Software Defined Networking Controllers and Forwarding Nodes

Eugen Borcoci, Tudor Ambarus, Marius Vochin
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
Emails: eugen.borcoci@elcom.pub.ro, tudorambarus@yahoo.com, mvochin@elcom.pub.ro

***Abstract* — Placement of Software Defined Networking (SDN) controllers and forwarders in large networks contexts is still a research open issue, given the different network contexts, providers' policies and possible optimization criteria. Multi-criteria decision algorithms can provide valuable solutions. This paper is an extension of a previous preliminary work, considering here large network environments and the additional problem of forwarding nodes assignment to SDN controllers.**

Keywords* — *Software Defined Networking; Multi-criteria optimizations; Controller placement; Forwarding nodes assignment; Reliability;

I. INTRODUCTION

Emergent Software Defined Networking (SDN) architectures and related technologies are of high interest for industry and operators, in wire-line and wireless networks and cloud computing environments, [2][3]. This paper considers the case of a Wide Area Network (WAN) owned by an operator and/or a Network/Service Provider (NP/SP).

For large SDN-controlled networks, multi-controller solutions are proposed to solve the scalability problems, related to SDN control centralization principle [5][6][7]. Flat or hierarchical organizations for multi-controller SDN are suggested in [6][7] (in the subsequent text, by "controller" it is understood a geographically distinct controller location). The data forwarding network nodes (called also "forwarders" or simply, "nodes") should be allocated to some controllers in a proactive or reactive way. Some design problems are: *What is the optimal number and placement of the controllers? How to allocate the forwarder nodes to controllers?*

The controller placement problem is a NP-hard one [9]. So, different solutions have been proposed, with specific optimization criteria, targeting performance in failure-free or more realistic scenarios. However, some criteria could lead to different solutions; so, a multi-criteria global optimization could be attractive. Some specific criteria could be defined as to: (a) maximize the *controller-forwarder* or *inter-controller communication* throughput, and/or reduce the latency of the path connecting them; (b) limit the controller overload (load imbalance) by avoiding too many forwarders per controller; (c) find an optimum controllers' placement and forwarder-to-controller allocation, offering a fast recovery after failures (controllers, links, nodes).

Also, other specific optimization goals could be added to the above list, depending on specific context (wire-line, wireless/cellular, cloud computing and data center networks) and on some specific business targets of the Service Provider.

The paper [1] provides a contribution on *multi-criteria optimization algorithms* for the controller placement problem. The target was *not to develop specific algorithms* to find an optimum solution for *a single given criterion* (several other studies already did that) but to *achieve an overall optimization on controller placement*, by applying *multi-criteria decision algorithms (MCDA)* [10]. The input of MCDA is the set of candidates (an instance of controller placement was called a *candidate solution*). Simple examples have been analyzed, proving the usefulness of the approach.

This paper is an extension of [1] by constructing a software simulation model; it considers larger realistic topologies, variation of the MCDA criteria and optimized allocation of the forwarders to controllers. Simulation experiments and novel results are presented.

The paper is organized as follows. Section II is an overview of related work. Section III revisits several metrics and algorithms used in optimizations and presents some of their limitations. Section IV develops the framework for MCDA-RL (reference level variant) to select the best controller placement solution. Section V presents a set of simulation experiments performed and the results obtained. Section VI presents conclusions and future work.

II. SDN CONTROLLER PLACEMENT -RELATED WORK

This short section is included for self containment of this paper. A more comprehensive overview on some previously published work on controller placement in SDN-managed WANs is given in [1]. The basic problem to be solved is on the number of controllers and their placement in a given network. The goal is to provide enough performance (e.g., low delay for controller-forwarder communications) and also create robustness to controllers and/or network failures.

The works [8][9] have shown that the above problem is theoretically not new. If *latency* is taken as a metric, the problem is similar to a known one, namely, the *facility or warehouse location problem*, solved, e.g., by using Mixed Integer Linear Program (MILP) tools.

Heller et al. [9] have shown that it is possible to find optimal solutions for realistic network instances, in failure-free scenarios, by analyzing the entire solution space, with off-line computations (the metric is latency). Going further, the works [8][11][14][15][16] additionally considered the resilience as being important with respect to events like: *controller* failures, *network links/paths/nodes* failures, *controller overload* (load imbalance). The *Inter-Controller Latency* is also important and generally it cannot be minimized while simultaneously minimizing controller-forwarders latency; a tradeoff solution could be the answer.

The works [8][15] developed several placement algorithms for some real topologies, trying to improve the reliability of SDN control, but still keep acceptable latencies. The controller instances are chosen as to minimize connectivity losses; connections are defined according to the shortest path between controllers and forwarding devices. Muller et.al. [16] try to eliminate some restrictions of previous studies, like: single paths, processing (in controllers) of the forwarders requests only *on-demand* and some constraints imposed on failover mechanisms.

As stated previously, *this paper does not aim to develop a new algorithm for optimized controller placement, based on a given particular metric, but extends a previous overall optimization work, while using multiple criteria.* The general part of the problem, exposed in [1] is summarized here for sake of self-containment.

III. SUMMARY OF CONTROLLER PLACEMENT METRICS ALGORITHMS

This section is a short presentation of a few typical metrics and optimization algorithms for controller placement. A more extended presentation can be found in [1]. Considering a particular metric (criterion) an optimization algorithm can be run, as in [8][9][11][16]. This paper goal is not to develop a new particular algorithm - but to search for a global optimization.

A. *Performance-only related metrics (failure-free scenarios)*

The network is represented by an undirected graph G(V, E), where V,E are the sets of nodes and edges, respectively and n=|V| is the number of nodes. The edges weights represent an additive metric (e.g., *propagation latency* [9]). The controllers will be co-located to some network nodes.

A simple metric is d(v, c): *shortest path* distance from a forwarder node $v \in V$ to a controller $c \in V$. In [9], two kinds of latencies are defined, for a particular placement C_i of controllers, where $C_i \subseteq V$ and $|C_i| \leq |V|$. The number of controllers is limited to $|C_i| = k$ for any particular placement C_i. The set of all possible placements is denoted by $C = \{C_1, C_2 \ldots\}$. One can define, for a given placement C_i:

Worst_case_latency:

$$L_{wc} = \max_{v \in V} \min_{c \in C_i} d(v,c) \tag{1}$$

Average_latency:

$$L_{avg}(C_i) = \frac{1}{n} \sum_{v \in V} \min_{c \in Ci} d(v,c) \tag{2}$$

The algorithm should find a placement C_{opt}, where *either average latency* or *the worst case latency is minimized.*

The limitations of this optimization process consist in: static values assumed for latencies, despite that delay is a dynamic value in IP networks; only free-failure case are considered; no upper limit on the number of forwarders assigned to a controller; not taking into account the inter-controller connectivity. Another possible metric to be considered in failure-free case is *Maximum cover*, [9][17]. The algorithm should find a controller placement, as to *maximize the number of nodes within a latency bound*, i.e., to find a placement of *k* controllers such that they cover a maximum number of forwarder nodes, while each forwarder must have a limited latency bound to its controller.

B. *Reliability aware metrics*

More realistic scenarios consider controller and/or network failures events. The optimization process aims now to find trade-offs to preserve a convenient behavior of the overall system in failure cases.

(1) Controller failures (cf): the work [11] observes that the node-to-controller mapping can change in case of controller failures. So, the latency-based metric should consider both the distance to the (primary) controller and the distance to other (backup) controllers. For a placement of a total number of *k* controllers, the failures are modeled by constructing a set C of scenarios, including all possible combinations of faulty controller number, from 0 of up to k - 1. The resulting maximum latency will be:

Worst_case_latency_cf:

$$L_{wc-cf} = \max_{v \in V} \max_{C_i \in C} \min_{c \in C_i} d(v,c) \tag{3}$$

The optimization algorithm should find a placement which *minimizes the expression* (3).

Note that in failure-free case, the optimization algorithm tends to rather equally spread the controllers in the network, among the forwarders. To minimize (3) and considering worst case failure, the controllers tend to be placed in the center of the network. Thus, in a worst case a single controller can take over all control. However, the scenario supposed by the expression (3) is very pessimistic; a large network could be split in some regions/areas, each served by a primary controller; then some lists of possible backup controllers can be constructed for each area, as in [16].

The conclusion is that an optimization trade-off should be found, for the failure-free or failure cases. This, once again, shows that a multi-criteria approach is attractive.

(2) Nodes/links failures (Nlf):

Links or nodes failures can cause some forwarders to lose access to all controller. An objective could be to find a controller placement that minimizes the number of nodes possible to enter into controller-less situations, in various scenarios of link/node failures. A realistic assumption is to limit the number of simultaneous failures at only a few (e.g., two [11]). If more than two arbitrary link/node failures happen simultaneously, then the topology can be totally disconnected and optimization of controller placement would be no longer useful.

For *any given placement C_i* of the controllers, an additive integer value metric *$Nlf(C_i)$* could be defined, as below: consider a failure scenario denoted by f_k, with $f_k \in F$, where F is the set of all network failure scenarios (suppose that in an instance scenario, at most two link/nodes are down); initialize *$Nlf_k(C_i) = 0$*; then for each node $v \in V$, add one to *$Nlf_k(C_i)$* if the node v has no path to any controller $c \in C_i$ and add zero otherwise; compute the maximum value (i.e., consider the worst failure scenario). One obtains the formula (4) where *k* covers all scenarios of F.

$$Nlf\left(C_i\right) = \max \; Nlf_k\left(C_i\right) \tag{4}$$

The *optimization algorithm* should find a *placement which minimizes (4).* It is expected that increasing the number of controllers, will decrease the *Nlf* value. However, the optimum solution based on the metric (4) could be very different from those provided by the algorithms using the metrics (1) or (2).

(3) Load balancing for controllers

A good balance of the node-to-controller distribution is desired. A metric *$Ib(C_i)$* will measure the degree of imbalance of a given placement C_i as the *difference between the maximum and minimum number of forwarders nodes assigned to a controller.* If the failure scenarios set S is considered, then the worst case should evaluate the maximum imbalance as:

$$Ib(C_i) = \max_{s \in S} \{\max_{c \in C_i} n_c^s - \min_{c \in C_i} n_c^s\} \tag{5}$$

where n_c^s is the number of forwarder nodes assigned to a controller c. Equation (5) takes into account that in case of failures, the forwarders can be reassigned to other controllers and therefore, the load of those controllers will increase. An *optimization algorithm* should find that *placement which minimizes the expression (5).*

(4) Multiple-path connectivity metrics

One can exploit the possible multiple paths between a forwarder node and a controller [16], hoping to reduce the frequency of controller-less events, in cases of failures of nodes/links. The goal in this case is to maximize connectivity between forwarding nodes and controller instances. The metric is defined as:

$$M\,(C_i) = \frac{1}{|V|} \sum_{c \in C_i} \sum_{v \in V} ndp\,(v, c) \tag{6}$$

In (6), *ndp(v,c)* is the *number of disjoint paths* between a node v and a controller c, for an instance placement C_i. An *optimization algorithm should find the placement C_{opt} which maximizes $M(C_i)$.*

C. Inter-controller latency (Icl)

The inter-controller latency has impact on the response time of the inter-controller mutual updating. For a given placement C_i, the *Icl* can be given by the maximum latency between two controllers:

$$Icl(C_i) = \max \; d(c_k, c_n) \tag{7}$$

Minimizing (7) will lead to a placement with controllers close to each other. However this can increase the forwarder-controller distance (latency) given by (1) and (2). Therefore, a trade-off is necessary, *thus justifying the necessity to apply some multi-criteria optimization algorithms, e.g., like Pareto frontier - based ones* [10].

IV. Multi-criteria optimization Algorithm applied for controller placement problem

While particular metrics and optimization algorithms can be applied (see Section III), some criteria lead to partially contradictory controller placement solutions. As shown in introduction (and [1]) an MCDA can provide an answer. It allows selection of a trade-off solution, based on several criteria. Note that partially such an approach has been already applied by Hock et.al. [11], for some combinations of the metrics defined there (e.g., max. latency and controller load imbalance for failure-free and respectively failure use cases).

This paper uses the same variant of MCDA implementation as in [1], i.e., the *reference level (RL) decision algorithm* [10] as a general way to optimize the controller placement, while considering an arbitrary number metrics. The MCDA-RL selects the optimal solution based on normalized values of different criteria (metrics).

Given *m* objectives functions (values to be minimized) one can identify the solutions as points in an objectives space R^m, where decision parameters/variables are: v_i, i = 1, ..m, with $\forall i$, $v_i \geq 0$; the image of a candidate solution is $Sl_s=(v_{s1},v_{s2}, ..,v_{sm})$, represented as a point in R^m and where S = number of candidate solutions.

The basic MCDA-RL [10], defines two reference parameters: r_i =*reservation level*=the upper limit, which the actual decision variable v_i of a solution should not cross; a_i=*aspiration level*=the lower bound beyond which the decision variables (and therefore, the associate solutions) are seen as similar. Applying these for each decision variable v_i, one can define two values named r_i and a_i, by computing among all solutions s = 1, 2, ..S:

$$\begin{aligned} r_i &= \max\,[v_{is}],\; s = 1,2,..S \\ a_i &= \min\,[v_{is}],\; s = 1,2,..S \end{aligned} \tag{8}$$

In [10], modifications of the decision variables are proposed: *replace each variable* with *distance from it to the reservation level*: $v_i \rightarrow r_i - v_i$; (increasing v_i will decrease the distance); normalization is also introduced, in order to get non-dimensional values, which can be numerically compared. For each variable v_{si}, a ratio is computed:

$$v_{si}' = (r_i - v_{si})/(r_i - a_i), \quad \forall s, i \tag{9}$$

The factor $1/(r_i - a_i)$ - plays also the role of a weight. The variable having high dispersion of values (max – min) will have lower weights and so, greater chances to determine the minimum in the next relation (10). So, if the values *min*, *max* are rather close to each other, then solution is chosen is "good", w.r.t. that respective decision variable.

The basic MCDA-RL algorithm steps are:

Step 0. Compute the matrix M$\{v_{si}'\}$, s=1…S, i=1…m

Step 1. Compute for each candidate solution s, the minimum among all its normalized variables v_{si}':

$$\min_s = \min \{v_{si}'\};\ i = 1...m \tag{10}$$

Step 2. Make selection among solutions by computing:

$$v_{opt} = \max \{ \min_s \}, s = 1, ..S \tag{11}$$

Formula (10) selects for each candidate solution s, the worst case, i.e., the closest solution to the reservation level (after searching among all decision variables). Then the formula (11) selects among the solutions, the best one, i.e., that one having the highest value of the normalized parameter. One can also finally select more than one solution (quasi-optimum solutions in a given range). The network provider might want to apply different policies when deciding the controller placement; so, some decision variables could be "more important" than others. A simple modification of the algorithm can support a variety of provider policies. The new normalized decision variables will be:

$$v_{si}' = w_i(r_i - v_{si})/(r_i - a_i) \tag{12}$$

where $w_i \in (0,1]$ is a weight (priority), depending on policy considerations. Its value can significantly influence the final selection. A lower value of w_i represents actually a higher priority of that parameter in the selection process.

The controller placement computing procedure (given the graph, link costs/capacities, constraints, desired number of controllers, etc.) is composed of two phases:

(1)*Phase 1*: Identify the parameters of interest, and compute the *values of the metrics for all possible controller placements*, using specialized algorithms and metrics like those defined in formulas (1) - (7). This Phase will produce the set of candidate solutions (i.e., placement instances). This procedure could be time consuming (depending on network size) and therefore, could be performed off-line [9].

(2)*Phase 2*: MCDA-RL: define r_i and a_i, for each decision variable; eliminate those candidates having parameter values out of range defined by r_i; define – if wanted – convenient weights w_i for different decision variables; compute the normalized variables (formula (12)); run the MCDA Step 0, 1 and 2 of the (formulas (10) and (11)).

The decision variables could be among those of Section III, i.e.: *Worst_case (1)* or *Average* (2) *latency* (failure-free case); *Worst_case_latency_cf (3); Nodes/links failures (Nlf) (4); Controller Load imbalance (5);Multi-path connectivity metric (6); Inter-controller latency (7).*

For a particular problem, a set of relevant variables should be defined. For instance, in a high reliable network environment one could consider only failure free metrics.

V. USE CASE STUDIES AND SIMULATION RESULTS

A proof of concept simulation program (written in Python language [18]) has been constructed by the authors, to validate the MCDA–RL based controller assignment procedure and allocation of forwarders to controllers.

The input information (of the current program version) are: the network (overlay or physical) topology graph and link costs (it is supposed an additive metric representing the estimated delays, or 1/bandwidth on network links); the number of controllers wanted; decision parameters – e.g., some of the metrics (1) – (7); priorities/weights (policy derived) assigned to the decision variables; the set of possible solutions (e.g., possible placement of the controllers - candidate solutions- resulted from some other specific metric algorithms). Note that the topology and costs can be deterministic or randomly generated. The program works on all possible placements and then selects the best solution based on weighted MCDA-RL.

In [1], very simple topologies have been considered as examples. In this study, real networks are considered (see Figure 1), taken from [12]. Note that usually the backbone nodes are not supposed to be simple forwarders, but such topology provides a relevant network graph, to illustrate the solving of the SDN controllers placement problem.

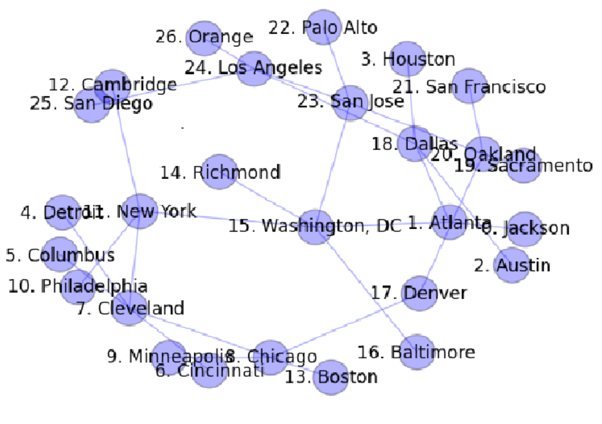

Figure 1. Real network topology example [19]

The average bandwidth of each link is estimated to 45 Mbps. From the computation point of view, one can estimate that such costs are equivalent to an additive metric normalized *to link_cost = 1*. In an equivalent way, one can assume that link latency is also normalized and can be measured relatively in few integer units. The network can be represented by an abstract graph, where each link has a cost equal to 1.

A. Controller placement

Suppose that for this network the metrics of interest and decision variables are: d1: ***W**orst latency (1,)* d2: ***A**verage latency* (2), (failure-free case); (failure-free case); d3: ***I**nter-controller latency (7).*

The reference levels are defined as (8) and in the first set of simulation we selected: r_1=6, a_1=0; r_2=3, a_2=0; r_3=6, a_3=0. For the first experiment we have chosen equal weights of the decision variables: w_i= *w*= *1*, w_2= *a*= *1*, w_3= *i*= *1*.

The total number of controllers is k=2. The first set of results illustrate the best controller placement.

In Phase 1 one should compute the metrics (1), (2) and (7) and then generate the populations of candidate solutions for controller placement. To do this, it is first necessary to know the distances between different nodes while adopting the *shortest path* approach. By using the classic *Dijkstra algorithm* the shortest distances from each node to any of others (bi-directional links are supposed) can be computed (these are shortest path trees). Then for each candidate placement solution the metrics (1), (2) and (7) are computed.

Note that for large networks the Phase 1 computing time could be large, given that a complete population of solutions should be generated. However, in this study such computations are considered to be offline, i.e., the objective is not to optimize the algorithm from this point of view (see Heller [11], for discussion of such aspects).

In Phase 2, the MCDA-RL is executed (launching command is *[atudor@localhost mcda]$ python mcda.py -w 1 -a 1 -i 1]*). The results are: *Optimum Ci placement is Ci = 176 (among the total number of solutions which is* C^{2}_{26} *= 351); Controllers are placed in node 23 and node 7.*

This is the best trade-off solution, while considering the three objectives defined by the metrics (1) (2) and (7) (see Figure 2).

B. Allocation of forwarder nodes to controllers

Once the placement of controllers is known, the allocation of the forwarder nodes to controllers should be performed. The solution applied here is a constructive one, given that a single natural criterion could be applied – i.e., to select for a forwarder the closest controller. Additionally, the allocation procedure might have a constraint: a limit for the maximum numbers of forwarders allowed to be allocated to a single controller, in order to prevent imbalances.

As an example, considering the shortest path criterion and placement of two controllers in nodes (C7, C23), the allocation of the forwarders to these controllers is shown in Figure 2, marked by different colors. No limit on the number of forwarders assigned to a given controllers have been considered in this scenario. Therefore, there is some imbalance (16 forwarders assigned to C23 and only 9 to C7). If the imbalance is considered to be too high then, additional optimizations are needed.

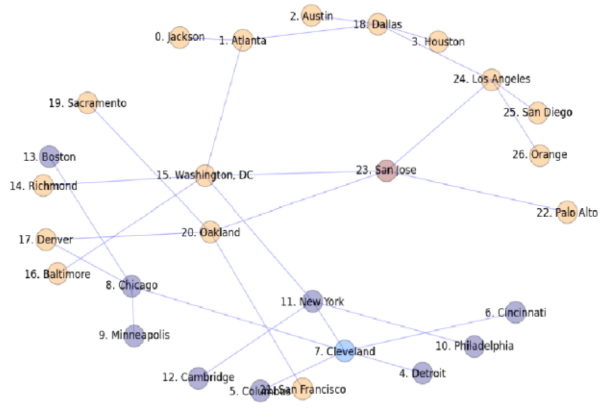

Figure 2. Controller placement and forwarder allocation (equal weights of decision variables: w_1= w_2= w_3=*1*); *k= 2 controllers*

C. Applying different weights – driven by policies

If policies should be enforced by the Network/Service Provider, then different weights can be assigned to the decision variables (see formula (12)). As an example, let us suppose that a low inter-domain latency should have higher priority than the latencies forwarders-controllers. In this case the metric (7) should have a weight <1; an example is given below, where one has the values: w= 1, a=1, i= 0.5 (the last weight is assigned to the metric (7)). The MCDA-RL will select as best, another solution for controller placement, i.e., C11 and C15 as presented in Figure 3. Note that C11 and C15 are closed to one another (C11-C15 distance = 1). However, the worst and average latencies in such case will be higher than for Figure 2 solution.

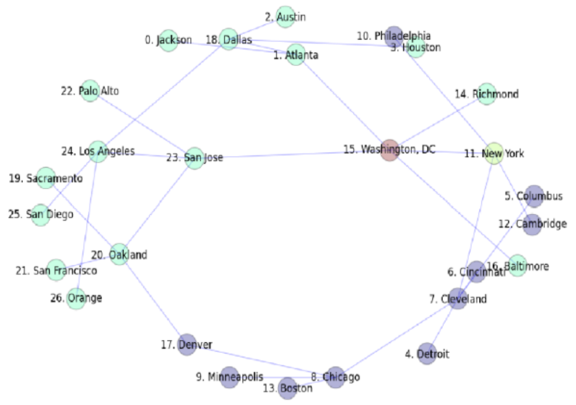

Figure 3. Controller placement and forwarder allocation with non-equal weights of decision variables: w_1= w_2=*1;* w_3=*0.5; k= 2 controllers*

D. Extension of set of objective functions

In this example, the set of metrics is more rich: in additionally to the previous three metrics, one considers the load imbalance metric (formula 5), with reference levels defined as r=6, a= 0. Also, backup controllers are assigned for each primary controller.

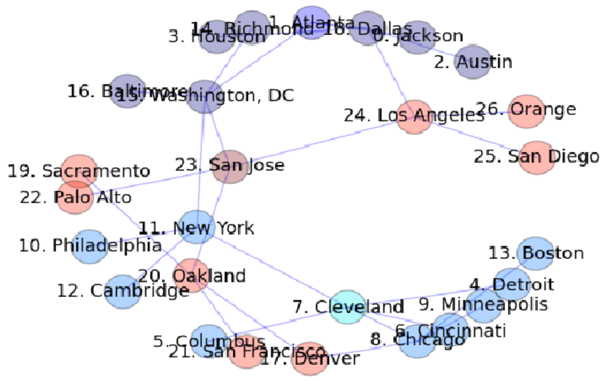

Figure 4. Controller placement and forwarder allocation with different weights of decision variables: $w_1= w_2= w_3=1; w_3=0.5; k= 3$ controllers; backup controllers are computed

To give more priority to the load imbalance metric, the weights have been selected as w_1=w=1; w_2=a=1; w_3=i=1; w_1=l=1 (for the load imbalance metric). The selected best controller placement is: C1, C7, C23. One can see (Figure 4) that allocation of forwarders to controllers is rather balanced: C1, C7, C23 have respectively 7, 9, 8 forwarders assigned to them. Also, the primary controllers C1, C7, C23 have as backup ones respectively C23, C1, C1.

VI. CONCLUSIONS AND FUTURE WORK

This paper extended the study [1], on using multi-criteria decision algorithms (MCDA) to optimally select among several controller placements solutions in WAN SDN, based on weighted criteria. The MCDA-RL can produce a tradeoff (optimum) result, while considering several criteria, part of them even being partially contradictory. The method proposed here is general and can be applied in various scenarios (including failure-free assumption ones or reliability - aware), given that it achieves an overall optimization. In this study, a simulation program has been constructed and real network topologies considered. The optimum controller placement has been found, while different weights policy-driven have been introduced. Also, forwarder-controller mapping optimization and backup controller selection have been also considered. The examples given demonstrate the flexibility of the approach in selecting the best solution while considering various criteria.

Future work will be done to apply the method proposed to other – metrics, considering multi-path approach for forwarder-controller paths hierarchical networks and studying the static/dynamic aspects of this approach.

REFERENCES

[1] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On Multi-controller Placement Optimization in Software Defined Networking -based WANs", The International Symposium on Advances in Software Defined Networks SOFTNETWORKING 2015, Barcelona, Spain, http://www.iaria.org/conferences2015/SOFTNETWORKING.html, [retrieved: 1, 2016]

[2] B. N. Astuto, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", Communications Surveys and Tutorials, IEEE Communications Society, (IEEE), 2014, 16 (3), pp. 1617 – 1634.

[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On Scalability of Software-Defined Networking", IEEE Comm. Magazine, February 2013, pp. 136-141.

[4] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in European Workshop on Software Defined Networks (EWSDN), Darmstadt, Germany, October 2012.

[5] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow" in Proc. INM/WREN, 2010.

[6] T. Koponen, et. al., "Onix: a distributed control platform for large-scale production networks," in Proc. OSDI, 2010.

[7] S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," Proc. HotSDN '12 Wksp., 2012.

[8] H. Yan-nan, W. Wen-dong, G. Xiang-yang, Q. Xi-rong, and C. Shi-duan, "On the placement of controllers in software-defined networks", ELSEVIER, Science Direct, vol. 19, Suppl.2, October 2012, pp. 92–97, http://www.sciencedirect.com/science/article/pii/S100588851 160438X, [retrieved: 1, 2016].

[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proc. HotSDN, 2012, pp. 7–12.

[10] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization". Lecture Notes in Economics and Mathematical Systems, vol. 177. Springer-Verlag, pp. 468–486.

[11] 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 ITC, Shanghai, China, 2013.

[12] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/, [retrieved: 1, 2016].

[13] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE JSAC, vol. 29, no. 9, 2011.

[14] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in GLOBECOM 2011, 2011.

[15] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability aware controller placement for software-defined networks," in Proc. IM. IEEE, 2013, pp. 672–675.

[16] L. Muller, R. Oliveira, M. Luizelli, L. Gaspary, and M. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability", IEEE Global Comm. Conference (GLOBECOM); 12/2014.

[17] D. Hochba "Approximation algorithms for np-hard problems", ACM SIGACT News, 28(2), 1997, pp. 40–52.

[18] https://www.python.org/doc/essays/blurb/, [retrieved: 1, 2016].