

On Multi-controller Placement Optimization in Software Defined Networking - based WANs

Eugen Borcoci, Tudor Ambarus, Marius Vochin

University POLITEHNICA of Bucharest - UPB

Bucharest, Romania

eugen.borcoci@elcom.pub.ro, tudorambarus@yahoo.com, mvochin@elcom.pub.ro

Abstract — Software Defined Networking (SDN) is a recent networking technology that promises important advantages in IP networking, related to flexibility at network and application level, together with powerful management and control. However, the SDN specific centralization creates scalability problems in large network environments. Multi-controller implementation of the SDN control plane for large networks can solve the scalability and reliability issues introduced by the SDN centralized logical control principle. There are still open research issues related to controllers placement, static or dynamic assignment of the network forwarding nodes to controllers, especially when network nodes/links and/or controllers failures appear or some constraints are imposed. This paper contains an analysis of some solutions proposed in the literature followed by a work in progress, on multi-criteria optimization methods applicable to the controller placement problem.

Keywords — *Software Defined Networking; Distributed Control Plane; Controller placement; Reliability; Multi-criteria optimizations.*

I. INTRODUCTION

The recently proposed Software Defined Networking (SDN) technology offers significant advantages to cloud data centers and also to Service Provider Wide Area Networks (WAN)[1]. The *basic principles* of the SDN architecture are [2][3][4]: *clear decoupling of the control and forwarding (data) planes; logically centralized control; exposure of abstract vision on network resources and state to external applications.* Thus, SDN offers an important advantage of independency of the control software w.r.t. forwarding boxes implementations offered by different vendors. Higher degree of programmability of the network control and also of applications is important consequences of the above principles.

This paper considers the case when SDN-type of control is applied in a WAN, owned by an operator and/or a Service Provider (SP).

Open research issues are related to the fact that the control-data plane separation can generate performance limitations and also reliability issues of the SDN controlled network [5][6] (note that in the subsequent text,

by “controller” it is understood a geographically distinct controller location):

(a) The forwarder nodes (called subsequently “forwarders” or simply “nodes”) must be continuously controlled, in a proactive or reactive way. The forwarders have to ask their master controllers and then be instructed by the latter, on how to process various new flows arriving to the forwarders. The result of instructions issued by the controller is filling appropriately the *flow tables* in the forwarder [2]. Such tables show what sequence of processing actions should be applied to a given data flow (identified by some specific parameters). The control communication overhead (and its inherent delay), between several forwarders and a single controller, can significantly increase the response time of the overall system. This happens because any type of controller would have eventually a limited processing capacity [5], w.r.t. the number of flow-related queries, or equivalently, the number of forwarders assigned to a controller could be too high. Therefore, the problem is: *how to distribute the controllers* as to minimize the controller-forwarder delay for a given network?

(b) The SDN control plane computes a single logical view upon the network; to this aim *the controllers must inter-communicate* and update/synchronize their data bases, in order to support the logical view construction and continuously updating of this unique vision upon the network [7][8][9]. One possible solution for inter-controller communication is to create an overlay network linking the controllers on top of the same infrastructure used by the data plane flows [10]. The problem is: how to distribute the controllers as to minimize the controller-to-controller communication delay, for a given set of controllers?

(c) Asynchronous events, such as controller failures or network disconnections between the control and data planes, may also lead to packet loss and performance degradation [5][11]. One can suppose that some forwarders are still alive (i.e., they can continue to forward the traffic flows, conforming their current flow table content). However, if the forwarders can no longer communicate with some controller, then they will have no knowledge on how to process the newly (in the future)

arrived flows. The problem is: how to distribute the controllers as to minimize the controllers and/or network links failures consequences?

There is a need to optimally place the controllers in the network. The overall objective will be an attempt to solve as much as possible of (a), (b), and (c) problems. However, this is a multi-criteria optimization problem and it was recognized as being a NP-hard one [11]. Consequently, different solutions have been proposed, targeting performance (problem (a), (b)), and performance plus reliability (problem (c)).

The paper [1] presented at The International Symposium on Advances in Software Defined Networks SOFTNETWORKING 2015, in Barcelona, Spain, contained an analysis of some solutions for (a), (b), (c) and then proposed a preliminary contribution on how *multi-criteria optimization algorithms* can be applicable to the controller placement problem. The study target was *not to develop specific algorithms*, dedicated to find an optimum solution for a *single given criterion* (several studies did that), but to *achieve an overall optimization of the controller placement*, by applying *multi-criteria decision algorithms (MCDA)* [12] [13]. The input data of the MCDA is a set of candidates (here an instance of controller placement is called a *candidate solution*).

This work is an extension of the paper [1], by refining more deeply some subjects and also by adding a novel section dedicated to simulation experiments and results.

The material is organized as follows. Section II is an overview of related work. Section III outlines several metrics and algorithms used in optimizations and present some of their limitations. Section IV develops the framework for MCDA usage as a tool for final selection of the control placement solution. Section V (which is new in this paper w.r.t. [1]) presents simulation model and a set of simulation experiments performed by the authors and results. Section VI presents conclusions and outlines the future work.

II. RELATED WORK ON SDN CONTROLLER PLACEMENT

This section is a short overview on some previously published work on controller placement in SDN - managed WANs. It is supposed that network topology and some metrics are known. The basic problems to be solved (in an optimum way) are: (1) *how many SDN controllers are needed* for that network and (2) *where the controllers should be placed* in the network. The goal is to provide enough performance (e.g., low delay for controller-to-forwarder communications) and also robustly preserve the performance level, when some controllers and/or network failures occur. Intuitively, it can be seen that some trade-off will be necessary.

In the cases of WANs having significant path delays, the controller placement determines the control plane convergence time. In other words, the placement affects

the controller's response to real-time events sensed by the forwarders, or, in case of proactive controller-initiated actions, how fast the controllers can push (in advance) the required actions to forwarding nodes.

Actually, it has been shown in [10][14] that such a problem is theoretically not new. If *latency* is taken as a metric, the problem is similar to the known one, as *facility or warehouse location problem*, solved, e.g., by using Mixed Integer Linear Program (MILP) tools.

The Heller et al. early work [11] motivates the controller placement problem and then quantifies the placement impact on real topologies like Internet2 [5] and different cases taken from Internet Topology Zoo [16]. Actually, their main goal was not to find optimal minimum-latency placements (they observed that generally, such a problem has been previously solved) – but to present an initial analysis of a fundamental design problem, still open for further study. It has been 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. This work also emphasized the fact (apparently surprising) that in most topologies, one single controller is enough to fulfill existing reaction-time requirements for some, reasonable size networks. However, resiliency aspects have not been considered in the above study.

Several works [10][14][17][18][19] have shown that resilience is important in the context of SDN and especially in the cases where additionally Network Function Virtualization (NFV) is wanted. Some resiliency-related issues have been considered in [14]:

(1) *Controller failures*: in case of a primary controller failure, it should be possible to reassign all its previously controlled nodes to a secondary closest controller, by using a backup assignment or via signaling, based on normal shortest path routing. Extreme case scenarios have been also considered, e.g., if at least one controller is still reachable; in such a case all nodes should keep functioning by communicating with it.

(2) *Network Disruption*: the failure of network links/nodes may appear, altering the topology. The routing paths (and their associated latencies) will change; a novel reassignment of nodes (forwarders) to other reachable controllers is needed. In the worst case, some parts of the network can be completely cut off, having no access to controllers. On short term this problem has no solution: such separated nodes can still forward traffic based on existing flow tables content, but they would have no more a controller to which send their request and from whom receive new instructions.

(3) *Controller overload* (load imbalance): shortest path-based assignment of the forwarders to controllers is natural. However, one should avoid that one controller have too many nodes to manage; otherwise its average response time will increase. Therefore, a well-balanced assignment of nodes to the different controllers is needed.

(4) *Inter-Controller Latency*: the SDN concepts ask for a centralized logic view of the network; therefore, inter-controller communications are necessary to synchronize

their data bases. No matter if a single flat level of controllers (e.g., like in Onix [8]) or a hierarchical topology (e.g., like in Kandoo [9]) of controllers is used, it is clear that inter-controller latency should be minimized. Therefore, an optimized controller placement should meet such a requirement.

The works [10][18] present a metric to characterize the reliability of SDN control networks. Several placement algorithms are developed and applied to some real topologies, claiming to improve the reliability of SDN control, but still keep acceptable latencies. The controller instances are chosen such that the chance of connectivity loss is minimized; connections are defined according to the shortest path between controllers and forwarding devices.

The work [19] identifies several limitations of previous studies:

- (1) forwarder-to-controller connectivity is modeled using single paths; yet, in practice, multiple concurrent connections may be available;
- (2) peaks in the arrival of new flows are considered to be only handled *on-demand*, assuming that the network itself can sustain high request rates;
- (3) failover mechanisms require predefined information which, in turn, has been overlooked.

The paper proposes the *Survivor*, i.e., a controller placement strategy that explicitly considers for network design the following elements: path diversity, controller capacity awareness, and failover mechanisms. Specific contributions consist in: significant reduction of the connectivity loss by exploring the path diversity (i.e., connectivity-awareness), which is shown to reduce the probability of connectivity loss in around 66% for single link failures; considering capacity-awareness proactively, while previous work handled requests churn on demand (it is shown that capacity planning is essential to avoid controller overload, especially during failover); smarter recovery mechanisms by proposing heuristics for defining a list of backup controllers (a methodology for composing such lists is developed; as a result, the converging state of the network can improve significantly, depending on the selected heuristic).

As stated previously, this paper does not aim to develop a new optimized placement algorithm based on a given particular metric, but to consider the multi-criteria aspect of the problem and attempt to find an overall optimization.

III. METRICS AND ALGORITHMS- SUMMARY

This section summarizes some typical metrics and objectives of the optimization algorithms for controller placement. The overall goal is to optimize the Control Plane performance. Note that, given the problem complexity, the set of metrics and algorithms discussed below is not representing an exhaustive view. Considering a particular metric (criterion) an optimization algorithm

can be applied, [10][11][14][19]. The goal of this paper is not to discuss details of such particular algorithms but searching a global optimization method. We only outline here their objectives and emphasize limitations of some particular cases.

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

The network is represented by an undirected graph $G(V, E)$ where V is the set of nodes, $n=|V|$ is the number of nodes and E is the set of edges. The edges weights represent an additive metric (e.g., *propagation latency* [11]). It is assumed that controller's locations are the same as some of the network forwarding 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 [11], 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 $C = \{C_1, C_2, \dots\}$. One can define, for a given placement C_i :

Average_latency:

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

Worst_case_latency:

$$L_{wc} = \max_{v \in V} \min_{c \in C_i} d(v, c) \tag{2}$$

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

Figure 1 shows a simple example of a network having six nodes.

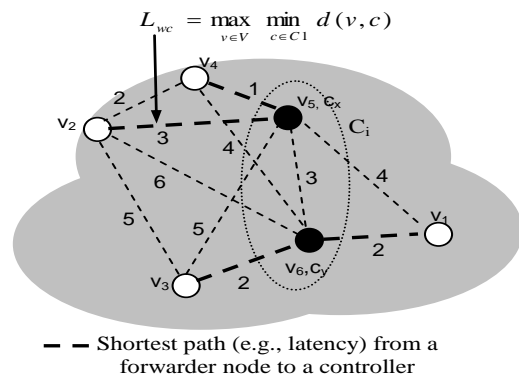


Figure 1. Simple network example of controller placement and nodes-to-controller assignment: v = forwarder node; c = controller; $C_1 = \{[c_x \text{ in } v_5 (v_5, v_2, v_4)], [c_y \text{ in } v_6 (v_6, v_1, v_3)]\}$

Note that Figure 1 could represent not quite the real network infrastructure, but an *overlay*, in which only a

subset of the total number of real nodes appear. This assumption makes the model more general in the sense that a hybrid network could be supposed as being the infrastructure supporting the overlay. A subset of nodes (routers) can be SDN forwarders and in some of them SDN controllers could be hosted. Two controllers $\{c_x, c_y\}$ can be placed in any location of the six nodes, e.g. in $\{v_5, v_6\}$. This placement instance is denoted by C_1 . The distances between different nodes (overlay paths) are marked on the graph.

How to assign the forwarders to controllers? If one considers C_1 and the “shortest path” as a criterion of selection, then the allocation of forwarders to controller will be:

$$C_1: c_x = v_5 (v_5, v_2, v_4), c_y = v_6 (v_6, v_1, v_3)$$

Note that a particular network node could play the both roles, as forwarder and controller.

Some limitations of this optimization process are:

- No reliability awareness: the metrics are pure distances, which in the simplest case are considered as being static values - despite that *delay* is usually a dynamic value in IP networks.
- There is no upper limit on the number of v nodes assigned to a controller; too many forwarders controlled by the same controller could exist, especially in large networks.

Other metric possible to be considered in failure-free case is *Maximum cover* [11][20]. 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.

All metrics and algorithms described above do not take into account the inter-controller connectivity, so their associated optimizations can be seen as being partial.

B. Reliability aware metrics

Several studies consider more realistic scenarios in which controller failures or network links/nodes failures might exist. The optimization process aims now to find trade-offs (related to failure-free scenarios) in order to preserve a convenient behavior of the overall system in failure cases.

(1) Controller failures (cf)

The work [14] observes that the node-to-controller mapping can change in case of controller outages. So, a realistic 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, in [14] the failures are modeled by constructing a set C of scenarios, including all possible combinations of faulty controller number, from 0 up to $k - 1$. The resulting maximum latency will be:

$$\text{Worst_case_latency_cf}:$$

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

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

Commenting the placement results based on the metric (1) or (2), one can observe that in failure-free case, the optimization algorithm tends to rather equally spread the controllers in the network, among the forwarders nodes. However, when attempting to minimize the expression (3) (and considering worst case failure), the controllers tend to be placed in the centre of the network. Thus, even if all controllers (except for one) fail, the latencies are still satisfactory (numeric examples are given in [14]). On the other side, one can criticize such an approach if applied to large networks; the scenario supposed by the expression (3) is very pessimistic; it is more realistic that a large network will 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 proposed in [19].

The conclusion is that a trade-off exists, between the placements optimized for the failure free case and those including controller failures. It is a matter of operator policies to assign weights to different criteria, before deciding (based on multiple criteria) the final selection of placement solution.

(2) Nodes/links failures (Nlf)

Links or nodes failures might produce network disruptions; some forwarders could have no more access to any controller. Therefore, an optimization objective could be to find a controller placement, which 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 [14]). 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 more efficient.

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 any 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). We get:

$$Nlf(C_i) = \max Nlf_k(C_i) \quad (4)$$

where k covers all scenarios of F .

The *optimization algorithm* should find that *placement which minimizes* (4). It is naturally expected that

increasing the number of controllers, will decrease the Nlf value. We also observe that 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 well designed system would require roughly equal load on all controllers, i.e., a good balance of the node-to-controller distribution. A metric can be defined to measure the degree of imbalance $Ib(C_i)$ of a given placement C_i as the difference between the maximum and minimum number of forwarder nodes assigned to a controller. If the failure scenarios set S is considered, then the worst case should evaluate the maximum imbalance as:

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

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 than the primary ones 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 [19], hoping to reduce the frequency of controller-less events, in cases of failures of nodes/links. The goal is to maximize connectivity between forwarding nodes and controller instances. The metric is defined as:

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

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:

$$Icl(C_i) = \max d(c_k, c_n) \quad (7)$$

Minimizing (7) will lead to a placement with controllers close to each other. However, this can increase the forwarder-to-controller distance (latency) given by (1) or (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.

D. Constraints

Apart from defining the metrics, the controller placement problem can be subject to different constraints. For instance, in [19], the input data for the optimal controller placement algorithm consists in the graph $G(V, E)$ information, set of possible controller instances C , request demand of a network device, each controller capacity, and a backup capacity for each controller. Integer Linear Programming (ILP) - based algorithm is applied; here the constraints can be split into three classes: placement-related, capacity related and connectivity-related. In general other limits can be defined, e.g., on maximum admissible latency, ratio number controller/trivial nodes, pre-defined regions for controllers, etc. They should be included in the respective algorithms.

IV. MULTI-CRITERIA OPTIMIZATION ALGORITHM

Sections II and III have shown that several criteria of optimum can be envisaged when selecting the best controller placement in a WAN. While particular metrics and optimization algorithms can be applied (see Section III), we note that some criteria lead to partially contradictory placement solutions. What approach can be adopted? The answer can be given by adopting a multi-objective optimization based on Multi-Criteria Decision Algorithms (MCDA). The good property of MCDA is that it allows selection of a trade-off solution, based on several criteria. Note that, partially, such an approach has been already applied in [14], for some combinations of the metrics defined there (e.g., max. latency and controller load imbalance for failure-free and respectively failure cases).

A. Reference level MCDA

This paper proposes to apply MCDA, as a general way to optimize the controller placement, while considering not only a single metric but an arbitrary number of them.

The multi-objective optimization problem [12][13] is to minimize $\{f_1(x), f_2(x), \dots, f_m(x)\}$, where $x \in S$ (set of feasible solutions), $S \subset \mathbb{R}^n$. The decision vector is $x = (x_1, x_2, \dots, x_n)^T$. There are ($m \geq 2$) possibly conflicting objective functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$, and we would want to minimize them simultaneously (if possible). In controller placement problem we might have indeed some partially conflicting objectives (e.g., to minimize the inter-controller latency and the forwarder-controller latency).

One can define Objective vectors as images of decision vectors. The objective (function) values are given by $z = f(x) = (f_1(x), f_2(x), \dots, f_m(x))^T$. We denote as feasible objective region $W = f(S) = \text{image of } S \text{ in the objective space}$.

Objective vectors are optimal if none of their components can be improved without deterioration to at least one of the other components.

A decision vector $x_- \in S$ is named *Pareto optimal* [12] if it does not exist another $x \in S$ such that $f_i(x) \leq f_i(x_-)$ for all $i = 1, \dots, m$ and $f_j(x) < f_j(x_-)$ for at least one index j .

We adopt here the MCDA variant called *reference level decision algorithm* [13]. It has the advantage to allow selection of the optimal solution while considering normalized values of different criteria (metrics).

We use a simplified notation:

- identify the solutions directly by their images in the objectives space R^m ,
 - decision parameters/variables are: v_i , $i = 1, \dots, m$, with $\forall i, v_i \geq 0$,
 - image of a candidate solution is $SI_s = (v_{s1}, v_{s2}, \dots, v_{sm})$, represented as a point in R^m ,
- S = number of candidate solutions.

Note that the value ranges of decision variables may be bounded by given constraints. The optimization process consists in selecting a solution satisfying a given objective functions and conforming a particular metric.

The basic *reference level algorithm* [13], defines two reference parameters:

- r_i =*reservation level*=the upper limit for a decision variable, which the decision variable v_i of a solution should not cross;
- a_i =*aspiration level*=the lower bound beyond which the decision variable (and therefore the associate solutions) are seen as similar.

Without loss of generality one may apply the definitions of [13], where for each decision variable v_i there are defined two values named r_i and a_i by computing among all solutions $s = 1, 2, \dots, S$:

$$\begin{aligned} r_i &= \max [v_{is}], s = 1, 2, \dots, S \\ a_i &= \min [v_{is}], s = 1, 2, \dots, S \end{aligned} \quad (8)$$

In [13], 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:

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

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). In other words, less preference is given to those decision variables having close values to each other (among different candidate solutions), i.e., if the values *min*, *max* are close enough, then it does not matter which solution is chosen by considering the respective decision variable.

The basic algorithm steps are:

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

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

$$\min_s = \min \{v_{si}'\}; i=1 \dots m \quad (10)$$

Step 2. Make selection among solutions by computing:

$$v_{opt} = \max \{ \min_s \}, s=1, \dots, S \quad (11)$$

Note that the formula (10) selects for each candidate solution s , the worst case, i.e., that solution being closest to the reservation level (after searching among all decision variables). Then the formula (11) selects among the solutions, the best one, i.e., that having the highest value of the normalized parameter.

This v_{opt} is the optimum solution, i.e., the MCDA selects the best value among those produced by the Step 1. In the case that several equal values exist in the Step 2, a random selection can be adopted or some other additional discrimination criterion. Note also, that it is no problem for the Step 2, to consider more than one solution, i.e., a set of several quasi-optimum solutions can be selected.

B. MCDA- Controller placement optimization

In this section, we *apply the reference level algorithm to the controller placement problem*. However, we modify the basic algorithm to be better adapted to controller placement problem, due to following remarks:

(1) The step 2 *compares values coming from different types of parameters/metrics* (e.g., max. latency, load imbalance, etc.) having different nature and being independent or dependent on each other. The normalization still allows them to be compared in the $\max\{ \}$ formula. *This is an inherent property of the basic algorithm.*

(2) However, the network provider might want to apply different policies when deciding the controller placement. In such policies, some decision variables (or metrics) could be "more important" than others. For instance, in some cases, the performance is more important, in others high resilience is the major objective.

A simple modification of the algorithm can support a variety of provider policies. We propose a modified formula:

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

where the factor $w_i \in (0, 1]$ represents a weight (priority) that can be established from network provider policy considerations, and can significantly influence the final

selection. Note that a lower value of w_i actually represents a higher priority of that parameter in the selection process.

The controller placement problem solving (given the graph, link costs/capacities, constraints, desired number of controllers, etc.) is composed of two macro-steps:

(1) *Macro-step1*: Identify the parameters of interest, and compute the values of the metrics for all possible controller placements, using specialized algorithms and metrics (1) - (7). In other word this step will produce the set of candidate solutions (i.e., placement instances).

This procedure could be time consuming (depending on network size) and, therefore, performed off-line [11].

(2) *Macro-step2*: MCDA

- define reservation and aspiration levels for each decision variable;
- eliminate those candidates having out of range parameter values defined by the reservation level;
- define appropriate weights (see formula (12)) for different decision variables - depending on the high level policies applied by the operator;
- compute the normalized variables (formula (12))
- run the Step 0, 1 and 2 of the MCDA algorithm (formulas (10) and (11)).

The decision variables can be among those of Section III, i.e.:

Average (1) or worst (2) case 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 selection of relevant variables should be done. For instance, in a high reliable network environment one could consider only failure free metrics.

C. Numerical example – MCDA optimization

A simple but relevant example is exposed to illustrate the MCDA power, based on the network in Figure 1. Suppose that for this network the metrics of interest and decision variables are (see Section III) on:

d1: *Average latency* (1), (failure-free case);

d2: *worst latency* (2,) (failure-free case);

d3: *Inter-controller latency* (7).

The reference levels are defined as in formula (8) and we propose: $r_1=3$, $a_1=0$; $r_2=6$, $a_2=0$; $r_3=6$, $a_3=0$.

Several placement samples can be considered:

$C_1 = \{ [c_{x_in_v5}(v_5, v_2, v_4)], [c_{y_in_v6}(v_6, v_1, v_3)] \}$

$C_2 = \{ [c_{x_in_v5}(v_5, v_1, v_2, v_4)], [c_{y_in_v3}(v_3, v_6)] \}$

$C_3 = \{ [c_{x_in_v3}(v_3, v_2)], [c_{y_in_v6}(v_6, v_1, v_4, v_5)] \}$

$C_4 = \{ [c_{x_in_v4}(v_4, v_2, v_5)], [c_{y_in_v6}(v_6, v_1, v_3)] \}$

1. *MCDA with equal priorities for $d1=1$, $d2=1$, $d3=1$* . The values of the metrics are computed using equations (1), (2) and respectively (7) for each placement: $C_1, ..C_4$.

A matrix $M(3 \times 4)$ is computed using the formulas (9). MCDA is applied by using formulas (10), (11). The final result is: $C_1 = \text{the best placement}$. Looking at Figure 1, we indeed can see that this placement is a good trade-off between node-controller latency and inter-controller latency.

2. *MCDA with different priorities for i.e. $d1=1$, $d2=0.5$, $d3=1$, i.e., the worst case latency $d2$ has highest priority, i.e., the solution minimizing the worst case controller - forwarder latency with high priority is desired*. After re-computing the matrix M and applying MCDA equations (1), (11), we find $C_4 = \text{the best placement}$. Indeed, we see in Figure 1 that worst case latency (node-controller) is minimized, however, the inter-controller latency is higher than in C_1 .

These examples proved how different provider policies can bias the algorithm results.

V. USE CASE STUDIES

A proof of concept simulation software program has been constructed to validate the above MCDA – based controller assignment procedure. This preliminary version of the program has been written in Python language [21] and uses the NetworkX software package [22] for the creation, manipulation and study of network graphs. The program has two running modes.

Static: in these modes the inputs are:

- the network (overlay) topology graph and link costs (it is supposed an additive metric representing the estimated delays on overlay network links);

- the number of controllers wanted;

- decision parameters – e.g., $d1$, $d2$, $d3$ of the previous section;

- priorities/weights assigned to the decision variables that comply with the network provider policy;

- the possible placement of the controllers (i.e., candidate solutions of the MCDA) (considering them as results of some other algorithms)

Dynamic: in this mode the following parameters can be selected:

- total number of network nodes (N)

- desired number of controllers

- the link costs (e.g., randomly assigned).

The program computes all possible placements and then selects the best solution based on weighted MCDA algorithm.

The program parses user arguments, constructs the weighted graph and associated candidate solutions,

computes Dijkstra's shortest path lengths between all nodes in the graph, and then applies the MCDA controller placement optimization algorithm. The flowchart of the program is depicted in Figure 2.

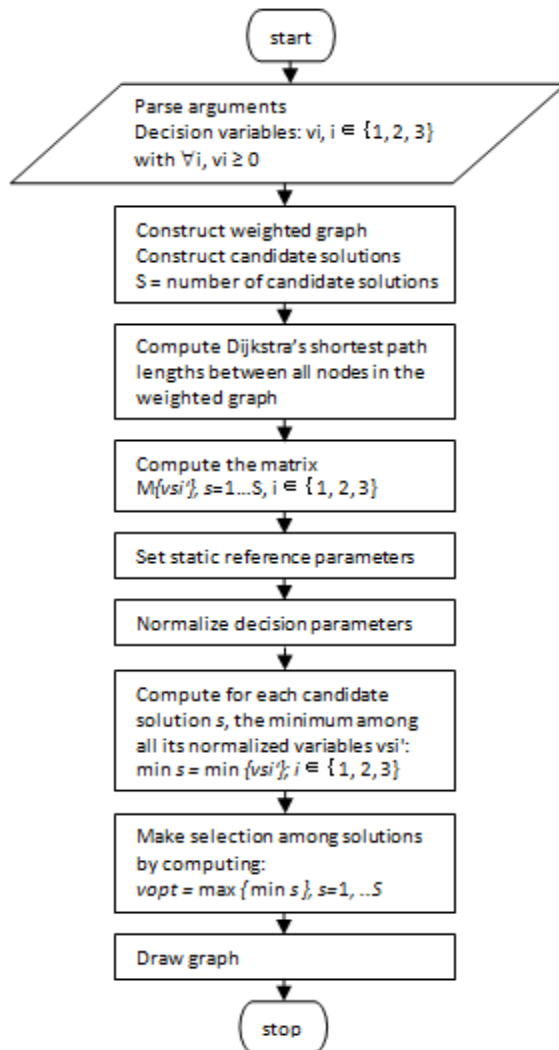


Figure 2. Flowchart for MCDA controller placement simulation program.

Generally the order of complexity of an algorithm is an important issue to be considered if its implementation is targeted. In our case the following particular characteristics of the optimization problem are valid:

(a) - this optimization computation has no real time requirements; it could be performed offline for a given fixed network (topology and costs are known or estimated).

(b) - the complexity order of the Dijkstra part of the algorithm is the well known to be $O(|V|^2)$, or $O(|E| + |V| \log|V|)$, if a more efficient implementation is chosen, where $|V|$ is the number of nodes and $|E|$ is the number of edges.

(c) the evaluation of the metrics (1)–(7) supposes that for every particular mono-criteria algorithm, all possible placements should be considered. For large networks this is given in the worst case by the number of combinations C_n^k , (n = number of nodes; k = number of controllers) which increases very much with the number n of network nodes (usually it is true that $k \ll n$). However this problem is common to all algorithms and it is not particular to the algorithm developed in this paper.

(d) the MCDA algorithm itself, has to construct a matrix having the dimension $NL * NC$, where:

- NL is the number of lines - equal to the number of decision variables (e.g. three in the example given in the Section IV.C);

- NC is the number of columns - equal to the number of candidates' solutions (variants of controller placements).

The number NL is usually small (e.g., $NL=7$ in this study). The number NC could be large, given by C_n^k , which is the number of possible combinations of controller placements. The Stirling formula $n! \approx (2\pi n)^{1/2} (n/e)^n$ shows a strong increase of the number of controller placements C_n^k with n . However, some practical considerations and/or policies could reduce significantly the actual number of combinations to be considered. It is expected that large networks will be split in some disjoint regions having significantly less than n nodes and a number of reduced number k_R controllers will be allocated for a given region. In the work [11] the authors even say that "...in many medium-size networks, the latency from every node to a single controller can meet the response-time goals of existing technologies, such as SONET ring protection...". If the number of regions is R , with $R > 1$ then the number of nodes n will be reduced at n/R , strongly decreasing the number of combinations. Some other restrictions imposed from policy considerations could also reduce the total number of nodes to be considered in the formulas (1) – (7).

The interface for running the simulation program is presented below.

```

[b38632@localhost mcda]$ python mcda.py --help
usage: mcda.py [-h] [-a A] [-w W] [-i I] [--dynamic] [-n N] [-c C]

Multi-criteria optimization algorithm

optional arguments:
  -h, --help show this help message and exit
  -a A       Average latency - failure free scenario. Expects a weight
            (priority) in interval (0, 1].
  -w W       Worst case latency - failure free scenario. Expects a weight
            (priority) in interval (0, 1].
  -i I       Inter controller latency. Expects a weight (priority) in
            interval (0, 1].
  --dynamic  Generate dynamic undirected graph
  -n N       Number of graph nodes. Valid only in dynamic mode.
  -c C       Number of controllers in graph. Valid only in dynamic mode.
            Allowed values are between N/3 and N/7
[b38632@localhost mcda]$
  
```

Some samples of simulation results are given below.

Figure 3 shows the results obtained for the network presented in Figure 1, while decision variables are d_1, d_2, d_3 defined in Section IV. In this use case, the highest priority is given to the worst case latency parameter ($d_2 = 0.5$), while the two others have $d_1=d_3=1$. One can see that the best solution selection is the setup C4, i.e., the same results as analytically estimated in Section IV.

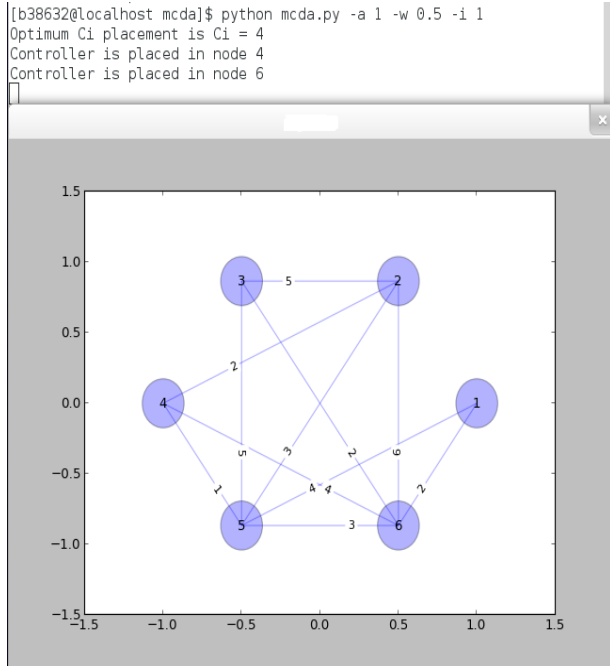


Figure 3. Simulation results for Figure 1 network; $d_1=1, d_2=0.5, d_3=1$.

Figure 4 shows another instance of use case, where the graph is dynamically generated, with $N=7$ nodes and $k=2$ controllers. The decision parameters have equal priorities: $d_1 = 1, d_2 = 1, d_3 = 1$. The best placement is denoted by C_3 and places the controllers in the nodes having the number 0 and 4 respectively. Note that the network of Figure 4 is a full mesh one. This is the effect of considering the overlay of paths. However, the algorithm and the program can work as well with partial mesh overlay graph.

Figure 5 shows a quantitative extension for a larger graph – dynamically generated with $N=14$ nodes and $k = 5$ controllers. The picture exposes a symmetrical figure due to full mesh connectivity between nodes. The link costs are randomly generated. The total number of possible placements is C_N^k . The MCDA selects the placement no. 55, indicated in the text associated to the figure.

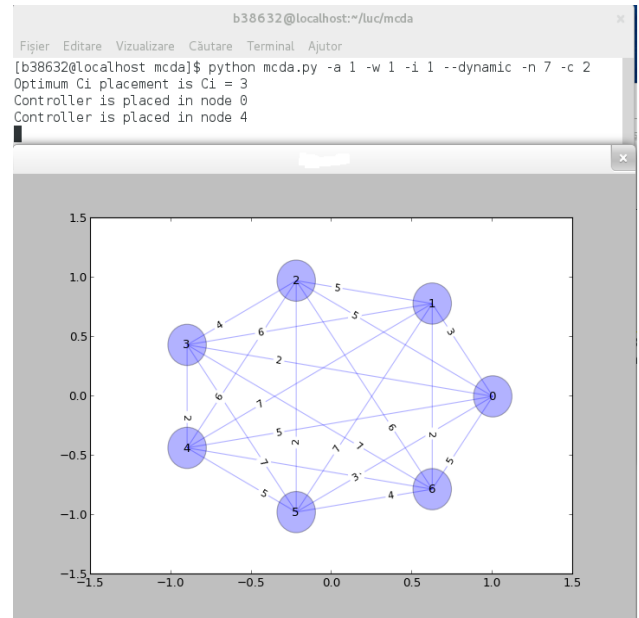


Figure 4. Simulation results for a network having $N=7$, network; $d_1=1, d_2=1, d_3=1$.

VI. CONCLUSIONS

This paper presented a study on using multi-criteria decision algorithms (MCDA) for final selection among several controller placements solutions in WAN SDN, while considering several weighted criteria. The MCDA quality is that it can produce a tradeoff (optimum) result, while considering several criteria, part of them even being partially contradictory.

A simulation program has been created to demonstrate the validity of results. The topology and link costs are generally overlay ones, and can be introduced in a particular way or randomly generated.

The method proposed is generic enough to be applied in various scenarios (including failure-free assumption ones or reliability aware), given that it achieves an overall optimization, based on multiple metrics supported by the reference model MCDA. Different network/service provider biases can be introduced in the selection process, by assigning policy-related weights to the decision variables. This simple algorithm modification creates a rather powerful tool to bias the selected solution, as to respond to the provider policy.

Future work will be done to apply the method proposed to very large networks - real life case studies (e.g., from Internet Topology zoo, [16]) and comparing the quality of trade-offs when defining different weights to decision variables.

```
[b38632@localhost mcda]$ python mcda.py -a 1 -w 1 -i 1 --dynamic -n 14 -c 5
Optimum Ci placement is Ci = 55
Controller is placed in node 0
Controller is placed in node 1
Controller is placed in node 3
Controller is placed in node 4
Controller is placed in node 5
```

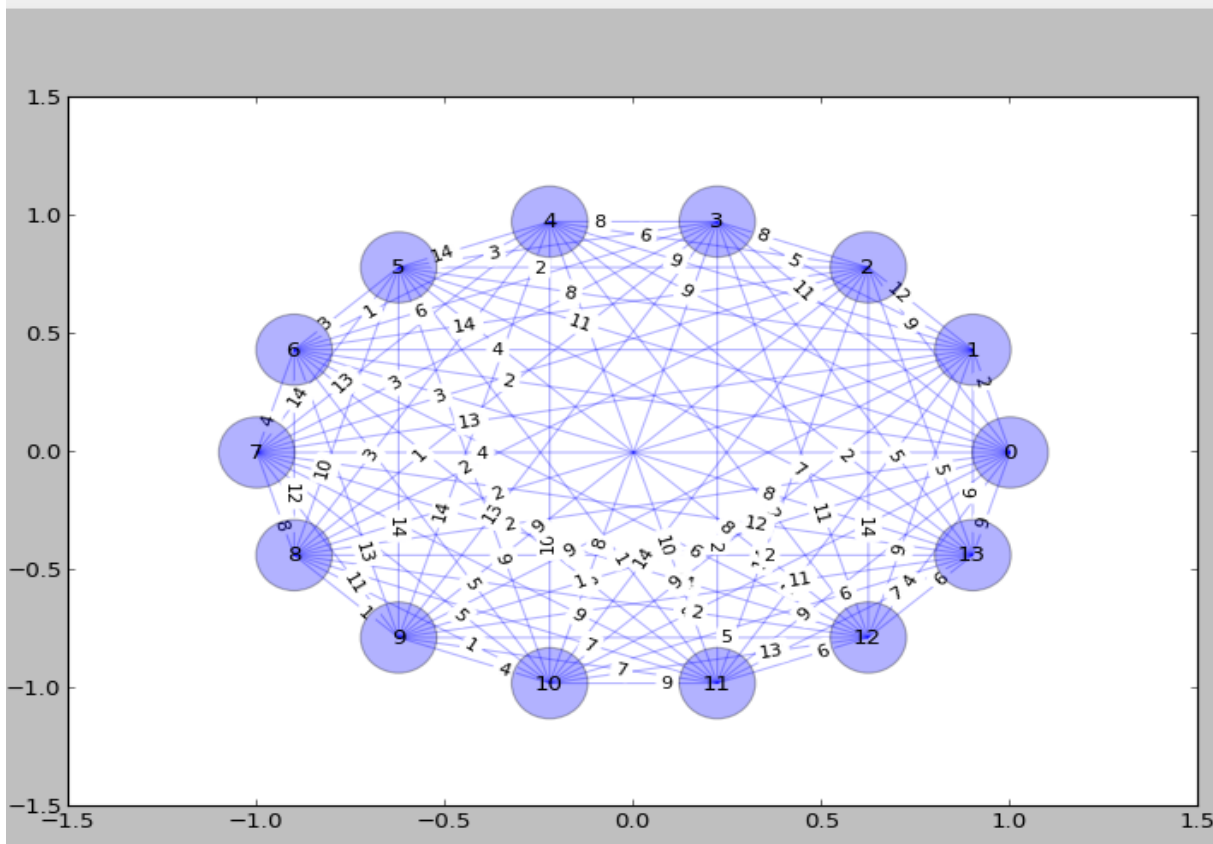


Figure 5. MCDA for a dynamic graph with 14 nodes (5 controllers) and equal priorities; $d1=1, d2=1, d3=1$.

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, 2015 - Barcelona, Spain, <http://www.iaria.org/conferences2015/SOFTNETWORKING.html>
- [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] "Software-Defined Networking: The New Norm for Networks" ONF White Paper 04.2012; retrieved: 02.2015 <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>.
- [4] "SDN: The Service Provider Perspective," Ericsson Review, 21.02.2013. retrieved: 02.2015 http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2013/er-software-defined-networking.pdf.
- [5] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, "On Scalability of Software-Defined Networking," IEEE Comm. Magazine, February 2013, pp. 16-141..
- [6] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in European Workshop on Software Defined Networks (EWSNDN), Darmstadt, Germany, October 2012.

- [7] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow" in Proc. INM/WREN, 2010.
- [8] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in Proc. OSDI, 2010.
- [9] S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," Proc. HotSDN '12 Wksp., 2012.
- [10] H. Yan-nan, W. Wen-dong, G. Xiang-yang, Q. Xi-rong, 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/S100588851160438X>.
- [11] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proc. HotSDN, 2012, pp. 7–12.
- [12] J. Figueira, S. Greco, and M. Ehrgott, "Multiple CriteDecision Analysis: state of the art surveys," Kluwer Academic Publishers, 2005.
- [13] 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.
- [14] 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.
- [15] Internet2 open science, scholarship and services exchange. <http://www.internet2.edu/network/ose/>.
- [16] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE JSAC, vol. 29, no. 9, 2011.
- [17] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in GLOBECOM 2011, 2011.
- [18] 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.
- [19] L. Muller, R. Oliveira, M. Luizelli, L. Gaspary, M. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability," IEEE Global Comm. Conference (GLOBECOM); 12/2014.
- [20] D. Hochba "Approximation algorithms for np-hard problems", ACM SIGACT News, 28(2), 1997, pp. 40–52.
- [21] <https://www.python.org/doc/essays/blurb/>
- [22] <https://networkx.github.io/>