Reliability-aware Optimization of the Controller Placement and Selection in SDN Large Area Networks

Eugen Borcoci, Stefan Ghita
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
Emails: eugen.borcoci@elcom.pub.ro, stalghita@gmail.com

***Abstract* — For large networks SDN-controlled, distributed control plane solutions are proposed, to solve the scalability problems generated by the SDN control centralization principle. In a multi-controller environment, the Controller Placement Problem (CPP) should be solved. Additionally, in a dynamic networking context, including possible failures of links or nodes, a forwarder node could try to select an available and reachable controller among those alive. Although several studies have been published, the above problems are still open research issues, given the various network contexts, providers' policies and possible multiple optimization criteria. Multi-criteria decision algorithms can provide valuable solutions. This paper extends a previous work, considering in the developed model some reliability aspects of the distributed SDN control and an extension to a dynamic controller selection method.**

Keywords — Software Defined Networking; Multi-criteria optimization; Controller placement; Controller selection; Forwarder nodes assignment; Reliability;

I. INTRODUCTION

Software Defined Networking (SDN) has as basic principles the decoupling of the architectural *Control Plane* (CPl) w.r.t *Data Plane* (DPl) and also CPl centralization in SDN controllers. In the case of large network environments, scalability problems of the CPl appear [1]. The usual solution for this is a distributed multi-controller implementation of the SDN control plane. Different flat or hierarchical organizations for a multi-controller SDN control plane have been developed, e.g., in [2][3].

Note that in a basic approach, the *SDN controller* (SDN-C) is understood as a control entity placed in a geographically distinct location, i.e., a particular physical network node. However, recently, the *Network Function Virtualization* technologies [4] allow that several logical SDN-Cs realized in a virtual manner (notation will be vSDN-C) can be collocated in the same physical node. In the following text we suppose the basic approach; however the models developed in this paper can be as well applied also to a virtualized environment.

Actually, several associated problems exist together with *controller placement problem* (CPP) itself. Some examples are: how the network topology is specified - flat or clustered; what criteria are considered to solve the CPP; number of controllers - predefined or not; failure-free or failure-aware metrics (e.g., considering backup controllers and node/link failures); how the DPl forwarders nodes are assigned to controllers (in static or dynamic way, i.e., depending on actual network conditions and network provider policies), and others. The evaluation of the degree of optimality of different approach can be studied on some simplified topologies – in order to compare the efficiency of approaches or, on real specific network topologies. Several studies [5-15] considered various aspects and solutions of the CPP problem.

In a real network environment, it has been apparent that there is no uniform and strict placement rule to be the best for any SDN-controlled network. Dynamic nodes addition and deletion can happen and, in such cases, a forwarder could dynamically select an appropriate controller, if it has enough pertinent and updated information. This is *called controller selection problem* (CSP) and can be considered as an extension of the CPP [11].

The CPP is a non-polynomial (NP) -hard problem [5]; therefore different pragmatic solutions have been proposed, many of them with specific optimization criteria, targeting performance in failure-free or failure-aware approaches. Examples of specific, individual criteria could be: to maximize the controller-forwarder or inter-controller communication throughput; reduce the latency of the path connecting them; limit the controller load imbalance; 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.

One main issue is that different optimization criteria could lead to different solutions; so, a multi-criteria global optimization could be a better approach.

The paper [12] provides a contribution on multi-criteria optimization algorithms for the CPP not by developing specific single-criterion algorithms (many other studies already did that) but to achieve an overall optimization by applying *multi-criteria decision algorithms* (MCDA) [16]. The input of MCDA is the set of candidates (an instance of controller placement is called a candidate solution). Examples have been analyzed, on some real network topologies, proving the usefulness of the approach.

 ISBN: 978-1-61208-669-9

This paper extends the model of [12]*; several reliability aware criteria have been added to the CPP solution. Also the novel CSP extension is introduced, being appropriate for a dynamic network context. It is shown that the same basic MCDA can be applied in both static and dynamic context, but with different sets of criteria. Simulation experiments and novel results are presented.*

The structure of the paper is described here. Section II is a short overview of related work. Section III revisits several metrics and optimization algorithms and presents some of their limitations. Section IV revisits the framework for MCDA-RL (the variant which is called "*reference level*") as a simple but powerful tool to solve the CPP and CSP problems. Section V presents the implementation performed to validate the MCDA proposed model in reliability-aware approach, and outlines the simulation experiments performed. Section VI offers few samples of simulation results to illustrate the validity of the approach. Section VII presents conclusions and future work.

II. RELATED WORK

This short section is included mainly for references. More comprehensive overviews on published work on CPP in SDN-controlled WANs are given in [10-13]. The goal is to find those controller placements that provide high performance (e.g., low delay for controller-forwarder communications) and also create robustness to controllers and/or network failures.

Heller et al. [5] have early 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 [6][7][8][9][14] 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 [6][8] developed several algorithms for real topologies, considering 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. [9] 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. Hock et.al. [7] adopted a multi-criteria approach for some combinations of the metrics (e.g., max. latency and controller load imbalance for failure-free and respectively failure use cases).

In a recent work [11], K.Sood and Y.Siang propose to transform the CPP problem into *Controller Selection Problem* (CSP), i.e., consider the dynamics of the network and make controller selection. They explore the relationship between traffic intensity, resources requirement, and QoS requirements. It is claimed that to optimize the control layer performance, the solutions must be topology-independent and adaptive to the needs of the underlying network behaviour. They propose a topology independent framework to optimize the control layer, aiming to calculate the optimal number of controllers to reduce the workload, and investigate the placement/location of the controllers. However, their first declared objective has been not to determine the optimal placement of controllers in the network, but to motivate the CSP.

In [12], a multi-criteria algorithm is used (applicable for an arbitrary number of decision criteria) to solve the CPP; validation of results have been presented for some real network topologies [17][18].

This paper extends the [12] work, by adding new reliability–aware metrics and also outlines the usage of the multi-criteria method to solve the controller dynamic selection problem (CSP).

III. EXAMPLES OF CONTROLLER PLACEMENT METRICS AND ASSOCIATED ALGORITHMS

This section is a short presentation of a few typical metrics and optimization algorithms for CPP and CSP. A more detailed presentation of them can be found in [12]. Considering a particular metric (criterion) an optimization algorithm can be run for a given metric, as in [5][6][7][9].

However, this paper goal is not to develop a new particular algorithm based on a given single metric, but to search for a global optimization. The individual metrics presented in this section can be embedded in a multi-criteria optimization algorithm.

The SDN-controlled network is abstracted by an undirected graph G(V, E), with V - set of nodes, E – set of edges and n=|V| the total number of nodes. The edges weights represent an additive metric (e.g., *propagation latency* [5]).

A basic metric is *d(v, c): shortest path* distance from a forwarder node $v \in V$ to a controller $c \in V$. We denote by C_i a particular placement of controllers; $C_i \subseteq V$ and $|C_i| < |V|$. 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 $C = \{C_1, C_2 \ldots\}$. Some metrics are basic, i.e., failure-free; others take into account failure events of links or nodes.

An important metric for SDN control is the *latency* between nodes. Note that, while it has a dynamic nature, in some simplified assumptions it is estimated as a static value.

A. Failure-free scenarios

- *Forwarder-to-controller latency*

In Heller's work [5], two (failure-free) metrics are defined for a given placement Ci: *Worst_case_latency* and *Average_latency* between a forwarder and a controller. An optimization algorithm should find a placement C_{opt}, where either *average latency* or *the worst case latency is minimized*.

The work [15] proposes an algorithm 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, but with an upper latency bound of each forwarder latency to its controller.

- *Inter-controller latency*

The SDN controllers should inter-communicate and therefore the inter-controller latency is important. For a given placement C_i, one can minimize the *maximum latency between two controllers*. Note that this can increase the forwarder-controller distance (latency).Therefore, a trade-off is necessary, thus justifying the necessity to apply some multi-criteria optimization algorithms, e.g., like Pareto frontier - based ones [7].

B. Failure-aware scenarios

In such scenarios controller and/or network failures events are considered. The optimization process aims now to find trade-offs to preserve a convenient behavior of the overall system in failure cases (controllers, or nodes, or links).

- *Multiple-path connectivity metrics*

If multiple paths are available between a forwarder node and a controller [9], this can exploited in order to reduce the occurrence 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. A special metric can be defined as:

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

The *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* $M(C_i)$.

- *Controller failures*

To minimize the impact of such failures, the latency-based metric should consider both the distance to the (primary) controller and the distance to other (backup) controllers. For a total number of *k* controllers, the failures can be modeled [7], by constructing a set C of scenarios, including all possible combinations of faulty controller number, from 0 of up to k - 1. The *Worst_case_latency_cf* will be:

$$L_{wc-cf} = \max_{v \in V} \max_{C_i \in C} \min_{c \in C_i} d(v,c) \quad (2)$$

The optimization algorithm should find a placement which *minimizes the expression* (2).

Note that in failure-free case, the optimization algorithm tends to rather equally spread the controllers in the network, among the forwarders. To minimize (2), the controllers tend to be placed in the center of the network, such that in a worst case, a single controller can take over all control. However, the scenario supposed by the expression (2) 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 [9]. The conclusion is that an optimization trade-off should be found, for the failure-free or failure cases. A multi-criteria approach can provide the solution.

- *Nodes/links failures*

For such cases, the 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 [7]). 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 a *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).

$$Nlf(C_i) = \max Nlf_k(C_i) \quad (3)$$

The optimization algorithm should find a placement *to minimize* (3), where *k should* cover all scenarios of F. It is expected that increasing the number of controllers, will decrease the *Nlf* value. However, the optimum solution based on the metric (3) could be very different from those provided by the algorithms using the latency-based metrics.

- *Load balancing for controllers*

It is desired 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:

$$Ib(C_i) = \max_{s \in S} \{ \max_{c \in C_i} n_c^s - \min_{c \in C_i} n_c^s \} \quad (4)$$

where n_c^s is the number of forwarder nodes assigned to a controller c. Equation (4) 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 (4).*

IV. MULTI-CRITERIA OPTIMIZATION ALGORITHMS

SDN controllers' placement and/or selection may involve several particular metrics (as summarized in Section III). If optimization algorithms for particular metrics are applied, then one can obtain different non-convergent solutions. Actually the CPP and CSP problems have naturally multi-criteria characteristics; therefore MCDA is a good way to achieve a convenient trade-off solution.

This paper uses the same variant of MCDA implementation as in [12], i.e., the *reference level (RL) decision algorithm* [16] as a general way to optimize the

controller placement, and controller selection, for an arbitrary number metrics. The MCDA-RL selects the optimal solution based on normalized values of different criteria (metrics).

The MCDA considers *m* objectives functions (whose values, assumed to be positive should be minimized). A solution of the problem is represented as a point in a space R^m of objectives; the decision parameters/variables are: v_i, i = 1, ..m, with $\forall i$, $v_i \geq 0$; so, the image of a candidate solution is $Sl_s=(v_{s1},v_{s2}, ..,v_{sm})$, represented as a point in R^m. The number of candidate solutions is S. Note that the value ranges of decision variables may be bounded by given constrains. The optimization process consists in selecting a solution satisfying a given objective function and conforming a particular metric.

The basic MCDA-RL [16], defines two reference parameters: r_i=*reservation level*=the upper limit, not allowed to be crossed by the actual decision variable v_i of a solution; a_i=*aspiration level*=the lower bound beyond which the decision variables (and therefore, the associate solutions) are seen as similar (i.e., any solution can be seen as "good"- from the point of view of this variable). Applying these for each decision variable v_i, one can define two values named r_i and a_i, i= 1, ..m, by computing among all solutions s = 1, 2, ..S:

$$r_i = \max [v_{is}],\ s = 1, 2, ..S$$
$$a_i = \min [v_{is}],\ s = 1, 2, ..S \quad (5)$$

An important modification is proposed in [16], aiming to make the algorithm agnostic versus different nature of criteria. The absolute value v_i of any decision variable is *replaced* with *distance from it to the reservation level*: r_i-v_i; (so, increasing v_i will decrease the distance); *normalization* is also introduced, in order *to get non-dimensional values, which can be numerically compared* despite their different nature. For each variable v_{si}, a ratio is computed:

$$v_{si}' = (r_i - v_{si})/(r_i - a_i), \quad \forall s,i \quad (6)$$

The factor $1/(r_i-a_i)$ - plays also the role of a weight. A variable for which the possible dispersion of values is high (max – min has a high value in formula (6)) will have lower weight and so, greater chances to be considered in determination of the minimum in the next relation (7). On the other side, if the values *min, max* are rather close to each other, then any solution could be enough "good", w.r.t. that respective decision variable.

The basic MCDA-RL algorithm steps are (see also [12]):
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}':

$$\min_s = \min\{v_{si}'\};\ i=1...m \quad (7)$$

Step 2. Select the best solution:

$$v_{opt} = \max \{ \min_s \},\ s=1, ..S \quad (8)$$

Formula (7) 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 (8) 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:

$$v_{si}' = w_i(r_i - v_{si})/(r_i - a_i) \quad (9)$$

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.

V. MCDA-BASED IMPLEMENTATION FOR SDN CONTROLLER PLACEMENT

A proof of concept simulation program (written in Python language [12]) has been constructed by the authors, to validate the MCDA–RL based CPP problem and allocation of forwarders to controllers. The program has been extended in this work with reliability evaluation features.

The simplifying assumptions (they could be also seen as limitations) of the model studied here, are: the network architecture is flat, i.e., no disjoint regions are defined; the network graph is undirected; any network node can be a forwarder but also can collocate a controller; when computing paths or distances, the metrics are additive; the number of controllers is predefined; the data traffic aspects and signaling interactions are not considered yet.

A. The MCDA basic model

The basic model to solve the CPP problem considered in this paper has two working modes:

a. static mode - the input data are: network graph (overlay or physical), link costs/capacities, shortest path distances between nodes (e.g., computed with Dijkstra algorithm based on additive metric), desired number of controllers, etc.).

Two phases are defined:

(1)*Phase 1*:

1.1. Define the criteria (i.e., the parameters of interest) and their priorities. The decision variables could be anyone, among those of Section III.

1.2. Compute all controller placements C1, C2, …. (i.e. the set of candidate solutions). The number of placements is C_n^k (n= total number of network nodes; k= number of controllers).

1.3. Compute the *values of the normalized metrics for each possible controller placement (i.e. future MCDA candidate solution)*, by using specialized algorithms and metrics like those defined in Section III.

The Phase 1 phase has as outputs the set of candidate solutions (i.e., placement instances) and values to fill the entries of the matrix M defined in Section IV. The Phase 1 computation could be time consuming (depending on network size) and therefore, could be performed off-line [5]. For instance, in a real network, a master SDN controller having all these information can perform these computations.

(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 (6)); run the MCDA Step 0, 1 and 2 of the (formulas (7) and (8)).

The Phase 2 provides the CPP solution.

b. dynamic mode – the input information is the total number of network nodes and desired number of controllers. The graph (which could be full-mesh or not) and costs of the links are randomly generated by a simulation program. The desired total number of nodes and the number of controllers should be specified as inputs in the program.

B. Reliability aware model

As shown in Section III, more realistic scenarios consider the possible occurrence of 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.

- *Backup controllers*

A simple static solution for assignment of forwarders to primary and backup controllers is presented below. We assume that CPP has been solved for a given network. Therefore the identities of controller nodes are known. The simplest assignment of forwarders to controllers is to consider the shortest paths between a forwarder to a controller. So, an algorithm computes all distances from a forwarder F_i to each controller CT_k and selects the closest CT_m as primary controller (based on shortest path between F_i and any controller) and the next (let it be CT_n in the ordered list of distances) as a backup controller.

However, while the primary controller placement after first run of the MCDA) is a global trade-off optimum, there is no guarantee that in case of node/link failures the placement of the backup controller is optimum, given the individual choice of the secondary/backup controller for each forwarder node.A natural solution is to add a novel criterion to the MCDA set of decision parameters.

An auxiliary algorithm is used to compute a simple metric (mean distance to a backup controller) to be added to MCDA. We introduce a novel decision variable *dist_backup* and perform the following computation (for each possible controller placement Ci containing the controllers CT_1, CT_2, ….CT_k):

```
For each forwarder Fi, i=1..N
 Do
   Dist_backup = 0;
   Compute dist. from Fi to any CTj, j=1..k;
   Dist_backup=Dist_backup + second_shortest_cost;
 Od
  Dist_backup_avg = Dist_backup/N;
```

This `Dist_backup_avg` can be added as a new decision variable to MCDA (maybe with appropriate wight) Therefore, the optimization will select a solution which considers also the backup controller nodes in the factors influencing the selection. Note that the inclusion of the backup controllers will increase the number of computations in the Phase 1.2 from C_n^k to C_n^{2k}.

- *Load balancing for controllers*

As shown in Section III, a good balance of the node-to-controller distribution is desired. If the number of nodes is N and the number of controllers is k, then the average number of nodes allocated to a controller is N/k. A simple new metric can be added to the set of MCDA criteria. This decision variable *D_avg* will measure the deviation of the number of nodes allocated to a controller CT_i, i.e., n_i from the average value N/k, and averaging this for all controllers.

$$D\_avg = (1/N)\ \text{Sum}\ |(n_i - N/k)|,\ i = 1 \ldots k \qquad (13)$$

Again, this variable can get an appropriate weight in the optimization process.

- *Nodes and link failures*

Nodes and link failures could appear in the network. Evaluation of effects of such events could be taken into account by adding new decision appropriate parameters in the set of MCDA input multi-criteria. Here, we adopted a different approach. Given that most important metrics are forwarder-controller latency, inter-controller latency, load balancing of the controllers, optimization of the placement of the primary and backup controllers, the MCDA has been first run to produce controllers' placement optimization based on these important parameters. Then the simulation program allows some events to happen (e.g., nodes or link failures). The MCDA has been run again and produce a new placement after removing the entities in failure. Finally the placement produced in the updated conditions can be compared with the initial one, to evaluate if significant changes appeared. In such a way one can evaluate the robustness of the initial placement, and decide if that can be preserved or must be changed.

Two input parameters have been defined in the model:

nf- number of nodes supposed to fail

ef – number of links supposed to fail.

The specific nodes and links which will fail will be selected as to to simulate the "worst case", i.e., those nodes having the lowest cost of the adjacent links and, respectively those links having the least costs. If after second run of the MCDA, the initial placement of the controllers does not change, this means that initial placement has enough good robustness properties. Of course, this result will depend on selection of nf and ef values, for a given N nodes of the graph.

 ISBN: 978-1-61208-669-9

C. Controller placement optimization- Simulation program

The user interface of the simulation program is presented in Figure 1.

```
stefan@mint ~/Desktop/simulator_mcda $ python mcda.py -h
usage: mcda.py [-h] [-a [A]] [-w [W]] [-i [I]] [-b [B]] [-l [L]] [--dynamic] [-n N] [-c C] [-nf NF] [-ef
EF] [--debug]

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].
  -b [B]      Average latency - failure scenario. Expects a weight (priority) in interval (0, 1].
  -l [L]      Controller load-balancing. 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
  -nf NF      Number of nodes that fail. Valid only in dynamic mode. Allowed  values: 1.. N-C.
  -ef EF      Number of edges that fail. Valid only in dynamic mode. Allowed  values: 1 ..N-C.
  --debug     Prints some computing results for debugging purposes.
```

Figure 1. The interface of the MCDA CPP simulation program

The decision parameters considered have been: *average* and *worst* latency between a forwarder and controller, *inter-controller* latency and *load balancing* related parameter. The program can be run isn static or dynamic mode, with any number and set of criteria among those presented in the interface. Note that if wanted, the set of decision parameter can be enriched; the only needed modification is the number of columns of the matrix M.

Several numerical examples and results of the basic CPP solutions have been already presented in the work [8]. The current version of the implementation added reliability feature presented in Section IV.B.

The pseudo-code of the simulation program for dynamic mode is presented below, in high level view.

```
Start
   Generate the random graph;
   Generate all controlers' placements;
   Run MCDA;
   If link_failures specified then eliminate from
the graph a number of ef links having the minimum
costs;
   If node_failures specified then eliminate from
the graph a number of nf nodes;
   If failures_produced
      Then {generate modified graph; Run MCDA;}
   Display the graphs;
Stop
```

D. Dynamic controller selection

In a dynamic network context, the controller choice (CSP) can be performed in a dynamic way. The multi-criteria algorithm can be as well applied in such cases. We consider here only the situations in which controller/node/link – related occur.

In the static approach the backup controllers are predefined; the placement is selected by the optimization algorithm. For a real network, the algorithm can be run offline in a management center (in a hierarchical organization of the control plane, this could be a master SDN controller). This center is supposed to know all information in order to run MCDA-RL algorithm. The aspects related of providing these information constitute a separate problem, which is not studied in this paper.

Supposing that a forwarder looses its connectivity with its controller, it can act in two ways; a. try to connect to a known backup controller; b. select among several by running a MCDA algorithm. The input information for MCDA (decision criteria) could be : identities/addresses of possible SDN controllers; degree of load for those controllers (this could be periodically communicated to the forwarder by a master SDN controller); local information observed by the forwarder, like connectivity to different nodes/controllers, etc. So, the forwarder can select based on MCDA-RL a novel controller.

VI. SAMPLES OF RESULTS

This section will shortly present samples of results, in order to prove the validity of approach. The experiments are reliability feature related.

- *Load balancing for controllers*

Figure 2 shows an example in which the network graph has been dynamically generated with N=6 nodes and k= 2 controllers. The decision criteria have been *inter-controller latency* (weight = 1) and *balancing criterion* (weight = 0.5, i.e. twice higher priority). The MCDA program is run with parameters :

```
stefan@mint$ python mcda.py -i 1 -l 0.5 --
dynamic -n 6 -c 2
```

The results obtained are: controllers in CT_0 and CT_3. The allocation of forwarders are :

```
Controller 0 has allocated node(s): 0, 2, 4.
Controller 3 has allocated node(s): 1, 3, 5.
```

One can see that while the inter-controller latency is not minimum, the allocation of the forwarders to controllers is balanced (3 forwarders per each controller).

- *Links and node failures*

If the unique parameter considered in MCDA would be the average latency of the forwarders to backup controllers, then one would expect that the resulting placement could be enough resilient to a low number of nodes and/or link failures.

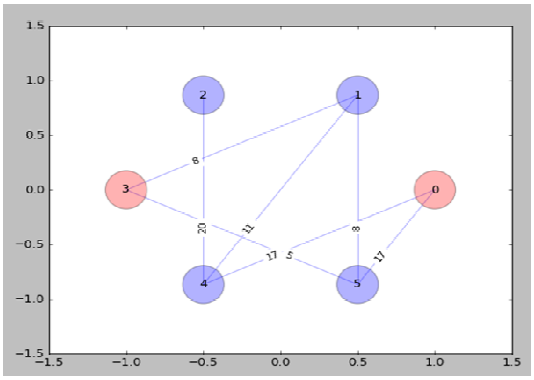

Figure 2. Simple example of a balanced allocation of the forwarders to controllers (after MCDA run)

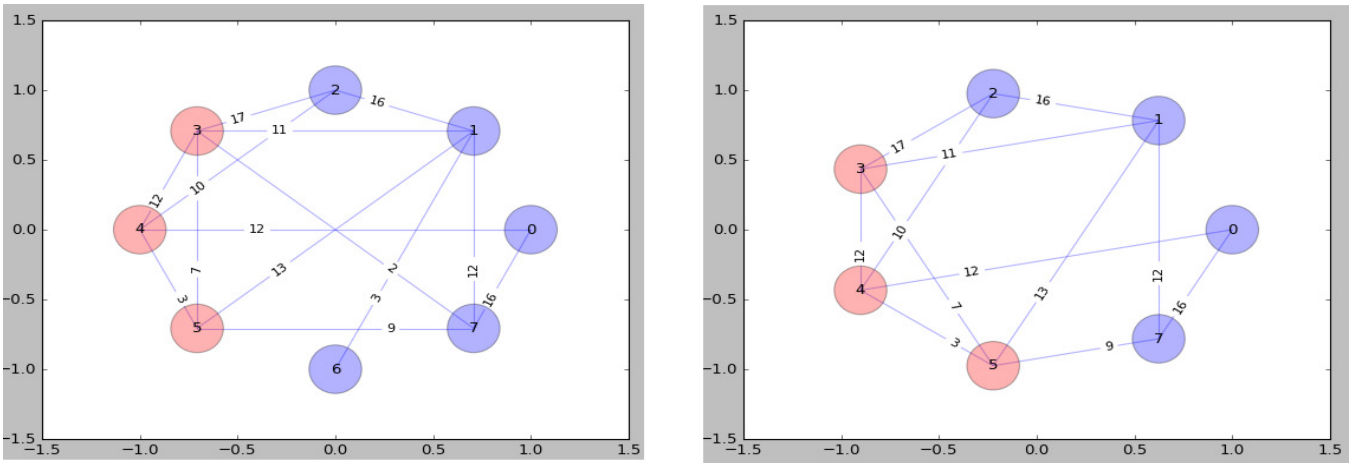

Figure 3. Example of placement resilient to link failures
Left: placement before link failures; Right: placement after some links failures.

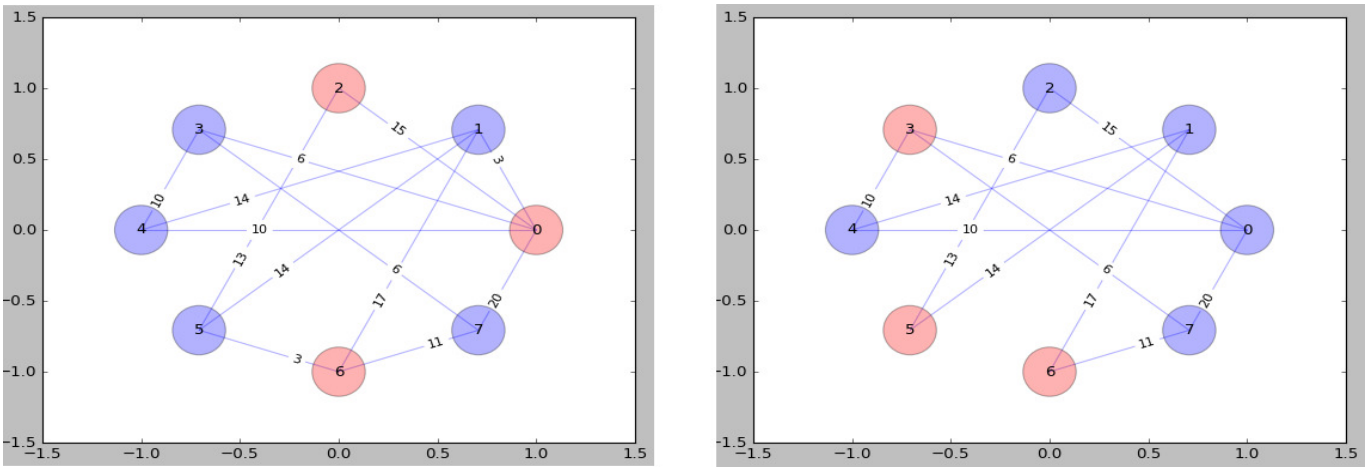

Figure 4. Example of placement non-resilient to link failures
Left: placement before link failures; Right: placement after some links failures.

Figure 3 shows such an example, by presenting the graphs resulted after running the program with the command:

```
python mcda.py -b --dynamic -n 8 -c 3 -ef 2
```

In this example, we have N=8 nodes and c= 3 controllers; the number of failure links ef=2.

One can see that after some links failure (1-6, 3-7) still the controller placement (after running MCDA on the reduced graph) is the same, i.e., 3,4,5.

On the other side, if the initial criterion of MCDA is the to minimize the average latency between the forwarders and controllers (parameter introduced with weight = 1) the optimum placement after some link/nodes failures will be different (Figure 4). The command for such a run is:

```
python mcda.py -a --dynamic -n 8 -c 3 -ef 2
```

These examples illustrate the power of the MCDA algorithm where various sets of criteria and different priorities (driven by policies) can be considered.

VII. CONCLUSIONS AND FUTURE WORK

This paper extended the study [12], on using multi-criteria decision algorithms (MCDA) to optimally place the controllers in large SDN, based networks. The MCDA advantage is that it can produce a tradeoff (optimum) result, while considering several weighted criteria, part of them even being partially contradictory.

In this study, a previous simulation program has been extended to include reliability aware metrics in the multi-criteria optimization algorithm. 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 a more deep study of the dynamic possibilities to apply the multi-criteria based selection of the SDN controllers and to consider also aspects of signaling traffic (inter-controller). Hierarchically organized SDN control planes are also open research topics for CPP and CSP problems.

REFERENCES

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

[2] A. Tootoonchian, and Y. Ganjali, "Hyperflow: a distributed control plane for openflow" in Proc. INM/WREN, 2010, https://pdfs.semanticscholar.org/f7bd/dc08b9d9e2993b363972b89e08e67dd8518b.pdf, [retrieved: 5, 2018].

[3] T. Koponen, , M.Casado, , N.Gude, J.Stribling, L. Poutievski, et.al., "Onix: a distributed control platform for large-scale production networks," in Proc. OSDI, 2010, https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Koponen.pdf, [retrieved: 5, 2018].

[4] B.Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualisation: Challenges and Opportunities for Innovations". IEEE Communications Magazine, pp. 90-97, February 2015.

[5] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proc. HotSDN, pp. 7–12, 2012, https://dl.acm.org/citation.cfm?id=2342444, [retrieved: 5, 2018].

[6] 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, pp. 92–97, October 2012, , http://www.sciencedirect.com/science/article/pii/S10058851160438X, [retrieved: 1, 2018].

[7] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," Proceedings of the ITC, Shanghai, China, 2013, https://ieeexplore.ieee.org/document/6662939/, [retrieved: 1, 2018].

[8] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability aware controller placement for software-defined networks," in Proc. IM. IEEE, pp. 672–675, 2013, https://ieeexplore.ieee.org/document/6573050/, [retrieved: 1, 2018].

[9] 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, https://ieeexplore.ieee.org/document/7037087/, [retrieved: 1, 2018].

[10] G.Wang, Y.Zhao, J.Huang, and W.Wang, "The Controller Placement Problem in Software Defined Networking: A Survey", IEEE Network, pp. 21- 27, September/October 2017.

[11] K. Sood and Y.Xiang, "The controller placement problem or the controller selection problem?", Journal of Communications and Information Networks, Vol.2, No.3, pp.1-9, Sept.2017.

[12] E. Borcoci, T. Ambarus, and M. Vochin, „Multi-criteria based Optimization of Placement for Software Defined Networking Controllers and Forwarding Nodes," The 15th International Conference on Networks, ICN 2016, Lisbon, http://www.iaria.org/conferences2016/ICN16.html, [retrieved: 5, 2018].

[13] S.Yoon, Z.Khalib1, N. Yaakob, and A.Amir, "Controller Placement Algorithms in Software Defined Network - A Review of Trends and Challenges", MATEC Web of Conferences ICEESI 2017 140, 01014 DOI:10.1051/matecconf/201714001014, 2017.

[14] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks" in GLOBECOM 2011, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.691.795&rep=rep1&type=pdf, [retrieved: 5, 2018].

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

[16] 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.

[17] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/, [retrieved: 4, 2018].

[18] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE JSAC, vol. 29, no. 9, 2011, pp.1765-1475..

 ISBN: 978-1-61208-669-9