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

Eugen Borcoci, Radu Badea, Serban Georgica Obreja, Marius Vochin
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
eugen.borcoci@elcom.pub.ro, radu.badea.@elcom.pub.ro, serban@radio.pub.ro, mvochin@elcom.pub.ro

***Abstract* — Multi-controller implementation of the Software Defined Networking (SDN) control plane for large networks environment can solve the scalability and reliability issues introduced by the centralized logical control principle of the SDN. However, there are still open research topics 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 analytical view 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) technologies offer significant advantages for cloud data centres and also for Service Provider Wide Area Networks (WAN)..The *basic principles* of the SDN architecture are [1][2] [3]: *decoupling of the control and forwarding* (data) planes; *logically centralized control*; exposure of *abstract network resources and state to external applications.* Thus SDN offers important advantages of independency of the control software w.r.t. forwarding boxes implementations offered by different vendors. Higher network programmability is also a consequence 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).

However, the control-data plane separation can generate performance limitations and also reliability issues of the SDN controlled network [4][5] (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 them, how to process various new flows arriving to them (by filling appropriately the *flow tables* [1]). 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 when the controller has a limited processing capacity [4], w.r.t the number of flow queries or the number of forwarders assigned to a controller is too high.

(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 construction and continuously updating of unique vision upon the network [6][7][8]. A frequent 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 [9].

(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 [4][10]. 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 they cannot communicate with some controller, they will have no knowledge on how to process the newly arrived flows.

There is a need to optimally place the controllers. This can be done by attempting to solve (a), (b), and (c). This is a multi-criteria optimization problem and it was recognized as an NP-hard one [10]. Consequently, different solutions have been proposed targeting performance, (problem (a), (b)), and performance plus reliability (problem (c)).

This paper contains an analysis of some solutions for (a), (b), (c) and then proposes a preliminary contribution on how *multi-criteria optimization algorithms* can be applicable to the controller placement problem. The target here is *not to develop specific algorithms* dedicated to find an optimum solution for *a given criterion* (several studies did that) but to *achieve an overall controller placement optimization*, by applying *multi-criteria decision algorithms (MCDA)* [11][12]. The input of MCDA is a set of candidates (here an instance of controller placement is called a *candidate solution*).

The paper 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 presents conclusions and future work.

 ISBN: 978-1-61208-398-8

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. The basic problem to be answered (in an optimum way) is *how many SDN controllers are needed* in a given network (topology and some metrics are defined) and *where they should be placed* in the network, as to provide enough performance (e.g., low delay for controller-forwarder communications) and robustly preserve the performance level when failures occur. Intuitively, it can be seen that some trade-off will be necessary.

In WANs having significant path delays the controller placement determines the control plane convergence time, i.e., affects the controllers' response to real-time events sensed by the forwarders, or, in case of proactive actions, how fast can the controllers push (in advance) the required actions to forwarding nodes.

Actually, it has been shown in [10][13] 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 [10] motivates the controller placement problem and then quantifies the placement impact on real topologies like Internet2 [4] and different cases taken from Internet Topology Zoo [15]. Actually, the main goal was not to find optimal minimum-latency placements (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. However, resiliency aspects have not been considered in the above study.

Several works [9][13][16][17][18] have observed that resilience is important in the context of SDN and especially if Network Function Virtualization (NFV) is wanted. Some resiliency-related issues have been considered in [13]:

(1) *Controller failures*: in case of a primary controller failure, it should be possible to reassign all its previously controlled nodes to their second closest controllers, by using a backup assignment or signaling based on normal shortest path routing. Extreme case scenarios have been also considered, e.g., if at least one controller is still reachable, 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 latencies) will change; some reassignment of nodes to other controllers is needed. In the worst case, some parts of the network can be completely cut off, having no access to controllers. Such nodes can still forward traffic, but they cannot anymore request or 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 might 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*: 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 [7]) or a hierarchical topology (e.g., like in Kandoo [8]) of controllers is used, it is clear that inter-controller latency should minimized. Therefore, an optimized controller placement should meet this requirement.

The works [9][17] 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 [18] identifies several limitations of previous studies: (1) forwarder-controller connectivity is modelled 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*, a controller placement strategy that explicitly considers path diversity, controller capacity awareness, and failover mechanisms at network design. Specific contributions consist in: significant reduction of the connectivity loss by exploring the path diversity (i.e., connectivity-awareness) which is shown to reduces 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).

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, [9][10][13][18]. The goal of this paper is not to discuss details of such particular algorithms (but searching a global optimization). We only outline here their objectives. Some limitations are emphasized for particular cases.

 ISBN: 978-1-61208-398-8

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* [10]). It is assumed that controller 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∈V to a controller c∈V. In [10], 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 :

Average_latency:

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

Worst_case_latency :

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

The optimization algorithm should find a particular placement C_{opt}, where *either average latency or the worst case latency is minimum.* Figure 1 shows a simple example of a network having six nodes. 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. On the graph are marked the distances between different nodes (overlay paths)

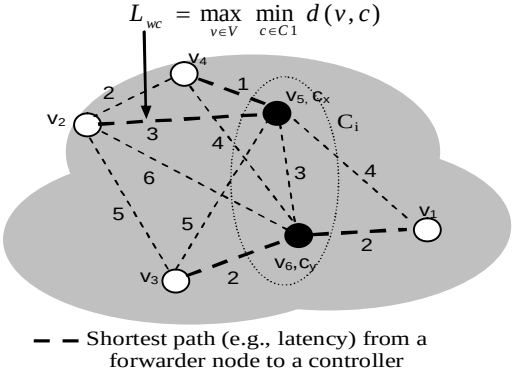

Figure 1. Simple network example of controller placement:v= forwarder node; c = controller; C_1= { [c_x_in_v_5 (v_5, v_2, v_4)], [c_y_in_v_6(v_6, v_1, v_3)]}

Some limitations of this optimization process are:

- No reliability awareness: the metrics are simply distances, which in the simplest case, are static.
- There is no upper limit on the number of v nodes assigned to a controller; too many forwarders to be controlled can exist, especially in large networks.

Other metric possible to be considered in failure-free case is *Maximum cover* [10][19]. The algorithm should find a controllers 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 as being partial..

B. *Reliability aware metrics*

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

(1) Controller failures (cf): the work [13] observes that node-to-controller mapping changes in case of controller outages. So, a realistic latency-based metric should consider both the distance to the (primary) controller and the distance to the other (backup) controllers. For a placement of a total number of k controllers, in [13] the failures are modelled 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 :

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

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

Commenting the placement results based on the metric (1) or (2) to (3), 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. When minimization of expression (3) (and considering worst case failure) controllers tend to be placed the centre of the network. Thus, even if all except for one controller fail, the latencies are still satisfactory (numeric examples are given in [13]). However, one can criticize such an approach, if applied to large networks; the scenario supposed by the expression (3) is very pessimistic; rather 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 [18].

The conclusion is that a trade-off exists, between the placements optimized for the failure free case and those including controller failure. 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 result in network disruption; 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 simultaneous failures at only a few (e.g., two [13]). If more than two arbitrary link/node failures happen simultaneously, then the topology can be totally

 ISBN: 978-1-61208-398-8

disconnected and optimization of controller placement would not be any more useful.

For any given placement C_i of the controllers, an additive integer value metric *Nlf(Ci)* 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 (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). 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 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 [18], 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 controllers instances. The metric is :

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

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-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.*

D. Constraints

Apart from defining the metrics, the controller placement problem can be subject to different constraints. For instance, in [18], 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, regions pre-defined for controllers, etc. They should be included in the respective algorithms.

IV. MULTI-CITERIA OPTIMIZATION ALGORITHM

The 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 partial 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 [13] for some combinations of the metrics defined there (e.g., max. latency and controller load imbalance for failure-free and respectively failure use cases).

A. Reference level MCDA

We propose 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 [11][12] is, to minimize $\{f_1(x), f_2(x), \ldots, f_m(x)\}$, where $x \in S$ (set of feasible solutions), $S \subset R^n$. The *decision vector* is $x = (x_1, x_2, \ldots, x_n)^T$. There are ($m \geq 2$) possibly conflicting objective functions $f_i : R^n \rightarrow R$, i= 1, ..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* = images of decision vectors. The objective (function) values are given by $z = f(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T$. We denote as feasible objective region W = f (S) = image of S 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 [11] if there does not exist another $x \in S$ such that $f_i(x) \leq f_i(x\_)$ for all i = 1, . . . , k and $f_j(x) < f_j(x\_)$ for at least one index j.

We adopt here the MCDA variant called *reference level decision algorithm* [12]. 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, ..m, with $\forall i, v_i \geq 0$,
- image of a candidate solution is $Sl_s=(v_{s1},v_{s2},...,v_{sm})$, 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 constrains. The optimization process consists in selecting a solution satisfying a given objective function and conforming a particular metric.

The basic *reference level algorithm* defines two reference parameters:

- r_i =*reservation level*=the upper limit for a decision variable, which the solution should not cross;
- a_i=*aspiration level*=the lower bound beyond which the reference parameters are seen as similar.

Without loss of generality one may apply the definitions of [12], where for each decision variable v_i there are defined r_i *and* a_i , by computing among all solutions s = 1, 2, ..S:

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

In [12], 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 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 variables having close values.

The basic 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}':

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

Step 2. Make selection among solutions by computing:

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

This v_{opt} is the optimum solution, i.e. it selects the best value among those produced by the Step 1.

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 some policies when deciding the controller placement. Some decision variables (or metrics) could be more important than others. 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.

The controller placement problem solving (given the graph, link costs/capacities, constraints, number of controllers desired, 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).

This procedure could be (depending on network size) time consuming and therefore performed off-line [10].

(2) *Macro-step2*: MCDA

- define reservation and aspiration levels for each decision variable;
- eliminate those candidates having parameter values out of range defined by the reservation level;
- define appropriate weights (see formula (9')) 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)).

 ISBN: 978-1-61208-398-8

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. E.g., in high reliable environment one could consider only failure free metrics.

C. Numerical example – MCDA optimization

Given the limited paper space, a simple but relevant example is exposed to illustrate the MCDA power, based on the Figure 1 network. Suppose that for this network the metrics of interest and decision variables are (see Section III) onl: d1:*Average latency* (1), 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_v_5 (v_5, v_2, v_4)], [c_y_in_v_6(v_6, v_1, v_3)]}
C_2= { [c_x_in_v_5 (v_5, v_1, v_2, v_4)], [c_y_in_v_3(v_3, v_6)]}
C_3= { [c_x_in_v_3 (v_3,v_2)], [c_y_in_v_6(v_6, v_1, v_4, v_5,)]}
C_4= { [c_x_in_v_4 (v_4,v_2, v_5)], [c_y_in_v_6(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, ..C4.

A matrix M(3x4) is computed using the formulas (9). MCDA is applied by using formulas (10), (11). The final result is : C_1 *= 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.

1. MCDA with priorities for i.e. d1=1, d2=0.5, d3=1, i.e., the worst case latency d2 has highest priority. After re-computing the matrix M and applying MCDA equations (1), (11), we find C_4*= 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.

V. Conclusions

This paper presented a work (in progress) 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 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.

Future work will be done to apply the method proposed to large networks - real life case studies (e.g. from Internet Topology zoo, [15]) and comparing the quality of trade-offs when defining different weights to decision variables.

References

[1] 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.

[2] “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.

[3] “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.

[4] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, “On Scalability of Software-Defined Networking”, IEEE Comm. Magazine, February 2013, pp. 16-141,.

[5] 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.

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

[7] 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.

[8] S. H. Yeganeh and Y. Ganjali, “Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications,” Proc. HotSDN ’12 Wksp., 2012.

[9] 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.

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

[11] J. Figueira, S. Greco, and M. Ehrgott, “Multiple CriteDecision Analysis: state of the art surveys”, Kluwer Academic Publishers, 2005.

[12] 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.

[13] 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.

[14] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/.

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

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

[17] 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.

[18] 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.

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

 ISBN: 978-1-61208-398-8