Multiple Conditions-aware Dynamic Switch Migration in SDN Large Area Networks

Eugen Borcoci, Silviu – Gabriel Topoloi, Serban Georgica Obreja
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
Emails: eugen.borcoci@elcom.pub.ro, silviu.topoloi@elcom.pub.ro, serban@radio.pub.ro

***Abstract* —Distributed control plane solutions are adopted in large SDN-controlled networks, to improve control plane scalability. Many studies exist, on Controller Placement or Selection Problem (CPP/CSP) using different optimization criteria. Most of them consider static solutions to optimize controller placement. However, in a dynamic context, (i.e., traffic flows variation, possible failures of links, nodes, or controllers, etc.) the initial controller placement and controller-to-forwarders mapping could be no more optimum. Additionally, in practice, the controllers have limited processing capacity so, their overload is possible. A solution for the above problems could be a partial dynamic switch migration, e.g., when detecting some controllers' overload. The contribution of this work-in-progress paper is an enhancement proposal of a dynamic migration solution (recently proposed in the literature), by introducing a multiple condition-aware migration complex decision. The proposed approach could solve the trade-offs between different optimization objectives of the network operator.**

Keywords — Software Defined Networking; Controller placement; Multi-condition-aware switch migration; Multi-criteria optimization; Forwarder nodes assignment; Reliability.

I. INTRODUCTION

The usual solution to assure the scalability of the *Software Defined Networking* SDN control plane is a distributed multi-controller implementation (flat or hierarchical organization), e.g., in [1][2].

Basically, a *SDN controller* (SDN-C) is placed in a geographically distinct location, i.e., physical network node. However, the recent *Network Function Virtualization* technologies [3] allow that SDN-Cs virtualization (notation for such a controller will be vSDN-C); several vSDN-Cs can be collocated in the same physical node. In the following text we suppose the basic approach, but the models developed in this paper can be as well applied to a virtualized environment.

A major issue in SDN large networks is the *Controller Placement Problem* (CPP). A lot of studies already exist dedicated to this problem [4-12]. Recent works are still elaborated, given that many associated issues exist together with CPP itself. Some examples are: network topology - flat or clustered; what are the criteria used to solve the CPP; number of controllers; failure-free or failure-aware metrics (controllers and/or node/link failures); controller-forwarder/switch mapping (in a static or dynamic way, i.e., depending on actual network conditions and network provider policies), etc. The optimality of the different solutions can be studied on some simplified topologies – in order to compare the approaches or, on real specific network topologies. The CPP is a non-polynomial NP-hard problem [4]; therefore, many pragmatic static/dynamic solutions have been proposed, using specific optimization criteria, targeting good performance in failure-free or failure-aware approaches.

Given the complexity of a real network environment, there is no unique best placement rule for CPP. Many of the current existing CPP solutions consider static mapping switches-controllers, thus having no capability of adaptation to dynamic load. However, during the network run-time, dynamic nodes addition and deletion can happen, or traffic variation (consequently, controller loads fluctuation appear), link/node/controller failures can appear, etc. In such cases, one can apply *dynamic switch migration* from the current controller to a new more appropriate controller, if enough pertinent and updated information exist at run-time [13-17]. This migration can be included in a more general *Controller Selection Problem* (CSP) and can be considered as an extension of the CPP [5].

The main parameter to be taken into account when deciding on switch migration is the current *load of the controllers*, dynamically evaluated by a monitoring system [13-17]. The switching objective is to achieve better load balancing and avoid controllers' overload. However, other individual parameters might be important, like controller-switch communication latency, inter-controller communication throughput, reliability-related properties, etc. Other specific optimization goals could be added to the above list, depending on specific network context (wire-line, wireless/cellular, cloud computing and data center networks) and on some specific business targets of the SDN-controlled network owner. A major problem is that different optimization criteria could naturally lead to non-convergent solutions; therefore, *a multi-criteria global optimization* could be a useful approach.

The contribution of this paper is a proposal to enhance a single criterion dynamic switch migration algorithm, recently proposed in the literature, [13][14], by *introducing a multiple condition-aware migration complex decision*. The procedure used is an extension of the method developed in [18] based on *multi-criteria decision algorithms* (MCDA)

 ISBN: 978-1-61208-753-5

[20]. Therefore, the goal here is not to develop some new optimization algorithm based on a single criterion, but to prove the value of multi-criteria CPP/CSP optimization approach, both statically, or performed during run-time. The input of MCDA is the set of candidates (e.g., an instance of controller placement and implicit a switch-controller mapping is called a candidate solution). Examples have been illustrated in the paper, on some simple network topologies, proving the usefulness of the approach. This work is still in progress; a simulation model is currently in development and its results will be reported in a future paper.

The paper structure is described here. Section II is a short overview of related work. Section III presents a flow-aware dynamic switching algorithm recently proposed [13][14]. Section IV shortly recalls the framework for MCDA-RL (the "*reference level*"). Section V contains the main paper contribution i.e., an enhancement proposal of the dynamic migration solution (described in Section III), by introducing a multiple condition-aware switch migration complex decision which could solve the trade-offs between different objectives of such optimization. Simple examples are given to prove the value of multi-criteria optimization. Section VI presents conclusions and future work.

II. RELATED WORK

This short section is included mainly for references. In works [4-8][19], specific optimizations based generally on a single criterion are proposed, while comprehensive surveys on CPP/CSP are overviewed in [9-12]. The general goal is to find those controller placements that provide high performance (e.g., low delay for controller-switch communications) and also create robustness to controllers and/or network failures. The studies [13-17] are oriented on CSP issues i.e., switch migration and load balancing algorithms. The work [18] applies MCDA [20] in order to consider several criteria in a static CPP approach.

An early work of Heller et al. [4], has found optimal CPP solutions for realistic network instances, in failure-free scenarios, by analyzing the entire solution space, with off-line computations (the metric is switch-to-controller latency). The works [6-8][19] 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; generally, it cannot be minimized while simultaneously minimizing controller-switch latency-therefore - a tradeoff solution could be the answer.

K.Sood and Y.Siang [5] propose to transform the CPP problem into *Controller Selection Problem* (CSP), i.e., consider the dynamics of the network and make controller selection for group of forwarders. They explore the relationship between traffic intensity, resources requirement, and QoS requirements. They search solutions which are topology-independent and adaptive to the needs of the underlying network behaviour. The optimal number of controllers is calculated, to reduce the individual workload, of a controller; the paper investigates the placement/location of the controllers. However, the first declared objective in [5] has been to motivate the CSP and not to determine the optimal placement of controllers in the network.

The work [6] they developed several algorithms for real topologies, considering the reliability of SDN control, but still looking for keeping 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.

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). Muller et.al. [8] eliminate some restrictions of previous studies, like: single paths, *on-demand* only processing (in controllers) of the forwarders requests, and some constraints imposed on failover mechanisms.

Yang Xu, Marco Cello et al., [13]14] recently developed a comprehensive solution for dynamic switch migration, based on run-time information delivered by a monitoring system. This approach will be further described in Section III as the starting point for work presented in this paper.

The paper [15] proposes a switch migration method, where switch migration is seen as a signature matching problem and is formulated as a 3-D earth mover's distance model to protect strategically important controllers in the network. A heuristic method is proposed, time-efficient and suitable to large-scale networks. Simulation results show that one can disguise strategically important controllers by diminishing the difference of traffic load between controllers. The proposed methods can relieve the traffic pressure of controllers and prevent saturation attacks.

In [18] a multi-criteria based algorithm is used (applicable for an arbitrary number of decision criteria) to solve the CPP.

This paper extends the solution of [13][14] and is based on [18] work. A multiple condition-aware switch migration complex decision is introduced which could solve the trade-offs between different objectives (thus solving a dynamic selection problem - CSP).

III. A FLOW-AWARE SDN DYNAMIC SWITCH MIGRATION ALGORITHM

This section will shortly present (as a starting point) a recent solution developed by Yang Xu, Marco Cello et al. in [13][14], for dynamic SDN switch migration, to avoid controller overload. Then, in the next section we will propose an enrichment of the decision for SDN switch migration, considering that in practice multiple conditions could actually exist, to influence the switch-to-controller assignment, i.e., not only the controller overload. Other criteria to take a decision can also be important, like switch-controller communication delay, reliability capabilities, etc. A multiple criteria optimization algorithm could offer a better trade-off solution.

A. The SDN switches migration problem

In [13][14], the switch migration scenario is analyzed, starting from a given switch-to-controller assignment and

partition (based on some criteria) of the network, in domains; each one is controlled by a single controller. Also, a realistic assumption is considered, i.e., limited processing capacity of the controllers. During run-time, if some controllers are overloaded (such events are dynamically observed by a monitoring system), then a heuristic algorithm is applied, to optimally move (re-assign) a number of the switches coordinated by the overloaded controller to another controller less loaded. In order to reduce the control plane signaling (needed to govern the migration) between controllers, the migration is *cluster-based*. In other words, not a single switch is migrated, but a cluster of switches are moved from an overloaded controller, e.g., CT_i, to another less loaded controller CT_j. Thus, the algorithm realizes a controller load balancing (the name *BalCon* is coined for this algorithm [13]). The [13][14] works do not assume a predictable traffic or well-known traffic patterns among the SDN switches. Instead, the network load level is learned from monitored values of the input or transited flows through the network of SDN switches.

The scenario supposes a set *S* of SDN switches, managed by a set *CT* of controllers. Let Si be a SDN switch controlled by the SDN controller CT_m. The following flow arrival rates are considered:

$f_{o,Si}$ at *Si* from outside (e.g., from some hosts) the SDN network,

$f_{Si,o}$ flows leaving the network from *Si*,

$f_{Si,Sj}$ - current arrival rate of new flows going on the link between the two connected switches, from *Si* to *Sj*.

All the above flows generate processing tasks in the controller CT_m. Here, it is assumed the case which produces the highest controller's load: reactive SDN control behavior is applied. This means that for each new flow coming to a switch, a *packet_in* message is uploaded to the controller. The message is asking the controller to process the flow information and then to install in the switch Si new rules for that flow.

The SDN network of switches was previously partitioned (using some algorithms) in disjoint "control-domains", each one controlled by a single controller. The total load $L(CT_m)$ at controller CT_m is composed of three main components:

- the path computation load of new flows arriving from
  - outside the SDN network, to the switches "belonging" to CT_m
  - another SDN domains of the same SDN network
- the rule installation load at each switch controlled by CT_m, for all flows crossing the domain controlled by CT_m.

To evaluate in a generic way the computational effort of a controller due to the instantiation of the new flows, the work [14] considered that path computation for a single flow requires α units of load, whereas the rules installation of a single flow in a single switch requires β units of load. So, the overall computation load for each controller can be evaluated, given the flow arrival rates at the switches coordinated by it.

If the CT_m has in its partition/domain a set $Sw(CT_m)$ of SDN switches, then its overall load $L(CT_m)$ (see detailed formulas in [14]), is:

$$L(CT_m)=L_{cmp}(CT_m) + L_{inst}(CT_m) \quad (1)$$

where the index *cmp* denotes the computation load and *inst* denotes the installation load. The two components are:

$$L_{cmp}(CT_m) = L_{cmp}(CT_m)_{in}+L_{cmp}(CT_m)_{transit}$$

$$L_{inst}(CT_m) = L_{inst}(CT_m)_{out}+L_{inst}(CT_m)_{transit} \quad (2)$$

The notations are explained below.

$L_{cmp}(CT_m)_{in}$ and $L_{cmp}(CT_m)_{transit}$ are the sums of all computation loads associated to input flows for all switches in $Sw(CT_m)$, related to $f_{o,Si}$ and respectively to $f_{Sj,Si}$, where $Si \in Sw(CT_m)$.

$L_{inst}(CT_m)_{out}$ and $L_{inst}(CT_m)_{transit}$ are the sums of the installation loads for all switches in $Sw(CT_m)$, in order to instruct each switch about flows going out of it, outside of the SDN network, or flows going out to another switch of the SDN network.

An SDN controller is overloaded or even congested when its overall computational load is $L(CT_m) > L$, where L is the maximum load to be admitted for CT_m. Note that in [14], the same value L is supposed for all controllers (this can be seen as a limitation of the method).

The *Optimal Controller Load Balancing (OCLB)* problem is defined in [14] as *to find the partition which minimizes the worst case of load of a controller CTm, among the set CT of all controllers.* It is stated in [13] that the OCLB problem is NP-complete. Therefore, a heuristic algorithm is proposed.

The OCLB problem is actually to partition a network graph. The computation of $L(CT_m)$ for each CT_m can be induced directly from the graph. The SDN network is modeled as a directed edge-weighted and vertex-weighted graph G(S, E) in which SDN switches are the vertices with weights $l(Si)$, where $Si \in S$ and edges $E = \{(Si, Sj) : Si, Sj \in S, l(Si, Sj) > 0\}$, are the inter-connections among SDN switches. The value $l(Si, Sj)$ is the edge weight of (Si, Sj). Equivalently, the overall load at CT_m is the sum of the weights of the vertices belonging to its partition plus the sum of weights of the edges directed to the partition of CT_m. Specifically, the switch Si placed in the partition of CT_m, produces the following load for its controller CT_m:

$$l(Si) = L_{cmp}(CT_m)_{in,Si} + {}_{inst}(CT_m)_{out,Si}+L_{inst}(CT_m)_{transit,Si,Sj} \quad (3)$$

The l(Si) is the sum of computing load in CT_m, for flows coming into Si from external networks added to the installation load of the CT_m, in order to install rules in Si for flows going out of Si (to external networks/hosts, or to other switches).

$$l(Sj, Si) = L_{cmp}(CT_m)_{transit,Sj,Si} \quad (4)$$

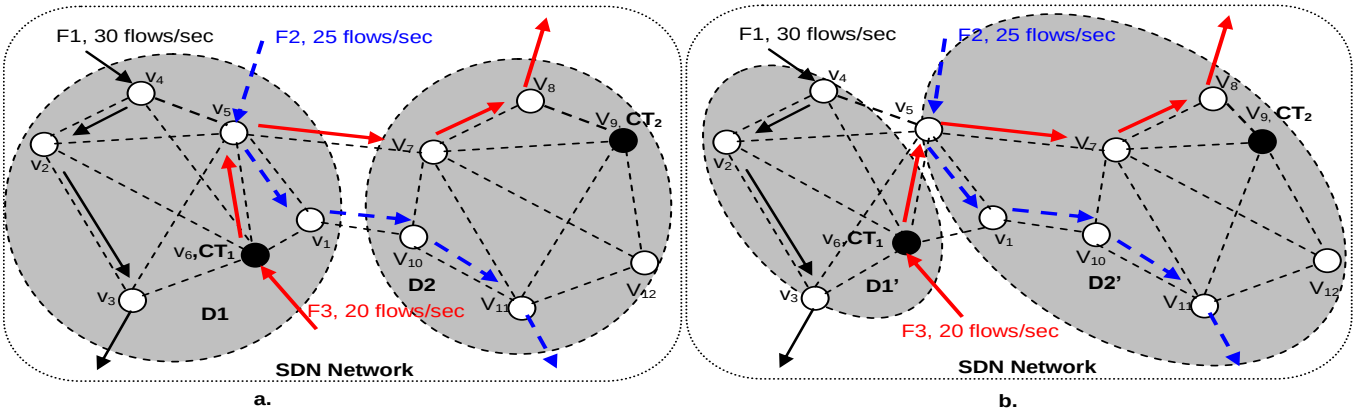

Figure 1. Switch migration from D1 to D2 – example; a. before migration; b. after migration

The weight l(Sj, Si) = $L_{cmp}(CT_m)_{transit,Sj,Si}$ is the computation load in CT_m, induced by the Si when it receives a new flow from Sj.

BalCon is a heuristic algorithm, operating during the network runtime. It detects and solves congestion at the SDN controllers through optimized SDN switch migrations. BalCon can be implemented as a northbound application of the SDN controller. BalCon consists of three phases:

1) *Monitoring and congestion detection*: BalCon monitors the congestion level at each controller. If the load of a controller CT_m reaches a given threshold, then BalCon computes a list of switches that may be migrated. The list is ordered by a priority based on a pre-determined metric.

2) *Clustering and migration evaluation*: starting from the SDN switches in the priority list, BalCon analyzes the traffic pattern among switches to find clusters of heavily connected switches.

3) *Cluster migration*: the best cluster is found and the migration is evaluated; the switches belonging to the cluster are migrated to the new SDN controller.

Figure 1 a. shows a SDN network partitioned in two domanins D1 and D2, controlled by the controllers CT_1 and respectively CT_2. Each vertice V_i of the graph is a network node accommodating a switch or a switch and a controller. The example shows three set of flows going into the network, routed as in the figure and producing F1: 40 flows/sec; F2:25 flows/sec and F3:20 flows/sec. We assume that α=1 and β = 0.1 [14]. Then, computing the loads for CT_1 and CT_2 (formulas (1) and (2) are applied), we get :

$L(CT_1)$ = (30+25+20)α + (3*30 +2*25 + 2*20)β = 75 + 18 =93 units of load

$L(CT_2)$ = (25 + 20)α + (2*25 + 2*20)β = 45+9 = 54 units of load.

Supposing that L=80 (maximum load for a controller) one can see that the CT_1 is overloaded. If for instance, V_5 and V_1 migrate to CT_2, then the loads will be modified (see Figure 1 b.). This simple example qualitatively proves that an appropriate migration can realize better load balancing.

$L(CT_1)$ = (30+20)α + (3*30 +2*25 + 2*20)β = 50 + 18 =68 units of load.

$L(CT_2)$ = (25 + 20)α + (3*25 + 4*20)β = 45+15.5 = 60.5 units of load.

This second partition provides better load balance.

B. The BalCon Algorithm

This sub-section summarizes the BalCon algorithm proposed in [14]. It is activated when an overload is detected for a controller, by a monitoring system. The input data in the algorithm are: the network graph G(S, E); the identity of a congested controller CTm and its load; the set of switches Sw(CTm) controlled by CTm. A cluster of switches to be migrated is started to be defined and then expanded via iterations (*IncreaseCluster* function). The migrations to different target SDN controllers of the selected cluster are evaluated by a function *ComputeMigrationAlternatives*. For each controller, it computes the controller load and the migration size (the number of switches to be migrated). Finally, the function *Evaluate-BestMigrationAlternative* evaluates the best alternative (based on some optimum criteria). The computation steps are [see details in 14]:

A=Ø; A is the set of cluster switches
A = ComputeStartingSwitchesList(Sw(CTm))
foreach $S_i \in A$ *do*
  T = {Si}; T is the cluster
  alternatives =
    alternatives $\cup$ *ComputeMigrationAlternatives*(*T*);
  while 1 do
    newT = IncreaseCluster(*T*);
    if *size*(*T*) >*mcs_ newT = T* then break;
    T = newT ;
    alternatives = alternatives $\cup$
      ComputeMigrationAlternatives(*T*);
  od
od
[T^0, Target_SDN_controller^0] $\leftarrow$
  EvaluateMigrationAlternatives (*alternatives*);

Starting from the cluster *T*, the function *IncreaseCluster* constructs the set *neighborsT* composed of all SDN switches

that are neighbors to *T*. An SDN switch *Si* is a neighbor of *T* if $\exists Sj \in T : l(Si, Sj) \neq 0, l(Sj, Si) \neq 0$. The function selects the neighbour that maximizes the relative density *Density* [11] of the newly created cluster. The rationale is that only SDN switches with strong inter-connections should be grouped into the same cluster. Then the cluster will be migrated between controllers as a whole, to reduce the overall computation complexity (related to migration) of controllers.

The algorithm halts if the cluster reaches a predefined size *mcs* (max cluster size), or the increased cluster is equal to the old one (*newT = T*). The next switch in *A* is then selected and inserted in an empty cluster *T*. When the max starting switch list size is reached, all the migration alternatives are evaluated (*AlternativeEvaluation)*. The best alternative composed by the cluster and the target SDN controller are chosen and the migration can be executed.

Given the alternatives vector, the function *EvaluateMigrationAlternatives* chooses the best alternative ([T^0,Target_SDN_controller^0]) among them, that optimizes one of the following *Evaluation-Method* like:

minMax - minimize the maximum controllers' load:

$$\underset{alternatives}{argmin}\ (\ max\ [L(CT_1), \ldots, L(CT|_{CT})]) \quad (5)$$

minSum - minimize the sum of controllers' load:

$$\underset{alternatives}{argmin} \sum_{CTm \in CT} L(CTm) \quad (6)$$

Note that the optimization criteria presented above is the unique *load balancing objective*, based on the current load of the controllers. This can be considered as a limitation of this method.

IV. MULTI-CRITERIA OPTIMIZATION ALGORITHMS

The placement of the SDN controllers and/or selection, or switch migration may involve several particular metrics. The migration of switches can use the metric defined by formulas (5) or (6). So, to achieve particular objectives, appropriate static and/or dynamic optimization algorithms can be applied. However, the CPP, CSP and switch migration problems have naturally multiple conditions characteristics; therefore, the MCDA is a good approach to achieve a convenient trade-off solution.

This paper uses the same variant of MCDA implementation i.e., the *reference level (RL) decision algorithm* (MCDA-RL) [20]. Here, the MCDA will be applied as to enhance the solution for dynamic switch migration, by adding the possibility to obtain a trade-off between load balancing objective and some other possible criteria. The MCDA-RL selects the optimal solution based on normalized values of different metrics. For the sake of completeness we summarize the MCDA-RL model.

The MCDA assumes *m* objectives functions (whose positive values, should be minimized). A solution is represented as a point in a space R^m of objectives; the decision parameters/variables are: v_i, i = 1, ..m, with $\forall i, v_i \geq 0$; 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. The value ranges of decision variables may be bounded by given constrains. The optimization selects a solution satisfying a given objective function and conforming to a particular metric.

The MCDA-RL [20], 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, below which the decision variables (i.e., the associate solutions) are seen as similar (i.e., any solution can be seen as "good"- from the point of view of this variable). 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:

$$\begin{aligned} r_i &= \max [v_{is}], s = 1, 2, ..S \\ a_i &= \min [v_{is}], s = 1, 2, ..S \end{aligned} \quad (7)$$

Normalization can make the algorithm agnostic versus different nature of criteria; the normalized non-dimensional values can be numerically compared despite their different nature. 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). For each variable v_{si}, a ratio is computed:

$$v_{si}' = (r_i - v_{si})/(r_i - a_i), \quad \forall s, i \quad (8)$$

The factor $1/(r_i-a_i)$ - plays also the role of a weight. A variable having high dispersion of values (i.e., *max – min* has a high value in formula (7)) will have lower weight and so, greater chances to be considered in determination of the minimum in the next relation (9). 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 [18]):

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}':

$$\min_s = \min\{v_{si}'\};\ i=1..m \quad (9)$$

Step 2. Select the best solution:

$$v_{opt} = \max\ \{\ \min_s\ \}, s=1, ..S \quad (10)$$

Formula (9) 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 (10) 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).

Different policies can be applied for selection; 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:

 ISBN: 978-1-61208-753-5

$$v_{si}' = w_i(r_i - v_{si})/(r_i - a_i) \quad (11)$$

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. MULTIPLE CONDITIONS-AWARE DYNAMIC SWITCH MIGRATION

This section will develop the main contribution of this paper, i.e., to consider several conditions to decide upon SDN switch migration and therefore, to transform the BalCon algorithm (shortly described in Section III, [13][14]) in a multi-criteria one, by making a trade-off optimization based on several weighted criteria. The idea is that the switch migration will change the controller-switch mapping and therefore, it can be seen as a new optimization problem derived from CPP/CSP.

Examples of several metrics of interest have been presented in [18]. A selection of them (at network provider choice and policies) can be included in the function *EvaluateMigrationAlternatives* of the BalCon algorithm, thus offering the possibility to decide upon a solution determined by multiple conditions and not only by the controller overload. The MCDA methodology will be applied. Some examples of such metrics are given below.

A. Other metrics examples

The SDN-controlled network can be 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. Note that this graph is different from the flow-based graph considered in the Section III. The edges weights may represent an additive metric (e.g., *average propagation latency*).

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 can be 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 could take into account failure events of links or nodes. It is assumed that the controllers are installed in particular positions of the set of network nodes V. A few examples are given below:

Example 1:

Worst_case_latency

$$L_{wc} = \max_{v \in V} \min_{c \in C_i} d(v,c) \quad (12)$$

Average_latency:

$$L_{avg}(C_i) = \frac{1}{n} \sum_{v \in V} \min_{c \in Ci} d(v,c) \quad (13)$$

The CPP algorithm should find a placement C_{opt}, where *either average latency* or *the worst case latency is minimized.* The limitations of the optimization process based on the above metrics (12) and (13) consist in: static values assumed for latencies, despite that delay is a dynamic value in IP networks; only free-failure case are considered; no upper limit exists on the number of forwarders/switches assigned to a controller; not taking into account the inter-controller connectivity. Another possible metric in failure-free case is *maximum cover*, [4]. 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.

Example 2:

Nodes/links failures (Nlf)

This example will consider a failure-aware metric. Links or nodes failures can cause some switches to loose access to controllers. Therefore, a particular optimization objective could be to find a switch-to-controller mapping that minimizes the number of switches possible to get into controller-less situations, in various scenarios of link/node failures. A realistic assumption is to limit the number of possible simultaneous failures at only a few (e.g., two [7]).

For *any given placement* C_i of the controllers, an additive integer value metric $Nlf(C_i)$ could be defined: 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; in other words, count the number of isolated nodes; compute the maximum value (i.e., consider the worst failure scenario). One obtains the formula (14) where *k* covers all scenarios of F.

$$Nlf(C_i) = \max Nlf_k(C_i) \quad (14)$$

The *optimization algorithm* should find a *placement which minimizes (14).* It is expected that increasing the number of controllers, will decrease the *Nlf* value. Note that the optimum solution based on the metric (14) could be very different from those provided by the algorithms using the metrics (12) or (13).

B. Multi-condition algorithm

Transformation of the BalCon algorithm in a MCDA type is realized by modifying the final BalCon phase and is summarized below. Note that positions of the controllers in the network is fixed; only the mapping switch-to-controller can vary.

The function *ComputeMigrationAlternatives* provides several solutions of switch assignment to controllers, to avoid controller overload.

The function *EvaluateMigrationAlternatives* will be replaced by a new function *Multi-condition_Eval_Migration_Alternative* in which the inputs are:

- migration alternatives provided by the original *ComputeMigrationAlternatives*
- solutions of switch-to-controller mapping provided by other algorithms based on several criteria of interest, selected from, e.g.: to minimize the maximum controllers' load (5); minimize the sum of controllers' load (6); worst case latency (12);

average latency (13); node/link failure (14) and maybe others. Note that the detailed algorithms for the metrics (12-14) are not presented here.

We also stress the idea that this study will not aim to select a given set of "best" criteria and use them for optimization. The reason is that such a selection is actually dependent on the particular SDN network characteristics and, more important on the policies of the SDN network provider/owner in defining the goals of the optimization process. So, this study shows the applicability and usefulness of multi-criteria in solving dynamic CSP problems.

The working phases are the following:

(1)*Phase 1*:

1.1. Define the additional criteria, (other than the controller load) i.e., the decision variables of interest and their priorities.

1.2. Consider all solutions C1, C2, .. (a solution is a particular switch-to-controller mapping provided by the *ComputeMigrationAlternatives* function (these would result after migration of a cluster).

1.3. Compute the values of the *normalized metrics for each candidate solution* (i.e. future MCDA candidate solution), by using specialized algorithms and their associated metrics.

The Phase 1 phase has as outputs the set of candidate solutions and values to fill the entries of the matrix M defined in Section IV.

(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 (8)); run the MCDA Step 0, 1 and 2 (formulas (9-11)).

The Phase 2 provides the migration solution.

C. Simple numerical example

This subsection will present a simple numerical example to illustrate the multi- criteria switch migration solution. The network has three domains D1, D2, D3, each one controlled by a controller.

The network is described by a graph, in which three sets of flow rates F1, F2, F3 are represented. The links could be physical or overlay links. The numbers written on some links show the estimated average latency between those vertices (this is an additive metric).

The loads for CT_1, CT_2, CT_3 (applying the formulas (1) and (2)) with α=1, β=0.1:

$L(CT_1)$ = (30+25+20)α + (3*30 +2*25 + 3*20)β = 75 + 20 =95 units of load.

$L(CT_2)$ = (25 + 20)α + (2*25 + 2*20)β = 45+9 = 54 units of load.

$L(CT_3)$ = (20)α + (3*20)β = 20+6 = 26 units of load.

Supposing that L=80 (maximum load for a controller) one can see that the CT_1 is overloaded. Therefore a cluster of switches migration from the domain of CT_1 would make a better load balance and solve the overload of CT_1.

Solution M1. Suppose that the cluster {V_1, V_5} would migrate to CT_2. The new loads would be:

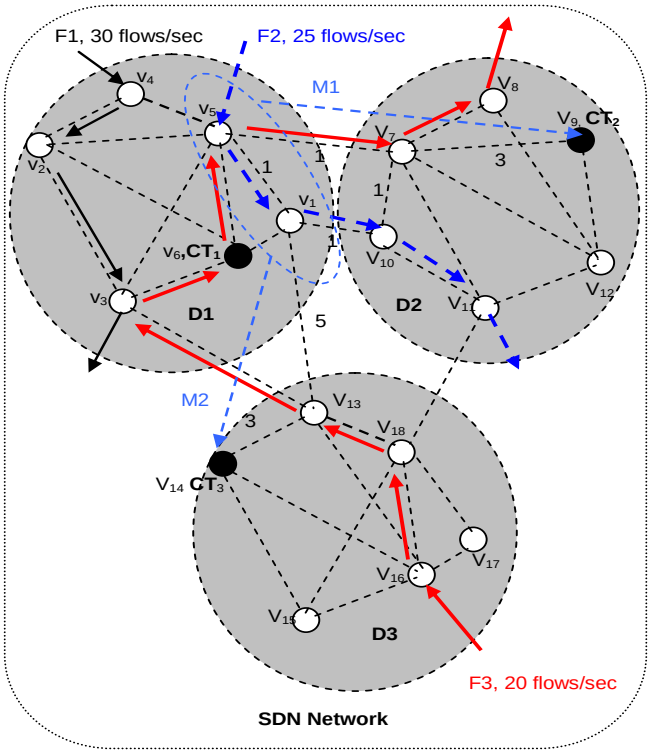

Figure 2. Switch migration example

$L(CT_1)$ = (30+20)α + (3*30 + 2*20)β = 50 + 13 = 63
$L(CT_2)$ = (25 + 20)α + (3*25 + 3*20)β = 45+13.5 = 58.5
$L(CT_3)$ = 26 (not modified)

Solution M2. If the cluster {V_1, V_5} would migrate to CT_3, then the new loads would be:

$L(CT_1)$ = (30+20)α + (3*30 + 2*20)β = 50 + 13 = 63
$L(CT_2)$ = 54 (not modified).
$L(CT_3)$ = (25+20)α + (2*25+3*20)β = 45+11 = 56

If a single criterion (BalCon: e.g., considering the formula (5)), then the migration solutions M1 and M2 are equally acceptable (max load = $L(CT_1)$ = 63).

However if one consider additional multi-criteria then the best solution selected by a multi-criteria algorithm could be different. Examples are given below (see Figure 2).

Worst case latency (formula (12))

Let us suppose that shortest path from vertices V_1, V_5 to controllers are: d(V_1, CT_2) = 5; d(V_5, CT_2) = 4; d(V_1, CT_3) = 8; d(V_5, CT_3) = 9. So, the MCDA for the criterion *worst case latency* will prefer the M1 solution as better.

Nodes/links failures (Nlf)(formula (14))

It can be seen that M2 solution is better than M1, if link failures are considered. In D2 domain, a failure scenario with links V_7-V_9 and V_9-V_{12} out of order, will isolate the controller CT_2 and consequently, *Nlf = 5*. On the other side in D3 domain, the maximum value of this metric is *Nlf* = 1.

If multiple other criteria are included in a MCDA based decision, then the final solution will depend on weights assigned to different criteria.

Of course, other criteria (see [18]) could be added to MCDA with different weights for the decision variables. We limited our numerical example above to a simple case, just to illustrate the idea of the approach, i.e., to prove the usefulness of the multi-criteria in deciding upon SDN switch migration. Complete calculations could be performed using the full BalCon algorithm enriched with MCDA procedures.

We have to acknowledge the limitations of this study as a work in progress. This paper does not include the study of the control plane signalling between controllers, in order to achieve awareness of the controllers about the new situation of their domains; this is a separate problem. No quantitative performance evaluations of the migration procedures are presented here. The limit of controller acceptable load has been considered to be the same for all controllers; actually these values can be different. These topics could be subjects of additional studies.

VI. CONCLUSIONS AND FUTURE WORK

This paper extended the studies [13][14] on dynamic switch migration, by enriching the final decision on the migration solution based, i.e., based not only on controllers load values but on multiple conditions and using multi-criteria decision algorithms (MCDA), as in [18]. The advantage of MCDA is that it can produce a tradeoff (optimum) result, while considering several weighted criteria, part of them even being partially contradictory.

The goal here was not to select a given set of "best" criteria and use them for optimization. The reason is that such a selection is actually dependent on the particular SDN network characteristics and, more important on the policies of the SDN network provider/owner in defining the goals of the optimization process. So, this study is focused to show the applicability and usefulness of multi-criteria in solving CPP/CSP problems not only in static context but also during run-time of the SDN network.

This is still a work in progress; a simulation model is currently in development. An initial proof on concept has been performed in Section V by using some simple but relevant examples.

Future work will be done to complete and run the simulation model and considering more extended network topologies.

A more deep study should consider the amount of signaling between controllers while switch migration occurs. Another open issue is to get a trade-off between the frequency of switch migration events and stability of the network (i.e., to avoid excessive migration of switches) - versus the rate of traffic changes in the data plane of the network. The dynamic of the data plane after switches migration (e.g. in multicast context) is still an open research issue.

REFERENCES

[1] 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, 2019].

[2] 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, 2019].

[3] 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.

[4] 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, 2019].

[5] 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.

[6] 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: 4, 2019].

[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, pp. 1-9, 2013, https://ieeexplore.ieee.org/document/6662939/, [retrieved: 4, 2019].

[8] 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), pp.1909 - 1915 12/2014, https://ieeexplore.ieee.org/document/7037087/, [retrieved: 4, 2019].

[9] 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.

[10] S.Yoon, Z. Khalib, 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.

[11] A.K. Singh and S. Srivastava, "A survey and classification of controller placement problem in SDN", Int'lJournal of Network Management, March 2018, DOI: 10.1002/nem.2018, pp.1-25, https://www.researchgate.net/publication/323974224 [retrieved: 4, 2019].

[12] A. Kumari and A.S. Sairam, "A Survey of Controller Placement Problem in Software Defined Networks", May 2019, https://arxiv.org/abs/1905.04649 [retrieved: 7, 2019]

[13] M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, and M. Marchese, "Balcon: A distributed elastic SDN control via efficient switch migration", in Proc. IEEE Int. Conf. Cloud Eng. (IC2E), pp. 40–50, April 2017.

[14] Yang Xu, et. al., "Dynamic Switch Migration in Distributed Software-Defined Networks to Achieve Controller Load Balance", IEEE Journal on Selected Areas in Communications , Vol. 37, No. 3, pp.515-528, March 2019.

 ISBN: 978-1-61208-753-5

[15] Y. Zhou, K. Zheng, W. Ni, and R. P. Liu, "Elastic Switch Migration for Control Plane Load Balancing in SDN", IEEE Access, Vol. 6, DOI 10.1109/ACCESS.2018.2795576, pp. 3609-3618, 2018.

[16] N. Mouawad, R. Naja and S. Tohmé, "Optimal and Dynamic SDN Controller Placement", July 2018, DOI: 10.1109/COMAPP.2018.8460361, https://www.researchgate.net/publication/326246703, [retrieved: 7, 2019].

[17] J. Cui, Q. Lu, H. Zhong,M.Tian, and L.Liu, 'A load-balancing mechanism for distributed SDN control plane using response time', IEEE Transactions on Network and Service Management, pp. 1-10, doi: 10.1109/ TNSM.2018.2876369, 2018.

[18] 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, 2019].

[19] 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: 6, 2019].

[20] A. P. Wierzbicki, "The use of reference objectives in multi-objective optimization". Lecture Notes in Economics and Mathematical Systems, vol. 177. Springer-Verlag, pp. 468–486, 1980.

 ISBN: 978-1-61208-753-5