Scalable Load Balancing Scheme for Distributed Controllers in Software Defined Data Centers

Mohamed Escheikh, Kamel Barkaoui

Scalable Load Balancing Scheme for Distributed Controllers in Software Defined Data Centers

Mohamed Escheikh
SYSCOM, ENIT
Tunis, Tunisia
Email: mohamed.escheikh@enit.utm.tn

Kamel Barkaoui
CEDRIC, CNAM
Paris, France
Email: kamel.barkaoui@cnam.fr

***Abstract*—We propose in this paper a hysteresis multiple-threshold-based load balancing (LB) system intended to distributed Software-Defined Networking (SDN) control plane network architecture. The considered LB system is based on Markov chain model and is governed by a control policy using a set of scaling out/in thresholds to evenly distribute traffic between an overloaded SDN controller (client controller) and a lightly loaded neighboring controller (server controller). This is achieved to circumvent large discrepancy in resource utilization through dynamically adapting the global available capacity. The proposed LB scheme aims to achieve multi-objectives tradeoff relevant to scalability, high availability, agility, flexibility, resource utilization, blocking probability and power saving without incurring significant overhead. We highlight through numerical investigations the effectiveness of our proposed model. This is achieved by means of transient and steady state analysis, based on appropriate performance metrics such as average aggregated capacity, transition rate (between client and server) and blocking probability. We show also how the proposed LB scheme performs the right scaling and resource provisioning decisions with respect to specific requirements.**

***Index Terms*—SDDC, SDN, NFV, Load balancing, Hysteresis, Markov chains.**

I. Introduction

The advent of cloud computing in IT environment and the spectacular growth of stringent applications, in the era of mobile, social, cloud, IoT and big data, has generated huge ever changing requirements of resources management and an urgent need to rationalize resource utilization.

In order to meet these requirements the trend of the cloud IT providers is to deploy massively in their data centers SDN paradigm and Network Functions Virtualization (NFV) technology to build Software Defined Data Center (SDDC) architectures. The objective is to offer on one hand agile and scalable resource provisioning and on the other hand highly flexible and elastic service delivery capabilities.

SDN and NFV when deployed in concert can bring several benefits to virtualized data centers [14] since they are complementary technologies. SDN is an emergent paradigm shifting from traditional network to the next generation Internet to provide programmability and more flexibility for introducing innovation in service as well as in network design and management [11]. Despite SDN brings a centralized control plane view enabling several advantages in terms of abstraction and control of the underlying network (i.e. data plane), achieving such a view through one central node is likely to cause critical scalability and reliability issues. In this regard the recent trend nowadays is to envisage multiple controllers based architecture. However, even though this solution empowers to cope with bottleneck and availability issues, traffic unbalance remains a potential weakness and a traffic engineering challenge to be met. The aforementioned problem is manifested whenever static configuration of the mapping between a forwarding element, in the data plane, and a controller is achieved. Hence scalable LB should be deployed in the SDN control plane.

On the other hand NFV enables to virtualize network functions such as ADCs (Application Delivery Controllers), WAFs (Web Application Firewalls) and load balancing. Implementing network functions as a software removes the need for hardware appliance and is achieved through virtual appliances hosted on commodity hardware. In a nutshell NFV together with SDN provides more flexibility, cost-effectiveness, scalability, security and better quality of experience to next generation data centers.

In this paper, we propose a modeling analysis of LB model for distributed SDDC controllers, with logically centralized view, enabling scalability and high availability while enhancing resource utilization and power saving. The proposed LB model is based on agile and flexible control policies hysteresis-based with multiple threshold structure based on Continuous Time Markov Chains (CTMC). It allows traffic flow management empowering dynamic accommodation of controller resources to unpredictable workload variations without incurring significant overheads.

The remainder of this paper is organized as follows. Section II presents the related work. Section III emphasizes on LB in distributed SDN controller networks. Section IV addresses the system description and client/server controller roles. Section V presents the proposed LB model. Related numerical analysis and performance evaluation is provided in Section VI. Section VII concludes this paper.

II. Related work

Authors in [18] present SDN survey describing SDN features and illustrating in detail its layers. They succinctly tackle multi-controllers issue in the context where control layer performance enhancements are needed. In [3] a comprehensive overview of SDN multi-controllers architectures is provided. Issue related to implementing LB mechanism in SDN control

plane, based on distributed multi-controllers architecture, had been investigated in several previous works where LB making decision is either centralized [4], [12] or distributed [19]. Authors in [17] introduce an overview of current research status in SDN and multi-domain SDN, focusing particularly on OpenFlow protocol, and its future related challenges. SDN control plane synchronization issue has been addressed in [20] and [1]. In [1], authors implement a Communication Interface for Distributed Control plane (CIDC) that allows synchronization and notifications exchanges. The aim of such implementation is to enable distributed services such as Firewall and Load Balancer between multiple distributed SDN controllers in order to enhance security and overall quality of service in distributed SDN architecture. Related results show the feasibility of CIDC in terms of performance compared to earlier models based on clustered controllers. [6] proposes a novel online algorithm to mitigate scalability and reliability concerns jointly considering dynamic association between switches and SDN controllers. The reason of such consideration is to dynamically devolve flows' control to switches leading to convenient cost-latency trade-off. Authors in [2] present different existing solutions and mitigation techniques that address SDN scalability, elasticity, dependability, reliability, high availability, resiliency, security, and performance concerns. In [13] authors present a mathematical model representing a multi-controllers loose management strategy dynamically adjusting interaction frequency and enhancing communication efficiency between controllers and network devices. In [5] an adaptive elastic distributed SDN architecture model is proposed as an optimization problem for selecting a minimum number of active controllers by changing mapping between switches and controllers through switch migration between domains according to network load. In [16], authors study two scheduling problems for delay tolerant applications on minimizing on one hand the peak resource usage for a given set of demands and time-varying resource cost while ensuring each demand service without missing their deadlines and on the other hand the demand completion time for a given maximum allowable resource. Wang et al. [15], formulated a joint optimization virtual machine placement problem with both delay and migration cost considered to leverage the spatial variation in the VMs migration for delay and cost optimization.

In literature hysteretic based models had been proposed to achieve multi-objective tradeoff. In [10] proposes an analytic framework for modeling Digital subscriber line DSL modems and providing optimal sleeping policies. This is achieved through finding a convenient tradeoff between energy consumption, delay performance and stability. The problem is formulated as a continuous time Markov Decision Process (MDP). Its shown that the optimal sleeping policy for the above MDP is a monotone hysteretic policy characterized by two queue length thresholds enabling to tune the modem from on to off or vice versa. Authors in [7] [8] [9] show that in a data center infrastructure based on multiple identical servers the optimal sleep energy efficient policy may be hysteretic and governed a double threshold.

III. Load balancing in distributed SDN controller networks

A. LB Activation/Deactivation through Horizontal Scaling (out/in) in SDN Control Plane

In order to provide scalable resource provisioning based on adding and removing controllers (or a set of instances in a controller), LB process is activated/deactivated adaptively according to load fluctuation and disparity between the set of controllers forming the control plane network. Activation and deactivation are governed by an horizontal scaling mechanism including two alternating phases. The first one is referred to as *scaling out* phase whereas the second is known as *scaling in* phase. Load scalability is enabled, in the considered distributed architecture, by flexible hysteretic thresholds empowering to easily expand and contract its resource pool to accommodate changing load. *Scaling in* and *scaling out* virtualized resources is performed based on a threshold-based scaling policy efficiently leveraging real time network status collected toward SDN monitoring tools.

1) Scaling out phase

Scaling out phase is implemented through scaling out technique activating the LB process. Whenever current controller load reaches a given threshold the controller behaves as a client and attempts to trigger LB process. This is achieved through sending a LB request to potential neighboring controller(s) (server(s)) able and willing to cooperate.

2) Scaling in phase

Whenever the global resource utilization of a set of active aggregated controllers, taking part in a LB process, drops under an acceptable lower limit corresponding to a power-inefficient utilization ratio, the *scaling in* phase is triggered. During the *scaling in* phase the LB process is then deactivated and additional resources provisioned by neighboring controller(s) are released.

B. LB Policy

LB policy defines how to distribute among various nodes (i.e controllers) according to specified weight values. appropriate choice of LB policy is of paramount importance to address scalability-availability-performance tradeoff issue. Policy's attributes should be closely related to workload dynamic, resource availability and QoS requirements. Indeed adopting inappropriate policy in a given context related to a given workload is equivalent to not taking the right decision at the right time which results in widening mismatch between workload assigned and resource provisioned. This would inevitably lead to providing unsatisfactory results not complying with the preset objectives of the adopted policy. Also oscillatory behavior should be avoided. This may be achieved by considering a LB hysteresis-based control policy with carefully chosen attributes and thresholds.

LB decision is based on calculating resource utilization and relies on comparing one controller's available capacity with that of one (or several other) controllers to determine whether a LB is enabled. The problem to be addressed concerns then when

and where to live-migrate controller instances straightforward a controller is detected overloaded. The LB decision in our modeling approach is locally enabled at the current controller whenever three major conditions are fulfilled. The first one concerns the level beyond which the current controller (i.e. client) is estimated overloaded, the second is related to the ability of potential neighbor(s) (server(s)) to handle traffic spikes of the overloaded controller and the last concerns the network capabilities to enable suitable communication conditions for traffic exchange between controllers.

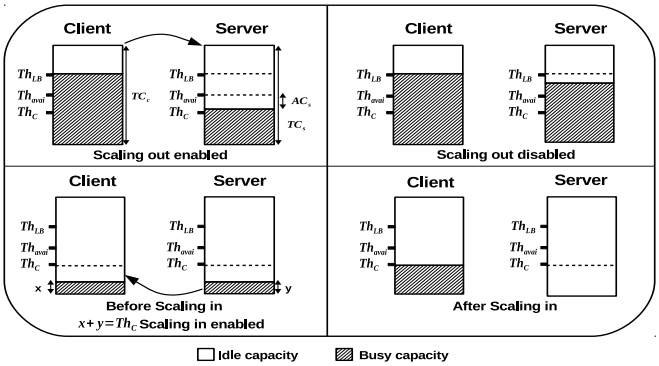


Fig. 1: LB (scaling out/in) conditions with respect to Client/server workload

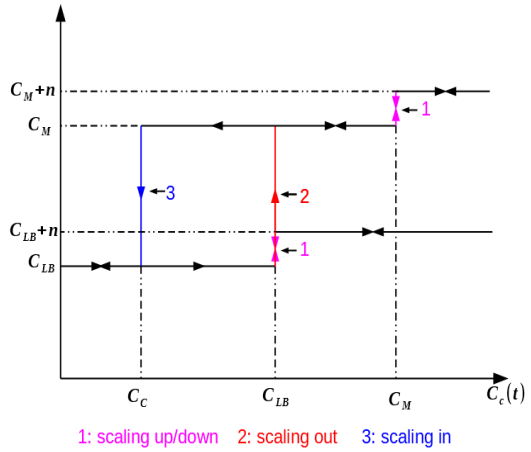


Fig. 2: IAC vs $C_c(t)$

IV. System Description and Client/Server Controller Roles

A. System Description

In what follows let us examine the main assumptions considered for modeling a distributed SDN control plane network with centralized logical view implementing a LB mechanism. We consider that:

◇ The system to model is a SDN control plane with logically distributed architecture controlling a data plane and a logically centralized view.

◇ The network (i.e. data) plane encompasses a set of forwarding switches and each switch is connected to the best controller in its proximity.

◇ The SDN control plane consists of a set of controllers (C_i where $1 \leq i \leq Nc_{max}$).

◇ Each controller C_i is composed of a set j of virtual instances V_{ij} where $1 \leq j \leq Nv_{max}$.

◇ Each controller C_i runs an instance of the LB scheme proposed in this paper, and may play the role of either a client (requesting a LB) or a server (able and willing to cooperate by sending back to the client a LB response).

◇ There exists a communication link between each pair of neighboring controllers and each link is subject to impairments: Whenever a LB process is going to be triggered, each link L_{ij} connecting a heavy-loaded controller (C_i) to a lightly loaded controller C_j may incur congestion or failure, with probability p_{ij}. This assumption allows us to take into consideration network impairments in our modeling approach.

◇ The controllers' network is partitioned into different non overlapping SDN domains to enable scalability and incremental deployment.

◇ The controller network is presented as a shared pool of instances.

◇ Each controller governs a specific (local) manageable SDN domain and each domain includes a subset of switches deployed in the data plane. The number of switches per domain do not exceed the controller capacity.

◇ The LB decision making is localized to each controller.

◇ SDN controller resources are pooled to serve multiple switches based on multi-tenant model with diverse physical and virtual resources dynamically and flexibly assigned and

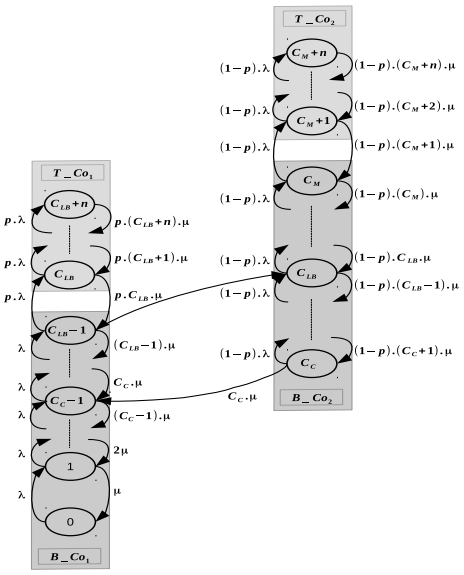


Fig. 3: LB model CTMC

reassigned in agreement with data plane requirements.

B. Client/Server Controller Roles

A controller may embody two kinds of LB roles (client or server) depending on its instantaneous capacity (load). A client is a controller reaching an overload threshold and requesting LB service from its neighboring server controllers. A server is a controller having enough available capacity and offering temporarily a portion of this capacity to the client for LB.
In what follows let's define some parameters describing the load of client/server controller. These parameters will be used to estimate the suitable load levels from which a controller will behave as a client to request or deactivate LB or behave as a server to offer a LB service.
$\diamond$ $C_c(t)$, the instantaneous client controller capacity at time t;
$\diamond$ TC_c, the total client controller capacity;
$\diamond$ $U_c(t) = \frac{C_c(t)}{TC_c}$, the instantaneous client controller utilization ratio (a parameter quantifying (in percent) the relative client load level).
$\diamond$ $C_{s_i}(t)$, the instantaneous capacity of the i^{th} neighboring (sever) controller at instant;
$\diamond$ TC_{s_i}, the total controller capacity of the i^{th} server controller;
$\diamond$ $U_{s_i}(t) = \frac{C_{s_i}(t)}{TC_{s_i}}$, the instantaneous utilization ratio of the i^{th} server controller, (a parameter quantifying (in percent) the relative load level of the i^{th} neighboring server);
$\diamond$ $U_c(t)$ (resp. $U_{s_i}(t)$) values are in the range between 0 and 100%. Hence, when the client (resp. i^{th} server) controller capacity becomes fully loaded, then $U_c(t)$ (resp. $U_{s_i}(t)$) reaches the value 100% whereas when the client (resp. i^{th} server) capacity is idle (no load), $U_c(t)$ (resp. $U_{s_i}(t)$) value is equal to 0%.
$\diamond$ $AC_{s_i}(t)$, the available capacity offered at the instant t by the i^{th} server to the client controller when this latter sends a LB request.

Hence each client controller, based on the knowledge of its own $U_c(t)$ and $U_{s_i}(t)$ of every neighboring i^{th} server controller, may ask to activate (resp. deactivate) the LB mechanism through triggering the scaling out (resp. in) phase. This is enabled toward sending a request to its neighboring controllers. These latter are invited to reply by sending back a positive or a negative response. In the first case each server i reserves a capacity $AC_{s_i}(t)$ to the client whereas in the second case no capacity will be reserved.

V. Client/Server LB Model

The *LB model*, (Fig.2), elucidates interaction between one client controller and one server controller. In the above model:
$\diamond$ Each controller capacity consists of a set of servers (instances) with service duration exponentially distributed with parameter μ.
$\diamond$ The client is fully represented through its capacity in terms of instances, whereas LB server is represented partially toward its reserved available capacity for LB.
$\diamond$ We focus on the traffic handled by the client controller. This traffic is assumed exponentially distributed with parameter λ.
$\diamond$ We denote also $\rho = \frac{\lambda}{\mu}$.

A. Threshold Capacities

$\diamond$ $TC_c = C_{LB} + n$: the maximum capacity of the client controller.
$\diamond$ TC_s: : the maximum capacity of the server controller.
$\diamond$ C_{LB}: a preset threshold capacity having value smaller than the total client capacity $C_{LB} + n$. C_{LB} corresponds to the minimum capacity initially reserved by the client controller before incurring an overload.
$\diamond$ n (Fig.2, column 1 (resp. column 2)): the remaining client controller capacity.
$\diamond$ $AC_s = AC_s(t)$: the amount of available capacity offered by the server controller to the client at instant t. t is the instant where the client sends a LB request to the server.
$\diamond$ $C_M = C_{LB} + AC_s$: the reserved capacity by both the client and the server given that LB is triggered and the offered server capacity, AC_s, is not fully consumed.
$\diamond$ C_C: a fixed threshold capacity ($C_C < C_{LB}$) below which the *scaling in* phase is triggered. Likewise we consider that: $C_M > C_{LB} > C_C$.

B. LB Policy

For this model (Fig.2, Fig.3) there is no LB policy since the client has no choice except to collaborate with a single server. A more versatile LB model with LB policy enabling LB between one client and several servers is also investigated however its presentation is out the scope of this paper.

C. Hysteresis Thresholds

The LB model (Fig.2, Fig.3) is hysteresis-based and involves different alternating phases. Every phase corresponds to a given state (described by workload level and power state). Switching between phases is enabled once reaching suitable workload thresholds. These thresholds (expressed in percent) are defined as follows:
$\diamond$ Th_{LB}: the load threshold beyond which a controller is considered overloaded (i.e. hot spot) and is going to trigger a LB request ($Th_{LB} = \frac{C_{LB}}{TC_c}$) . This indicates that the client controller is heavily loaded and hence one or more instances running on it should be transferred away through migration techniques in order to mitigate bottleneck. Whenever $U_c(t)$ reaches this threshold (i.e. Th_{LB}) the controller behaves as a client and may trigger *scaling out* phase.
$\diamond$ $Th_{avai} = \frac{AC_s(t)}{TC_s}$: the load threshold below which a controller is considered as a server able to offer a part of its available capacity (i.e. $AC_s(t)$) to overloaded controllers (i.e. client) to balance workload. For a server controller having $U_s(t) <$ Th_{avai} may play a server role.
$\diamond$ $Th_C = \frac{C_C}{TC_c}$: the load threshold below which controller is considered lightly loaded (i.e. cold spot) and is going to trigger *scaling in* phase. Th_C indicates that the global resource utilization ratio of a controller is below a computing threshold. In other words, it's advocated to run controller consolidation and switch the idle controller into sleep power

state if this may lead to power saving. Th_C may be chosen equal to an estimated acceptable lower limit leading to power-inefficient utilization ratio. Fig.1 summarizes the LB conditions (hysteresis-based) with respect to client/server workload. Notice that every controller may be assigned a set of specific and dynamic thresholds.

D. Instantaneous Aggregated Capacity (IAC) vs $C_c(t)$

The proposed LB model (Fig.2, Fig.3) describes, from a client controller perspective, interaction between one client and one server. The instantaneous aggregated capacity (i.e. IAC) is defined as the instantaneous reserved capacity for a given incoming traffic, $C_c(t)$, to the client controller.

IAC is governed by the following rules (Fig.2):
For $C_c(t)$<C_{LB}, C_{LB} capacity is initially reserved by the client controller; Whenever $C_c(t)$ reaches the capacity C_{LB}, two alternatives may be considered:
◇ **Alternative1 (LB is enabled)**: In this case the client may profit from LB, with probability $(1-p)$, the *scaling out* phase may be triggered and the reserved capacity switches from C_{LB} (reserved by the client) to C_M. This means that the initially reserved capacity C_{LB} by the client is extended by AC_s (Fig.3, B_Co_2). Hence the total reserved capacity becomes equal to $C_M = C_{LB} + AC_s$. In such case and as $C_c(t)$ increases and reaches C_M, a *scaling up* phase is triggered and the global reserved capacity switches from C_M to C_M+n. If the reserved capacity is $C_M + n$ and $C_c(t)$ decreases below C_M a *scaling down* phase is started and the reserved capacity switches down from C_M+n to C_M. On the other hand if the reserved capacity is C_M and as $C_c(t)$ decreases until reaching $C_c(t)$ a *scaling in* phase is triggered and the reserved capacity switches from C_M to C_{LB}.
◇ **Alternative2 (LB is not enabled)**: In such case the client fails in finding available neighbor to ensure LB with probability p and in such case, if the reserved capacity is C_{LB} and $C_c(t)$ increases and goes beyond $C_{LB}-1$, a *scaling up* phase is initiated and the reserved capacity (in the client) switches from C_{LB} to $C_{LB}+n$. If the reserved capacity is $C_{LB}+n$ and $C_c(t)$ decreases below C_{LB} a *scaling down* phase is triggered and capacity switches down from $C_{LB}+n$ to C_{LB}.
It's worth mentioning also that whenever $C_c(t)$ exceeds $C_{LB}+n$ or C_M+n the excessive incoming traffic to the client controller will be lost.

E. Markov Chain Description

The different macro-states of the CTMC may be summarized as follows (Fig.3):
◇ The bottom part, (Fig.3, column 1, B_Co_1), includes states S_{1i} (0≤i¡C_{LB}) and describes a client not yet overloaded.
◇ The top part (Fig.3, column 1, T_Co_1), includes states S_{1i} (C_{LB}≤i≤$C_{LB}+n$) describing the client with $C_c(t)$ reaching and exceeding C_{LB} without finding any LB service with probability p.
◇ The bottom part (Fig.3, column 2, B_Co_2), includes two subsets of states. The first one consists of states S_{2i}, (C_C≤i≤C_{LB}) representing client states (given that the LB is active) whereas the second subset includes states S_{2i}, ($C_{LB}+1$≤i≤C_M) corresponding to the offered available capacity AC_{s_1} by the server for the client during LB.
◇ The top part (Fig.3), column 2, T_Co_2), includes states S_{2i} (C_{LB}≤i≤$C_{LB}+n$) describes client states with $C_c(t)$ reaching and exceeding C_{LB} after fully consuming AC_{s_1} instances.

We choose to represent the system states in the infinitesimal generator of the CTMC with two columns according to the following rules: columns are represented in order from the left to the right and the states of each column are represented also in order from the bottom to the top. The above description is necessary to understand the steady state probability distribution evolution versus ρ and IAC versus time describing the system behavior with respect to time.
If we consider R, the amount of reserved capacity given by:

- R=C_{LB} (resp. $C_{LB}+n$) for the set of states represented by B_Co_1 (resp. T_Co_1),
- R=C_M (resp. C_M+n) for the set of states represented by B_Co_2 (resp. T_Co_2),

and A the number of active flows, the pair (R, A) represents a Markovian process. Based on the above assumptions, we model the LB scheme using hysteresis-based CTMC. Given that the CTMC is irreducible and aperiodic its ergodicity is verified and the resolution of the state equation leads to a unique solution. We define a state by a couple (i,j), where i refers to the index of column in the Markov Chain and j refers to the client/server controller capacity states. let also be $\Pi_(i,j)$ is the steady state probability of the state (i,j).

F. LB Performance Metrics AC, TR and BP

Let's define in this subsection some LB performance metrics referred to as Average Aggregated Capacity (AC), Transition Rate (TR) and Blocking Probability (BP). These metrics will be used subsequently to capture LB model behavior through steady state analysis:

$$AC = C_{LB}.\sum_{i=0}^{C_{LB}-1}\Pi_{(1,i)} + (C_{LB}+n).\sum_{i=C_{LB}}^{C_{LB}+n}\Pi_{(1,i)} + C_M.\sum_{i=C_C}^{C_M}\Pi_{(2,i)} + (C_M+n).\sum_{i=C_M+1}^{C_M+n}\Pi_{(2,i)} \quad (1)$$

$$TR = (1-p).\lambda.\Pi_{(1,C_{LB}-1)} + C_C.\mu.\Pi_{(2,C_C)} \quad (2)$$

$$BP = \Pi_{(1,C_{LB}+n)} + \Pi_{(2,C_M+n)} \quad (3)$$

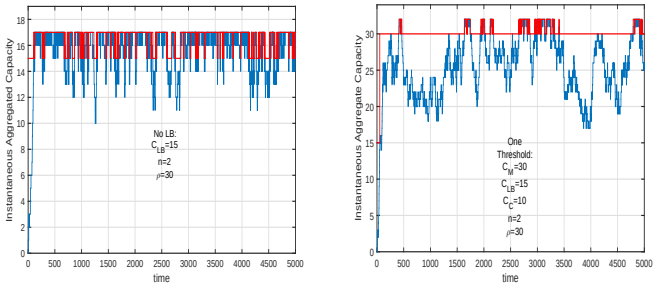
G. Optimal Values for AC and TR

In addition to the rules governing the capacity reservation and the switching between different capacities, a decisive criterion that should be taken into consideration concerns the choice of optimal parameters C_{LB} and C_C enabling to minimize both average aggregated capacity (AC) and ping pong overheads (due to adverse scaling out/in (i.e. TR)). In order to achieve this goal let's define a cost function f_C (eq.

(4)).

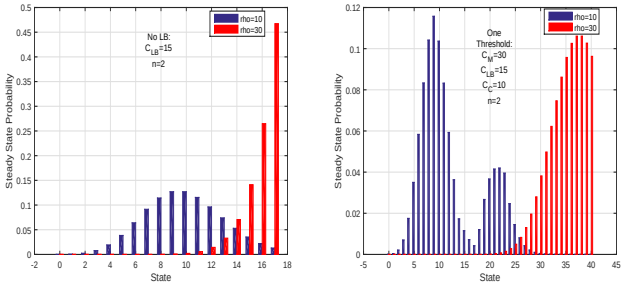
$$fc = \frac{AC}{max(AC)} + \theta . \frac{TR}{max(TR)} \quad (4)$$

where θ is a coefficient enabling to adjust how important is altered TR cost compared to altered BP cost for a data center service provider. Notice that minimizing the cost function leads to find for a given θ a unique couple with minimum values of respectively AC and TR.



(a) $NoLBmodel$ (b) *LB model*

Fig. 4: IAC vs time for different LB models ((No LB (*LB model*, p=1)), (*LB model*, $p = 0.3$) (ρ=30)



(a) $NoLBmodel$ (b) *LB model*

Fig. 5: Steady state probability distribution for different models (No LB model, *LB model* (ρ=10-30)

VI. Numerical Results

In order to assess the performance of the proposed LB model, we conduct several numerical investigations. The first one is based on transient analysis whereas the second is based on steady state analysis. The following parameters are used for illustrative numerical LB models' resolution:

◇ For *LB model* (Fig.2, Fig.3): $C_M = 30$, $C_{LB} = 15$, $C_C = 10$, $n = 2$, $p = 0.3$. Hence AC may alternate between the following capacities: $C_M + n$, C_M, $C_{LB} + n$, C_{LB}.

◇ For *No LB model*: *LB model* is also used to describe controller without LB by setting $p = 1$. In this case AC may alternate between $C_{LB} + n$ and C_{LB}.

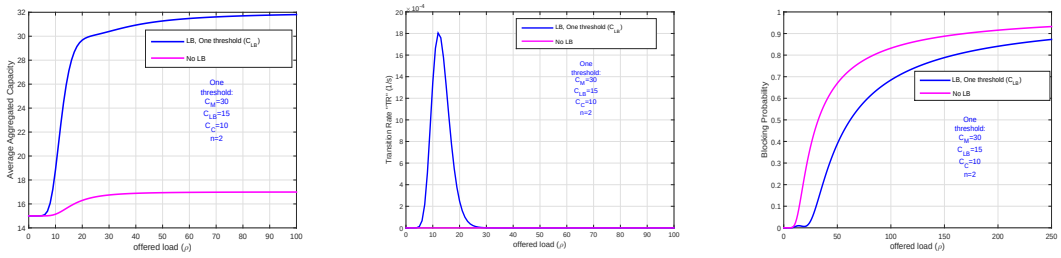
In order to investigate numerical results related to both transient analysis (Fig.4) and steady state analysis (Fig.5) and since the Markov chain of *LB model* is hysteresis-based, it's necessary to take into consideration in which order are represented the states of the Markov chains in the infinitesimal generator (see subsection V-E).

A. Transient Analysis

Fig.4 shows the evolution of IAC vs time, for a given $\rho = 30$. This is given respectively for *No LB model* and *LB model*. We have chosen ρ sufficiently high in order to highlight the impact of $C_c(t)$ on IAC (i.e. on the switching frequency between different threshold capacities). For *No LB model* (Fig.4.(a)) we observe that IAC alternates between only two threshold capacities ($C_{LB} = 15$ and $C_{LB} + n = 17$). This is obvious since in such scenario only *scaling up* and *scaling down* phases are enabled and there is no neighboring server controller able to reserve capacity for the overloaded client controller. In other words the maximum capacity reserved in this case corresponds to the client capacity controller. For *LB model* (Fig.4.(b)) we notice that as $C_c(t)$ fluctuates, IAC switches between four threshold capacities (32, 30, 17, 15). This is justified by the fact that the different scaling phases (out/in/up/down) may be activated and deactivated. However despite the maximum reserved capacity is the same as in *LB model*, the reservation is achieved with finer granularity and more threshold capacities are used (i.e. 32, 30, 27, 25, 22, 20, 17, 15).

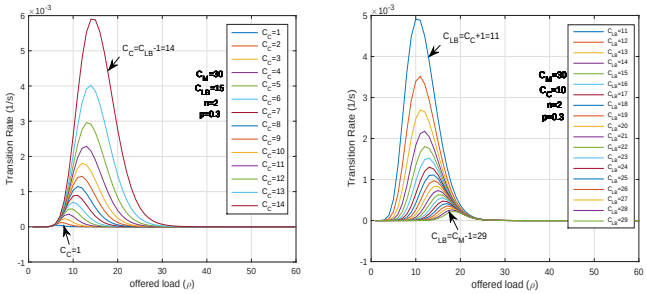
B. Steady State Analysis

Fig.5 shows how workload increase impacts the steady state probability distribution for different LB models (*No LB model* (Fig.5.(a)), *LB model* (Fig.5.(b))) and for different ρ values (i.e. 10, 30). For light incoming traffic (i.e. ρ=10), the most visited states (for LB model) are those of the first column in the CTMC (Fig.5). This means that for such traffic there is almost no LB and the reserved capacity corresponds only to the client controller capacity. However for heavy traffic (i.e. ρ=30) the states of the last column (Fig.5) in the CTMC have higher probability values since the system (i.e. client+server(s)) is heavily loaded. This means that the server controller(s) is/are the most solicited to handle the incoming traffic to the client. Fig.6 plots AC, TR, BP metrics vs ρ for different models: *No LB model* (Fig.6.(a)), *LB model* with $p = 1$ (Fig.6.(b)) and *LB model* with $p = 0.3$ (Fig.6.(c)). A first observation from Fig.6shows that the lack of LB (Fig.6.(a)) yields less AC, no TR and much more BP when compared to cases wher LB is activated (Fig.6.(b), Fig.6.(c)). This is trivial since the absence of LB involves less reserved resources, no scaling out/in phases and greater risk to loose exceeding load. A second observation from Fig.6.(b) and Fig.6.(c)) shows that AC is an increasing function of ρ whenever LB is activated. In Fig.7(a) (resp. Fig.7(b)) we plot AC with respect to C_C (resp. C_{LB}) for different C_{LB} (resp. C_C) values. Similarly Fig.8(a) (resp. Fig.8(b)) highlights TR behavior with respect to C_C (resp. C_{LB}) for different C_{LB} (resp. C_C) values. On the other hand Fig.9(a) (resp. Fig.9(b)) investigates AC (resp.



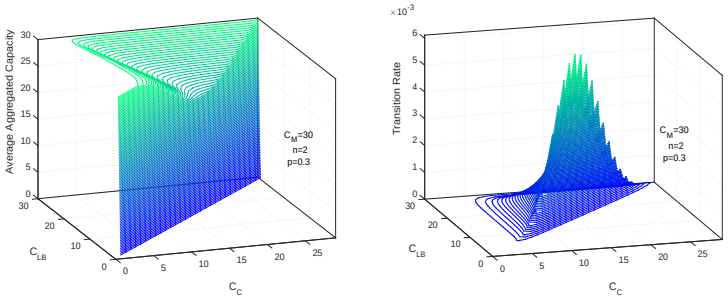
(a) AC vs ρ (b) TR vs ρ (c) BP vs ρ

Fig. 6: LB performance metrics vs ρ for different LB models ((No LB model (*LB model*, $p = 1$), (*LB model*, $p = 0.3$)) (ρ=30)



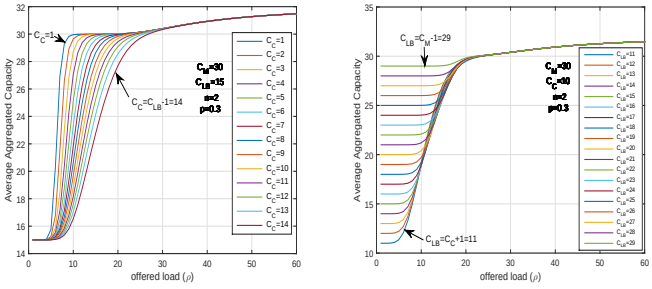
(a) Fixed C_{LB}, varied C_C (b) Fixed C_C, varied C_{LB}

Fig. 7: TR vs ρ



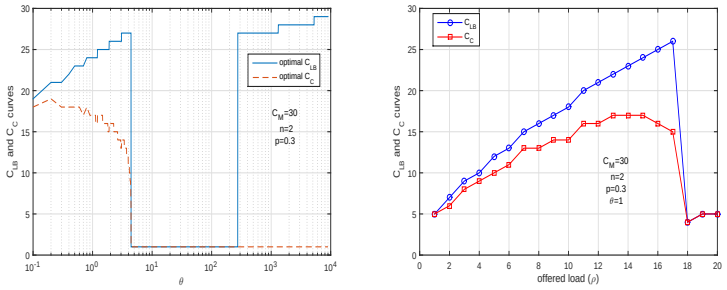
(a) Average aggregate load (b) Transition Rate

Fig. 9: AC and TR as function of C_{LB} and C_C



(a) Fixed C_{LB}, varied C_C (b) Fixed C_C, varied C_{LB}

Fig. 8: AC vs ρ



(a) Optimal C_{LB} and C_C as function of θ (b) Optimal C_{LB} and C_C as function of ρ

Fig. 10: Optimal C_{LB} and C_C as function of θ and ρ

TR) with respect to both C_C and C_{LB}. The above curves clearly shows that the same peer C_{LB} C_C minimizes (resp. maximizes) AC (resp. TR). It's obvious that minimizing the cost function f_c (eq. 4) and thus minimizing both AC and TR cannot be achieved unless a tradeoff is made. Fig.10(a) (resp. Fig.10(b)) investigates optimal C_{LB} and C_C values with respect to θ (resp. ρ). Notice that from Fig.10 we can deduce the optimal hysteresis thresholds to trigger *scaling out* and *scaling in* phases.

VII. Conclusion

In this paper, the main objective is to enable through hysteresis-based scheme efficient LB scheme for LB in SDN with multi-controllers distributed architectures with logical global view. To this end, we investigate LB toward two hysteresis-based LB models. The first one relatively simple and concerns LB between one client controller and one server controller whereas the second, more versatile, describes LB interactions between one client and multiple (three) servers according to a given LB policy. The proposed LB models fully leverage hysteretic capabilities to bring agile scalability while ensuring low operational costs. Numerical evaluations of the two proposed models are conducted through two kinds of investigation. The first one is based on transient analysis and shows how controller resources, of both the client and servers, are dynamically allocated during activation and deactivation of

the LB process with respect to variable and unpredictable spike. The second is based on steady state analysis and highlights performance metrics with respect to workload such as the average aggregated capacity (AC) (measuring the global capacity reserved during LB, the transition rate (quantifying the frequency of scaling out/in) and the blocking probability (BP). Notice also that all numerical investigations are based on comparative evaluation between controller models with and without LB. In future work we will be interested to jointly use auto-scaling with LB mechanisms (based on similar hysteretic concept used in this paper), enabling to combine vertical scaling with horizontal scaling. We will also generalize the proposed model in this paper by investigating the problem of optimal LB policies between one client and multiple servers in the SDN control plane.

REFERENCES

[1] F. Benamrane, M. Ben Mamoun, and R. Benaini. An east-west interface for distributed sdn control plane: Implementation and evaluation. *Computers and Electrical Engineering*, 57:162–175, 2017.

[2] K. Benzekki, A. El Fergougui, and A. E. Elalaoui. Software defined networking (sdn): a survey. *Security and Communication Networks*, 2016.

[3] O. Blial, M. Ben Mamoun, and R. Nguyen Benaini. An overview on sdn architectures with multiple controllers. *Journal of Computer Networks and Communications*, (4), 2016.

[4] A. A. Dixit, S. Mukherjee, T.V. Lakshman, and R. Kompella. Elasticon: an elastic distributed sdn controller. In *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*, pages 17–28, 2014.

[5] Ligang Dong, Jing Zhou, Tijie Xu, Dandan Yang, Ying Li, and Weiming Wang. Loose management for multi-controller in sdn. volume 177, pages 3–13, 06 2017.

[6] X. Huang, S. Bian, Z. Shao, and H. Xu. Dynamic switch-controller association and control devolution for sdn systems. In *proceedings of IEEE ICC 2017*, 2017.

[7] Ioannis Kamitsos, Lachlan Andrew, Hongseok Kim, and Mung Chiang. Optimal sleep patterns for serving delay-tolerant jobs. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 31–40. ACM, 2010.

[8] Ioannis Kamitsos, Lachlan Andrew, Hongseok Kim, Sangtae Ha, and Mung Chiang. Better energy-delay tradeoff via server resource pooling. In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 611–616. IEEE, 2012.

[9] Ioannis Kamitsos, Sangtae Ha, Lachlan LH Andrew, Jasika Bawa, Dana Butnariu, Hongseok Kim, and Mung Chiang. Optimal sleeping: models and experiments for energy-delay tradeoff. *International Journal of Systems Science: Operations & Logistics*, 4(4):356–371, 2017.

[10] Ioannis Kamitsos, Paschalis Tsiaflakis, Sangtae Ha, and Mung Chiang. Stable sleeping in dsl broadband access: Feasibility and tradeoffs. In *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pages 1–6. IEEE, 2011.

[11] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: a comprehensive survey. *Proceedings of the IEEE*, 103:14–76, 2015.

[12] C. Liang, R. Kawashima, and H. Matsuo. Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers. In *Proceedings of the 2014 Second International Symposium on Computing and Networking*, pages 171–177, 2014.

[13] H. Sufiev and Y. Haddad. Dcf: Dynamic cluster flow architecture for sdn control plane. in consumer electronics (icce). In *2017 IEEE International Conference on Consumer Electronics*, pages 172–173, 2017.

[14] F. De Turck, P. Chemouil, R. Boutaba, M. Yuand, C. E. Rothenberg, and K. Shiomoto. Introduction: Special issue on management of softwarized networks. *IEEE Transactions on Network and Service Management*, 13:362–365, 2016.

[15] X. Wang, X. Chen, C. Yuen, W. Wu, M. Zhang, and C. Zhan. Delay-cost tradeoff for virtual machine migration in cloud data centers. *Journal of Network and Computer Applications*, 78:62–72, Jan 2017.

[16] X. Wang, C. Yuen, X. Chen, N. Ul Hassan, and Y. Ouyang. Cost-aware demand scheduling for delay tolerant applications. *Journal of Network and Computer Applications*, 53:173–182, July 2017.

[17] F. X. Wibowo, M. A. Gregory, K. Ahmed, and K. M. Gomez. Multi-domain software defined networking: Research status and challenges. *Journal of Network and Computer Applications*, 87:32–45, 2017.

[18] W. Xia, Y.Wen, C.Heng Foh, D.Niyato, and H. Xie. A survey on software-defined networking. *IEEE Communications Surveys and Tutorials*, 17:27–51, 2015.

[19] Y. Zhou, M. Zhu, and L. Xiao. A load balancing strategy of sdn controller based on distributed decision. In *Proceedings of the 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 851–856, 2014.

[20] Y. Zou, Y. Tian, S. Guo, and Y. Wu. Active synchronization of multi-domain controllers in software defined networks. *Concurrency and Computation: Practice and Experience*, 2016.