A Compositional Modelling Approach for Live Migration in Software Defined Networks

Elisa Maini and Nicola Mazzocca
Department of Electrical Engineering and Information Technology
University of Naples Federico II, Via Claudio 21, 80125 Naples, Italy
Email:{elisa.maini, nicola.mazzocca}@unina.it

Abstract—**Recent advances offered by Software Defined Networking and virtualization techniques are creating the favourable conditions to design and develop Future Networks where network functions and services can be virtualized, dynamically instantiated and moved across networks. The ability to dynamically allocate virtual nodes across distributed physical hosts and even more the live migration of the Virtual Machines which perform such virtual network functions are driving current network infrastructures towards "programmable" networks. To be adopted as a deployable feature on a Carrier's Network, live migration performance need to be analysed and tested. Therefore a compositional modelling approach could provide early measures by evaluating the impact of these new technologies on the performance of Future Network systems. The main contribution of the paper is the definition of a general modelling framework to integrate simple models representing the main components and features of a Future Network architecture. Starting from this model composed by atomic sub-models, we conduct the performance analysis of the live migration of a single Virtual Machine between two hosts located in different networks. The Möbius tool has been used for developing the model as well as for studying its behaviour and performance. Finally, some simulative results are provided in order to show the feasibility of this approach.**

Index Terms—**Software Defined Networking, Network Function Virtualization, OpenFlow, Virtual Machines Live Migration, Stochastic Activity Network.**

I. Introduction

The progress of the IT technology is impacting on the evolution of networks, which are becoming less hierarchical and with a limited number of simplified nodes inter-connecting multiple local areas. In turn, this local areas are becoming more and more populated by small aggregation-nodes and also by a huge number of mobile devices, smart objects and "things".

The cost reductions of devices due to advanced technology in terms of computing, storage and communication-networking, is creating the ideal conditions to increase the users capability to "drive the network intelligence" towards the edges of the networks where this innovation will take place [1]. However, the "ossification" of Internet makes some difficulties for Service Providers and Network Operators to develop and deploy new network functionality, services and management policies which are essential to cope with the increasing dynamic of the ICT markets. Actually, today most of the Network Functions (typically from the layer 4 to the layer 7) are provided by specific nodes called middle-boxes. These nodes offer important services such as the improving security (e.g. firewalls and intrusion detection systems) and performance (e.g. proxies) as well as the reduction of bandwidth costs (e.g. WAN optimizers). However, they also represent most of the CApital EXpenditures (CAPEX) and Operating EXpenditures (OPEX) due to the effort management that they require [2].

In this evolution towards the Future Network systems, emerging paradigms such as *Software Defined Networking* (SDN) [3] and *Network Function Virtualisation* (NFV) [4] offer the possibility to develop such network functions (and all middle-boxes) in software, by separating and abstracting the logic from the underlying resources. Moreover, *OpenFlow* [5] protocol operates directly on network flows, simplifying the creation and installation of new "rules" into network resources such as switches and routers. Therefore, the network is more flexible, adapting to market dynamics. "Programmable" networks [6] offer many advantages for the management of the middle-boxes such as enabling dynamically adaptive polices [7] and an elastic execution of virtual middle-boxes [8]. The above analysis brings to the main assumption that network functions and services should mainly be executed as *virtual* network functions in an ensemble of Virtual Machines (VMs) dynamically placed into distributed platforms at edges of the network [9], [10]. This implies, also, the possibility to move such VMs according to requirements and constraints of the physical machines where they are hosted.

This paper argues one of the main challenge behind this vision which is the capability of dynamically instantiating, orchestrating and relocating VMs across networks. In particular, it provides a formal method based on Stochastic Activity Network (SAN) which aims to analyse the live migration performance of a single VM between two different physical hosts in the network. In particular, we introduce a generic model of a Future Network infrastructure based on SDN/OpenFlow and NFV solutions. This generic model is implemented by defining SAN sub-models and interfaces among them to share information.

This work is mainly intended to proof of the advantages that could gain Network Operators in adopting the proposed modelling approach. In particular, our model intends to support the design phase of the network infrastructure, providing a valid tool to define its features and configuration parameters.

Finally, such model could help Network and Service Operators for the evaluating the impact of the new technologies on the live migration, widely used as conventional method for changing the placement of the VMs in the network.

The paper is organized as follows: Section II provides some related works and a brief background knowledge that will help the reader to understand the paper easily; Section III implements the modelling approach by mean of SAN models; Section IV presents some results while Section V ends the paper with some remarks and future developments.

II. Related works and background knowledge

VM live migration within (and across) data centers has traditionally been a complex task due to the requirements of storage accessibility and moving the network-level policies associated with each VM to its new location. Traditional approaches studied the live migration behaviour by using some virtualization platforms (e.g. Xen) to evaluate the migration performance in terms of migration time and energy consumption given a resource available [11]. Others adopted some prediction techniques [12] and probabilistic frameworks [13], [14] to build a performance model representing the performance features of the live migration.

Conventional VM live migration involves the transfer of the CPU and memory state as well as the storage data. However, storage data migration in WAN is still an important challenge on which researchers are spending more efforts [15]. In the other side, the live migration inter-data centers presents the problem to allow IP change; solutions like tunnelling [16] and later-2 expansion [17] work around the problem of connection loss due to a change of the IP address. Using OpenFlow VMs are allowed keep their original IP addresses, maintaining all existing connections [18], [19]. Moreover, this protocol (i) allows the mobility between the layer-2 and layer-3, (ii) allows a programmable SDN, and (iii) network resources can be remotely configured, controlled and monitored. Therefore, SDN-based solutions could improve management operations and performance as well as enable the VM migration in intra and inter data centers [20], [21] Recent works [22], [23] present a SDN architecture to enable the VM live migration and evaluate its performance.

Generally speaking, the VM live migration involves several key factors such as the VM memory size, page dirty rate, the network transmission rate and the migration algorithm. The page dirty rate is the rate which the VM memory page change with. These factors and even more the migration algorithm used, can introduce relevant variations of the migration performance. There are different techniques for the live migration [24], [25], [26] that trade-off two important performance parameters: the migration time and the downtime. Migration time refers to the time required to move the VM between physical hosts while downtime is the portion of the time when the VM is not running. *stop&copy* [25] designs halt the original VM and copy its entire memory to the destination. This techniques minimise the migration time but suffers from high downtime as the VM is suspended during the entire transfer. On the other side, *on-demand* [26] migration operates by stopping the VM copy only essential kernel data to the destination. The remainder of the VM address space is transferred when it is accessed at the destination. While this technique has a very short downtime, it suffers from high migration time.

Among the several techniques used, the iterative pre-copy algorithm [27] minimizes the total migration time and downtime than other algorithms used, such as on demand migration and the stop&copy. Using an iterative approach, the hosted VM in the source physical machine can be kept in an active state during the migration towards the new destination. Since the VM is running, some memory pages are changed during the migration and must be re-sent. The term "iterative" means that pre-coping occurs in several rounds and the data to be transmitted during a round are the dirty pages generated in the previous round. The pre-copy phase terminates if: 1) the memory dirtying rate exceeds the memory transmitted rate; 2) the remaining dirty memory becomes smaller than a predefined threshold value; 3) the number of iterations exceeds a given value. After several rounds of synchronization, a very short stop-and-copy phase is performed to transmit the remaining dirty pages. Then, the VM is halted for the final state transfer and re-starts it in the new location.

III. A SAN-based implementation

This Section describes our modelling approach in order to support the design of a Future Network by defining a "generic" modelling framework for peformance analysis of the live migration. Our approach exploits the usage of the formal models, allowing to create the overall model of the system under analysis in a well-structured way. We adopte a component-based view of the network applying *divide-et-impera* techniques: the overall system model is decomposed recursively into sub-models until reaching an atomic sub-model. The effectiveness of this approach has been showed in [28] where a modelling framework has been defined and then implemented exploiting the flexibility of the SAN formalism.

The analysis of the VM live migration over a SDN/OpenFlow network requires to consider the entire application field by modelling four different components: the *Controller*, the *OpenFlow Switch*, the *VM* which are the traditional components in SDN and the *VM Live Migration Orchestrator*. This latter is a soft-component in charge of the resources management and orchestration as well as for the starting and stopping of the migration. This is argued by many efforts that the ETSI [29] is spending in this direction; however, in this work, the capability of the orchestrator is limited to the triggering the migration for a given VM. Other features will investigated in next developments.

The implementation of the system has been realized using the SAN formalism [30]. The choice of this approach is due to the great flexibility and power in the modelling given by using such formal method. One of main advantages of the SAN formalism is the possibility to map each component of the designed architecture into "atomic" models. Component

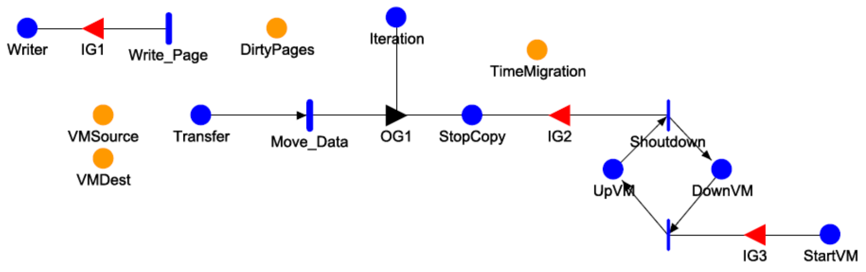


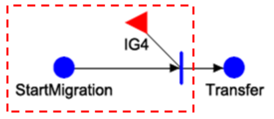
Fig. 1: Live migration algorithm model

interfaces have been realized by sharing extended places, which contain complex data structures used by atomic models to read and write shared information. In the follow subsections, each atomic model is described in detail.

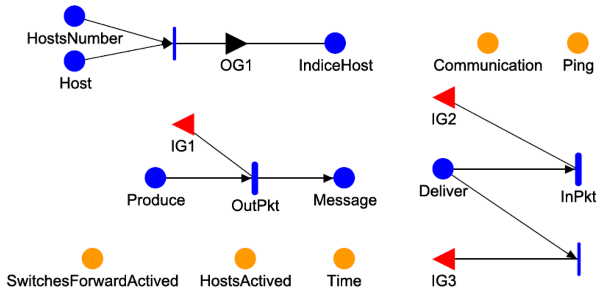
A. *Virtual Machine model*

This model consists of three main parts: the live migration algorithm (Fig. 1), a trigger signal (Fig. 2(a)) and one part that represents the typical production and delivery of packets (Fig. 2(b)). The pre-copy algorithm executes in several rounds and each iteration is modelled using the place `Iteration`.

When a new iteration starts, a new token is added to this place. The extended place `VMSource` is an input parameter set as the memory size of the hosted VM into the source physical machine. `VMDest` represents the fraction of memory of such VM that is transferred to the destination machine during the migration. The memory size of the hosted VM into the source physical machine, the memory dirtying rate during the migration and the elapsed time at each round are modelled as `VMSource`, `DirtyPages` and `TimeMigration`, respectively.



(a) Trigger signal



(b) VM model

Fig. 2: Trigger signal and Virtual Machine model

In the beginning, `VMDest` is zero and one token in `Transfer` is used to start the live migration. All the memory of the hosted VM in the source physical machine is transferred to the destination. The timed transition `Move_Data` models the channel and the such transition expires when all memory of the hosted VM in the source physical machine is transferred to the destination. When the migration starts, some memory pages (called dirty pages) change during the process and they must be re-sent to the destination. This is modelled using the place `Writer`, the gate `IG1`, the timed transition `Write_Page` and the extended place `DirtyPages`. In particular, when the migration starts, one token is in `Writer` and some memory pages change at a given rate. The timed transition is modelled as an exponential distribution with parameter equal to the page dirty rate (that is an input parameter). Pages changed during this phase are modelled by an extended place `DirtyPages`.

The place `Iteration` increases one token until when one of the stop-conditions is verified. Such event is modelled using the gate `OG1`. In aprticular, when a stop condition occurs, one token is in `StopCopy` and the hosted VM in the source physical machine is halted for the final transfer round. During last iteration one token is in the place `DownVM`. When last fraction of memory has been transfered to the destination, the VM hosted on the physical machine of destination is re-activated. In this case, `DownVM` is zero and one token is in `StartVM`. Morevoer, the `VirtualHost` model represents the ability to handle a trigger signal (Fig. 2(a)) to start the migration. Fig. 2(b) shows the `host` model representing the typical production and delivery packet operations. The timed transition `OutPkt` and `InPkt` are modelled as a deterministic distribution.

B. *Live migration orchestrator model*

This atomic model represents the entity in charge for enabling the live migration (Fig. 3). To run it, a token is in the place `Trigger`. The hosted VM in the source physical machine is selected for the migration and, after one token is in the place in `MigrateHost`. When the migration is completed, one token is added at `Completed`. The timed transition `Trigger_signal` is modelled as a deterministic distribution with a transition firing every $10s$.

C. *Controller model*

The sub-model for the controller is shown in Fig. 4. Basically, it represents the interaction

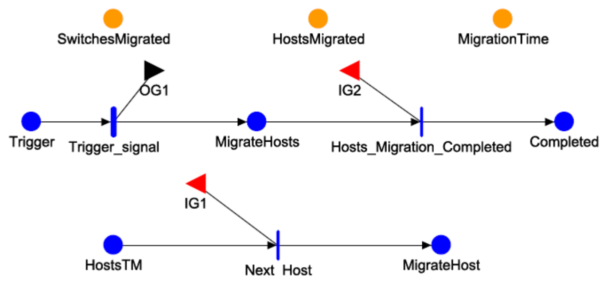


Fig. 3: Live migration orchestrator model

occurring between the Controller and the OpenFlow Switch. In particular, `OpenFlow_Switch` and `OpenFlow_Controller` share a common place `SwitchToControllerRequest` and the timed transition `Matching` has two cases (i.e. `Flow_mod` and `Packet_Out`).

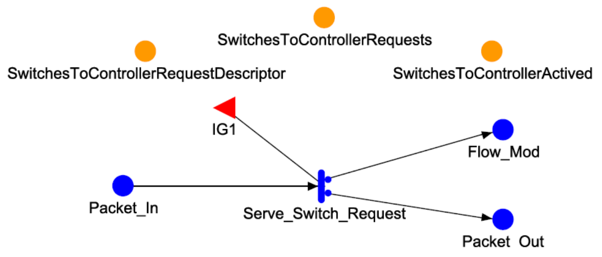

SwitchesToControllerRequests
SwitchesToControllerRequestDescriptor
SwitchesToControllerActived
IG1
Flow_Mod
Packet_In
Serve_Switch_Request
Packet Out

Fig. 4: Controller model

D. *Open Switch model*

The sub-model that represents a OpenFlow Switch is shown in Fig. 5. It is argued that the behaviour of such switch is based on set of rules installed into itself. Basically, if a rule matches an incoming packet, the forwarding decision is instantaneously executed on the switch. Otherwise, if there is no matching rule, the switch asks the controller for an action to execute. We model these events by using two cases associated to the timed transition `Serve_switch_request`. For such transition a normal distribution with a mean of $4\mu s$ and standard deviation of 101.43 has been used according to the reference [31]. Moreover, `Write_Flow_Entry`, `Output_Action` and `Matching` are modelled as a uniform distribution with lower bound $4\mu s$ and upper bound $16.5\mu s$. Finally, `FlowExpires` is modelled as a normal distribution with mean 1 and standard deviation $0.2ms$.

IV. Experimental results

In this Section we report some results obtained by using Möbius tool [32], a software tool for modelling the behaviour of complex systems. The modelling language is based either graphical or textual representations supporting several

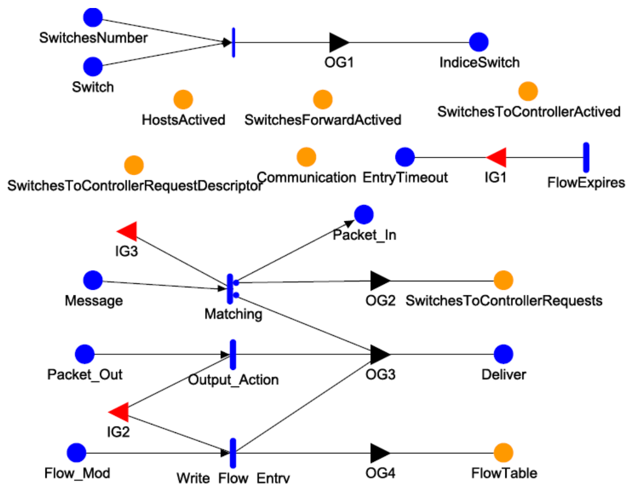


Fig. 5: Open Switch model

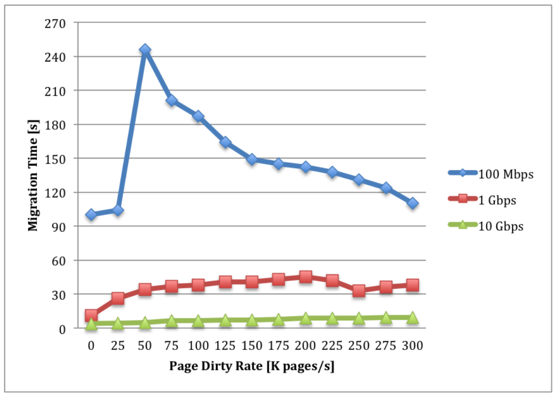
modelling formalisms including SAN, Markov chains and extensions as well as stochastic process algebras. Functionality of the system can be defined as model input parameters, and then the behaviour of the system can be automatically studied across wide ranges of input parameter values. In particular, we use Möbius for the analysing and evaluating the VM live migration in a SDN/OpenFlow infrastructure.

To perform useful results, we evaluate the migration time and downtime by changing the page dirty rate. Fig. 6 shows the effect of varying the page dirty rate on migration time and downtime for three different link speed values: 100 Mbps, 1 Gbps and 10 Gbps. The memory size VM migrated experimentation is 1024 MB. This value is used for all experiments. We observe an interesting relationship between page dirty rate and migration performance: specifically, such relationship is not linear this occur because of the stop conditions defined in the migration algorithm. In line with some results presented in [27], if the page dirty rate is below the link capacity, all modified pages are transferred in a timely fashion resulting in a low migration time and downtime. In the other side, if the page dirty rate starts approaching towards the link capacity, the migration performance degrades significantly.

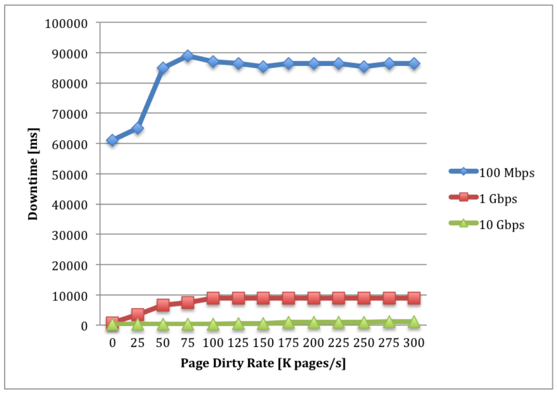
Moreover, it is relevant to observe the impact of the link speed on the live migration performance. Results have been obtained considering a page dirty rate set as 300.000 pages/second. Fig. 7 shows clearly that the migration performance are influenced by link speeds; moreover such figure highlights that relationship between the link capacity and the migration time is inversely proportional. Same consideration about the relationship between the link capacity and the downtime.

V. Conclusion and future steps

In this paper we have defined a general modelling framework to support the design of a network architecture based on SDN/NFV principles. The framework is general enough to model a design a SDN architecture by defining independent sub-models. In particular, such framework has been used to

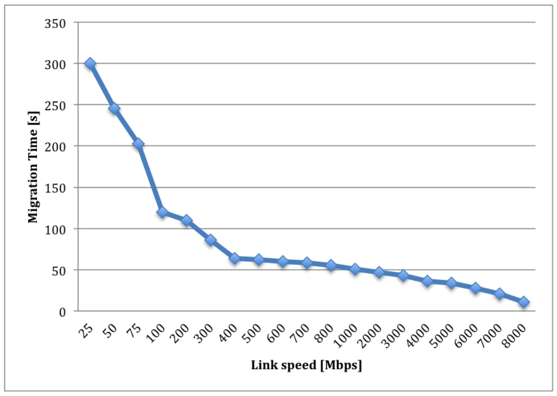


(a) Migration Time

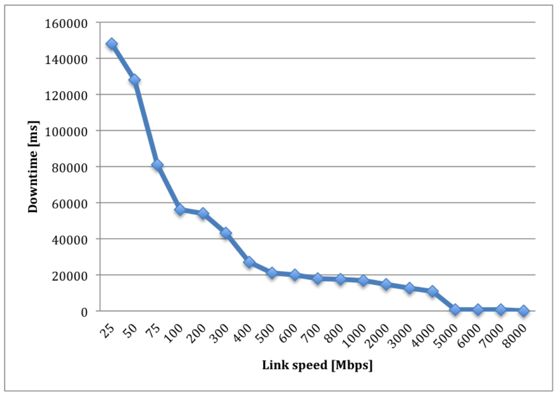


(b) Downtime

Fig. 6: Migration time and downtime versus page dirty rate



(a) Migration Time



(b) Downtime

Fig. 7: Migration time and downtime versus link speed

evaluate the performance of the virtual machine live migration in terms of migration time and downtime. The main goal of our analysis is to highlight how some parameters such as link speed of the network and the page dirty rate of the migration algorithm can influence significantly the migration time and downtime. This is confirmed from some results obtained by Möbius tool that show how the link speed and the page dirty rate are the main factors impacting on the live migration performance. In particular, link speed and the page dirty rate cause not linear effects on such performance indicators due to the use of some stop conditions in the migration algorithm which force the migration to the final round. Future research efforts will be oriented on the development of other aspects related to architecture components. Moreoveor, we will plan to investigate other migration algorithms in order to migrate an ensemble of VMs across networks providing other performance indicators including throughput, latency, and jitter.

REFERENCES

[1] A. Manzalini, R. Minerva, E. Dekel, Y. Tock, E. Kaemfer, W. Tavernier, K. Casier, S. Verbrugge, D. Colle, F. Collegati, A. Campi, W. Cerroni, R. Vilalta, R. Munoz, R. Casellas, R. Martinez, N. Mazzocca, E. Maini, *Manifesto of Edge ICT Fabric*, 17th International Conference on Intelligence in Next Generation Networks (ICIN), Italy, 2013

[2] J. Sherry and S. Ratnasamy, *A Survey of Enterprise Middlebox Deployments*, http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-24.html, 2012.

[3] Open Networking Foundation: https://www.opennetworking.org/

[4] Network Function Virtualization: http://www.etsi.org/technologies-clusters/technologies/nfv

[5] Open Networking Foundation: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow

[6] A. Galis, S. Clayman, L. Mamatas, J. Rubio-Loyola, A. Manzalini, S. Kuklinski, J. Serrat, T. Zahariadis, *Softwarization of Future Networks and Services Programmable Enabled Networks as Next Generation Software Defined Networks*, IEEE Software Defined Networks for Future Networks and Services (SDN4FNS), Trento, Italy, 2013.

[7] S. Rajagopalan, D. Williams and H. Jamjoom, *Pico Replication: A High Availability Framework for Middleboxes*, 4th annual Symposium on Cloud Computing (SoCC), CA, 2013.

[8] S. Rajagopalan, D. Williams, H. Jamjoom and A. Warfield, *Split/Merge: System Support for Elastic Execution in Virtual Middleboxes*, 10th USENIX Symposium on Networked System Design and Implementation (NSDI), Chicago, USA, 2013.

[9] S. Clayman, E. Maini, A. Galis, A. Manzalini, N. Mazzocca, *The Dynamical Placement of Virtual Network Functions*, 1st IEEE / IFIP International Workshop on SDN Management and Orchestration (SDNMO), Krakow, Poland, 2014.

[10] E. Maini and A. Manzalini *Management and Orchestration of Virtualized Network Functions*, 8th International Conference on Autonomous Infrastructure, Management and Security (AIMS), Brno, Czech Republic, 2014.

[11] H. Liu, C. Xu, H. Jin, J. Gong and V. Liao, *Performance and Energy Modeling for Live Migration of Virtual Machines*, 20th International Symposium on High Performance Distributed Computing (HPDC), CA, 2011.

[12] B. Hu, Z. Lei, Y. Lei, D. Xu and J. Li, *A Time-series Based Precopy Approach for Live Migration of Virtual Machines*, 17th International Conference on Parallel and Distributed Systems (ICPADS), Taiwan, 2011.

[13] S. Kikuchi and Y. Matsumoto, *Performance Modeling of Concurrent Live Migration Operations in Cloud Computing System using PRISM Probabilistic Model Checker*, 4th International Conference on Cloud Computing, Washington DC, USA, 2011.

[14] F. Farahnakian, P. Liljeberg and J. Plosila, *LiRCUP: Linear Regression based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers*, 39th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA), Santander, Spain, 2013.

[15] H. Lai, Y. Wu and Y. Cheng, *Exploiting Neighbourhood Similarity for Virtual Machine Migration over Wide-Area Network*, 7th IEEE International Conference on Software Security and Reliability (SERE), Gaithersburg, MD, USA, 2013.

[16] R. Bradford, K. Kotsovinos, A. Feldmann and H. Schioberg, *Live Wide-Area Migration of Virtual Machines Including Local Persistent State*, 3rd International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments, SIGPLAN VEE, San Diego, CA, USA, 2007.

[17] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal *et al.*, *VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*, Internet Draft, 2013.

[18] R. Erickson, G. Gibb, B. Heller, D. Underhill *et al.*, *A Demonstration of Virtual Machine Mobility in a OpenFlow Network*, SIGCOMM (*Demo*), Seattle, WA, USA, 2008.

[19] G. Stabler, A. Rosen, K. Wang and S. Goasguen, *Elastic IP and Security Groups Implementation Using OpenFlow*, 6th International Workshop on Virtualization Technologies in Distributed Computing Date (HPDC), Delft, Netherlands, 2012.

[20] C. Baker, A. Anjum, R. Hill, N. Bessis and S. Kiani, *Improving Cloud Data centre Scalability, Agility and Performance using OpenFlow*, 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS), Bucharest, Romania, 2012.

[21] S. Jain, A. Kumar, S. Mandal, J. Ong *et al.*, *B4: Experience with a Globally-Deployed Software Defined WAN*, SIGCOMM, Hong Kong, China, 2013.

[22] S. Ghorbani, E. Keller, M. Caesar, J. Rexford, C. Schlesinger and D. Walker, *Transparent, Live Migration of a Software Defined Network*, ACM Symposium on Cloud Computing (SoCC), Seattle, WA, USA, 2014.

[23] E. Keller, D. Arora, D. P. Botero and J. Rexford, *Live Migration of an Entire Network (and its Hosts)*, 11th ACM Workshop on Hot Topics in Networks (HotNets-XI), Redmond, WA, USA, 2011.

[24] C. Clark, K. Fraser, S. J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, *Live migration of virtual machines*, USENIX Symposium on Networked System Design and Implementation (NSDI), Boston, MA, USA, 2005.

[25] M. Kozuch and S. Satyanarayanan, *Internet suspend/resume*, IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), New York, USA 2002.

[26] E. Zayas, *Attacking the process migration bottleneck*, ACM Special Interest Group on Operating Systems, SIGOPS, Vol. 21, Pages 13-24, 1987.

[27] S. Akoush, R. Sohan, A. Rice, A. W. Moore and A. Hopper, *Predicting Performance of Virtual Machine Migration*, 18th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Miami Beach, Florida, USA, 2010.

[28] S. Marrone, N. Mazzocca, R. Nardone, R. Presta, S.P. Romano, and V. Vittorini, *A SAN based Modeling Approach to Performance Evaluation of an IMS-Compliant conferencing framework*, Transaction on Petri Nets and Other Models of Concurrency VI, Springer, Pages 308-333, 2012.

[29] ETSI: http://www.etsi.org

[30] W. H. Sanders and J. F. Meyer, *Stochastic Activity Networks: Formal definition and Concepts*, Lectures on formal methods and performance analysis, Pages 315-343, 2002.

[31] M. Jarschel, S.Oechsner, D. Schlosser, R. Pries, S. Goll and P. Tran-Gia *Modeling and Performance Evaluation of an OpenFlow Architecture*, 23rd International Teletraffic Congress (ITC), San Francisco, USA, 2011.

[32] Mobius modelling tool: https://www.mobius.illinois.edu