A Modular Simulation Tool of an Orchestrator for Allocating Virtual Resources in SDN

Aurelio Javier García, Cristina Cervelló-Pastor, and Yury Jiménez

***Abstract*—Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) are technologies for enabling innovative network architectures. Nevertheless, a fundamental problem in instantiation of Virtual Networks (VNs), performed by NFV, is an optimal allocation of resources offered by one or more SDN domain networks. The process of instantiation of VNs is performed in several phases, including splitting and mapping algorithms. For each one of these phases, researchers have developed algorithms, being possible to obtain different results combining them. This paper introduces a modular and flexible graphical discrete event simulation tool for solving the complete virtual resource allocation in SDN domain networks problem. A Java-based tool has been developed to integrate existing and future algorithms related to each phase of the process. The simulator is a test-bed in which researchers can select the appropriate algorithm in each phase and display the results in a graphical form, while obtaining a performance evaluation of the selected and proposed algorithms.**

***Index Terms*—Network functions virtualization, network modeling, optimization, software-defined networking.**

I. Introduction

Traditional network architectures are ill-suited to meet the requirements of today's enterprises, carriers, and end users. Thanks to a broad industry effort spearheaded by the Open Networking Foundation (ONF), SDN is transforming the network architecture. In the SDN architecture, the control and data planes are decoupled. Network intelligence is centralized in software-based SDN controllers, which maintain a global view of the network. As a result, enterprises and carriers gain unprecedented programmability, automation, and network control, enabling them to build highly scalable, flexible networks that readily adapt to changing business needs. The ONF is a non-profit industry consortium that is leading the advancement of SDN and standardizing critical elements of the SDN architecture such as the OpenFlow™ protocol, which structures communication between the control and data layers of supported network devices. OpenFlow is the first standard interface designed specifically for SDN, providing high-performance, granular traffic control across multiple vendors' network devices. OpenFlow-based SDN is currently being rolled out in a variety of networking devices and software, delivering substantial benefits to both enterprises and carriers.

NFV aims to transform the way that network operators architect networks by evolving standard Information Technology (IT) virtualization technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in Data centers, Network Nodes and in the end user premises. It involves the implementation of network functions in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need for installation of new equipment. With NFV, SDN networking adds a new layer named orchestration. The orchestrator has a view of all the different platforms in the network and can monitor the resource utilization (CPU, RAM and storage).

NFV is highly complementary to SDN, but not dependent on it (or vice-versa). NFV can be implemented without a SDN being required, although the two concepts and solutions can be combined and potentially greater value accrued. NFV goals can be achieved using non-SDN mechanisms, relying on the techniques currently in use in many data centers. But approaches relying on the separation of the control and data forwarding planes as proposed by SDN can enhance performance, simplify compatibility with existing deployments, and facilitate operation and maintenance procedures. NFV is able to support SDN by providing the infrastructure upon which the SDN software can be run. Furthermore, NFV aligns closely with the SDN objectives to use commodity servers and switches.

In this paper we introduce a global simulator, which is a graphical tool that facilitates development and simultaneous checking of different types of algorithms to VNs instantiation. The simulator models the functions of the orchestrator using the global view of the network provided by the controllers. The contribution of this paper is threefold. First, it presents a simulator architecture oriented to modeling the VNs instantiation process [1]-[6]. Second, the paper describes networks modeling. Third, it explains the model used for implementing existing algorithms, i.e., network splitting, and node and link mapping algorithms [7]–[11]. This simulator is available in [12].

The rest of this paper is organized as follows. Section II presents the programming technologies used for the simulator development, the simulator options, how networks are created and incorporated in the simulator, how to run simulations and the networks results. Section III explains the evaluation of simulation. Section IV describes the programming structure used for modeling networks, nodes and links. Section V presents the network algorithms: network splitting and node and link mapping. Section VI provides other works in this

Manuscript received December 10, 2013; revised February 12, 2014. This work has been supported by the Government of Spain through the project TEC2010-20527-C02-01 and through a predoctoral FPI scholarship.

The authors are with Department of Telematics Engineering. Universitat Politècnica de Catalunya. BarcelonaTech. Esteve Terradas, 7, 08860, Castelldefels, Barcelona, Spain (e-mail: aurelioj.garcia@entel.upc.edu, e-mail: cristina@entel.upc.edu, e-mail: yury.jimenez@entel.upc.edu).

DOI: 10.7763/IJMO.2014.V4.352

area. Finally we conclude in Section VII identifying future simulator options and planning new algorithms formulation for both splitting and mapping problems.

II. Simulator Architecture, Design and Overview

A. Simulator Architecture

This tool is a modular graphical discrete event simulator for allocation of virtual resources in SDN domain networks. The SDN domain networks include SDN nodes, i.e. routers, switches and servers, all managed by a controller. The switches interconnect the servers with the routers, and the routers interconnect the SDN domains as it is shown in Fig. 1.the servers provide services to end-users.

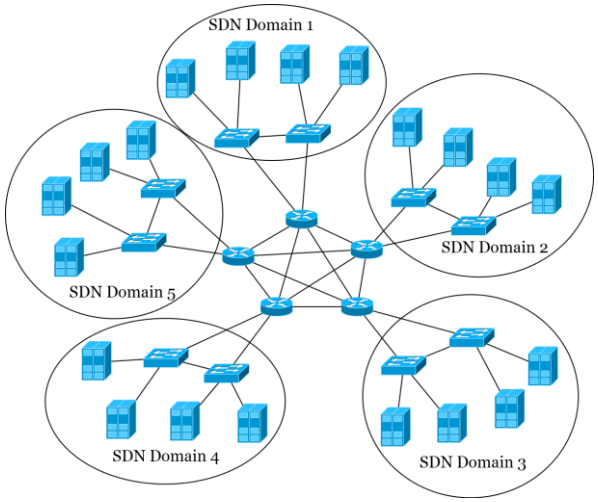

Fig. 1. Interconnection of SDN domains.

The tool models the operation of a system as a discrete sequence of events in time, i.e., the start and end times of the VN requests. At any instant of time the system receives VN requests with a specific lifetime and serves them one by one. Each VN request passes through three phases. Fig. 2 shows these phases. In the *Network Splitting* phase, virtual nodes are distributed over SDN domain networks by the criteria of tenant cost of SDN resources, but not yet assigned to concrete SDN domain nodes. In the *Nodes Mapping* phase, the virtual nodes are finally assigned to the SDN domain nodes, and in the *Links Mapping* phase the virtual nodes are interconnected through SDN domain links according to the virtual links. During the VN request lifetime, the selected SDN domain resources are reserved. At the end of the VN request lifetime these resources are released and can be reused for future VN requests. During this process, a VN request is rejected if there are not enough SDN resources.

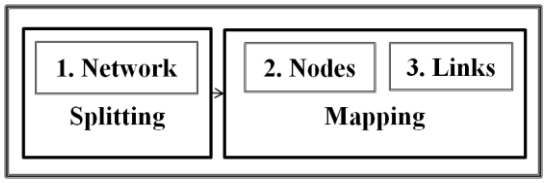

Fig. 2. Phases of simulator for each VN request.

B. Simulator Design

Simulator is based on Java programming language and it has been developed and tested using the Eclipse Integrated Development Environment (IDE). The Graphical User Interface (GUI) of our simulator has been implemented under Swing. Swing classes (part of the Java™ Foundation Classes (JFC) software) implement a set of components for building GUIs and adding rich graphics functionality and interactivity to Java applications. Graphs are developed by Java Universal Network/Graph Framework (JUNG) software library. It is a software library that provides a common and extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network. Library CPLEX has been used to solve the relaxed Mixed Integer Programming (MIP) problem. The ObjectAid UML (Unified Modeling Language) Explorer is an agile and lightweight code visualization tool for the Eclipse IDE and it has been used for representing the simulator class diagrams.

C. Simulator Overview

The main window of the simulator has three parts: a menu bar on the top, tabs with tables of SDN and virtual networks on the left, and tabs with graphics of SDN and virtual networks on the right. Menu bar has four elements: *Simulator*, *SDN Networks*, *Virtual Networks* and *Simulation*. *Simulator* has two options: *About* and *Exit*. SDN networks and Virtual Networks have three options: *Open*, *Create* and *Save*. *Simulation* has the option *Create*.

SDN Networks. SDN networks can be inserted by default having them embedded into the simulator; they can also be inserted opening an excel file with the SDN networks information (*SDN Networks/Open*); or they can be created with the wizard (*SDN Network/Create*). The SDN networks creation parameters are: the number of networks, the number of routers, switches and servers, server operating systems (*Linux*, *Windows*, *Solaris* or *Android*), CPU, RAM and storage, their corresponding range of prices, available number of virtual nodes in a node, link bandwidth and distance, and his corresponding range of price. Once the SDN networks are created, the results are showed by the simulator. The list with the created SDN networks is on the left and the graphics on the right. Clicking on one SDN network or on one tab on top, its graphic is showed on the right. Right-clicking on one node or link its color change to green and the information is displayed. SDN networks can be saved in an excel file using the option *SDN Networks/Save* on the menu bar.

Virtual Networks. VNs can be inserted into the simulator by default, they can also be generated opening an excel file with the virtual networks information (*Virtual Networks/Open*) or created with the wizard (*Virtual Network/Create*). The virtual networks creation parameters are: the number of networks, the number of window time units, the number of networks per window considering Poisson arrivals, the average units of time with an exponential distributed lifetime, the number of routers, switches and servers, server operating systems (Linux, Windows, Solaris or Android), CPU, RAM and storage, and link bandwidth. Once the virtual networks are created the results are showed by the simulator. The graphical details are similar to those explained for SDN networks.

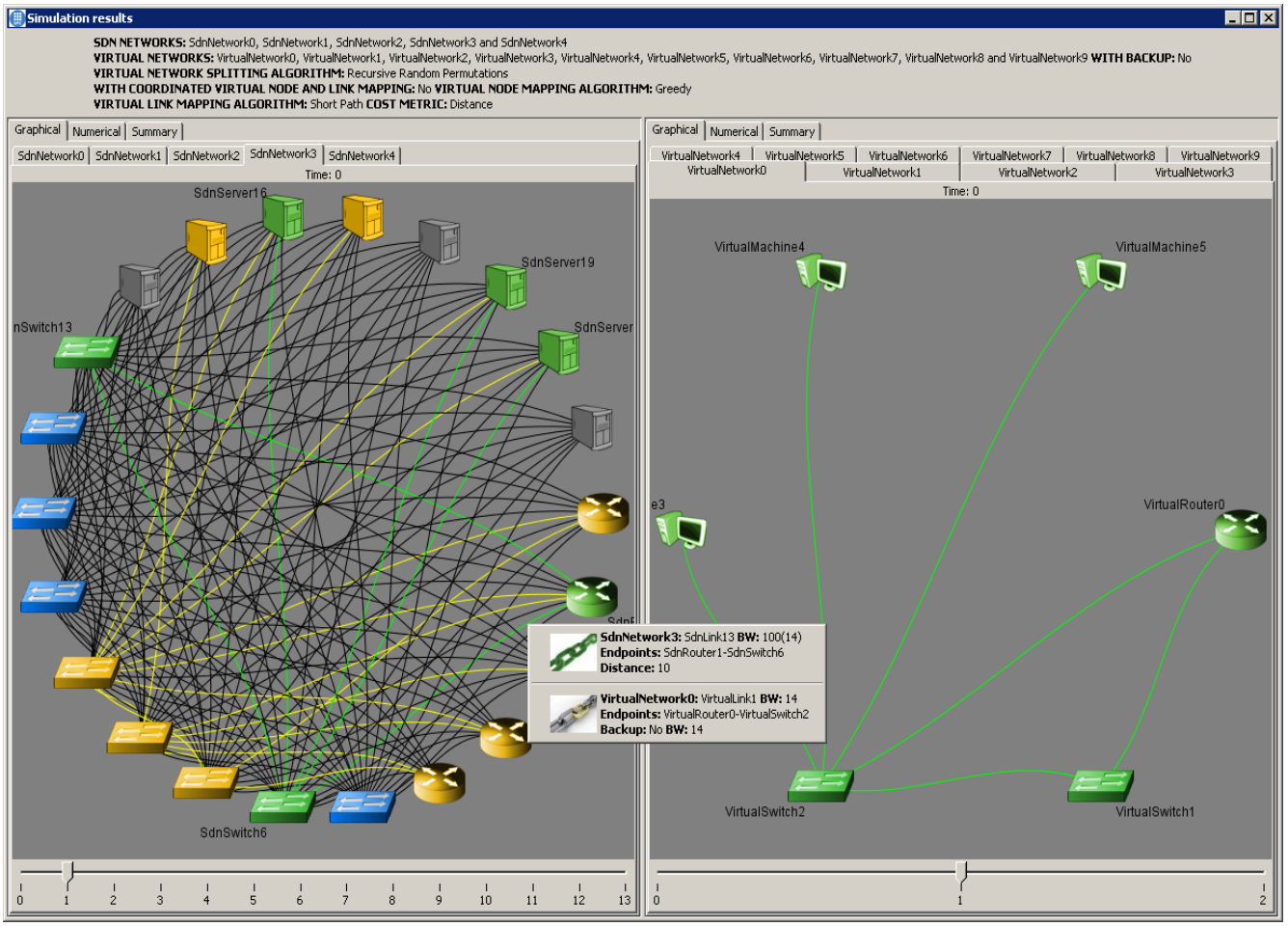

Fig. 3. Graphical results of simulation.

III. Simulation Evaluation

The simulation is created clicking on the menu option *Simulation/Create*. The existing simulation options are: the SDN networks, the virtual networks, the inclusion or not of virtual networks backup, the virtual network splitting algorithm, and if the virtual network embedding is with coordinated virtual node and link mapping or not. In case of selecting with coordinated virtual node and link mapping, the next option is the virtual network embedding algorithm and the rounding virtual nodes mapping algorithm. On the other hand, in case of selecting without coordinated virtual node and link mapping, the next option is the virtual nodes mapping algorithm. Finally, the process ends up selecting the virtual link mapping algorithm and its cost metric, i.e., distance or available bandwidth.

Once the options of the simulation are selected, it runs automatically. A window with the results of our simulation is displayed. This window is divided into three parts. On the top, there are the selected options for the simulation; on the left, the results for the SDN networks; and on the right, the results for the virtual networks. On top of both networks results there are two tabs: the graphical and the numerical networks results. Each network can be selected by clicking on the corresponding tab. Both results are presented instant by instant. The time instant is showed on top of the graphic and using the slider below it is possible to change the instant of time. It is feasible to create and compare as many simulations as you want.

SDN Networks Results. The graphical results of the SDN networks show the allocation of SDN resources to virtual resources at each instant of time. The grey color of the nodes and links means that they have no virtual resources assigned. The yellow color means that they contain virtual resources. The red color means that their occupation is above 80%. Finally, the green color means that they have been selected for showing information about them. Selecting a node and right-clicking on it, it is shown information of the SDN node and of the virtual nodes allocated, if that is the case. The color of node changes to green when the information of the SDN node is pressed. The color of the node and the color of the virtual node in the graphical results of the VNs change to green color when the information of the virtual node is pressed. The same happens with links.

Virtual Networks Results. The graphical results of the virtual networks show the allocation of SDN resources to virtual resources at each instant of time. In this case, the code color is the following. The grey color of nodes and links means that they are not assigned to SDN resources; the yellow color means that they have been assigned; the green color means that they have been selected to know information; and, the blue color represents a link between two different SDN networks. Selecting a node and right-clicking, the simulator shows information about the virtual node and the SDN node that have been assigned. When the information of a virtual node or a SDN node are pressed, the node color changes to

green color and the color of SDN node in the graphical results of the SDN networks results, changes to green also. Clicking on *All Virtual Network* the color of all SDN nodes and links, assigned to the virtual network, in the graphical results of the SDN networks, they change to green. Selecting a virtual link and right-clicking, the information of the virtual link is showed. This information is more extensive than the others.

The numerical results of the virtual networks show a table with the general allocation of SDN resources to virtual resources at each instant of time. Fig. 3 shows the graphical results of one simulation.

IV. Network Modeling

A network consists of nodes and links and can be physical (SDN) or virtual. Here is explained the model of physical and virtual nodes, links and networks.

A. SDN and Virtual Nodes Modeling

Node class models a node and contains the main characteristics of all nodes. The general attributes of a node are: network name, node name, identifier number, time stamp, type (*Router*, *Switch* or *Server*), operating system (*Linux*, *Windows*, *Android* or *Solaris*), CPU, RAM and storage. The main class is *Node*. *SdnNode* and *VirtualNode* extend *Node*. Fig. 4 shows the node classes hierarchy.

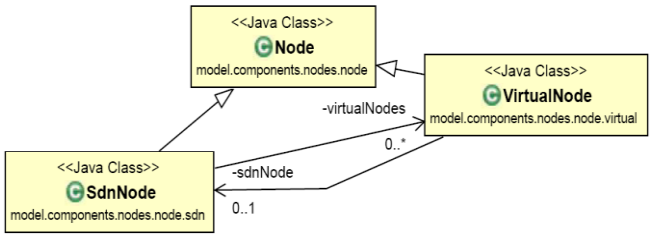

Fig. 4. Hierarchy of the classes of a node.

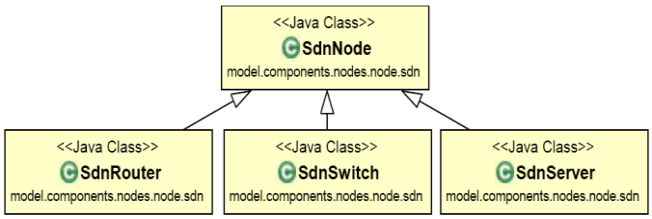

Fig. 5. Hierarchy of the classes of a SDN node.

SDN Nodes. *SdnNode* class models a physical node and contains the main characteristics of all SDN nodes together with the *Node* class. One SDN node can host more than one virtual node and they will use its CPU, RAM and storage. The general attributes of a SDN node are: the available CPU, the available RAM, the available storage, the maximum number of virtual nodes that it can host, and the available number of virtual nodes and virtual nodes that it can host. *SdnRouter* models a SDN router, *SdnSwitch* models a SDN switch and *SdnServer* models a SDN server. These classes extend *SdnNode*. The difference between these three classes, apart from the type, is that a server has storage resource while routers and switches do not. Fig. 5 shows the hierarchy of the classes of a SDN node.

Virtual Nodes. *VirtualNode* class models a virtual node and contains the main characteristics of all virtual nodes together with the *Node* class. One virtual node will be hosted in one SDN node. A general attribute of virtual nodes is the SDN node that hosts it. *VirtualRouter* models a virtual router, *VirtualSwitch* models a virtual switch and *VirtualServer* models a virtual server. These classes extend *VirtualNode*. The difference between these three classes, apart from the type, is that a server has storage resource while routers and switches do not. Fig. 6 shows the hierarchy of the classes of a virtual node.

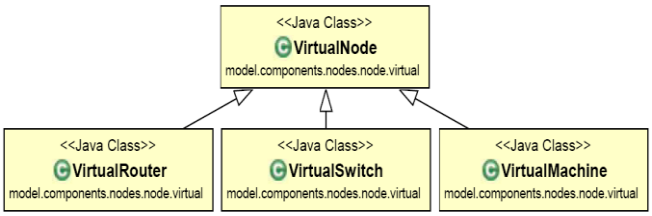

Fig. 6. Hierarchy of the classes of a virtual node.

B. SDN and Virtual Links Modeling

Link class models a link and contains the main characteristics of all links. The general attributes of a link are: a network name of the link, a name, an identifier number, a time stamp, the bandwidth and two connected nodes. The main class is *Link*. *SdnLink* and *VirtualLink* extend Link.

SDN Links. *SdnLink* class models a SDN link and contains the main characteristics which have all SDN links together with the *Link* class. One SDN link can host more than one virtual link and they will use its bandwidth. The general attributes of a SDN link are: the available bandwidth, the physical distance and the virtual links that it hosts.

Virtual Links. *VirtualLink* class models a virtual link and contains the main characteristics which have all virtual links together with the *Link* class. One virtual link will be hosted in one or more SDN links. The general attribute of virtual links is the SDN links that host the virtual link. Fig. 7 shows the hierarchy of the classes of a virtual link.

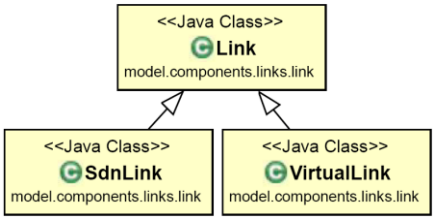

Fig. 7. Hierarchy of the classes of a link.

C. SDN and Virtual Networks Modeling

Network class models a network and contains the main characteristics that all networks have. The general attributes of *Network* are: name, identifier number, a time stamp, state (*Accepted*, *Rejected*, *Available* or *Ready*) and its graph. The main class is *Network*. *SdnNetwork* and *VirtualNetwork* extend *Network*.

SDN Networks. *SdnNetwork* class models a SDN network and contains the main characteristics which have all SDN networks together with *Network* class. One SDN network can host more than one virtual network or a part of it, and they will use its nodes and links.

Virtual Networks. *VirtualNetwork* class models a virtual network and contains the main characteristics of the virtual networks together with the *Network* class. One virtual

network will be hosted in one or, in parts, in more than one SDN networks. General attributes of virtual networks are: the start and end dates, the name of SDN networks that host the virtual network, subnetworks forming the virtual network, the mapping of virtual nodes in SDN nodes, the mapping of virtual links in SDN links and the flag of accepted or denied virtual network.

Fig. 8 shows the hierarchy of the classes of a network.

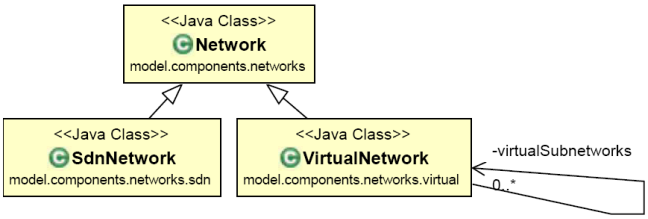

Fig. 8. Hierarchy of the classes of a network.

V. Network, Nodes and Links Algorithms

Once we have the SDN and virtual networks selected, the next step is the selection of the algorithms for the simulation. The process by which a VN request is allocated into various network infrastructures is performed in three steps. Fig. 9 shows the required steps for the selection of the algorithms.

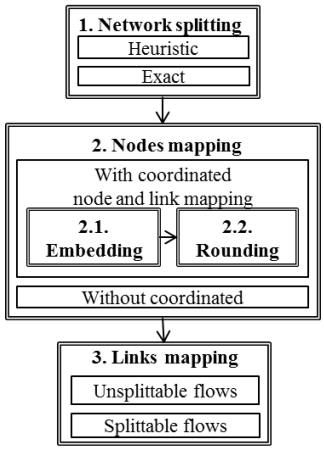

Fig. 9. Steps for the selection of the algorithms.

In *Network Splitting* the result is a virtual network divided into multiple virtual networks that were assigned to different network infrastructures. This step can be performed in two ways, with *heuristic* or *exact* algorithms [8]. The next step, *Nodes Mapping*, selects the best SDN nodes for the virtual nodes of the virtual subnetworks of the first step. This step can be done in two ways, *with* or *without coordinated node and link mapping*. In case of *with coordinated node and link mapping* it uses MIP formulation [13] to solve the embedding problem. Since solving an MIP is known to be NP-Hard [13], it relaxes the integer program to obtain a linear programming formulation that can be solved in polynomial time. Library CPLEX has been used to solve the relaxed MIP problem. It uses *deterministic* and *randomized rounding* techniques on the solution of the linear program to approximate the values of the binary variables in the original MIP [7]. In case of *without coordinated node and link mapping* it assigns virtual nodes using some greedy heuristics, e.g. assigns virtual nodes with higher processing requirements to SDN nodes with more available resources [9],[10]. In the final step, *Links Mapping*, once all the virtual nodes have been mapped, we embed virtual links onto SDN paths using shortest path algorithms [9] in case of unsplittable flows, or using *multi-commodity flow* algorithms [10], [14] in case of splittable flows.

A. Network Splitting Algorithms

A network splitting algorithm divides a virtual network in subnetworks for different network infrastructures optimally. The implemented network splitting algorithms [8] are: Exact, Recursive Random Permutations and Iterated Local Search. The main class is *SplittingAlgorithm* and models the main characteristics of the network splitting algorithms. *Exact*, *IteratedLocalSearch* and *RecursiveRandomPermutations* extend *SplittingAlgorithm*. The method *run* of *SplittingAlgorithm* is called by the simulation for runnig the algorithm and returns the subnetworks. Inside this method there is a call to the method *bestNodesMapping* that is empty in *SplittingAlgorithm* but differently implemented in *Exact*, *IteratedLocalSearch* and *RecursiveRandomPermutations*. This method returns the best mapping of virtual nodes for doing the network splitting. Fig. 10. Shows the hierarchy of the classes of the networks splitting algorithms.

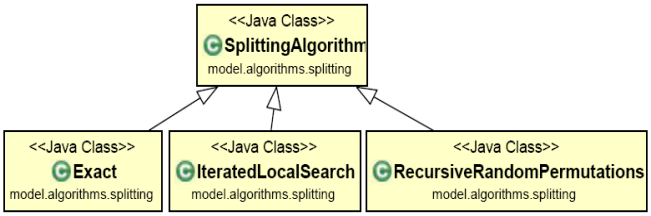

Fig. 10. Hierarchy of the classes of the network splitting algorithms.

B. Nodes Mapping Algorithms

Once we have the network split, the next step (step 2 in Fig. 9) is to allocate virtual nodes to SDN nodes. It can be done in two ways, *with* or *without coordinated node and link mapping*. The main class is *NodesMappingAlgorithm* and models the main characteristics of the nodes mapping algorithms. This class has the method *run*() not implemented. This method has two input parameters: a virtual network and a SDN network, and the output are the nodes mapping. Fig. 11 shows the hierarchy of the classes of the nodes mapping algorithms.

Without Coordinated Node and Link Mapping. Greedy node mapping algorithm extends *NodesMappingAlgorithm* (Fig. 11.) and implements its method *run* with greedy heuristics [9], [10].

With Coordinated Node and Link Mapping. The main class is *Rounding*, and extends *NodesMappingAlgorithm*. *Deterministic* and *Randomized* extend *RoundingAlgorithm* (Fig. 11.). This class has a different method *run*(), neither implemented. This method has four input parameters: a virtual network, a SDN network, *x* and *f* variables; and the output is the mapping of the nodes. The variables *x* are binary set to 1 if there is traffic flow on the virtual links via SDN links. The *f* variables are the amount of traffic for the virtual links routed over the SDN links. *x* and *f* variables are the result of a linear program, obtained by the class *EmbeddingAlgorithm*. This is the step *2.1. Embedding* in Fig. 9. The implemented network embedding algorithms are Networked Cloud Mapping (NCM) [11] and Virtual Network

Embedding (ViNE) [7]. The main class is *EmbeddingAlgorithm*. NCM and Vine extends it. The method *run* of this class is called by the simulation for running the algorithm and sets the *x* and *f* values. Inside this method there is a call to the method *minimize*, which is empty in *EmbeddingAlgorithm* but implemented in *NCM* and *ViNE*. This method minimizes the cost of embedding a virtual network in a SDN network. Fig. 12 shows the classes hierarchy of the network embedding algorithms.

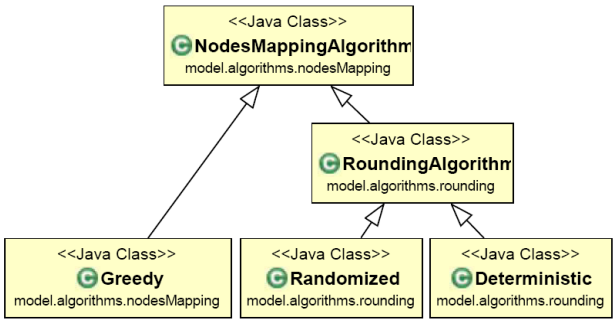

Fig. 11. Hierarchy of the classes of the nodes mapping algorithms.

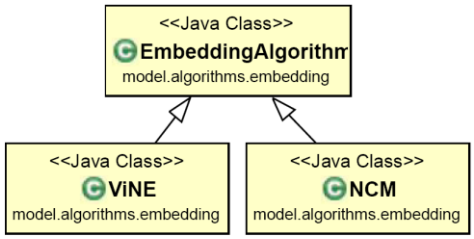

Fig. 12. Hierarchy of the classes of the network embedding algorithms.

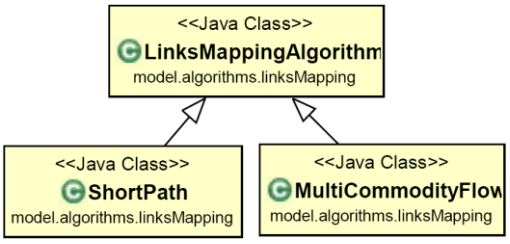

Fig. 13. Hierarchy of the classes of the links mapping algorithms.

Once we have the *x* and *f* values, the next step in Fig. 9 is *2.2*, performed by *RoundingAlgorithm*, in Fig. 11. This algorithm is responsible for obtaining integer values and with these values assigns virtual nodes to network infrastructures. The method *run()* of *RoundingAlgorithm* is called by the simulation for running the algorithm and returns the nodes mapping. This method is empty in *RoundingAlgorithm* but implemented in *Deterministic* and *Randomized*.

C. Links Mapping Algorithms

Once we have the nodes mapping, the next step is to incorporate virtual links to SDN links, i.e., step 3 in Fig. 9. It can be done in two ways, for unsplittable flows or splittable flows. The main class is *LinksMappingAlgorithm* and models the main characteristics of the links mapping algorithms. This class has the method *run*() not implemented. This method has three input parameters: a virtual network, a SDN network and the nodes mapping, and as output parameter, the links mapping. The class *ShortPath* implements the shortest distance path algorithm (Dijkstra's algorithm) for cases of unsplittable flows [9]. The class *MultiCommodityFlow* implements multi-commodity flow algorithm for the case of splittable flows [10], [14].

VI. Related Work

Georgia Tech Internetwork Topology Models (GT-ITM) [15] is a distribution that contains code to generate graphs that model the topological structure of internetworks [16]. In Performance results of [8] the proposed exact embedding algorithm is developed using C++ with the CPLEX library and the GT-ITM tool [15] is used to randomly generate physical topology and VN requests. In [7], the authors implement a discrete event simulator to evaluate the performance of algorithms [17] and the physical network topologies are randomly generated using the GT-ITM tool [15]. In [9] and [10] a discrete event simulator is implemented to evaluate the performance of algorithms and the physical network topologies are randomly generated using the GT-ITM tool [15].

In the simulator presented in this paper, the generation of SDN and virtual networks and algorithms are developed inside the same structure, in different blocks, enabling a total coordination between all phases of the virtual resource allocation in physical networks problem, allowing measuring the algorithms performance.

Network virtualization can enable multiple researchers to evaluate new network protocols simultaneously on a shared experimental facility like [18] and [19]. Planet-Lab [18] is a group of computers available as a test-bed for computer networking and distributed systems research. It was established in 2002 and in June 2010 it was composed of 1090 nodes at 507 sites worldwide. Planet-Lab experiences have become critical in the formulation of the US National Science Foundation's Global Environment for Network Innovations (GENI) initiative [19]. GENI, a virtual laboratory for exploring Future Internet at scale, creates major opportunities to understand, innovate and transform global networks and their interactions with society. Dynamic and adaptive, GENI opens up new areas of research at the frontiers of network science and engineering, and increases the opportunity for significant socio-economic impact. In our simulator the VN embedding algorithms can be tested previously in a local form.

VII. Conclusion and Future Work

SDN and NFV are a promising way to de-ossify the current Internet by providing a shared platform for a variety of new network services and architectures.

A major challenge in building the diversified Internet is to perform efficient and on-demand VN assignment.

In this paper, we have developed a modular and flexible graphical discrete event simulation tool for solving the complete virtual resource allocation in SDN networks problem. This Java-based tool has been developed to integrate existing and future algorithms related to each phase of the process. The simulator is a test-bed in which researchers can select the appropriate algorithm in each phase and display the results in a graphical form, while obtaining a

performance evaluation of the selected and proposed algorithms.

We have evaluated the performance of the proposed tool through experiments and our findings are summarized as follows:

1) The benefits of subdividing the virtual resource allocation problem in different phases or parts are the independent evaluation of each one. We can do research on one part without needing to change other parts of the simulator.
2) Different approaches of one part can coexist in the simulator and can be selected in order to evaluate and make a systematic comparison between them.
3) Another advantage of the proposed tool is that networks, algorithms and the graphical part have been developed in the same extended language programming (Java).
4) The proposed tool, in the graphical part, shows all assigned resources in each instant of time for performing an exhaustive evaluation. The results are shown in graphical and numerical form.

Future work will consist in improving the simulator adding new features like to save and to open simulations, to obtain numerical graphs from numerical results and to perform a large number of simulations oriented to numerical results without graphical results.

Moreover, we also plan to explore new algorithms formulation for both splitting and mapping problems.

REFERENCES

[1] N. Niebert, I. E. Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs, "Network virtualization: a viable path towards the future internet," *Springer Wireless Personal Communications*, pp. 511–520, March 2008.

[2] A. Haider, R. Potter, and A. Nakao, "Challenges in resource allocation in network virtualization," ITC Specialist Seminar on Network Virtualization, May 2009.

[3] L. Peterson, S. Shenkar, and J. Turner, "Overcoming the internet impasse through virtualization," *IEEE Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[4] J. Turner and D. Taylor, "Diversifying the Internet," in *Proc. the IEEE Global Telecommunications Conference* (GLOBECOM'05), vol. 2, 2005.

[5] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proc. SIGCOMM*, pp. 3-14, 2006.

[6] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61-64, 2007.

[7] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *Networking, IEEE/ACM Transactions*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[8] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate network," *Computer Networks*, vol. 55, no. 2, pp. 1011–1023, 2011.

[9] Y. Zhu, M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE International Conference on Computer Communications* (Infocom'06), pp. 1–12, Apr. 2006.

[10] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM Sigcomm Computer Communication Review*, vol. 38, no. 2, pp. 17–29, Apr. 2008.

[11] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. C. Pastor, and A. Monje. "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Transactions on Computer, special section on Optimizing the Cloud*, vol. 62, no. 6, pp. 1060–1071, June 2013.

[12] Simulator. [Online]. Available: http://147.83.118.145/Simulator.zip

[13] A. Schrijver, *Theory of linear and integer programming*, New York NY, USA: John Wiley & Sons, Inc., 1986.

[14] W. Szeto, Y. Iraqi, and R. Boutaba, "A multi-commodity flow based approach to virtual network resource allocation," in *Proc. the IEEE Global Telecommunications Conference* (GLOBECOM03), vol. 6, pp. 3004-3008, 2003.

[15] GT-ITM. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[16] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE International Conference on Computer Communications* (Infocom' 96), vol. 2, pp. 594–602 1996.

[17] ViNE-Yard. [Online]. Available: http://www.mosharaf.com/ViNE-Yard.tar.gz

[18] Planet-Lab. [Online]. Available: http://www.planet-lab.org/

[19] GENI. [Online]. Available: http://www.geni.net/

Aurelio Javier García received his degree in telematics engineering from the Cartagena School of telecommunications engineering (ETSIT), Universidad Politécnica de Cartagena (UPCT), Murcia, Spain. He is a member of bampla research group (design and evaluation of broadband networks and services), where he develops his PhD thesis with cristina cervelló-pastor as supervisor. His research interests focus on routing algorithms, network resources management, sofware-define network (SDN), network functions virtualization (NFV), network discovery protocols and virtualization.

Cristina Cervelló-Pastor received her MSc degree in telecom engineering and Ph.D degree in telecommunication engineering, both from the Barcelona School of telecommunications engineering (ETSETB), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. She is an associated professor in the Department of Telematics Engineering of UPC (ENTEL), and currently, she is the head of this Department. She had also been the director's assistant, the head of castelldefels section and the academic secretary of ENTEL. She is part of the research group on modelling and evaluation of broadband networks (BAMPLA). Her research trajectory has been centered on the field of routing in high speed networks, resource management, virtualization, and quality of service and the development of new protocols in optical networks. She has been involved in diverse national and European projects (NOVI, FEDERICA, ATDMA, A@DAN, Euro-NGI, Euro-FGI, EURO-NF) being responsible of various public and private funding R&D projects. She has been member of the technological committee of i2CAT during more than nine years. She has been reviewer of the Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo (CYTED) and reviewer of several journals and conferences. She has also contributed to the reviews of the XXXIII COIT for their awards in the best Master Thesis and PhD Thesis. In parallel, she has published diverse papers in National and International Journals and Conferences and she has supervised two thesis in the field of routing and contention resolution in high speed networks. She has also several patents in the area of OBS networks. Currently she is supervising two additional theses in the field of management and optimal resource assignment in software defined networks.

Yury Jiménez received her MSc degree in telecommunications engineering from the National University of Colombia. Since 2012 she is a PhD student in the Universitat Politècnica de Catalunya (UPC) at Barcelona (Spain) through a scholarship provided by the Catalonian Government. She is a member of bampla research group (design and evaluation of broadband networks and services), where she develops her PhD thesis. Her research interests focus on routing algorithms, network resources management, software-define network (SDN), network discovery protocols and virtualization.