Improving Productivity and Reducing Cost Through the Use of Visualizations for SDN Management

Vinícius Tavares Guimarães*, Oscar Caicedo Rendon†, Gléderson Lessa dos Santos*, Guilherme Rodrigues*, Carla Maria Dal Sasso Freitas*, Liane Margarida Rockenbach Tarouco*, Lisandro Zambenedetti Granville*
*Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre/RS, Brazil
{vtguimaraes,glsantos,gcrodrigues,carla,tarouco,granville}@inf.ufrgs.br
†Universidad del Cauca, Popayán, Cauca, Colombia
omcaicedo@unicauca.edu.co

Abstract— **Visualization means the use of Information Visualization (InfoVis) techniques to augment human cognitive capacity to support users tasks more efficiently. According to the current literature, visualization is a relevant requirement to support the management of Software-Defined Networking (SDN). For example, visualization can help network administrators in everyday tasks such as in the identification of traffic patterns, or in the implementation and test of new networking services. However, in most cases, network administrators are unskilled on InfoVis, which makes them naturally reluctant in the use of visualizations. Also, building up visualizations from scratch can spend a significant amount of effort, which directly impacts on the increase of cost and the decrease of administrators' productivity. Thus, in this paper, we introduce a reuse-based approach to facilitating the employment of visualizations in the SDN context with the focus on improving productivity and reduce costs. We developed a prototype to evaluate and demonstrate the feasibility of our approach. The obtained results to one of the usage scenarios show, for example, that our solution can decrease in 264 hours the administrators' workload (*i.e.*, around U$ 9,976.56) when employing visualizations in daily tasks.**

I. Introduction

Nowadays, the Software-Defined Networking (SDN) paradigm has attracted attention from both academia and industry mainly because it is a way to overcome traditional network challenges such as the "Internet Ossification" [1]. In SDN, the network control plane is decoupled from the forwarding plane (*i.e.*, data plane), making the control plane directly programmable [2]. Thus, the network intelligence is logically centralized in software-based controllers, and the network devices (*e.g.*, switches) perform packet forwarding based on rules installed by the controllers [3].

Despite the advances on SDN over the last years (*e.g.*, OpenFlow), the management of SDN-based networks is still an open issue [4]. In this sense, Wickboldt *et al.* [5] present the most relevant requirements to foster the development of SDN management. Among such requirements is visualization, which means the use of Information Visualization (InfoVis) [6] techniques to augment human cognitive capacity in order to support users tasks more efficiently. We advocate that InfoVis is an essential tool to help network administrators in SDN management tasks, mainly, due the high flexibility provided by this paradigm (*e.g.*, to deploy and test new networking services).

InfoVis for network management is an old research topic [7], and several investigations were developed throughout the years [8] [9] [10] [11] [12] [13] [14]. Specifically in the SDN context, there are some investigations on visualization techniques to help in everyday management tasks [15] [16] [17] [18] [19]. In general, these investigations have explored network topology views, 2D time-series charts, and graph representations to assist in, for example, monitoring, configuration, and security. In this paper, however, we take another perspective regarding InfoVis for SDN management. Specifically, our investigation focuses on facilitating the employment of InfoVis by network administrators to improve productivity and reduce costs. To the best of our knowledge, up to now, there is no investigation with this aim.

Our primary motivation is based on two key promises of SDN [20]: (*i*) simplified network management through the network automation; and (*ii*) the reduction of costs by decreasing the Capital Expenditure (CapEx) and the Operating Expenditure (OpEx). For the first one, we argue that InfoVis provides helpful tools to support the network automation. Thus, facilitating the employment of visualizations can contribute to achieving a simplified network management. For the second, our first insight is that, in most cases, network administrators are unskilled on InfoVis. So, they are naturally reluctant in the use of InfoVis because it can be laborious by spending a significant amount of effort, which directly impacts on the increase of costs and the decrease in productivity. In fact, developing visualizations from scratch can be very expensive because (*i*) it is usually time-consuming, in special, for network administrators that, in general, are non-experts in programming [21]; and (*ii*) it leads administrators to lose focus on their core tasks, consequently decreasing their productivity.

In this paper, we introduce an approach to facilitating the employment of visualizations in the SDN context by encouraging their reuse. We claim that a reuse-based approach is promising because (*i*) it takes advantage of current visualizations (*e.g.*, visualizations for traditional networks), (*ii*) it provides an easy path to enabling administrators to reuse groundbreaking visualizations; and (*iii*) it has as ultimate goal the decreasing of costs. Also, we understand that the SDN architecture provides proper mechanisms (*e.g.*, the northbound interface) to the design of a reuse-based approach.

In a nutshell, our approach uses principles and concepts of software reuse because we have realized that the building up of visualizations is essentially a software development task. So, we have explored the Reuse Asset Management Process of the IEEE Std 12207-2008 [22], and the principles of the Component-Based Software Engineering (CBSE) to reuse, in special, the Component-Based Development (CBD) process [23]. We developed a prototype to evaluate and demonstrate the feasibility of our approach. The obtained results to one of the usage scenarios show, for example, that our solution can decrease in 264 hours the administrators' workload (*i.e.*, around U$ 9,976.56) when employing visualizations in daily tasks.

The remainder of this paper is organized as follows. In Section II, we provide a brief description of the main background concepts associated with our approach. Section III introduces related work. In Section IV, we introduce our proposal. In Section V, we present the evaluation. Finally, in Section VI, we conclude the paper with final remarks and perspective for future work.

II. Background

This section presents an overview of SDN management and, after, it outlines InfoVis for network management.

A. SDN Management Overview

SDN has two major characteristics [24]: (*i*) the separation of the control plane (that defines how to handle the traffic) from the data plane (that forwards traffic according to the rules installed by the control plane); (*ii*) the consolidation of the control plane where a software-based program controls multiple network elements in the data plane through an Application Programming Interface (API). The current literature has also introduced the concepts of *southbound* and *northbound* APIs [4]. The southbound API is an interface between the data plane and the control plane, whereas the northbound API offers a common interface to upper layers (*e.g.*, the management plane) operate the control plane.

Based on that, one of the main promises of SDN is to allow flexible and efficient network management via software-based programs. However, this promise is not a reality yet. In this sense, Wickboldt *et al.* [5] introduce a set of requirements to foster the adoption and development of the SDN management. The requirements are bootstrap and configuration, availability and resilience, network programmability, performance and scalability, isolation and security, flexibility and decoupling, network planning; and monitoring and visualization. Although most of such requirements are not new and have been investigated in the scope of traditional networks, they must be revisited to encompass the SDN paradigm and, then, to achieve the promise of flexible and efficient management. In this paper, as previously presented, we focus on the ***visualization*** requirement.

B. InfoVis for Network Management

Essentially, InfoVis comprises visualization techniques for "abstract" data, *i.e.*, data in a non-spatial domain and structures like multi-dimensional databases, text, graphs, and trees [25]. A widely known InfoVis reference model [6] describes the flow of information in InfoVis applications. Initially, raw data (*e.g.*, network management data) are gathered in data tables together with other derived information. Next, data tables are processed and transformed into one or more visual representations. Finally, the end-user (*e.g.*, the network administrator) manipulates and interacts with the visual representation in one or more views.

Throughout the years, several works have proposed InfoVis techniques for network management [8]. In the 90s, for example, visualizations were used to assist in the management of the AT&T's network [14], and to display the Internet graph using routing and reachability data [26]. More recently, visualizations have been investigated for network traffic analysis [27], to assist in understanding the use of Simple Network Management Protocol (SNMP) [11], to help in the management of Wireless Sensor Networks (WSN) [9] and Mobile Ad-hoc Networks (MANET) [10], and to aid in comprehending the Internet routing data based on Border Gateway Protocol (BGP) information [12]. Moreover, many investigations have focused on assisting in security management [13].

III. Related Work

Current investigations on visualizations for SDN have focused on different management aspects. Afaq *et al.* [15] introduce an approach to classifying, detecting, handling, and visualizing elephant flows in SDN. They use 2D line charts to display the volume of traffic of flows passing through a switch, and the volume of traffic of an elephant flow in a given queue of the proposed queue-based classification scheme. Isolani *et al.* [16] present an approach based on monitoring and visualization to help administrators to make decisions about configuration/reconfiguration in OpenFlow-based networks. A physical topology view shows information such as the levels of resource usage, control traffic, and data traffic through the size and color of devices and links. Also, 2D interactive line charts (integrated with the topology view) display resources usage in terms of installed rules, traffic rates, and packet processing rates.

Pantuza *et al.* [17] use a real-time graph representation to display the SDN network topology where the graph is updated and rendered according to events happened into the controller such as the join/leave of network elements. Watashiba *et al.* [18] introduce a visualization tool that assists OpenFlow controller developers in the understanding of the physical network topology and flows behavior. Briefly, a physical topology view allows to superimposing the information of network flows like the traffic amount of each link. Wang *et al.* [19] propose an OpenFlow-based security management architecture where a topology view helps in the identification and location of security events by showing users join/leave, the load condition of links and service elements, who is accessing which application service, and where attacks happen.

These previous works focus on the development of visualizations to help network administrators in specific tasks like monitoring, configuration, and security of SDN. We take another path that, in essence, aims at facilitating the employment of visualizations by network administrators under the perspective of decreasing cost and improve their productivity.

IV. Proposal

This section presents the context and motivation, stakeholders, solution overview, and architecture, elements and its interactions.

A. Context & Motivation

Visualization is a helpful tool to assist network administrators in everyday tasks, in special for SDN management [5]. For instance, by using visualizations administrators can get insights (*i*) to identify anomalous conditions more promptly and, then, lead the managed infrastructure back to a consistent state; and (*ii*) to properly understand the main network scenarios, which is fundamental to the development of network automation routines. On the other hand, the expertise on InfoVis is not a mandatory skill for network administrators. Thus, the employment of visualizations by them requires, in most cases, diving into the unknown that can be too risky regarding increase cost and decrease the productivity. In this sense, we check three alternatives that the administrator may follow to employ visualizations in their daily tasks.

1) He/she elects a commercial tool (*e.g.*, Hyperglance). Although commercial tools can save the time of setup, deployment, and maintenance, we realize two major drawbacks: (*i*) it can strongly increase the CapEx; and (*ii*) it can affect the Total Cost of Ownership (TCO) significantly. This alternative may also be expensive in terms of effort and money since it can require staff training on proprietary technologies that are not in the public domain.

2) He/she selects an open-source tool (*e.g.*, the Floodlight Web UI). This alternative does not generate direct financial expenses. However, it can bring hidden costs that may increase the Operational Expenditures (OpEx) with the maintenance costs and the costs associated with the spending time to deploy and setup an open-source solution.

3) He/she decides to build up from scratch. Thus, he/she must have skills to choose suitable InfoVis techniques, select the development framework, and seek for supporting material for programming, deploying and maintaining. Besides being time-consuming and expensive, building up visualizations is out of the scope of network administrators' responsibilities, which can take them to lose focus on their core tasks, decreasing their productivity.

The first and second alternatives presented above have another important weakness related to the fact that the visualizations available by those tools are "as is". Thus, administrators are not able, for example, to explore groundbreaking visualizations that can better fit with their management needs. Regarding the third alternative, it can be replacing by outsourcing, where the administrator consults InfoVis designers and developers to build up custom visualizations. Although it can be less expensive in terms of effort, it can strongly affect CapEx and OpEx.

By observing such a context, we have realized that administrators can be reluctant in the employment of visualizations mainly because it can spend a significant amount of effort to design, develop, deploy, and maintain. This effort is directly translated in the decrease of productivity and increase of costs. Thus, our main contribution is to provide a solution that, by means of reuse, facilitates the employment of visualizations by network administrators for SDN management, focusing on improving productivity and cost reduction.

B. Stakeholders

Our proposal is based on three stakeholders that can collaborate with one another for improving the reuse (see Figure 1). The ***network administrator*** is the primary stakeholder of the architecture that takes advantage of reusable assets to facilitate the employment of visualizations in his/her daily tasks. The ***InfoVis expert*** has excellent skills in InfoVis, especially to build up visualizations. Here, the ***InfoVis expert*** can explore the SDN management domain to design and develop groundbreaking visualizations to improve the *Visualization Repository* with novel *Visualization* components. The ***wrapper developer*** has advanced skills in programming, in special, on manipulating and developing APIs. Wrapper developers can contribute to increasing the *Data Wrapper Repository* by implementing *Data Wrapper* components. We also assume that the ***network administrator*** can play the role of ***InfoVis expert*** (*e.g.*, to develop a *Visualization* component) and ***wrapper developer*** (*e.g.*, to write a *Data Wrapper* component) if he/she has skills to do that.

C. Solution Overview

As previously presented, we argue that the building up of visualizations is essentially a software development task. Thus, we understand that the knowledge provided by the software reuse discipline can be leveraged for our proposal once it is a traditional and well-established practice in software development focusing on, among other things, improving quality, increasing productivity, and decreasing cost [28]. In this paper, specifically, we investigate the Software Reuse Process defined in the IEEE Std 12207-2008 [22], and the CDB. The Software Reuse Process is divided into three main axes: Domain Engineering Process, Reuse Asset Management Process, and Reuse Program Management Process. Here, we focus on the Reuse Asset Management Process that aims to manage the life of reusable assets from conception to retirement. In our proposal, a reusable asset is a software component that is designed and developed for facilitating the use/reuse of visualizations for SDN management. Currently, we envisage two types of reusable assets: *Data Wrapper* and *Visualization* (see Section IV-D).

Regarding the Reuse Asset Management Process, we have adapted two expected outcomes from a successful implementation of it: (*i*) the **asset classification scheme**, and (*ii*) the **asset storage and retrieval mechanism**. This adaptation is feasible because the IEEE Std 12207-2008 describes the outcomes in a general manner, *i.e.*, there are no low-level specifications concerning techniques or technologies to achieve them. Thus, in the **asset classification scheme**, we have used the Simple Knowledge Organization System (SKOS) [29] to classify reusable assets. SKOS allows, among other things, an easy port of existing knowledge organization systems that enable us to take advantages of, for example, well-known taxonomies and classifications of SDN [30] and InfoVis [31].

In the design and development of the **asset storage and retrieval mechanism**, we rely on the concepts of CBD. CBD is an approach for building up a system by using components [23] where a component is a unit of composition with well-defined interfaces, which can be deployed independently and used by third parties [32]. CBD is part of the CBSE that has emerged as a more organized and focus approach to reuse [23]. In fact, the main advantage of CBD is reusability, and Sinha and Jain [33] show, through an empirical study, that IT professionals find it easier to reuse components than objects (*i.e.*, an object-oriented approach). Based on that, we advocate that CBD is also a suitable approach to facilitate the reuse of visualizations by administrators mostly because we understand that the profile of IT professionals and network administrators is too similar.

In CDB, Web services have become the dominant standard for delivering components as well as for providing a black-box reuse [33]. The black-box reuse is a type of reuse that does not require code modification, which allows a significant reduction of cost, in special, after several reuse instances [34]. In this sense, we argue that the black-box reuse through Web services is another important advantage of CBD for our proposal. First, it aims at reducing cost. Second, it can improve the productivity of network administrators when reusing because, in general, they are non-experts in programming [21]. Thus, we have designed reusable assets as Web services uniquely identified by a Uniform Resource Identifier (URI). This approach is also effective to (*i*) enabling platform-independent access for humans through a Web browser; and (*ii*) allowing interaction with other systems (*e.g.*, mashups) in an easier way since Web services are machine-readable.

D. Architecture, Elements & Interactions

Figure 1 shows the proposed architecture, its elements and interactions, and the stakeholders. The architecture is placed in the management plane of SDN [5], and it takes advantage of the northbound APIs of current SDN controllers.

Before explaining the elements of the architecture and its interactions, we describe the reusable assets *Data Wrapper* and *Visualization*. Essentially, the *Data Wrapper* is a software component responsible for accessing the northbound API of SDN controllers to retrieve data of interest (*e.g.*, flow statistics), and deliver such data to the *Visualization* component. The *Visualization* component, in turn, receives data from the *Data Wrapper* component to build up the visualization itself. Therefore, the *Data Wrapper* operates as a gateway between the northbound API of an SDN controller and the *Visualization* component. Thus, the launching of a visualization is performed by composing a *Data Wrapper* with a *Visualization* component where the only requirement is the *Data Wrapper* output be compatible with the *Visualization* input.

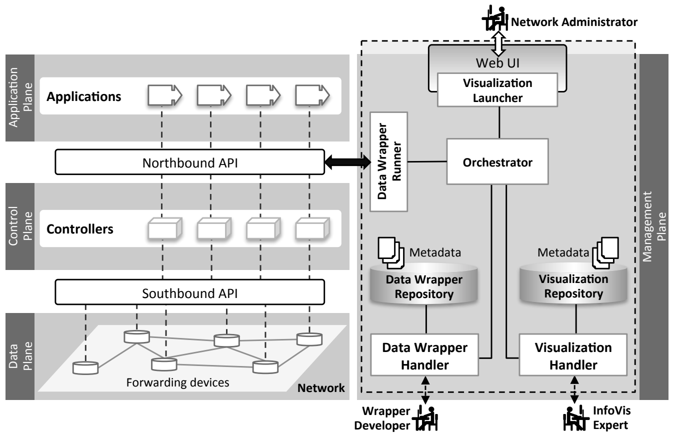


Figure 1. Conceptual architecture (dashed area) at the management plane

Regarding the proposed architecture, the *Data Wrapper Repository* and *Visualization Repository* elements store, respectively, the *Data Wrapper* and *Visualization* components together with their metadata. These metadata include information such as the input and output parameters, components description and classification, and aim to support stakeholders to use/reuse components or to develop new ones. The metadata information help, for example, in the design and development of a new *Data Wrapper* (*e.g.*, to access a new SDN controller) compatible with an existing *Visualization* component (*e.g.*, a topology view) by providing, among other things, the input format expected by the *Visualization* component.

The *Data Wrapper Handler* and *Visualization Handler* elements are responsible for manipulating the *Data Wrapper Repository* and *Visualization Repository*, respectively. These handlers are able, for example, to insert, read and update metadata information, retrieve the existing software components, and record new ones. The *Data Wrapper Handler* and *Visualization Handler* are integrated with the *Orchestrator* element. The *Orchestrator* is the main software element of the architecture because it is responsible for coordinating the execution of all other elements and, then, to display the visualization in the Web User Interface (UI). The *Orchestrator* works as explained below.

The user requests a visualization (*e.g.*, network topology) through the Web UI by selecting particular reusable assets, *i.e.*, a *Data Wrapper* (*e.g.*, to retrieve topology data from a Floodlight controller) and a *Visualization* (*e.g.*, link-node display) software component. Then, the information about the selected components is sent to the *Orchestrator* that invokes the *Data Wrapper Handler* and *Visualization Handler* to retrieve them from the repositories. The *Data Wrapper Handler* and *Visualization Handler* return the requested software components to the *Orchestrator*, which calls the *Data Wrapper*

Runner. In a nutshell, the *Data Wrapper Runner* executes the *Data Wrapper* component and delivers the output data provided by the *Data Wrapper* to the *Orchestrator*. We have designed the *Data Wrapper Runner* as a separate element of the *Orchestrator* to ensure more flexibility once currently there is no *de facto* standard for northbound APIs.

When the *Orchestrator* receives the output data coming from the *Data Wrapper*, it calls and runs the *Visualization* component by passing these data as input parameters. The *Visualization* component processes the received data, build the visualization, and returns it to the *Orchestrator*. As the last step, the *Orchestrator* sends the output generated by the *Visualization* component to the *Visualization Launcher*. Briefly, the *Visualization Launcher* is responsible for rendering and display the visualization in the Web UI.

V. Evaluation

This section outlines the experimental environment, prototype implementation, methodology, usage scenarios and, finally, results and discussion.

A. Experimental Environment

Figure 2 depicts the experimental SDN environment which is based on a university campus network. The emulated topology is composed of 13 OpenFlow switches connecting 112 hosts from 6 laboratories (each one with 16 hosts) and 2 administration offices (each one with 8 hosts), one Web Server, and one File Server. In summary, we use a simulation of real traffic of HTTP and FTP services between hosts and servers. The HTTP traffic is characterized by all hosts accessing the Web Server to request Web pages. Such Web pages vary in size according to the average page weight provided by the HTTP Archive Report[1] (*i.e.*, between 1.5MBytes and 2MBytes). The FTP service was used to generate traffic by transferring large files (*e.g.*, 8GBytes) between hosts of interest and the FTP server. Our goal is to create usage scenarios to demonstrate the employment of the proposed approach as well as the usefulness of visualizations to help in management tasks. The evaluation topology was created using the Mininet[2] emulator. We also used two remote controllers (Floodlight[3] and Ryu[4]) to highlight the reuse of *Data Wrapper* components. The experiment itself was performed in one 2.26 GHz Intel Core 2 Duo with 8GB of RAM.

B. Prototype Implementation

To support the evaluation, we developed a prototype that uses Web technologies and follows the Model-View-Controller (MVC) design pattern in a client-server architecture. The server-side runs Apache on a Linux-based server. *Data Wrapper* components were developed in PHP and Shell Script whereas *Visualization* components were implemented using

[1]http://httparchive.org/
[2]http://mininet.org/
[3]http://www.projectfloodlight.org/
[4]https://osrg.github.io/ryu/

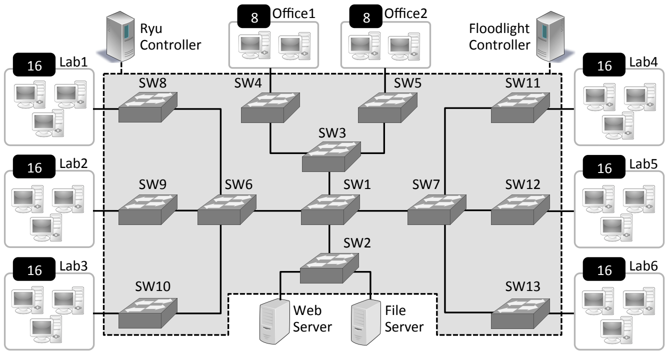


Figure 2. Evaluation topology

JavaScript and the D3.js[5] library. The *Visualization Repository* and *Data Wrapper Repository* are accessed through RESTful Web services, and the metadata and settings are stored in a relational database. The Python language was used to develop scripts that use the Mininet API to automating the build up of the evaluation topology as well as the simulation of the real traffic of HTTP and FTP services.

C. Methodology

Here, we explain the methodology to measure cost and productivity. Essentially, we estimate the effort in terms of consuming of time (in work hours) to build up a set of *Data Wrapper* (DW_n) and *Visualization* (VIS_n) components (see Table I). Based on the effort estimation, we can derive costs and analysis the impact on the productivity of network administrators. The software components (SC_n) (Table I) were also used to create usage scenarios that help us to demonstrate the effectiveness of the proposed approach.

As previously introduced, we advocate that building up a *Data Wrapper* and a *Visualization* is a software development activity. Thus, we have adopted the Common Software Measurement International Consortium (COSMIC) Functional Size Measurement (FSM) method that measures software sizing through Function Points (FP). We relied on the COSMIC measurement manual version 4.1.0 [35], which is the COSMIC implementation guide for the standards defined in ISO/IEC 19761:2011. We have adopted the COSMIC method for three main reasons: (*i*) it has generic applicability that comprehends software from different domains, (*ii*) it is used worldwide in various industries; and (*iii*) it is entirely open, *i.e.*, all method documentation is available in the public domain.

The COSMIC unit of measurement is 1 CFP (**C**osmic **F**unction **P**oint), which is the size of one *data movement*. A *data movement* moves a single *data group* within elementary software components called *functional processes*. There are four types of *data movement* that can be briefly described as follows: (*i*) an "*Entry*" moves data from a functional user into a *functional process*, (*ii*) an "*Exit*" moves data from a *functional process* to a functional user, (*iii*) a "*Read*" moves data from a persistent storage to a *functional processes*; and

[5]https://d3js.org/

Table I
DEVELOPED SOFTWARE COMPONENTS, AND THE ESTIMATION OF SIZE, EFFORT, AND COST

SC_n	$Size(SC_n)$	$Effort(SC_n)$	$Cost(SC_n)$	**Description**
DW_1	3 CFPs	33 hours	U$ 1,247.07	Retrieves topology data (*i.e.*, nodes and connections) from the Ryu controller.
DW_2	3 CFPs	33 hours	U$ 1,247.07	Retrieves flow statistics of each switch (*e.g.*, bytes count) from the Ryu controller.
DW_3	3 CFPs	33 hours	U$ 1,247.07	Retrieves flow statistics of each switch (*e.g.*, bytes count) from the Floodlight controller.
VIS_1	4 CFPs	44 hours	U$ 1,662.76	A simple dashboard that displays port data of a switch (*e.g.*, packets and bytes count). VIS_1 loads data both from DW_2 and DW_3.
VIS_2	5 CFPs	55 hours	U$ 2,078.45	A treemap view that displays and details the total traffic on each port of each switch in the topology. VIS_2 loads data from DW_2 and DW_3.
VIS_3	10 CFPs	110 hours	U$ 4,156.90	Interactive parallel coordinates view integrated with a grid panel that details the information presented in the view. VIS_3 loads data from DW_2 and DW_3.
VIS_4	17 CFPs	187 hours	U$ 7,066.73	An interactive topology view. The topology view enables users to click on a switch of interest to call VIS_1. VIS_4 loads data from DW_1.

(*iv*) a "*Write*" moves data lying inside a *functional process* to persistent storage.

Based on the COSMIC method, we can estimate the total size in CFPs of DW_n and VIS_n components. We use the equation $Effort(SC_n) = Size(SC_n) * DPR$ to estimate the total effort in work hours where the DPR means the Productivity Delivery Rate. DPR is defined in terms of hours per function point, *i.e.*, the average number of hours spent to produce a function point. Here, we used $PDR = 11$ hours per function point that is based on the average PDR for development team effort [36]. To estimate costs ($Cost$), we have adopted the mid-range wage (in U$) of network administrators provided by the 2015 career reviews[6]. The 50th percentile within the salary range of network administrators is U$ 72,560 annually. Thus, we used the earning per hour (in U$) as $E_h = 37.79$, *i.e.*, 160 hours a month, twelve months per year. Therefore, the total cost of a SC_n is $Cost(SC_n) = Effort(SC_n) * E_h$.

D. Usage Scenarios

In order to better illustrate our results and discussion, we first introduce three usage scenarios that use the SC_n presented in Table I.

- ***Scenario 1*** (S_1) uses DW_2 and VIS_2 components to display a treemap view (see Figure 4).
- ***Scenario 2*** (S_2) is composed of DW_3 and VIS_3 components to launch the parallel coordinates visualization (see Figure 5).
- ***Scenario 3*** (S_3) combines the DW_1 and VIS_4 components to show the topology view, and VIS_4 uses VIS_1 to display the simple dashboard with information about a switch of interest (see Figure 6).

E. Results & Discussion

Table I shows the number of CFPs, the $Effort$, and $Cost$ of each developed SC_n. The score in CFPs of *Data Wrapper* components has produced the same result because these elements have the same *functional processes* (*i.e.*, gathering data from the controller, processing data, and outputting data) that result in 3 *data movements*. The COSMIC Generic Software Model defines that *data manipulation* is not separately measured, *i.e.*, any *data manipulation* is assumed to be accounted for by the *data movement* with which it is associated. This definition helps us to clarify that, although these software components have the same *functional processes*, it does not mean that they have the same logical processing. We also highlight a significant increase in the number of CPFs (*i.e.*, $Effort$ and $Cost$) of the *Visualization* components. For example, the VIS_1 component (that is a simple dashboard) accounts more CFPs than the *Data Wrapper* components. Also, the VIS_3 and VIS_4 components have achieved a high number of CFPs mainly due to their interactive features (*e.g.*, linking and brushing and zooming). In essence, based on these results, we corroborate our insights that the building up of InfoVis techniques can be laborious and require a significant amount of effort, which can directly impact on the increase of costs and the decrease in productivity of network administrators.

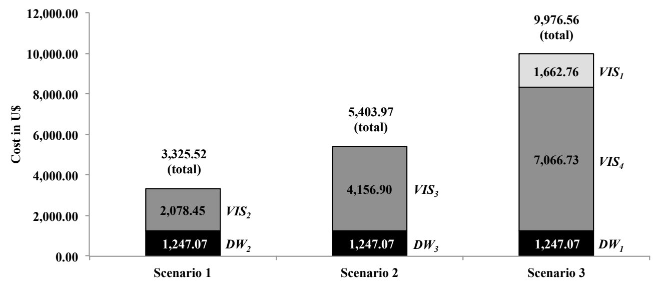


Figure 3. The estimated cost for each usage scenario

Regarding the usage scenarios previously presented, Figure 3 shows the cost estimated for each one based on the cost of each software component individually. In S_3, for example, the total cost to build up from scratch achieves U$ 9,976.56 by summing up the cost of DW_1, VIS_4, and VIS_3. Also, considering that one network administrator works 160 hours per month, the effort spent in S_3 (264 hours) reaches approximately one and a half month of work. Thus, we can state that our reuse-based approach decreases the administrators' workload when they employ visualizations to help in SDN

[6]http://computer-careers-review.toptenreviews.com/

management. By decreasing the workload, it reduces cost and, ultimately, saves money. Finally, the proposed architecture provides proper mechanisms (*e.g.*, the asset classification scheme) to facilitate and encourage the reuse.

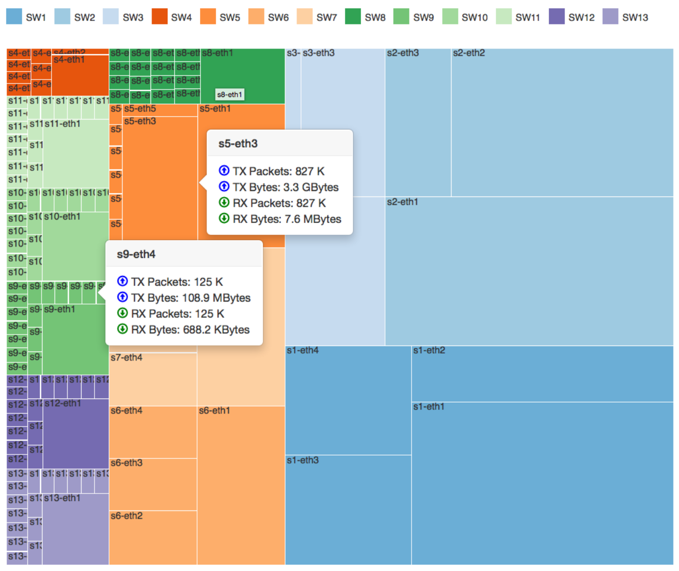
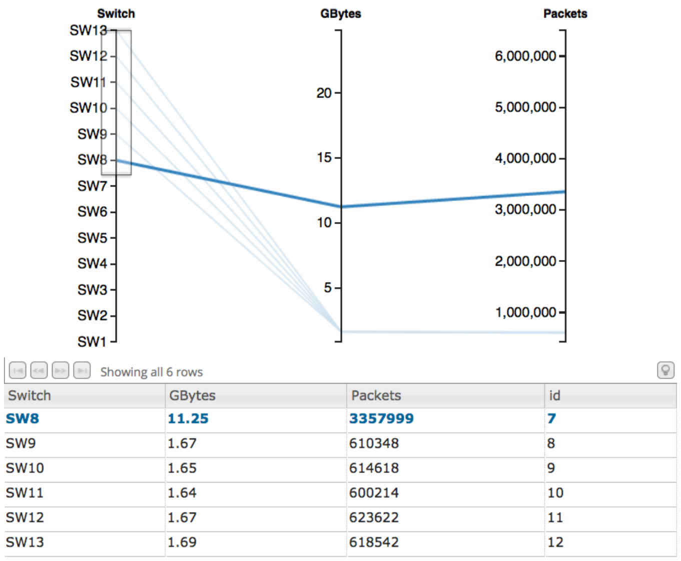


Figure 4. The treemap view (S_1) enables administrators to visualize detailed information by clicking on a point of interest. We highlight the comparison between the traffic load in s5-eth3 (Office2) and s9-eth4 (Lab2) because we simulated the download of a large file from the File server to the host on the port s5-eth3. Thus, the total traffic of s5-eth3 is bigger than the others while s9-eth4 follows the expected load for Labs and Offices. Such insight can be obtained by observing the size of the rectangles.



Switch	GBytes	Packets	id
SW8	**11.25**	**3357999**	**7**
SW9	1.67	610348	8
SW10	1.65	614618	9
SW11	1.64	600214	10
SW12	1.67	623622	11
SW13	1.69	618542	12

Figure 5. The parallel coordinate (S_2) shows information about aggregate traffic in each switch of the topology. Here, we selected only the switches of Labs. With this visualization, the network administrator can promptly identify that SW8 (Lab1) has approximately ten times more aggregate traffic than the others switches (see the dark line in the parallel coordinates system and the highlighted row in the data grid).

VI. Conclusions and Future Work

In this paper, we introduced a reuse-based approach that focuses on facilitating the employment of InfoVis by network administrators to improve productivity and reduce costs in the context of SDN management. We described the proposed approach outlining its motivation, conceptual architecture, elements and interactions, and stakeholders. We also developed a prototype to evaluate and demonstrate the feasibility of our approach, where we have corroborated that it can significantly reduce costs and improve productivity (in terms of effort) of network administrators in the employment of visualizations for SDN management. To achieve that, we propose an experimental SDN environment and develop a set of software components (*i.e.*, *Data Wrapper* and *Visualization*) to create three usage scenarios. By estimating the effort and costs to build up each software component, our results demonstrate that the development from scratch can be too time-consuming and expensive. For example, the third usage scenario sums up 264 hours of effort that achieves U$ 9,976.56 in cost. In short, we conclude that our reuse-based approach is a promising path to reducing costs and improve the productivity of network administrators in the employment of visualizations for SDN management.

As future work, we plan to extend the proposed architecture to provide *Data Wrapper* components that integrate with the SDN application plane. In summary, we believe that the number of SDN-based applications will grow in the next few years together with the need of useful visualizations that help network administrators in the management tasks to support such applications.

Acknowledgment

This work has been supported by PROPESP/IFSul, INF/UFRGS, CNPq, and CAPES-SETEC.

References

[1] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, July 2009.

[2] ONF White Paper, "Software-Defined Networking: The New Norm for Networks," April 2012.

[3] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[4] H. Kim and N. Feamster, "Improving network management with software defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 114–119, February 2013.

[5] J. Wickboldt, W. De Jesus, P. Isolani, C. Both, J. Rochol, and L. Granville, "Software-defined networking: management requirements and challenges," *Communications Magazine, IEEE*, vol. 53, no. 1, pp. 278–285, January 2015.

[6] S. K. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*, 1st ed. San Diego, CA, USA: Academic Press, 1999.

[7] A. Pras, J. Schoenwaelder, M. Burgess, O. Festor, G. Martinez Perez, R. Stadler, and B. Stiller, "Key Research Challenges in Network Management," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 104–110, October 2007.

[8] V. T. Guimarães, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco, and L. Z. Granville, "A survey on information visualization for network and service management," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 285–323, Firstquarter 2016.

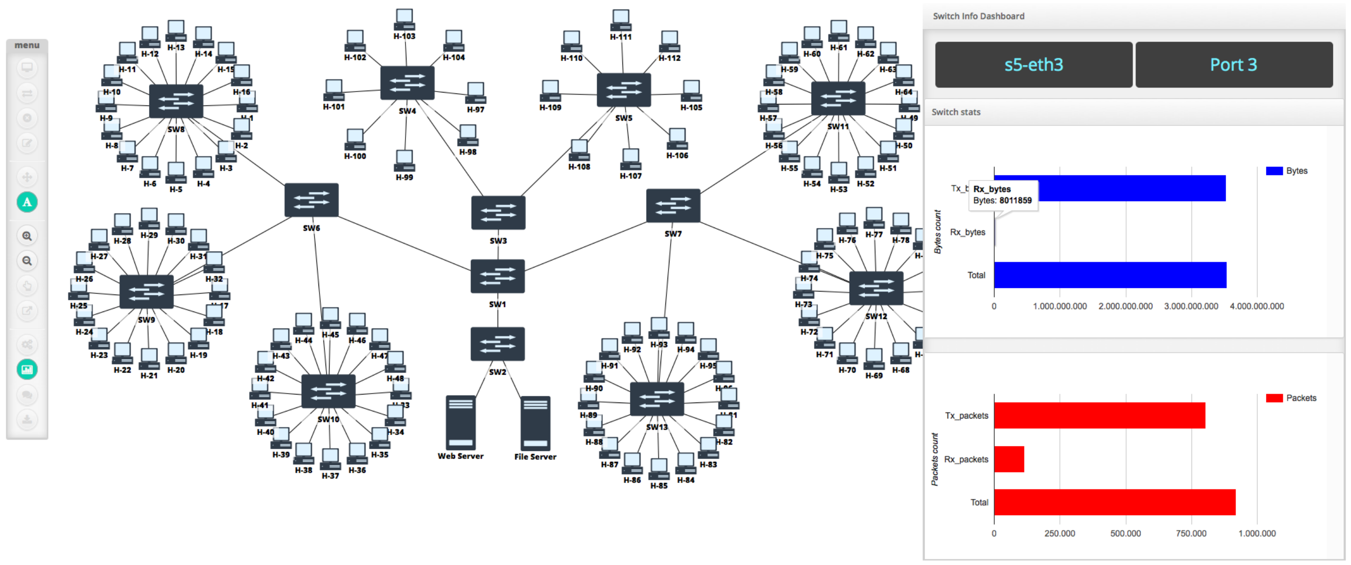


Figure 6. The topology view (S_3) helps the network administrator to understand the network topology. Here, we show the topology and the simple dashboard that displays information about s5-eth3 that could be used to complement the visualization presented in S_1. This visualization uses different glyphs to display each type of device (*e.g.*, switch, host, and server). It also has interactive features like zooming and moving elements. Such features allow the network administrator, for example, to organize elements according to his/her way (unlike the topology view available in the Floodlight Web UI which is based on a simple force-directed graph, for example).

[9] S. N. Sato, K. and H. Shimada, "Visualization and management platform with augmented reality for wireless sensor networks," *Wireless Sensor Network*, vol. 7, no. 1, pp. 1–11, January 2015.

[10] Y. Tsutsui, Y. Kitaura, K. Kikuchi, T. Ohta, and Y. Kakuda, "Development of a Real-Time System for Visualizing Network Topology and Packet Flow in MANET Field Experiments," in *Mobile Ad-hoc and Sensor Networks*, Dec 2014, pp. 151–157.

[11] V. Tavares Guimaraes, G. Lessa dos Santos, G. da Cunha Rodrigues, L. Zambenedetti Granville, and L. Rockenbach Tarouco, "A collaborative solution for SNMP traces visualization," in *ICOIN 2014*, Feb 2014, pp. 458–463.

[12] S. Papadopoulos, G. Theodoridis, and D. Tzovaras, "BGPfuse: Using Visual Feature Fusion for the Detection and Attribution of BGP Anomalies," in *VizSec'13*, New York, NY, USA, 2013, pp. 57–64.

[13] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1313–1329, Aug. 2012.

[14] R. A. Becker, S. G. Eick, and A. R. Wilks, "Visualizing network data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 1, pp. 16–28, Mar. 1995.

[15] M. Afaq, S. Rehman, and W.-C. Song, "Visualization of elephant flows and qos provisioning in sdn-based networks," in *APNOMS 2015*, Aug 2015, pp. 444–447.

[16] P. Heleno Isolani, J. Araujo Wickboldt, C. Both, J. Rochol, and L. Zambenedetti Granville, "Interactive monitoring, visualization, and configuration of openflow-based sdn," in *2015 IFIP/IEEE IM 2015*, May 2015, pp. 207–215.

[17] G. Pantuza, F. Sampaio, L. Vieira, D. Guedes, and M. Vieira, "Network management through graphs in software defined networks," in *CNSM 2014*, Nov 2014, pp. 400–405.

[18] Y. Watashiba, S. Hirabara, S. Date, H. Abe, K. Ichikawa, Y. Kido, S. Shimojo, and H. Takemura, "Openflow network visualization software with flow control interface," in *COMPSAC 2013*, July 2013, pp. 475–477.

[19] K. Wang, Y. Qi, B. Yang, Y. Xue, and J. Li, "LiveSec: Towards effective security management in large-scale production networks," in *ICDCSW 2012*, June 2012, pp. 451–460.

[20] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 1955–1980, Fourthquarter 2014.

[21] O. Caicedo Rendon, F. Estrada-Solano, and L. Zambenedetti Granville, "A mashup ecosystem for network management situations," in *GLOBECOM 2013*, Dec 2013, pp. 2249–2255.

[22] "ISO/IEC/IEEE standard for systems and software engineering–software life cycle processes," *IEEE STD 12207-2008*, pp. c1–138, Jan 2008.

[23] S. Mahmood, R. Lai, and Y. Kim, "Survey of component-based software development," *Software, IET*, vol. 1, no. 2, pp. 57–66, April 2007.

[24] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[25] M. Tory and T. Möller, "Rethinking visualization: A high-level taxonomy," in *INFOVIS '04*, Washington, DC, USA, 2004, pp. 151–158.

[26] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the internet," in *USENIX 2000*, 2000, pp. 1–12.

[27] R. Hofstede and T. Fioreze, "SURFmap: A network monitoring tool based on the Google Maps API," in *IFIP/IEEE IM '09*, June 2009, pp. 676–690.

[28] W. Frakes and K. Kang, "Software reuse research: status and future," *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 529–536, July 2005.

[29] W3C, "SKOS Simple Knowledge Organization System Reference," 2009, available at: http://www.w3.org/TR/skos-reference/. Acessed: March 2016.

[30] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 1955–1980, Fourthquarter 2014.

[31] T. Munzner, *Visualization Analysis and Design*, ser. AK Peters Visualization Series. CRC Press, 2014.

[32] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[33] A. Sinha and H. Jain, "Ease of reuse: An empirical comparison of components and objects," *Software, IEEE*, vol. 30, no. 5, pp. 70–75, Sept 2013.

[34] T. Ravichandran and M. A. Rothenberger, "Software reuse strategies and component markets," *Commun. ACM*, vol. 46, no. 8, pp. 109–114, Aug. 2003.

[35] COSMIC, "The COSMIC functional size measurement method - version 4.0.1," 2015.

[36] P. Morris, "The cost of speed," available at: http://www.totalmetrics.com/resources/software-measurement-articles/total-metrics-articles Acessed: March 2016.