A Mashup Ecosystem for Network Management Situations

Oscar Mauricio Caicedo Rendon
Computer Networks Group
Institute of Informatics
University Federal do Rio Grande do Sul
Email: omcrendon@inf.ufrgs.br

Felipe Estrada-Solano
Telematics Engineering Group
Telematics Department
University of Cauca
Email: festradasolano@unicauca.edu.co

Lisandro Zambenedetti Granville
Computer Networks Group
Institute of Informatics
University Federal do Rio Grande do Sul
Email: granville@inf.ufrgs.br

***Abstract*—Current network management approaches and their implementations are not intended to address dynamic situations that need rapid delivery of good-enough and comprehensive solutions. In this paper, we introduce a novel mashup ecosystem, called Mashment Ecosystem, that allows Network Administrators to conduct on a Mashment Maker the activities and interactions necessary to provide Mashments. Mashments are mashups aimed to tackle network management situations. We evaluate the Mashment Ecoystem by estimating with the Keystroke-Level Model and measuring in a test scenario the time that Network Administrators take to perform the activities of creating, launching, and publishing Mashments. Similarly, we evaluate the time for retrieving information about a network management situation by using or not Mashments. The evaluation results corroborated that Network Administrators, in our ecosystem, need short-time to deal with network management situations.**

I. Introduction

Nowadays, the use of computer networks has become vital to most enterprises. Up to now, in these networks, many management tasks require manual intervention by Network Administrators, mainly, to manage dynamic *Situations* that need rapid delivery of good-enough and comprehensive solutions [1]. In general, a *Situation* is a collection of entities (*i.e.*, things in a domain), their attributes, and relations in a time interval [2]. Hereinafter, we call a network management *Situation* as NMSit. Information technology departments do not provide situational solutions for NMSits because the requirements of these types of *Situations* are usually located in the long-tail of enterprise needs [3] [4]. As result, Network Administrators must create by themselves solutions for NMSits.

The Situation Management (SM) is an approach to provide solutions that enable to analyze, correlate, and coordinate the interaction between people, information, technologies, and actions intended to overcome *Situations* [2][5]. SM includes three aspects always linked to the time axis [2]: (*i*) the investigative aspect is related to retrospective cause analysis of *Situations* (*e.g.*, finding the root of a packet transmission failure in an OpenFlow-based network), (*ii*) the control aspect is directed to change or preserve *Situations* (*e.g.*, migrating a virtual switch from an overloaded server); and (*iii*) the predictive aspect is aimed to presage *Situations* (*e.g.*, projecting the appropriate time to migrate a critical router before a service disruption happen). In addition, as *Situations* are dynamic, SM emphasizes an adaptive management style.

Current network management solutions, such as Ganglia[6] and Nagios [7], are not intended to address the NMSits. Regarding these solutions, it is important to point out that they are not compatible, difficulting the information collection and information fusion that are necessaries to deal with a NMSit. Furthermore, the referred solutions are created without taking into account their rapid integration, extension, and improvement by Network Administrators, making hard the coping of NMSits. Thus, during a NMSit, the job of Network Administrators is hindered, since they are not able to enhance their workspace and must use numerous mismatched solutions from the Web and the network management.

Different research works have been developed about network management in traditional [8], virtual [9], Software Defined (SDN) [10], and cloud [11] networks. Such researches do not focus on face the NMSits and are developer centric. In the last years, mashups [12], that are end-user centric solutions formed by combining resources from different providers, have been applied in several domains as situational projects [13] and natural disasters [14] [15]. Nevertheless, mashups have not been used for tackling the NMSits. In this way, we raise the following question: how to tackle the NMSits by focusing on the Network Administrator?. In order to answer this question, we introduce a mashup ecosystem, named Mashment Ecosystem, which allows to carry out SM in network management. To the best of our knowledge, this work is the first to use an approach centric in the Network Administrator, SM, and mashup-based to deal with NMSits.

The Mashment Ecosystem, first, helps and encourages to Network Administrators to build up Mashments (*i.e.*, mashup used to deal with a NMSit) by themselves. Second, it allows Network Administrators to collect, correlate, and fuse information from heterogeneous resources offered by diverse providers. Third, it promotes the sharing and reuse of Mashments to avoid their wasting and to push the rise of innovative ones. Summarizing, the key contributions presented in this paper are to: (*i*) propose a novel Mashment Ecosystem for rapidly tackling NMSits by focusing in the Network Administrator; and (*ii*) demonstrate the short time that the Network Administrator needs to address a NMSit by building up and using a Mashment in our ecosystem.

The remainder of this paper is organized as follows. In Section II, we present SM and the mashup technology. In Section III, we introduce the Mashment Ecosystem. In Section IV, we expose and analyze the case study developed to evaluate

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

our proposal. In Section V, we provide some conclusions and implications for future work.

II. Background

In this section, we present a SM background. Also, we describe mashups and research works about mashups–SM.

A. Situation Management

SM is an emerging approach to provide solutions that need the planning and implementing of actions aimed to overcome a determined Situation [2]. SM requires the use of novel techniques [5], first, to collect time/state information about *Situations*. Second, to correlate and fuse multi-source information for timely and correct decision making in *Situations*. Third, to analyze past and predict future *Situations*. Fourth, to present information aiming at the human comprehension maximization.

Solutions based on the SM concepts are found in several domains. For instance, in the domain of polyester film base manufacturing, a situational solution, based on an expert system, has been proposed for monitoring the non-steady state events and assisting human operators with the event tasks [16]. In the aviation domain, a scalable and distributed situational system has been introduced for the management of air security incidents such as terrorist attacks that need, to be overcomed, the coordination and sharing of information from different organizations [17]. An architecture, based on SM, the Service Oriented Architecture, and developer centric, has been outlined for supporting demand response aspects of the smart grid domain [18]. Althoug SM has been used in several domains, there is not a SM-based approach to deal with NMSits.

B. Mashups

Mashups are web applications centered in end-users and built up by combining several resources (*e.g.*, data, application logic, and user interfaces) from one or more providers [12]. Here, end-user centric means that mashups can be built by users without advanced programming skills. In addition, regarding the mashups is to noteworthy [19]. First, they encourage the sharing among end-users. Second, the providers that supply resources, the end-users/developers that create mashups, and the end-users that use mashups act as a single unit known as mashup ecosystem.

If a mashup is developed for rapidly coping an immediate need of one or a set of end-users, it can be considered as a situational solution [20]. Mashups have been useful to manage *Situations* in diverse domains. For instance, Mashups were used to help to overcome a fire emergency in San Diego (California, United States) by sharing weather and rescue information among civil organizations and the government [14]. In situational projects that involve a small number of users and have a short lifespan, a mashup environment has been introduced in order to support management tasks. In such environment, the project manager is able to quickly develop a mashup for visualizing and filtering the information of his/her project [13]. An architecture, based on the Web 2.0 and wireless sensor networks, has been proposed in order to estimate the speed and timing of possible floods. A mashup prototype that collects, correlates, and presents data from multiple wireless sensors was developed to test the architecture [15]. Despite the use of mashups in several domains, there is not a mashup-based approach for tackling the NMSits.

III. Mashment Ecosystem

In order to better explain our proposal, we present an overview of the Mashment Ecosystem. Subsequently, we describe its Resources, Stakeholders, Activities&Interactions, and Software Entities.

A. Overview

Current network management approaches do not focus on dealing with NMSits. In the same way, although the mashup technology provides good basis for developing composite situational solutions by end-users, it has not been used for tackling the NMSits. Therefore, there is a gap in the mashup and network management related research and, consequently, there is a chance for innovation. Hereinafter, we present how a Mashment Ecosystem, based on the abstraction of resources, the mashups composition model, and a Network Administrator centric approach, can be targeted to address the NMSits. In particular for coping the NMSits, the Mashment Ecosystem faces three issues: (*i*) the complexity and heterogeneity to collect, correlate, and fuse information from multiple resources of the Web and the network management, (*ii*) the demand by functionalities that allow Network Administrators to rapidly create adaptable solutions for NMSits; and (*iii*) the need by visualization functionalities that enable Network Administrators to get NMSit information, in a very understandable way.

Before detailing the Mashment Ecosystem, we introduce the Mashment concept and a motivating scenario. A Mashment is a tunable situational mashup that allows Network Administrators (their end-users) to tackle a NMSit by combining diverse types of resources from multiple providers. Tunable means that Mashments are adaptable and easily customizable. Since Mashments are a special type of mashup, they can be created by Network Administrators. Regarding a Mashment is also relevant to point out. First, it hides the heterogeneity, complexity, and stiffness of resources used to deal with a NMSit. Second, it bears the easy collection, correlation, and fusion of information about a NMSit. Third, it presents NMSit information, in a visual and clear way. Fourth, it can be rapidly created to cope a determined NMSit.

Motivating Scenario. Let's suppose the following NMSit: in a virtual network formed by OpenFlow-based heterogeneous Slices from different providers (NP_a, NP_b, and NP_c), a packet failure transmission occurs because of sudden and unidentified errors. To tackle this NMSit, the Network Administrator needs to found errors from Slices in NP_a, NP_b, and NP_c. As every NP uses a different OpenFlow Controller, aiming at overcoming this NMSit, the Network Administrator has three options. The first one is to collect, correlate, and visualize network monitoring information by using disparate solutions, such as command line interfaces to execute specific commands on each Controller, distinct web user interfaces to monitor virtual switches, and external web tools to display non-integrated information about packet traffic. A drawback of first option is that the use of several mismatched solutions consumes more time than use an integrated solution. The

second option is to develop a low-level script to integrate the aforementioned commands, user interfaces, and web tools. This option also consume a lot of time because, the Network Administrator usually does not have advanced knowledge in programming. The third option is to participate in the Mashment Ecosystem. A Network Administrator in our ecosystem is able to quickly build up, in a high-level abstraction, by him/herself a Mashment to face the described NMSit. This Mashment hides the resources heterogeneity from NP_a, NP_b, and NP_c. Furthermore, the Mashment presents the network management information of virtual network in an integrated and intelligible way.

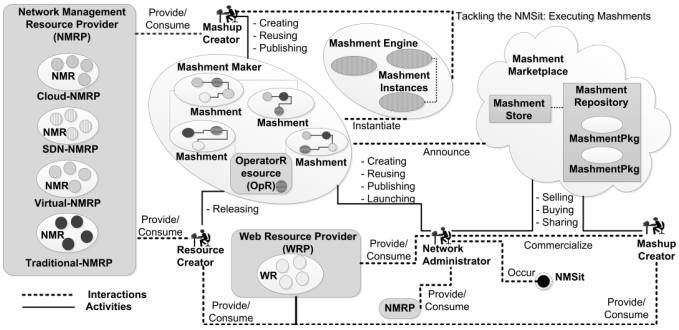

Fig. 1: Mashment Ecosystem

The Mashment Ecosystem (see Figure 1) is formed by: resources (Network Management Resources, Web Resources, and Operator Resources), Mashments, stakeholders (Network Administrator, Mashup Creator, Resource Creator, Web Resource Providers, Network Management Resource Providers, and Software Entity Providers), software entities (Mashment Maker, Mashment Engine, Mashment Repository, and Mashment Store), activities performed by stakeholders, interactions between stakeholders, and interactions between software entities. In this ecosystem, Network Administrators and Mashup Creators build up Mashments by using the Mashment Maker. Mashments are made up of resources from different providers and are executed in the Mashment Engine. The resources are released by the Resource Creator. In the Martketplace, the Mashments are shared, sold, and purchased by Mashup Creators and Network Administrators.

If a NMSit occurs, a Network Administrator can address it as follows: (*i*) buying or getting free of charge a Mashment(s), rapidly creating one or more Mashments, or quickly reusing a Mashment(s) previously built; and (*ii*) executing the purchased or created Mashment(s). It is important to highlight that the Mashment Ecosystem evolves over time because of the emerging and perishing of resources, the sharing and commercialization of Mashments, and the dynamic interactions between stakeholders.

The Mashment Ecosystem can be expressed as $MEco = \{M, St, A, I, Se\}$. Where: St, A, I, and Se are sets of stakeholders, activities, interactions, and software entities, respectively. In turn, the set of Mashments $M = \{m_i | m_i = (R_{used}, r_{root}, \delta, nmsit) : R_{used} \subset R, r_{root} \in R, nmsit \in NMSit\}$. Here, R_{used} is the set of resources on m_i, r_{root} is the root resource that starts the m_i execution, δ is the execution flow of resources on m_i, and $nmsit$ is the specific NMSit tackled by m_i. The other sets forming our ecosystem are described in next subsections.

B. Resources

A resource is a clearly identifiable entity in a time interval, which is conceived or can be adapted to tackle a NMSit. The set of resources is $R = \{r_i | r_i \in NMR \cup WR \cup OpR\}$. Where, Network Management Resources (NMR) are entities intended for the network management. NMR examples are Ganglia to manage traditional networks, Citrix Center for monitoring virtual resources, NetOpen to control OpenFlow-based networks, network monitoring systems based on the Simple Network Management Protocol, and all Application Programming Interfaces (API) that provide interaction with network elements.

Web Resources (WR) are Internet entities conceived or useful (via adaptation) for the network management. WR examples are the Google Maps API to show the geographic location of several network devices, the Multi Router Traffic Grapher (MRTG) to generate web pages with images presenting the traffic of network links, and the RRDTool to display over time the performance data of routers.

Operator Resources (OpR) are entities for combining resources (*i.e.*, NMR, WR, and even Mashments). There are two classes of OpR. Configuration_OpR to set up parameters for both access and communication to resources. An example of Configuration_OpR is a service to configure the security credentials required to monitor a virtual router. Control_OpR are composition patterns, such as $Split$, $Merge$, $Aggregate$, $Invoke$, $Trigger$, and $Receive$, useful, for instance, to collect, correlate, and fuse resources.

C. Stakeholders

A stakeholder affects and is affected by the activities and interactions performed by other one. The set of stakeholders is $St = \{st_i | st_i \in NMRP \cup WRP \cup SEP \cup ResourceCreators \cup MashupCreators \cup NetworkAdministrators\}$. Where, The Network Management Resource Provider (NMRP) is in charge of supplying NMR. Citrix Systems and Cisco Systems, providing solutions and programming interfaces to manage virtual servers and network devices, are examples of NMRP. The Web Resource Provider (WRP) is responsible for supplying WR. An example of WRP is a big player as Yahoo Inc. that provides visualization libraries and map services useful to present network management information. Another example is Oetieker&Partners Inc. that supplies web solutions intended for network monitoring, such as RDDTool and SmokePing.

The Software Entity Provider (SEP) is in charge of offering one or more software entities. In general, the Mashment Maker (containing visual representations of NMR, WR, OpR, and Mashments) and the Mashment Engine are provided, in an unified way, by the same SEP. In turn, the Mashment Repository and Mashment Store are usually offered, in a distributed way, by different SEP. In the Mashment Ecoystem can exist several Makers, Engines, Repositories, and Stores.

Before participating in the building of a Mashment, many WR and NMR need of adaption, in data format and/or communication protocol. The Resource Creator is responsible for

this adaptation that we called releasing. Since such a releasing requires strong programming skills, Resource Creators are usually software companies and professional developers. The Open Software community, that provides APIs to interact via standardized protocols with network devices and servers containing virtual routers, is an example of Resource Creator.

The Mashup Creator creates, publishes, and launches Mashments by means of the Mashment Maker and the Mashment Engine. Also, he/she is able to share, sell, and buy Mashments in the Marketplace. A Mashup Creator can get profits by commercializing Mashments. Software companies, professional developers, and end-users are examples of this class of stakeholder.

The Network Administrator is responsible for tackling the NMSits in traditional, virtual, SDN, and cloud networks. The Mashment Ecosystem allows Network Administrators to: (*i*) create and execute Mashments, (*ii*) reuse existing Mashments, NMR, OpR, and WR, (*iii*) improve their workspace as result of (*i*) and (*ii*); and (*iv*) get profits through publishing and selling Mashments.

D. Activities and Interactions

The activities are actions conducted by stakeholders in the software entities. The set of activities is $A = \{Releasing, Creating, Reusing, Publishing, Launching, Selling, Buying, Sharing\}$. Where, $Releasing$ is carried out by Resource Creators to enable the combination of heterogeneous resources through adapting NMR and WR. After $Releasing$, the adapted resources can be used to build up Mashments.

Mashup Creators and Network Administrators perform $Creating$, $Reusing$, $Publishing$, $Sharing$, $Selling$, and $Buying$. $Creating$ allows to build up a Mashment (*i.e.*, define δ or execution flow), which involves: (*i*) discover the available resources (NMR, WR, OpR, and Mashments), (*ii*) select the suitable resources to address a NMSit, (*iii*) orchestrate a static or dynamic plan for tackling a NMSit, by combining the previously selected resources, (*iv*) monitor if the WR, NMR, and Mashments used on the orchestrated plan are available and flawless; and (*v*) reconfigure the orchestrated plan if any resource is in flaw or unavailable.

$Reusing$ enables to take advantage of existing Mashments, aiming at the creation of more complex and innovative Mashments. $Publishing$ allows to package and put Mashments in the Mashment Repository, aiming at their sharing, selling, and buying. $Sharing$ enables to offer and get Mashments free of charge. $Selling$ and $Buying$ allow the commercialization of Mashments. $Sharing$, $Selling$, and $Buying$ promote the reuse of Mashments and encourage the evolution of the Mashment Ecosystem. Mashments (built and purchased) are sent to execute through performing $Launching$, which, in turn, is conducted by Network Administrators. Every Mashment launched is called Mashup Instance.

The interactions take place in the relationships: stakeholder/stakeholder and software entity/software entity. The set of interactions can be expressed as $I = \{Provides, Consume, Tackling, Commercialize, Occur, Instantiate, Announce\}$. Where, $Provide$ and $Consume$ occur from the need of supplying and consumption of NMR and WR, during: the building up of the Mashment Maker, the resources releasing, and the Mashments creation that enhances and improves the Mashment Maker. $Provide$ and $Consume$ take place among: Resource Providers, Resource Creator, Mashup Creators, and Network Administrators.

$Instantiate$ is conducted among the Mashment Maker and the Mashment Engine when a Network Administrator launches a Mashment. $Announce$ is performed among the Mashment Maker and the Mashment Marketplace, aiming at publishing of Mashments that can be later shared, purchased, and sold in $Commercialize$. Mashup Creators and Network Administrators carry out $Commercialize$. $Occur$ and $Tackling$ are special interactions used to represent the emerging of a NMSit and the corresponding responses offered by Mashment Instances. During $Tackling$, Mashment Instances interact with NMR, WR, and OpR in order to face a NMSit.

E. Software Entities

The software entities are responsible for supporting and automating the activities and interactions aforedescribed. The set of software entities is $Se = \{Maker, Engine, Marketplace\}$. Where, the Mashment Maker allows Network Administrators and Mashment Creators to build up Mashments, in an easy and rapid way. The Maker is a development environment that provides a composition approach, high-level programming tools, and a lightweight development process. The composition approach consists of four phases related to $Creating$ and $Reusing$ activities: (*i*) discover and select, (*ii*) orchestrate, (*iii*) monitor and reconfigure; and (*iv*) reuse. The above phases enable, first, to use the last information retrieved from available resources. Second, to avoid the use of redundant, incomplete, or irrelevant information provided by resources in failure. Third, to avoid the lack of information because of unavailable resources.

The high-level programming tools are visual facilities, based on drag-and-drop and wire mechanisms, that allow to perform the composition approach, $Publishing$, and $Launching$. Such tools are responsible for hiding the data mapping among WR, NMR, and OpR. The data mapping is a problem particularly daunting for Network Administrators, because, generally, they are not expert developers. As the Mashment Maker is devoted to Network Administrators, we define a simple process to tackle whatever NMSit: (*i*) *conduct the approach of composition* above described and so build up Mashments for the NMSit or *buy the Mashment* for the NMSit, (*ii*) *use the Mashments* to deal the NMSit; and (*iii*) *maintenance of Mashments* to avoid their malfunctioning.

The Mashment Marketplace allows to establish a new value chain in which revenues are shared not only by WRP and NMRP but all stakeholders. Marketplaces that involve end-users (as Network Administrators) and professional developers (as Mashup Creators) have proved valuable to promote the evolution over time of Service Ecosystems (as the Mashment Ecosystem). The Android Market and the Apple Store are succesful examples of solutions marketplaces. The Mashment Marketplace is make up of the Mashment Store(s) and the Mashment Repository(s). In the Mashment Store are performed

$Selling$, $Sharing$, and $Buying$. As result of $Announce$, in the Mashment Respositoy are stored MashmentPkgs (a packaged Mashment) to be sold or shared. The reuse of existing Mashments, by $Selling$ and $Sharing$, is a key aspect for the evolution of proposed ecosystem.

The Mashment Engine is responsible for the lifecycle (*i.e.*, $Instantiate$) of Mashment Instances that are Mashments on run time. A Mashment Instance allows to tackle (*i.e.*, $Tackling$) a NMSit. As the Mashments are make up of WR, NMR, OpR, and even Mashments, that function in back-end or front-end, the Mashment Engine is an enabler to create, destroy, and cache such resources into both web servers and web/mobile clients.

IV. Case Study

To assess our proposal, we perform a test environment for the Mashment Ecosystem. This section describes the test conditions and analyzes the obtained results.

A. Test Environment

The Figure 2 depicts the test environment for the raised case study. To set up such environment, first, we created a heterogeneous virtual network. Second, we developed the Mashment Maker, the Mashment Engine, and the Marketplace Repository (as a Database). Third, we released into the Mashment Maker the resources to create, launch, and publish a Mashment, hereinafter called MVN. When suddenly and unidentified transmission errors occur (*i.e.*, the NMSit) in the virtual network built, MVN allows the Network Administrator to visually look for them (*i.e.*, tackling the NMSit) by presenting, in a visual and integrated way, the collected, correlated, and fused information about packet traffic from switches.

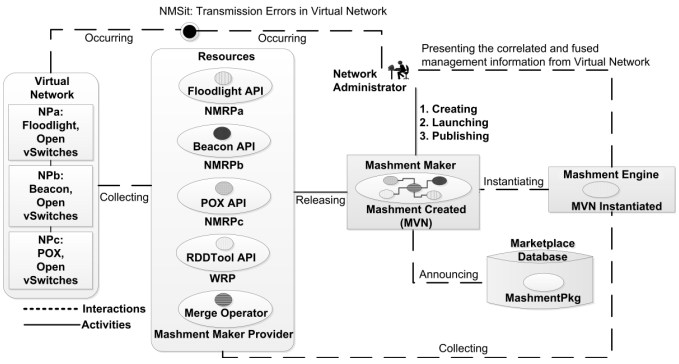

Fig. 2: Test Environment

The virtual network was created by using Open vSwitch and one different OpenFlow Controller in each network provider (NP_a, NP_b, and NP_c), namely, Floodlight, Beacon, and POX. Beacon and Floodlight are controllers based on the Java programming language. In turn, POX is based on Python. The controllers and switches were deployed on the Mininet that is a software for emulating OpenFlow networks. The Mininet was executed on Oracle VM VirtualBox. The Mashment Maker was developed by using Asynchronous Javascript and XML (AJAX), web services based on the Representational State Transfer (REST), and APIs for Floodlight, Beacon, POX, and RRDTool. The Mashment Maker was deployed on the Apache Tomcat Server. This Apache Tomcat was used as the Mashment Engine in the server-side. In the client-side, the Mashment Engine used was Firefox. The Marketplace Database was implemented on a MySQL Server. Network Administrators created the MVN by using the Mashment Maker.

B. Evaluation and Analysis

To evaluate the Mashment Ecosystem, we estimated and experimentally measured the time that Network Administrators take to perform $Creating$, $Launching$, and $Publishing$ MVN. MVN (see Figure 3) is formed by five visual components (*i.e.*, R_{used}): BeaconController, FloodlightController, and POXController representing the OpenFlow controllers (*i.e.*, NMR) used in the virtual network, RRDTool (*i.e.*, WR) representing the web tool used to generate images that present packet traffic information, and Monitoring Panel that is a merge operator (*i.e.*, OpR) and the root resource (*i.e.*, r_{root}). Subsequently, we also estimated and experimentally measured the time that Network Administrators take to retrieve, by using MVN, information about the NMSit above presented.

The time estimation was made by using the Keystroke-Level Model (KLM). In KLM, each activity is modeled as a sequence of actions. The time average for KLM actions is [21]: (*i*) Press and release a key $\rightarrow K = 0.2s$, (*ii*) Type a string $\rightarrow T_n = n * K$, (*iii*) Hold or release the mouse $\rightarrow B = 0.1s$, (*iv*) Point the mouse $\rightarrow P = 1.1s$, (*v*) Move the hand from mouse to keyboard or viceversa $\rightarrow H = 0.4s$; and (*vi*) Mental preparation $\rightarrow M = 1.35s$. In addition to these actions, we used, drag-and-drop a visual element $\rightarrow T_{dnd}$ and wire two visual elements $\rightarrow T_{wire}$. T_{dnd} and T_{wire} are given by [4]: $T_{dnd} = P + 2B = 1.3s$ and $T_{wire} = 3P + 8B = 4.1s$.

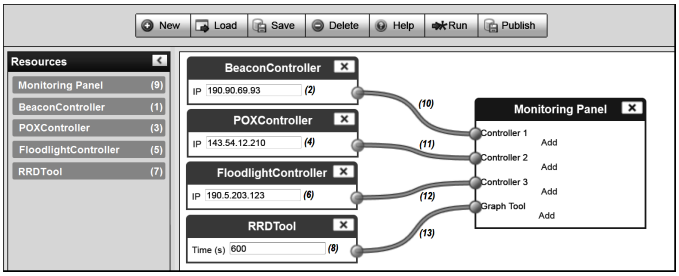

Fig. 3: Mashment Maker - Creating MVN

To tackle the NMSit, the Network Administrator creates MVN, the corresponding actions sequence (*i.e.*, δ) is as follows (see Figure 3): ***(1)*** Drag-and-drop Beacon $\rightarrow T_{dnd}$, ***(2)*** Configure Beacon (IP Address=190.90.69.93) $\rightarrow T_{con12}$, ***(3)*** Drag-and-drop POX $\rightarrow T_{dnd}$, ***(4)*** Configure POX (IP Address=143.54.12.210) $\rightarrow T_{con13}$, ***(5)*** Drag-and-drop Floodlight $\rightarrow T_{dnd}$, ***(6)*** Configure Floodlight (IP Address=190.5.203.123) $\rightarrow T_{con13}$, ***(7)*** Drag-and-drop RRDTool $\rightarrow T_{dnd}$, ***(8)*** Configure RRDTool (Time in seconds = 600) $\rightarrow T_{rrd}$, ***(9)*** Drag-and-drop Monitoring Panel $\rightarrow T_{dnd}$, ***(10)*** Wire Beacon to Monitoring Panel $\rightarrow T_{wire}$, ***(11)*** Wire POX to Monitoring Panel $\rightarrow T_{wire}$, ***(12)*** Wire Floodlight to Monitoring Panel $\rightarrow T_{wire}$; and ***(13)*** Wire RRDTool to Monitoring Panel $\rightarrow T_{wire}$. Where, $T_{con38} = P + 2H + T_{n=38} = 8.8s$ and $T_{rrd} = P + 2H + T_{n=3} = 2.5s$.

According the previous sequence, the estimated time for *Creating* MVN is $\mathbf{C_{est} = 13M + T_{con38} + T_{rrd} + 5T_{dnd} + 4T_{wire}}$. Then, it is expected that, by using the Mashment Maker, Network Administrators take $56.25s$ to build up MVN. We consider this $\mathbf{C_{est}}$ is good because, for instance, just typing the example script (10 lines with 40 characters each one) to generate a single RRD image takes $T_{script} = 10M + T_{n=400} = 93.5s$. In this way, we can state that Network Administrators, participating in the proposed ecosystem, can rapidly create Mashments aimed to tackle a NMSit. Generalizing, the estimated time to create any Mashment can be expressed as: $\delta(t) = T_{sel} + T_{conn} + T_{conf} + T_{ment}$. Where, $T_{sel} = \sum_1^i T_{dnd}$, $T_{conn} = \sum_1^j T_{wire}$, $T_{conf} = P + H + (nt * K)$, and $T_{ment} = M * (i + j + o)$. Here, i is the number of elements in the set R_{used} (drag-and-dropped), j is the number total of wires linking R_{used}, nt is the number total of characters to configure R_{used}, and o is the number of R_{used} to be configurated. In $\delta(t)$ a dynamic composition model could be used to decrease both $Tsel$ and T_{conn}. This dynamic composition for Mashments is a future work.

The Network Administrator performs the following actions sequence for *Launching* MVN. This sequence is the same for every Mashment: ***(1)*** Drag-and-drop MVN (when a Mashment is created, it is represented as a resource in the Mashment Maker) $\rightarrow T_{dnd}$, ***(2)*** Point the mouse to Run button $\rightarrow P$; and ***(3)*** Press and release Run button $\rightarrow 2B$. Therefore, the estimated time for *Launching* is given by $\mathbf{L_{est} = 3M + P + T_{dnd} + 2B}$. As result, it is expected that Network Administrators take $6.65s$ to start MVN by means of the Mashment Maker.

The Network Administrator performs the following actions sequence for *Publishing* MVN. This sequence is the same for every Mashment. ***(1)*** Point the mouse to Save button $\rightarrow P$, ***(2)*** Point the mouse to dialog that asks the Mashment name $\rightarrow P$, ***(3)*** Type the string MVN $\rightarrow T_{n=3}$, ***(4)*** Mouse press and release to store MVN in the Mashment Maker $\rightarrow 2B$, ***(5)*** Point the mouse to Publish button $\rightarrow P$, ***(6)*** Point the mouse to dialog that asks the Marketplace location $\rightarrow P$, ***(7)*** Type a repository string, for instance, http://www.mashments.mplace.com/repos $\rightarrow T_{n=37}$; and ***(8)*** Mouse press and release to store MVN in the Marketplace Database $\rightarrow 2B$. Therefore, the time estimated for the *Publishing* activity is given by $\mathbf{P_{est} = 8M + 4(P + B) + T_{n=3} + T_{n=37}}$. As result, it is expected that the Network Administrator takes $23.60s$ to publish any Mashment. Afterwards, MVN and, in general, any Mashments can be shared, sold, and purchased in the Marketplace Store that will be presented in a future work.

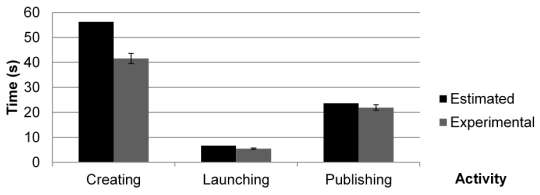

Fig. 4: Activities Time: Estimated vs Experimental

We also conducted an experimental study to measure the time that Network Administrators take to perform *Creating*, *Launching*, and *Publishing*. In the study participated 30 Network Administrators whose age ranged from 22 to 35. Although all participants frequently had used web tools none of them had used a mashup maker before. Thus, each participant was trained to use the Mashment Maker by 45 minutes. We took the experimental average time in seconds with a 95% confidence level. The results of estimated and experimental times (see Figure 4) corroborate the short time that Network Administrators need to tackle a NMSit by performing activities in the proposed ecosystem. Furthermore, as the experimental times of *Creating* ($41.55s$), *Launching* ($5.46s$), and *Publishing* ($21.92s$) were always less than the corresponding estimated times, we can state that the implementation of proposed ecosystem had a good behavior in front of KLM estimations.

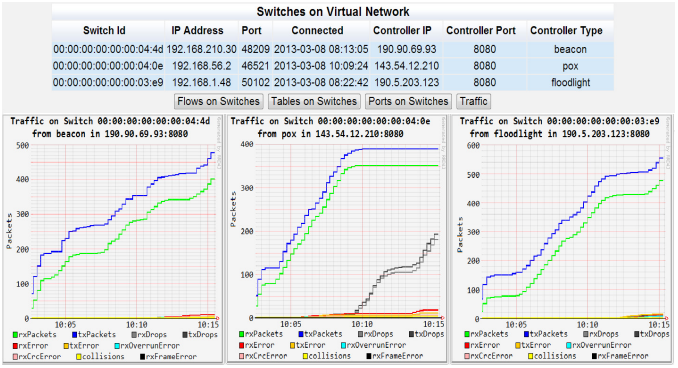

Fig. 5: MVN on runtime

On runtime, MVN (see Figure 5) allows Network Administrators to tackle the raised NMSit. MVN during its execution presents, simultaneously in an integrated user interface, the information of flows, links, and packect traffic of Open vSwitches, regardless of controllers from network providers. For instance, into MVN, the actions sequence to retrieve packet traffic information of three switches, each one in a different controller, is as follows: ***(1)*** Point the mouse to controllers list $\rightarrow P$, ***(2)*** Mouse press and release to select three controllers $\rightarrow 6B$; ***(3)*** Point the mouse to Switches button $\rightarrow P$, ***(4)*** Mouse press and release the Switches button $\rightarrow 2B$, ***(5)*** Mouse press and release to select three switches $\rightarrow 6B$, ***(6)*** Point the mouse to Traffic button $\rightarrow P$, ***(7)*** Mouse press and release the Traffic button to open the RRDTool images that contain the packet traffic information $\rightarrow 2B$. The estimated time to the above sequence is given by $\mathbf{R_{est} = 7M + 3P + 16B}$. Thus, it is expected that, by using MVN, Network Admistrators take $14.35s$ to deal the raised NMSit, by analyzing, in an integrated user interface, three RRDTool images that present information about packets received, transmitted, dropped, and with error. This result was corroborated by the experimental study in which $\mathbf{R_{exp} = 9.01s < R_{est}}$.

If the Network Administrator does not participate in the Mashment Ecosystem, he/she performs the following actions sequence to retrieve the information about the packet traffic on one switch from a specific controller web tool: ***(1)*** Point the mouse to Switches tab $\rightarrow P$, ***(2)*** Mouse press and release to select the Switches tab $\rightarrow 2B$, ***(3)*** Point the mouse to select a switch $\rightarrow P$, ***(4)*** Mouse press and release to select a switch $\rightarrow 2B$, ***(5)*** Point the mouse to Ports button $\rightarrow P$; and *(6)*

Mouse press and release to select ports of switch $\rightarrow 2B$. These actions must be repeated three times, one by each controller web tool. Therefore, without MVN the estimated time to retrieve non-integrated information about the packet traffic on three switches is: $\mathbf{R_{3s} = 3(6M + 6B + 3P) = 36s}$. Therefore, $\mathbf{R_{exp} < R_{est} < R_{3s}}$. In this sense, it is important to highlight that the retrieving time for MVN is significantly smaller (a 60% taking into account the estimated time) than for the non-MVN case. According this result, we can state that a Network Administrator in the Mashment Ecosystem can tackle a NMSit faster than one out of it.

V. Conclusions and future work

In this paper, we introduced a mashup ecosystem (Mashment Ecosystem) that allows to tackle network management situations (NMSit). The Mashment Ecosystem and its implementation are based on the high-level abstraction of NMR, WR, and OpR, the composition model of mashups, and an approach centered in the Network Administrator for building up of composite solutions. Our ecosystem empowers the Network Administrator with the important ability to rapidly create, launch, and publish Mashments that are mashups devised to collect, correlate, fuse, and present integrated information about a NMSit. We also presented experimental measured and KLM estimation of time that the Network Administrator take to: (*i*) create, launch, and publish MVN that is a Mashment aimed to address a specific NMSit: transmission errors in a heterogeneous virtual network; and (*ii*) retrieve, by using MVN, the integrated information about the referred NMSit.

The aforementioned NMSit has a particular challenge: it needs the fast development of a solution (MVN) able to retrieve, merge, and rapidly present, in an integrated way, network management information from different OpenFlow controllers and their underlying virtual network elements. The Mashment Ecosystem allowed the Network Administrator to overcome such a challenge, corroborating its significance and the relevance of Mashment concept. Through an experimental and KLM evaluation, we have confirmed, first, the short time that a Network Administrator takes to MVN: create (estimated=$56.25s$, experimental=67.24), launch (estimated=$6.65s$, experimental=$5.46s$), and publish (estimated=$23.60s$, experimental=21.92). Second, the short time that a Network Administrator, that is using MVN, takes to retrieve (estimated=$14.35s$, experimental=$9.01s$) the integrated information about the raised NMSit. The experimental evaluation confirmed KLM predictions and, consequently, the feasibility of using our ecosystem to tackle any NMSit. Furthermore, it is important to highlight that the time to retrieve non-integrated information about this NMSit, by using mistmached solutions, is $36s$. Thus, it is expected that a Network Administrator in the Mashment Ecosystem can tackle a NMSit 60% faster than one out of it.

As future work, we plan to propose and implement a Mashment dynamic composition model in order to tackle the NMSits more rapidly. Furthermore, we are interested in evaluating the productivity of Network Administrators participating in the Mashment Ecosystem. We also plan to implement the Mashment Marketplace to evaluate its feasibility. The acceptance of Mashments by Network Administrators is a topic to explore too.

Acknowledgment

The research of PhD(c) Caicedo is funded by two scholarships one PECPG of the CAPES (Brazil) and another one of the University of Cauca (Colombia).

References

[1] Z. Zhao, S. Bhattarai, J. Liu, and N. Crespi, "Mashup services to daily activities: end-user perspective in designing a consumer mashups," in *iiWAS '11*. New York, NY, USA: ACM, 2011, pp. 222–229.

[2] G. Jakobson, J. Buford, and L. Lewis, "Situation Management: Basic Concepts and Approaches," in *Information Fusion and Geographic Information Systems*, ser. Lecture Notes in Geoinformation and Cartography, W. Cartwright, G. Gartner, L. Meng, M. . Peterson, V. . Popovich, M. Schrenk, and K. . Korolenko, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. 2, pp. 18–33.

[3] G. Bader, W. He, A. Anjomshoaa, and A. Tjoa, "Proposing a context-aware enterprise mashup readiness assessment framework," *Information Technology and Management*, vol. 13, pp. 377–387, 2012.

[4] S. Tian, G. Weber, and C. Lutteroth, "A tuplespace event model for mashups," in *OzCHI '11*. New York, NY, USA: ACM, 2011, pp. 281–290.

[5] G. Jakobson, L. Lewis, C. Matheus, M. Kokar, and J. Buford, "Overview of situation management at sima 2005," in *MILCOM '05. IEEE*, 2005, pp. 1630 –1636 Vol. 3.

[6] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: Design, implementation and experience," *Parallel Computing*, vol. 30, p. 2004, 2003.

[7] W. Barth, *Nagios: System and Network Monitoring*, 2nd ed. San Francisco, CA, USA: No Starch Press, 2008.

[8] G. Pavlou, "On the evolution of management approaches, frameworks and protocols: A historical perspective," *J. Netw. Syst. Manage.*, vol. 15, no. 4, pp. 425–445, Dec. 2007.

[9] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–20.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[11] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, apr 2010.

[12] E. M. Maximilien, A. Ranabahu, and S. Tai, "Swashup: situational web applications mashups," in *OOPSLA '07*. New York, NY, USA: ACM, 2007, pp. 797–798.

[13] N. Ozkan and W. Abidin, "Investigation of mashups for managers," in *ISCIS 2009*, sept. 2009, pp. 622 –627.

[14] A. Majchrzak and P. H. B. More, "Emergency! web 2.0 to the rescue!" *Commun. ACM*, vol. 54, pp. 125–132, April 2011.

[15] E. Tosti and W. Smari, "Sensors integration in a grid-based architecture for emergency management systems," in *DEST '10*, April, pp. 435–442.

[16] J. Adams and C. Reynolds, "A complex situational management application employing expert systems," in *Systems, Man, and Cybernetics, 2000. IEEE*, vol. 3, 2000, pp. 1959 –1964 vol.3.

[17] R. Koelle and A. Tarter, "Towards a distributed situation management capability for sesar and nextgen," in *ICNS '12*, april 2012, pp. O6–1 –O6–12.

[18] B. Magoutas, G. Mentzas, and D. Apostolou, "Proactive situation management in the future internet: The case of the smart power grid," in *DEXA '11*, 29 2011-sept. 2 2011, pp. 267 –271.

[19] K. Huang, Y. Fan, and W. Tan, "An empirical study of programmable web: A network analysis on a service-mashup system," in *ICWS '12*, june 2012, pp. 552 –559.

[20] R. Latih, A. Patel, A. Zin, T. Yiqi, and S. Muhammad, "Whip: A framework for mashup development with block-based development approach," in *ICEEI '11*, july 2011, pp. 1 –6.

[21] D. Kieras, "Using the keystroke-level model to estimate execution times," in *University of Michigan*, 2001.