THESARD: on The road to resiliencE in SoftwAre-defined networking thRough self-Diagnosis

José Manuel Sánchez, Imen Grida Ben Yahia
Orange Labs
Paris, France

Noël Crespi
Institut-Mines Télécom, Télécom SudParis, CNRS UMR5157
Evry, France

***Abstract*—This demonstration presents THESARD, the implementation of a self-diagnosis platform for SDN based networks. This platform automates the diagnosis by building and updating on-the-fly the fault propagation model of a streaming application. Self-healing actions are also shown to illustrate the recovery process for both the SDN underlying network and the streaming application, once the root cause is identified via this model.**

I. Introduction

Software Defined Networking (SDN) promises flexibility and elasticity on services through programmability and network abstraction. However, two challenges arise in SDN: 1) the resilience of the SDN controller, which becomes a single point of failure, and 2) the dynamicity of the SDN infrastructure in terms of continuous changes in the forwarding flows, network topology and type of control (in-band and out-of-band). In this paper, we propose THESARD, a self-diagnosis platform able to cope with resiliency and dynamicity challenges in SDN.

The innovation of THESARD comes from two angles: 1) an automated generation of the fault propagation model as well as its update and 2) the identification of the root cause with finer granularity based on this generated model. The self-healing actions are also shown so to complete the feedback management loop.

The structure of the paper is as follows: section II details the THESARD architecture and section III details the goal of the demo, its implementation environment, scenario and the demonstrated uses cases.

II. THESARD Architecture

THESARD platform is part of the management plane of the SDN infrastructure. It is then technology-agnostic and independent from the type of SDN controller in use. Indeed, THESARD platform is independent from the southbound interface and obtains a global view of the network topology from the controller's northbound interface. THESARD builds on-the-fly and updates a fault propagation model and then identifies the root cause of service and network resources failures (with network component granularity) by exploiting this model. THESARD is composed of three blocks (Fig. 1 in yellow):

1) Wrapper and classifier block: an SDN application that receives the network topology from the SDN controller through its northbound interface in a JSON format and provides as output with a machine-readable file that contains the classified network elements: hosts, switches, logical ports, control links, access links and inter switch links.

2) Self-Modeling block: it takes this machine-readable file and generates the fault propagation model by instantiating and assembling the templates of the discovered network elements. The control links and the SDN controller are included in this fault propagation model.

3) Root Cause analysis block: it pinpoints the root cause at a finer-granularity (the faulty network element and its faulty inner component) by propagating a set of network observations through the generated fault propagation model.

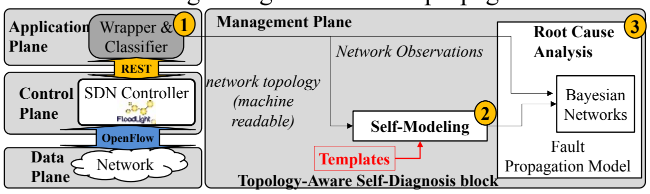

Figure 1. Simplified THESARD Architecture

III. Demonstration

THESARD platform ensures two levels of diagnosis based on two approaches: 1) a topology-aware self-diagnosis approach [1] to model and diagnose the dynamic software-defined infrastructure and 2) a service-aware self-diagnosis approach [2] to include on that model the overlying networking services utilizing virtual resources (VNFs and virtual links) allocated over the software-defined infrastructure, which was presented to the current NetSoft2016 edition. The THESARD platform in this demonstration enables the diagnosis of a streaming application and the underlying resources of the software-defined infrastructure involved in that streaming application. Indeed, THESARD completes the self-healing control-loop composed of detection, diagnosis, and recovery blocks to automatically recover the video streaming application and the SDN infrastructure, with recovery actions such as instantiating on-the-fly the SDN controller when is faulty and restoring faulty control and data links.

Steps of this demo: THESARD platform is based on three key steps, shown in Figure 2 in red:

Step 1: Transformation of the network topology into a machine-readable format containing the classified network elements

Step 2: On-the-fly construction and continuous update of the fault propagation model from the machine-readable format and running applications. This model contains the network nodes, their internal logical and physical components such as ports or running applications to ensure a fine-granular diagnosis.
Step 3: Root cause analysis with Bayesian networks to calculate the probability of faulty networked elements with component-level granularity by exploiting this generated fault propagation model. The eventual recovery is based on the root cause analysis.

Implementation Environment: The self-diagnosis framework integrates different open source software packages. The SDN controller is based on Floodlight [3] and the SDN infrastructure is emulated with Mininet [4]. The Bayesian Network algorithm is based on the Kevin Murphy's Bayesian Networks Toolbox [5], running in MATLAB [6]. We implemented the Graphical User Interface (GUI) in Python with the Qt software library [7]. The fault propagation model is visualized in 3-D with UbiGraph [8], which allows for visualizing the dynamic and interactive dependency graph encompassing the interactions among SDN resources and their components.

Demo scenario: The scenario of the demonstration is shown in Figure 2. A new client demands the video content to the streaming server (1), which starts sending it. However, for this content to reach the client, the SDN controller must install the necessary flows on the switches (2). The GUI monitors the current network topology provided by the SDN controller in a periodic basis and it classifies the network elements (3) into different nodes (hosts, switches, and controllers) and links. The self-modeling algorithm takes as input this list of classified network elements, instantiates their templates, and assembles them to generate the fault propagation model (4). This model remains stable unless there are topological changes, in which case the self-modeling block regenerates the fault propagation model by incorporating the new elements. Once a malfunction occurs, the root cause analysis block is triggered to correlate the alarm of a faulty streaming service with the state of the elements in the software-defined infrastructure and updates the fault propagation model with the root cause(s)(5), pinpointing the most probable root causes i.e. a network node and its internal component (CPU, port, card, application, etc.). Once the root cause is identified, the GUI suggests a recovery action (6) based on the root cause that will be validated by a human administrator (7) once is proved this action re-establishes the streaming service.

Demonstrated use cases: We demonstrate the three aforementioned steps with three use cases:
Case 1: Diagnosis and recovery of faults affecting several streaming clients as consequence of a faulty SDN controller application.
Case 2: diagnosis and recovery of faults affecting one streaming client such as consequence of a single faulty data link.
Case 3: update of the fault propagation model with new network nodes.

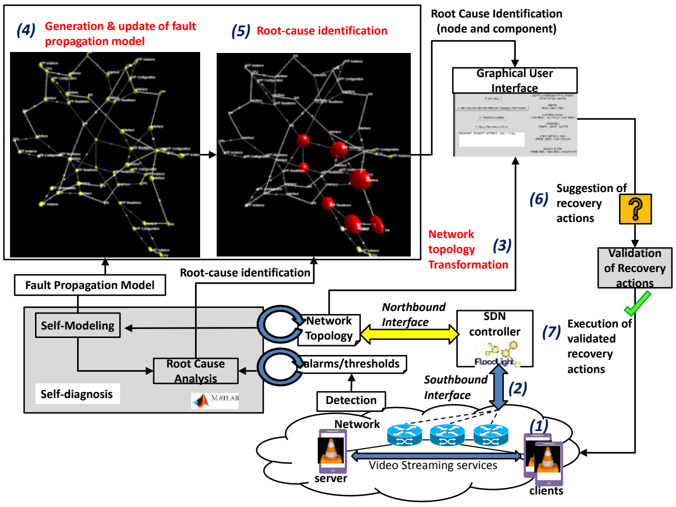

Figure 2. Demo Scenario of the THESARD platform

A preview of THESARD platform is available in [10], where the fault propagation is generated for different network topologies and several faults are generated and recovered by the self-healing system with a set of predefined recovery actions. This demo was exhibited at Orange Labs Research exhibition 2015, and it aroused the interested of many different actors such as third-parties, network operators, or vendors, as an enabler to ensure automated and intelligent resilience in SDN.

TECHNICAL REQUIREMENTS

One 30-40" TV screen (HDMI) and one 15-20" PC monitor (VGA).

REFERENCES

[1] J. Sanchez, I. Grida Ben Yahia, N. Crespi, "Self-Modeling Based Diagnosis of Software-Defined Networks," Workshop MISSION 2015 at 1st IEEE Conference on Network Softwarization, London, 13-17 April 2015.

[2] J. Sanchez, I. Grida Ben Yahia, N. Crespi, "Self-Modeling based Diagnosis of Services over Programmable Net-works," 2nd IEEE Conference on Network Softwarization, Seoul, Korea, 6-10 June 2016.

[3] "Floodlight OpenFlow Controller." [Online]. Available:http://www.projectfloodlight.org/floodlight/

[4] "Mininet: An Instant Virtual Network on your Laptop (or other PC)" [Online]. Available: http://mininet.org

[5] Kevin Murphy's Bayesian Networks Toolbox, MIT AI lab,200 Technology Square,Cambridge. Available at: http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html

[6] MATLAB Release 2013A, The MathWorks, Inc., Natick, Massachusetts, United States.

[7] "Qt software package for Python." [Online]. Available: http://www.qt.io/download/

[8] "UbiGraph 3-D graph representation tool." [Online]. Available: http://www.ubietylab.net/ubigraph/

[9] "VLC Media Player." [Online]. Available: http://www.videolan.org/vlc/

[10] THESARD plaftorm. Available: https://www.youtube.com/watch?v=xNudu48quRM