Network Troubleshooting with SDN-RADAR

Tigran Avanesov*, Gabriela Gheorghe*, Maria Rita Palattella*,
Miroslaw Kantor*, Ciprian Popoviciu† and Thomas Engel*
*SnT, Université du Luxembourg, Email: first-name.last-name@uni.lu
†Nephos6 Inc., Email: chip@nephos6.com

***Abstract*—Despite increasing deployment of software-defined networks (SDN), little is yet known about how to actually manage such kind of networks. When service degradation (generically termed here "trouble") happens, the first step is to localise underperforming network paths ("troubleshooting"). SDN-RADAR gives a novel approach to ease large network troubleshooting by leveraging SDN features and incorporating distributed monitoring of network traffic. The demo shows how the SDN-RADAR tool can help network administrators understand which is the most likely faulty network link.**

I. Introduction

The more complex and heterogeneous networks become, the harder the job of IT administrators. It is generally quite difficult to understand when network faults occur, how to diagnose and how to mitigate them. Faults typically expose themselves by delays or downtime in service delivery. Faced with such symptoms, the administrator has to use a flurry of different tools, specialised for different network protocols (HTTP, DNS, ICMP, UDP, etc). Such solutions can no longer keep up with the complexity of network protocol interactions, especially in the face of virtualization and the deployment of SDN technologies.

SDN technologies are already incorporated into operation [1], [2]. Therefore, it makes sense to investigate how SDN features can help network administrators to do their job. With SDN-RADAR we make the case that SDN delivers essential functionality for the human operator to understand and investigate network issues. SDN-RADAR is intended as an early warning system to show service degradation from a centralised network "dashboard". The novelty of the approach taken by SDN-RADAR is that it combines end-user service monitoring, with network configuration supplied by SDN devices.

This demo will present the SDN-RADAR proof of concept tool, that puts together state of the art SDN technologies (the OpenDaylight controller [3] (ODL), and Mininet [4] as a testing environment) with existing cutting edge monitoring of service quality-of-experience provided by Nephos6 [5]. Nephos6's Sonar tool implements the concept of distributed monitoring with its Sonar agents. These are scripts that collect key data needed in calculating synthetic user experience for specific services, and are dynamically managed by a controller (not necessarily in the sense of an SDN controller). To the best of our knowledge, this is the first demonstration of how SDN can be operationalised and can help optimise hybrid environments containing both SDN and traditional networks.

This publication is supported in part by the CoSDN project, INTER/POL-LUX/12/4434480, funded by the Fonds National de la Recherche Luxembourg. Partly it has also been supported by the GEN6 EU-project 297239.

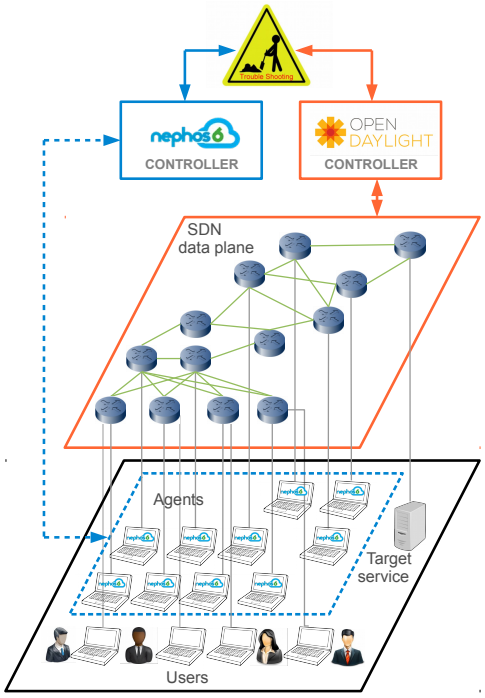

Fig. 1. SDN-RADAR architectural elements.

II. SDN-RADAR Design

SDN-RADAR supports network operations when a target service is offered to users at various locations. At all times, SDN-RADAR gets information about the end-users quality-of-experience, as well as updated information about the network topology. The former comes from a monitoring infrastructure that targets several service endpoints; the latter comes from an SDN controller that has a complete view over the network in its reach. We envision that the service-level-agreement, settled at the time of service provision setup, contains a specification of how much service degradation is acceptable by users in exchange for service access. For the moment, service degradation reflects into latency, but the variation of latency (i.e. jitter) and packet loss are also targeted with this approach. Based on such expectation, SDN-RADAR computes the differential experience from several locations, which can give an early warning of a network issue for users that access the service from a particular location. As a next step, SDN-RADAR localises the service degradation by inferring the actual path taken by test packets from the degradation location, and calculates the most likely underperforming segments on

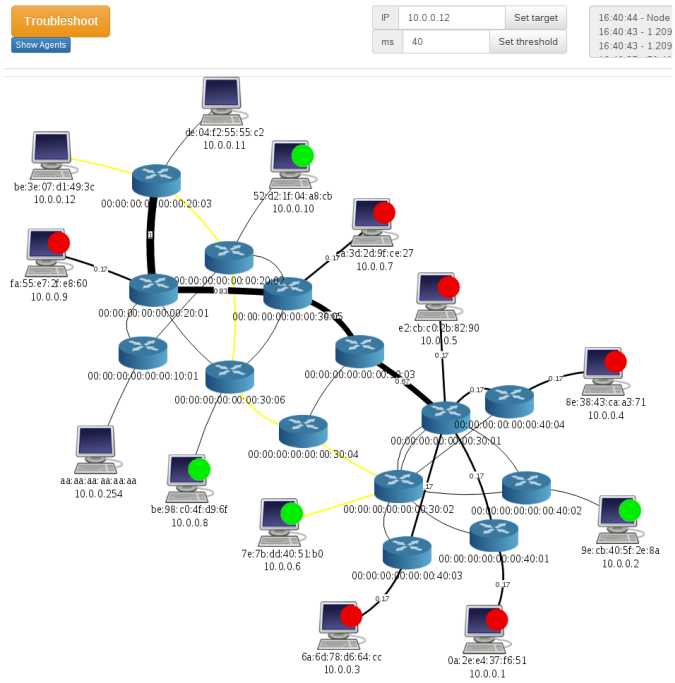

Fig. 2. Screenshot from the SDN-RADAR proof of concept indicating the most likely underperforming network links towards the target IP 10.0.0.12.

that path where the fault occurred. The SDN-RADAR architecture is shown in Figure 1 and consists of:

The target service is the set of infrastructure elements responsible for delivering the end service to users. In most cases the service delivering infrastructure consists of multiple servers, load-balancers, proxies, etc.

The measurement agent essentially represents a user of the target service, and it periodically measures context-related user experience over the target service.

The measurement controller coordinates the work of measurement agents and collects their reports. The measurement controller can request the measurement agents to monitor specific targets, what kind of measurements the agent(s) should perform and their periodicity. The reported measurement values are stored in the measurement controller database.

SDN Controller and SDN switches. The controller communicates with OpenFlow switches and exposes a unified interface to program OpenFlow switches.

The troubleshooting application is the software that takes into account the measurements reported by the measurement agents and any other features provided by SDN controller to identify problematic links in the network. The results can be displayed to the network administrators allowing them to take further measures for improving the network performance.

III. SDN-RADAR - Proof of Concept and Demo

The SDN-RADAR proof of concept brings together cutting edge SDN technology (the OpenDaylight controller, the Mininet environment that is OpenFlow-enabled) with the Sonar quality assurance tool (Sonar agents and the measurement controller). On top of these technologies, this approach gathers network topology information and infers path extraction:

Topology discovery is a building block network service that is typical for SDN controllers. These controllers (of which OpenDaylight is one) contain modules to discover and monitor network topology. Topology discovery typically includes the discovery of the links and of the hosts.

Path extraction. The proof of concept assumes that the routing rules are installed by the default sample routing application of the OpenDaylight controller in the Hydrogen release. For every host H in the network, the default SimpleForwarding routing app installs flows that forward packets to the next switch in a shortest path to host H based on the destination IP address. Knowing this template allowed us to deduce the paths of the packets. For determining the path of a packet from host A to host B, first all flows of the switch connected to A are examined. When a rule with the destination IP of B is found, the output port of this rule is known. The OpenDaylight topology contains the link connected to that particular output port, hence it is known which is the next switch and its corresponding port where the packet will arrive next. By continuing this procedure until reaching host B, the actual path from A to B is inferred.

Demonstration. We deployed our system in several virtual machines (VM) in a local OpenStack [6] cluster. For simulating a network of OpenFlow [7] switches we used Mininet [4], a simulated network environment running in a VM. On another VM, we implemented the Hydrogen release of the OpenDaylight [3] controller, one of the most advanced SDN open-source SDN controllers with a wide industrial support. The monitoring was performed by Sonar measurement agents.

The troubleshooting application can run in any VM that has access to both Sonar and SDN controllers. In our demo, this application runs on a laptop connected to the University network in order to access OpenDaylight. Figure 2 shows the SDN-RADAR graphical user interface (GUI), able to: (i) infer the topology and identifying the measurement agents, (ii) allow for the target service selection, and (iii) obtain measurements and show the reaction when the degradation threshold is set up. The purpose of the demo is to show how SDN-RADAR proceeds to localise a network problem, once service degradation is detected. Network problems will be artificially injected in the Mininet setup, e.g., by adding delays on a link with tc tool. The troubleshooting procedure can successfully infer on which path the service degradation happened. In Figure 2, red circles indicate the hosts with measurement agent software running that report degradation. In this way, SDN-RADAR provides to network administrators an efficient tool for detecting service degradation, and performing troubleshooting in a scalable way.

References

[1] Telefónica PressOffice, http://pressoffice.telefonica.com/documentos/nprensa/20011013_TEF_NEC_vCPE_over_NFV.pdf, 2014.

[2] Huawei, http://pr.huawei.com/en/news/hw-391065-sdn.htm, 2014.

[3] "OpenDaylight project," http://www.opendaylight.org.

[4] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010.

[5] Nephos6 Inc., "Sonar Solution Overview," http://www.nephos6.com/pdf/Sonar-Brochure.pdf, 2014.

[6] "Openstack cloud platform," https://www.openstack.org/, 2014.

[7] "OpenFlow Switch Specification (version 1.1.0)," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf, Feb. 2011.