SAKURA a Model Based Root Cause Analysis Framework for vIMS

Sihem Cherrared, Sofiane Imadali, Eric Fabre, Gregor Gössler

POSTER: SAKURA a Model Based Root Cause Analysis Framework for vIMS

Sihem Cherrared[1,2], Sofiane Imadali[1], Eric Fabre[2], and Gregor Gössler[3]

[1]Orange Lab Networks, 92320 Chatillon, France
[2]INRIA, Campus de Beaulieu. F-35042 Rennes, France
[3]Univ. Grenoble Alpes, INRIA, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

***Abstract*—Model based machine learning (MBML) techniques solve novel diagnosis problems and provide explanations for their decisions. However, current MBMLs suffer some limitations, since virtualization of network brings new challenges such as the dynamic topology and elasticity. Those limitations include the high dependency on previous knowledge and the difficulty to represent the model. To face those limitations, we propose SAKURA: a root cause analysis framework for the virtual Ip Multimedia Subsystem (vIMS). SAKURA is composed of a self-modeling and a constraints solver algorithm.**

***Index Terms*—Service Function Chain, Model-based machine learning, self-modeling, root cause analysis.**

I. MOTIVATION AND APPROACH

Virtualization of networks is considered to be a driving force and basis of the design process for both computing and networking 5G architecture. Specifically, Network Function Virtualization (NFV) facilitates dynamic creation, adaptation and termination of networking services [1]. Those services are chained or connected to each other to achieve an end-to-end 5G service called a Service Function Chain (SFC). Many open source projects are involved in the design of virtual network functions. Metaswitch proposed Clearwater Virtual Ip Multimedia Subsystem (vIMS) a virtual version of the IMS IETF and 3GPP standard for Voice over IP (VoIP) for 4G and 5G [2]. Clearwater includes most of the infrastructure and service specifications that we can identify in a virtual network environment. In our work we consider the docker version of Clearwater, which provides a lightweight encapsulation to maintain and update each component independently [3].

The fault management of virtual networks needs to be addressed at an early stage to take full benefits of the upcoming NFV architectures. Fault management includes detection, root cause localization, and healing of abnormal states of a network. This requires to have an accurate knowledge of the entire network components and services [4]. MBML techniques, such as probabilistic graphical models, are considered as white box solutions mostly applied in the Root Cause Analysis (RCA) process [5]. MBMLs reason about a system from an explicit representation of its structure and functional behaviour. MBMLs solve novel problems and provide explanations for their decisions. The model can be constructed with different node types: network resources, alarms and events. However, MBMLs suffer some limitations, such as a high dependency on previous knowledge and some difficulties to represent the model, particularly virtual networks bring more challenges in deriving such models. Including the difficulty to reason in a huge dynamic model with several network layers and segments due to the physical and virtual dependencies, the lack of network visibility, and the dynamic network topology.

SAKURA is a model based RCA framework for vIMS. SAKURA framework is based on a self-modeling and RCA algorithm. The self-modeling algorithm apply a model defined for vIMS. The proposed model is a dependency graph constituted of Boolean variables linked with logical dependencies. The model is a multi-layer and a multi-resolution graph that represents the constraints between network components from the physical to the application and service level.

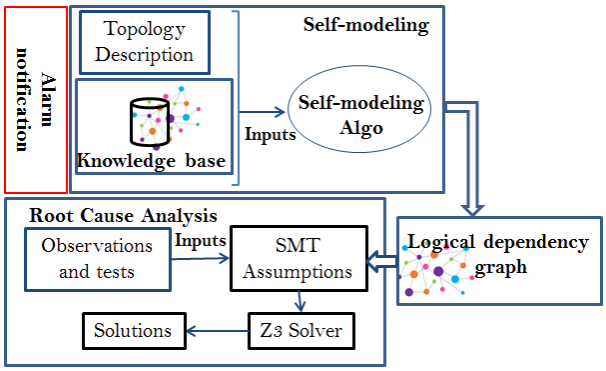

Figure 1. SAKURA RCA Framework.

Our main motivation in deriving such a model regarding the number of challenges introduced by the virtual networks is the existence of two types of knowledge: *acquired* and *learned* knowledge. Acquired knowledge represents the knowledge about the service components requirements and capabilities that we can obtain from description languages and standards such as TOSCA [6], expert operators or network protocols. It also represents the available observations in a running environment such as the deployed services, the virtual and physical hosting environment and components status. While, the *learned knowledge* is acquired by tests and faults injection. The faults are performed to obtain information about the health of network components and learn the components dependencies.

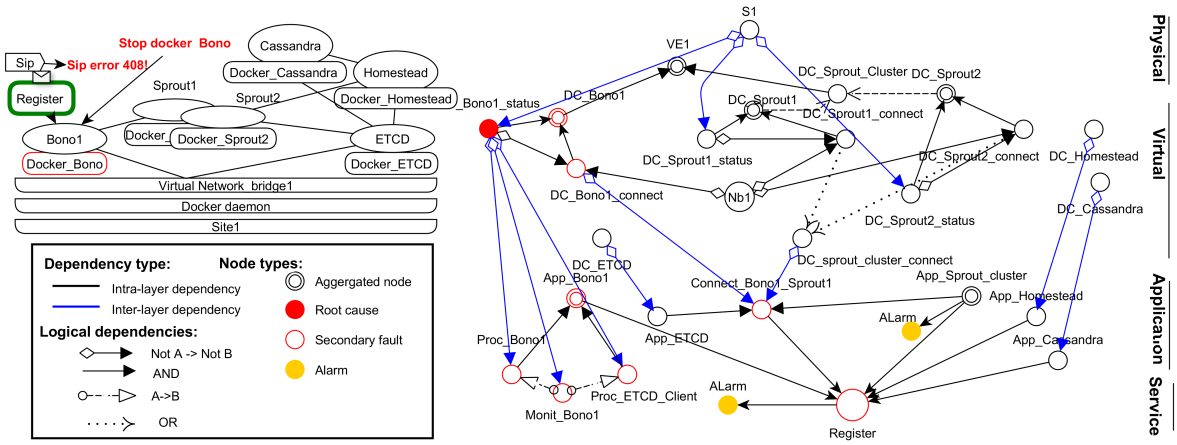

Figure 2. SAKURA RCA Fault propagation use case.

II. SAKURA RCA

Figure 1 depicts the different steps of SAKURA framework. As inputs to our framework, we consider a topology description file, a knowledge base and observations and tests described in the following:

- Topology description: The real time network topology is described in a YAML file [7]. The network topology information could be retrieved by interrogating the orchestration platforms or open source monitoring tools.
- Knowledge base: contains the description language of the vIMS model. The knowledge base is constructed by the acquired and learned knowledge. The dependencies are modeled for each vIMS layer (i.e physical, virtual, application and service), and are divided into two types inter and intra layer. The defined model is a multi-layer and a multi-resolution logical model extendable to a probabilistic model. It includes the description of dependencies in a case of a presence of an auto-recovery mechanism, and the case of elasticity of functions (i.e the number of deployed dockers for the same virtual function). Note that the defined model is generic in the sense that it describes the different types of network resources involved, how they depend on each other and how they interact. This generic patterns are then instantiated according to the current network topology.
- Observations: represent the alarms and logs of Clearwater components that we collect from each network function, the real-time status of components and the network health reports collected from the Clearwater live-test [8] [9].

The main procedures of SAKURA framework are the self-modeling and the RCA algorithm:

1) Self-modeling: represents the action of generating automatically a model. The dependency graph model is constructed based on both the topology description and the knowledge base. The self-modeling procedure is called in each failure detection to create the model that corresponds to the current network topology to launch the RCA procedure.

2) The rca algorithm: takes into consideration the given observations about the network and the model generated by the self-modeling procedure to retrieve a number of constraints in a SMT file and resolves them using the Z3 solver [10]. More observations can be added with tests.

III. SAKURA Fault propagation use case

Figure 2 depicts a fault propagation use case. The dependency model is generated through the self-modeling algorithm. In this use case, only the BONO application details are provided, the other functions are represented as aggregated variables. To illustrate a fault propagation scenario, we injected a fault in the proxy BONO docker. The sip register injected traffic returned two types of alarms: the sip protocol code error and an alarm in the docker SPROUT connected to BONO. As represented in the defined model, the fault started in the virtual layer from the faulty component docker BONO and then propagated to other non-faulty components via inter and intra layers interactions.

IV. Conclusion and Future work

We proposed SAKURA a model based RCA framework for vIMS. SAKURA models the multi-layers of virtual networks and includes the auto-recovery and elasticity aspects. In our future work we propose to release the logical dependencies to a probabilistic model and develop further the RCA module.

References

[1] M. Huang *et al.*, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, 2018.

[2] "Clearwater project," http://www.projectclearwater.org/., [Online; accessed 16/04/2019].

[3] C. Anderson, "Docker [software engineering]," *IEEE Software*, vol. 32, no. 3, pp. 102–c3, May 2015.

[4] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Jun 2018.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[6] "Tosca," https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca, [Online; accessed 16/04/2019].

[7] "YAML," https://yaml.org/., [Online; accessed 11/04/2019].

[8] S. Cherrared *et al.*, "Lumen: A global fault management framework for network virtualization environments," in *ICIN*, 2018.

[9] "Clearwater-live-test," https://github.com/Metaswitch/clearwater-live-test, [Online; accessed 16/04/2019].

[10] "Z3 Solver," https://github.com/Z3Prover/z3, [Online; accessed 17/04/2019].