A framework to facilitate management of services in cloud based 5G environments

Nikos Koutsouris, Apostolos Voulkidis and Kostas Tsagkaris
WINGS ICT SOLUTIONS
336 Syggrou Avenue, 17673 Athens, Greece
{nkouts,avoulkidis,ktsagk}@wings-ict-solutions.eu

***Abstract*—The main target of 5G network technologies is to offer radically increased user capacity and quality of service, while saving energy and reducing investment and operating costs. A key technology in this pursue is the cloudification of the virtualized network functions, which can eventually create an environment where services for network operators and application / content providers are chained, configured, deployed and orchestrated as in a common plane. The ARCADIA framework provides the necessary functionalities for facilitating the development and management of component services that can fully exploit the underlying programmable 5G infrastructure. ARCADIA supports policy definition even at the developer level, as well as unified monitoring and dynamic reconfiguration of service parameters.**

Keywords—Highly distributed applications, micro-service, cloud, unikernel, virtualization, devops, reconfiguration, SDN, NFV, annotations.

I. INTRODUCTION

The future 5G network environments will comprise a plethora of interconnected objects, from sensors and devices with multiple radio interfaces to vehicles and smart home equipment. The 5G infrastructure will be able to support service provision to a huge number of users, the great majority of which will be non-human users. This requires a significant change in the deployment, coordination and management of networks and relevant processes. Network Operators have started to prepare for a transition to a new era, where networks are software defined and network functions are virtualized, so as to be able to satisfy the radically increased traffic demand while ensuring QoS and QoE levels, and minimizing capital and operational costs.

The exploitation of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies adds flexibility and at the same time complexity to the 5G networks, which can be compensated by the use of a suitable framework for the management and orchestration of the virtual functions, virtual appliances, software nodes and every other softwarized network resource. The ARCADIA framework, which is developed in the ARCADIA project [1] and is funded by the H2020 EU programme, is a novel software development paradigm that enables the chaining, configuration, deployment and orchestration of software components in a smart and dynamic way. The proposed framework addresses the challenge of managing and orchestrating virtualized 5G functions. ARCADIA is driven by the Service Oriented Architecture concepts [5] and its main novelty resides in the introduction and implementation of the Smart Controller, whose functionalities can ensure the trustworthy chaining and configuration of components, based on an extensible context model that describes requirements and available options.

More information on the role and the modules of the Smart Controller is provided in section IV D. Beforehand, sections II and III highlight in brief the main concepts and technologies behind ARCADIA, while sections IV A, B and C introduce the basic parts of the ARCADIA framework. Finally there are some conclusions on how the work should evolve in the mid and the long term. It has to be noted that this paper presents the current work in progress in the context of the ARCADIA project and all the described concepts will be elaborated and validated in a set of selected use cases before the first official release of the ARCADIA framework.

II. VIRTUALIZATION AND CLOUDIFICATION

Virtualization enables the optimized utilization of resources, as more applications and services can be packed onto the infrastructure. On the other hand, cloud computing offers through a broad network access, a pool of resources that can be assigned dynamically and on demand, while their usage can be monitored, controlled and optimized. To fully exploit the merits deriving from a virtualised cloud environment, it is required to go further than just porting applications and services from running on bare metal to running on Virtual Machines (VMs) Technologies such as containers and unikernels allow better resource and service management by further exploiting the concept of autonomous applications and micro-services. Unikernels are highly optimised, specialised machine images constructed by only using the minimum required set of operating system libraries to run an application. Their small footprint is an important feature for a cloud application as it reduces the cost of the deployment by using only minimal resources and increases the security of the application by shrinking the attack surface. Moreover, the lack of unnecessary operating system libraries allows unikernels to boot extremely fast making them ideal for mission critical and highly available applications.

The softwarization of the network functions should not be done on a monolithic basis, e.g. by just creating one virtualized building block for all the functions of a network element. More recent software design paradigms, used currently for cloud applications, such as the micro-service architecture, have to be exploited, so that the resulting network applications can be

scaled up or down in a matter of seconds without extreme differences in cost. In this case, every VNF can be seen as a micro-service, and the various network operations are carried out by the corresponding network applications that are built by chaining the necessary VNFs. Modular applications consisting of several stateless micro-services are perfect for scaling operations and cost-effective deployment due to their autonomous nature. Stateless micro-services are by design horizontally scalable since by using load balancers, more instances of said services can be deployed without the need of heavy reconfiguration. Such load balancers could be either actual bare metal machines or even deployed Virtual Functions (VF) controlled by an appropriate orchestrator. In addition, by separating data from functionality services are more agile and fault tolerant which is an important requirement for highly distributed and highly available network applications.

III. Management and Orchestration

SDN and NFV-based networks and services will be actually large software systems, with thousands software modules distributed at tenths or even hundreds of different hardware machines. The various network functionalities will be actually network applications, namely pure software. Network operation based on VNFs implemented as micro-services that are communicating and interacting in a cloud environment is a complex task that is getting harder and harder. The related network applications should be Reconfigurable-by-Design, infrastructure independent and at the same time, resilient to failures and easily scalable.

To overcome all the aforementioned difficulties, solutions and tools originating from the world of software design and development should be adopted. Moreover, management tools that are trying to simplify the deployment and scaling process by automating different aspects of the work-flow should be embraced. Service modeling tools, like Juju [4], enable developers and IT professionals to automate mundane tasks and reduce workloads, by undertaking a big part of the deployment process on a private or public cloud. Developers can use such tools to create the blueprint of their application called "service graph", where they can define how micro-services are interacting with each other and have an overview of their application data-flow. Moreover, DevOps environments are getting more and more difficult to manage due to the increasing amount of fragmentation between infrastructure providers.

Deploying a complex, micro-service based, network application on top of different hardware infrastructure leads to service modeling issues, different network requirements and configurations as well as different policies. New development paradigms are trying to tackle such issues by leveraging the power of SDN and NFV. Network orchestration should be performed in an automatic way, based on policies defined throughout the network application life-cycle. It is important to support policy creation at all levels, starting from the developer, continuing to the provider of a service and the provider of the infrastructure, and reaching the final user of an application.

IV. The ARCADIA Framework

The ARCADIA framework is a Novel Reconfigurable-By-Design Highly Distributed Applications (HDA) Development Paradigm. It takes care of multi-infrastructure deployment, high availability requirements and automatic real-time reconfiguration of applications. To tackle such tasks, ARCADIA applications are based on a micro-service model and are governed by a sophisticated policy manager. In other words, each ARCADIA application consists of several autonomous components, which can communicate with each other based on a service graph defined by the developer and policy rules defined throughout their life-cycle. Each component can be stored in a public or private registry on the ARCADIA platform and it can be re-used by other applications. As a framework for software development, ARCADIA can satisfy the needs of the transition to the 5G networks and can support the implementation of the new 5G network architectures. Particular focus is given to the management of networks and services through the introduction of the Smart Controller, a module that has the most significant role in the framework, as it is responsible for orchestrating the numerous modules that have to cooperate, such as the unikernel generator, the deployment manager, the policy manager etc.

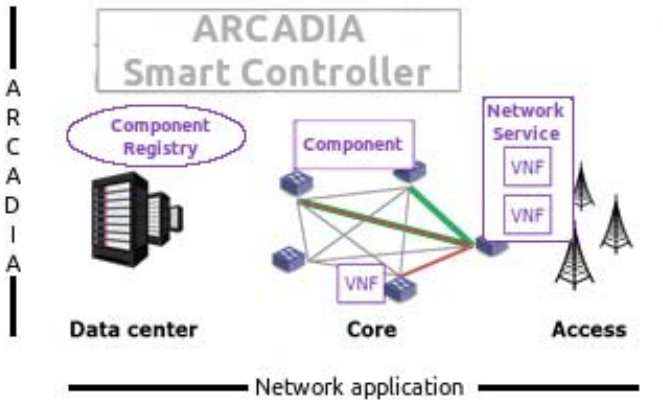

Fig. 1. The ARCADIA framework

Network Operators or more specifically, VNF developers, will create and push the necessary 5G network functions as components to the ARCADIA registry, where they can publish them with public or private access. They can use all publicly available components to create a service graph for their network application through an innovative web-based user interface. The deployment module will ensure the smooth interaction with the underlying Software Defined Network on top of different infrastructure according to the policies defined by the developers and the operator.

A. The ARCADIA components

Following the micro-services design pattern, an application should be a combination of smaller, independent and autonomous modules called components. In addition, most modern applications need to be agile and reactive to changes in load or other aspects that affect the performance of the whole application. To achieve that, components should be stateless by design and highly tested in unusual scenarios like for example

a sudden spike in connected clients. By discarding state, a management tool like ARCADIA's Smart Controller can horizontally scale out (or in) only the affected part of the application by starting (or stopping) component instances and configuring required load balancers in front of them based on defined policies and thus reducing operation costs and resources wasting while maintaining quality of experience for the end users. Moreover, to fully exploit the virtualized environment, ARCADIA components can either be fully-featured virtual machines or unikernels, which are a special type of virtual machines that contain only the minimum required operation system libraries to run the component. Native and/or Docker-powered containers will be available in the next iterations of the framework.

ARCADIA components must follow and implement some features that will allow them to take part in the ARCADIA ecosystem. Every component based on the framework must be orchestrable and governable, or in other words, able for providing a way of reconfiguring it without having to re-instantiate it. Another important feature is the ability to be monitored by providing specific and meaningful metrics to the framework such as connected clients or cpu usage. Finally, components must be able to provide a way to communicate with other components by exposing chainable binding interfaces. Managing such features is a complicating task due to the different requirements and varied interfaces each component has. To tackle these issues, ARCADIA wraps components with a thin interfacing layer, based on a context model and supported by JAVA annotations, that is responsible for exposing component interfaces and metrics and providing real time configuration options to the Smart Controller.

To create a component, developers can transform their legacy applications by using specific JAVA annotations during development. JAVA annotations are used to provide meta-data to a java application, without affecting the execution of the application itself, although they can be used for that as well. They are pre-defined words preceded by the "@" symbol and they can be written in many different parts of the code depending on their configuration, for example whether they annotate methods, classes, fields, etc. Annotations are used during three stages of the application life-cycle determined by their defined retention policy; before compilation, during build time or on runtime. Most of the natively supported annotations are discarded during compilation stage, however, ARCADIA annotations are configured to stay past that stage and during runtime. Using the Reflection API, provided by JAVA, other applications such as ARCADIA Smart Controller can read those annotations and give instructions to the application.

Validation of the correct usage of the annotations and exposed interfaces is done in two steps; the first one is before the final component submission by using the provided ARCADIA plugin for the Eclipse Che which is an open source web-based IDE. The second validation is executed by the smart controller before generating the actual VM or unikernel that is saved in the component repository. Each component is bundled with a separate process called Agent that is the main communication tunnel between the component and the Smart Controller, which sends commands for orchestration.

B. Service Graphs

In order to coordinate how different components are communicating with each other, developers or service providers must define the high level blueprint of the application called service graph. There are two kinds of service graphs in ARCADIA; one that defines component communication and dependencies before deployment called "static service graph" and a second that represents not only the actual deployed components but also the administrative components that are required and managed by the framework like virtual routers and load balancers called "grounded service graph". In order to compose static service graphs, ARCADIA offers an innovative web-based drag and drop graph editor where service developers can choose which components their application requires and connect them based on their respective exposed interfaces. The editor does not allow connections between components that they can't have meaningful communication according to their interfaces. After the creation of the service graph, it is validated in order to follow ARCADIA paradigm and if the validation is successful it is saved in a graph repository available to be used by other users based on its permissions. On the other hand, grounded service graphs are generated and managed by the smart controller based on the static graphs. They are dynamic and they reflect the real-time topology of the application, for example in case of a horizontal scaling out based on defined policies, the new instances of the scaled component will be shown in the grounded graph. Moreover, service developers can define service graph-wide (or application-wide) metrics that will be helpful for having end-to-end policies and reconfiguration.

C. Policies

ARCADIA offers a robust policy editor and enforcer based on "If This Then That" philosophy where service or infrastructure providers can add per component and per application policies. In addition to the metrics advertised by components and service graphs, policy editor has also access to the infrastructure metrics provided by the interfacing layer, like for example cpu usage and memory capacity. In addition, an extended context model will allow developers to use policies through annotations in the source code of their component. The policy manager needs to validate all the defined rules and check for conflicts and that is a very complicated task especially considering the multiple infrastructure nature of the framework and the different policies infrastructure providers may have. Policies are taken into consideration by the optimization module of the Smart controller which offers both pro-active optimization, realized before deployment, and re-active optimization, which is executed during runtime and throughout the application life-cycle.

D. The role of the Smart Controller

The Smart Controller is the most sophisticated module of the framework. It contains many sub-modules that are important for different periods of the applications life-cycle starting from the development to monitoring and reconfiguration. During development, Smart Controller, in cooperation with other tools provided by the framework, is responsible for validating the usage of annotation and reporting

any logical and syntactic errors. After that, the component is pushed to the ARCADIA registry where the Smart Controller performs another series of validations and prepares it for bundling with an agent and unikernel generation. Moreover, before deploying the component, Smart Controller is searching and deploying any required dependencies. Smart Controller is infrastructure agnostic and can deploy components on different infrastructure providers according to the defined policies. In addition, by monitoring the components during runtime, Smart Controller is responsible for scaling and reconfiguring the application in real time with complex optimization algorithms.

E. Deployment

ARCADIA is a development paradigm that embraces Highly Distributed Cloud Applications and one of the most important requirement of such applications is the ability to deploy and orchestrate them across multiple infrastructure providers or domains and sites owned by one network operator. Currently, ARCADIA only supports infrastructures managed by OpenStack and other implementations based on the OpenStack API, however, adapters for Amazon Web Services API and Google Compute Engine API will be provided upon the ending of the project. To allow deployment and orchestration over different data centers and network domains, ARCADIA leverages the power of SDN and NFV. OpenOverlayRouters instances are deployed as components on the different domains and allow for better control over both management and data plane of the application. In addition, by natively using virtual networks the framework is able to facilitate Service Function chaining and Virtualized Functions for 5G compatible applications.

When the virtual network has been setup, the deployment continues with instantiating the individual components that are part of the service graph. Deployment of micro-service based applications is not straightforward as it requires careful orchestration flow in order to avoid racing condition problems and missing dependencies. To do so, virtual machines or unikernels are injected with the ARCADIA agent which is responsible for communicating with the Smart Controller and its specific modules and for enforcing the required deployment orchestration protocol. Each component is pushed asynchronously to the different infrastructure selected for the execution, according to the defined policies. When all the components have been instantiated Execution manager which is a module of the Smart Controller responsible for managing the service graph takes over and realizes the deployment plan. During that phase, components are instantiating and appear in a "deployed" state waiting for dependencies resolution and any required endpoint chaining. Finally, a synchronous message is sent by the Execution manager to change the state to "Started". Meanwhile, the ARCADIA agent keeps listening for messages from the modules of the Smart Controller through the application life-cycle. In addition, Execution manager is responsible for advertising the monitoring streams and metrics and provide details about the status of the components to the Smart Controller. In case of a required change of the service graph either due to policies or component failure, the execution manager will request a new deployment plan and push it to the components through the ARCADIA agents. Component to execution manager communication is asynchronous through a pub/sub system while Execution manager to component is always synchronous.

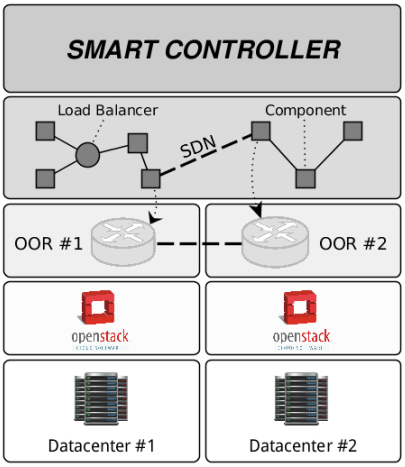

Fig. 2. The ARCADIA ecosystem deployed

F. Real time reconfiguration and monitoring

As we defined in the previous sections, ARCADIA components must be orchestrable and reconfigurable by design though the thin interfacing layer provided by the framework. The Smart Controller can scale specific components according on internal metrics (cpu usage, memory usage) or metrics published explicitly by components (total-connected-users, latency etc.) based on thresholds defined as policies by the developer or the infrastructure provider. Scaling is performed by redirecting the virtualized network flow through virtual load balancers in front of "scaled" components. Moreover, components can provide endpoints for reconfiguration through ARCADIA annotations. Throughout the application lifecycle, smart controller can change those exposed parameters according to optimization algorithms and defined policies.

CONCLUSIONS

In the remaining duration of the project, the developed functionalities of the Smart Controller will be tested and evaluated. In addition they will be enriched with knowledge building capabilities so as to further improve their performance. The Policy Management and Service Chaining parts will also be finalized and a fully functional release is planned to be made available for download by the end of 2017.

REFERENCES

[1] The ARCADIA Horizon 2020 Project, http://arcadia-framework.eu/

[2] ARCADIA Project deliverables D2.3 - Description of the ARCADIA Framework and D2.2 – Definition of the ARCADIA Context Model, http://www.arcadia-framework.eu/wp/documentation/deliverables/

[3] Eclipse Che Next-Generation IDE, http://www.eclipse.org/che/

[4] Juju Orchestrator by Canonical Ltd, https://jujucharms.com/about

[5] M.P. Papazoglou, Service-oriented computing: concepts, characteristics and directions, in proc. of the 4th Intl Conf. on Web Information Systems Engineering (WISE), 2003