Managing the Availability of VNFs with the Availability Management Framework

Pradheba C. Rangarajan, Ferhat Khendek
Electrical and Computer Engineering
Concordia University,
Montreal, Canada
{p_rangar, khendek@encs.concordia.ca}

Maria Toeroe
Ericsson
Montreal, Canada
Maria.Toeroe@ericsson.com

***Abstract*—A Virtualized Network Function (VNF) is deployed as a cluster of VMs in the Network Functions Virtualization Infrastructure (NFVI). As in traditional telecom, VNFs are expected to provide the required functions at the requested level of availability. For this, one has to incorporate proper redundancy, and appropriate recovery and coordination mechanisms among the redundant entities. The Service Availability Forum (SA Forum) has standardized such mechanisms into a set of middleware services. Among them, the Availability Management Framework (AMF) has the responsibility of managing the availability of application services based on a configuration, called AMF configuration. We propose the use of AMF to manage the availability of the services provided by VNFs. For this, we map the AMF concepts to the Network Function Virtualization (NFV) domain and propose a method for the generation of AMF configurations for AMF managed VNFs. The approach generates an AMF configuration that meets the required level of availability of the requested service workload while aims at maximizing the resource utilization.**

Keywords—Network Function Virtualization (NFV), Virtual Network Function (VNF), Availability, Availability Management Framework (AMF), Configuration, Generation

I. INTRODUCTION

With the growing popularity of Network Function Virtualization (NFV) [1], the trend of deploying Network Services (NS) using the NFV architecture is increasing. Its goal is to automate the deployment and management of NSs in a seamless manner. To provide a desired NS one or more Virtualized Network Functions (VNFs) [1] may be chained together. Each VNF is deployed as a cluster of VMs in the Network Functions Virtualization Infrastructure (NFVI) [1]. As in traditional telecom, VNFs are expected to provide the required functions at the requested level of availability. The design of such VNFs is a challenging task. One has to incorporate proper redundancy and appropriate recovery mechanisms. More importantly, in the event of failure, coordination among the redundant entities is vital to ensure service continuity. The Service Availability Forum (SA Forum) [3] abstracts the different availability mechanisms (i.e. managing the life cycle of application component instances, coordinating the redundant entities and executing the appropriate recovery mechanisms in the event of failure) into the Availability Management Framework (AMF) [4]. To manage the availability of some services such as of a network function implemented as a VNF, AMF requires some information like the number of software entities providing those services, their relations and distribution, provided as a configuration – called the AMF configuration [4].

In this paper, we propose the use of AMF as middleware to manage the availability of the services provided by VNFs. Accordingly, we propose a method for the generation of AMF configurations for such VNFs. In this method, we map the NFV concepts to the AMF domain and design VNF Components (VNFC) [2] for a VNF. We calculate the number of AMF entities as well as the number of VNFC Instances (VNFCI) for each VNFC from both availability and resource utilization perspective to meet the requested level of availability for each requested service and to deploy the VNF using minimum number of physical hosts, i.e. optimize resource utilization. The availability of a service depends on the availability of all the entities involved in providing the service and the interferences caused by the collocation of entities (for example collocated virtual machines (VMs)). To minimize such interferences, the entities can be grouped into fault isolation units (for example VMs) in different ways, which in turn may increase the resources (e.g. number of physical hosts) needed. Therefore, the challenge is to calculate the number and the arrangement of entities that will meet both availability and resource goals. The configuration and deployment of the VNF in an NFVI, the number of VNFCIs, their anti-affinity relationship can be extracted from the generated AMF configurations.

Before describing the mapping and the method we first provide some background in the next section.

II. BACKGROUND

A. Network Function Virtualization

NFV is changing the way NSs are being designed, deployed and managed [1]. It leverages virtualization and cloud technologies to roll out NSs faster as opposed to traditional networks. For this purpose, NFV unveils a new set of concepts called VNFs, NFVI and NFV Management and Orchestration (NFV-MANO) [1]. VNFs are network functions that have been virtualized and can run over shared compute, storage and networks in a NFVI [2]. The NFVI encompasses heterogeneous

978-3-901882-98-2 (c) 2017 IFIP

physical hardware, software and networking elements necessary to run VNFs [1]. NFV-MANO is responsible for managing the life cycle of the NS and its constituent VNFs [6].

A VNF may consist of a single or several software components that collaborate to provide the network function. These software components are referred to as VNF Components (VNFCs) [2]. A VNFC Instance (VNFCI) represents the runtime instantiation of a VNFC [2]. A VNF may instantiate more than one VNFCI of each VNFC. Each VNFCI runs in a VM of the NFVI and a VM may host only one VNFCI.

B. Availability Management Framework (AMF)

An AMF configuration consists of two groups of entities: the service provider entities and the service entities [4]. The service providers include Components, Service Units (SU), Service Groups (SG) and AMF nodes, whereas the service entities include Component Service Instances (CSI) and Service Instances (SI). The information about the service provider entities, their collocation relations are described in the AMF configuration [4].

For AMF a component may represent a hardware or a software resource that provides some service [4]. It is the smallest building block for an AMF application. It is also the smallest fault-zone within an AMF managed system [4]. A component starts providing a service only when a workload is assigned to it. AMF abstracts the workload unit as CSIs [4]. Components may collaborate to provide a desired service functionality. For these reasons, components working together to provide a service are grouped into logical unit called SUs [4]. Accordingly, from the service side, CSIs are composed into higher level workload units called SIs. At run-time AMF assigns SIs to the SUs [4] by assigning the CSIs of each SI to the components of an SU. This tight collaboration of components may allow faults to propagate easily [7]. The SU is the next fault-zone identified by AMF that can be isolated and repaired independently [4].

To protect the service in spite of failures, redundant SUs work together and form a protection group called Service Group (SG) [4]. If any of the SUs actively providing a service fails, the service is failed over to a redundant SU. The SGs follow one of the following redundancy models: 2N; N+M; N-way active; N-way; and No-redundancy redundancy models. Typically, one or more SGs form an AMF application [4].

AMF identifies the AMF node as the logical entity that is used to host the SUs [4]. This could be mapped to a physical host or a VM. SGs are deployed over a group of AMF nodes which form the AMF cluster [4]. AMF entities are typed, except for nodes and clusters. The common characteristics of AMF entities are captured in their respective types, like component types for components, SU types for SUs, Component Service Types (CST) for CSIs, Service Types (ST) for SIs, etc.[4].

III. RELATING THE TWO DOMAINS

Each VNF exposing a specific network functionality may consist of one or more VNFCs. Each VNFC provides a specific service type (i.e. sub-functionality of the network function) and it is packaged as a software image [9]. Such a VNFC is mapped to an AMF Application (App) type which provides a single service type, i.e. the service type of the VNFC mapped to it. For the NFV domain we introduce a new concept called AMF node type. This AMF node type represents a collection of software images of AMF components necessary for the SU type that provides the service type of a VNFC. Each VNFCI runs in a dedicated virtualization container (e.g. VM) [2]. In the AMF domain, each AMF node can be mapped into a VM. Thus, the VNFCI can be mapped to the AMF node, and the VNFCIs of a given VNFC to an AMF node group.

We generate an AMF configuration for a service type, i.e. the service type to be provided by a VNFC. We design a VNFC by grouping one or more AMF component types into a service unit type so that it can provide the service type. Based on the number of workload units (i.e. SIs) to be provided for the given service type (i.e. service capacity) and the requested level of service availability we determine the number of AMF entities (i.e. SUs, SGs, VMs) so that the VNFCIs providing the service can be deployed using the minimum number of physical hosts. The affinity and anti-affinity relations between the AMF entities are defined by AMF and therefore are reflected in the AMF configuration. In particular, nodes of an AMF node group are redundant entities, therefore they cannot be collocated. As a result, the VNFCIs form an NFVI anti-affinity group. During deployment, the required number of VNFCIs of a VNFC are deployed using the AMF configuration.

The proposed AMF configuration generation approach is applied for each of the service types of the VNF to configure the VNFCIs of each VNFC. In this paper, the terms VNFCI, AMF node and VM are used interchangeably.

IV. AMF CONFIGURATION GENERATION FOR VNFS

The AMF configuration generation process for VNFs requires three inputs: 1) the Configuration Requirements (CR); 2) the extended Entity Types File (ETF) model; and 3) the Infrastructure file. In the CR, the service capacity to be provided in terms of number of SIs of a service type and the number of CSIs of a component service type in each SI is described. It also includes the required level of service availability. An ETF describes the software component types from the perspective of AMF. In this file, a software vendor describes the dependencies, capabilities and limitations of software component types and the CSTs it can provide [8]. An ETF model may include ETFs from different software vendors. It has been extended to include reliability information (e.g. failure rates) [5] and also the memory required per CSI of a given CST. The Infrastructure file captures the information related to the capacity of the physical hosts and the VMs. It also includes the details regarding the Mean Time to Fail (MTTF) of physical hosts, host OS, hypervisor, VMs and guest OS.

Similar to the AMF configuration generation for a cluster [5], the AMF configuration generation for VNFs involves four steps: 1. ETF prototype selection; 2. AMF types creation; 3. AMF entities creation; 4. Distribution of the AMF entities for deployment. Step 1, Step 2 and Step 4 remain more or less the same as in [5]. In this section we discuss only Step 3.

As a result of the Step 1 and Step 2, each potential solution that can meet the CR is given as an AMF type stack, i.e. a set of AMF types that together can provide the requested functionality. Before configuring the instances of the AMF types, one has to determine the number of instances of each type necessary for handling and protecting the workload. This consists of determining the number of SUs, SGs and AMF nodes (VMs), their distribution among the VMs and the distribution of these VMs among a minimal number of physical hosts as opposed to the approach proposed in [5]. Since we are designing a VNFC, the SUs and VMs need to be identical as they provide one service type. In this paper, we also consider identical physical hosts.

Three factors can influence the number of physical hosts: redundancy, interferences due to collocation of entities and the capacity of the VMs and physical hosts. Redundancy requires additional resources to protect the service against failures [7]. The number of SUs per SG determines the minimum number of physical hosts required because each SG is deployed over VMs of an anti-affinity group, which therefore are hosted on different physical hosts. To minimize the number of physical hosts we can collocate within a VM or a physical host SUs belonging to different SGs. The physical hosts have finite resources in terms of memory, disk, number of cores and so on, and a limited number of VMs of a given flavor can be hosted on a physical host. All of which limits the collocation. Also, when several entities are collocated within a fault-zone when they fail they affect each other. To minimize the interferences, components can be grouped into different fault-zones (i.e. fault isolation units) such as VMs, but this may increase the number of physical hosts required. It is also well known that virtualization introduces some overhead that has to be taken into account in the calculation of collocated instances.

Before considering the distribution and the availability perspective one has to determine the maximum number of SIs (capacity) that can be supported by each VM flavor from computation resources perspective. In this paper, we only consider the memory perspective. The memory required for an active SI assignment is the total memory required by the components collaborating to provide the CSIs of the SI and this information is provided in the ETF model. Knowing the total memory of a VM flavor, the memory required for the guest OS, and the memory required for an SI provided by a given type stack, one can determine easily the maximum number of SIs a VM flavor can support. VM flavor-type stack combinations that are not able to support at least one SI are discarded.

Once the maximum capacity in terms of SIs is determined, the next step is to determine for each type stack and each VM flavor the number of SUs, SGs, VMs needed, and the resulting SIs per SU, and other entity distributions, etc. that meet the required level of availability with a minimal number of physical hosts. We follow an iterative process driven by the number of SIs per VM. For each AMF type stack and VM flavor we have a possibility of a minimum number of SIs per VM (lower bound equal to one initially) and a maximum (upper bound equal to the capacity of the VM flavor in terms of SIs initially). Each iteration will reduce the interval to the lower half or the upper half depending on the estimated availabilities for the configurations corresponding to the lower bound, upper bound and mid—interval point. For estimating the availability at each iteration we extend the method in [5] to take into account the availability of the infrastructure and the interferences between the collocated entities. For a given type stack and VM flavor, the process terminates if the lower bound does not meet the required level of availability or when the upper bound meets or exceeds it. In the first case there is no configuration meeting the required level of availability with the current VM flavor for the considered type stack, while in the second case we have determined the "optimal" configuration using the considered type stack and VM flavor.

The "optimal" configuration meeting the required level of availability and using the minimum number of hosts is the configuration that uses the minimum number of hosts among all the "optimal" configurations for all the type stacks and the VM flavors.

A. Calculating the number of entities (SUs, SGs, VMs, physical hosts)

For each type stack and VM flavor (with a fixed SI capacity), we determine the number of SIs per SU, the number of SUs per SG, the number of SGs, the number of SUs per VM, the VM groups, the number of VMs per physical host and the number of physical hosts.

The setting of the SU capacity depends on the actual recovery of the components and the redundancy models of the SGs in the type stack. If the impact of the failure of any component goes beyond the component itself and therefore may impact another SI then the SU capacity is set to a single SI so that the SU provides fault isolation between SIs. Otherwise, we use the fault isolation feature of the components and try to pack them together into a single SU up to the current SI capacity of the VM. In either case we check for vendor limitations, i.e. if the SU cannot provide at least one SI the type stack is discarded and if it cannot provide the theoretical maximum, then the SU capacity is set to the vendor specified maximum.

Based on the redundancy model we calculate the number of SUs in an SG so that all the assignments of an SI can be assigned to different SUs and that there is a spare SU when needed (for the case of No-redundancy and N-way redundancy models). The redundancy model also determines the relation between the SU capacity and the SG capacity, which are used therefore to calculate the number of SGs required to serve all the SIs.

Depending on the redundancy model the VM capacity may or may not be used to the fullest. For redundancy models with standby assignment(s) it is important that no two SUs of the same SG are hosted on the same VM or even physical host to avoid data loss and therefore service discontinuity. So, in these cases the calculated number of SGs puts a limit on the number of SUs per VM and per host as well, otherwise a VM or a host failure could cause such a loss.

Besides the number of SGs, the number of VMs per host is also limited by the resources of the host as mentioned earlier. Both need to be taken into account when calculating the number of physical hosts required to host all the SGs needed to provide all the SIs.

V. RELATED WORK

Work reported in [11] [12] [13] [14] considers meeting the availability requirement while scheduling the VMs or applications in the cloud. For example, [11] proposed a structural constraint aware VM placement for a set of related VMs to improve the performance and availability of the services. Authors in [14] generate a configuration for a set of redundant VMs. This is done to protect the application services against physical host failures. This work highlights the trade-off that exists between availability and optimal VM placement. Most of the researchers mapped a single-process application to a VM. In our approach, while this application could be mapped to an AMF component it is not limited to be single-process. However, since the VMs are providing the same service type of a VNFC, we only considered collocating entities of the same type. It should be noted that there are a few works [15] [16] that focus on deploying applications in multi-cloud environments with the goal of maximizing resource usage and availability.

The solutions in [5] and [11] are closely related to our work. Researchers in [11] proposed a placement approach to improve the availability of the applications in the cloud. It determines the placement of application components in VMs by considering the trade-off between improving the availability of services and minimizing the required resources. This solution takes into account the interferences between different applications. However, it is limited to stateless applications and considers only the N-Way active redundancy model. The approach proposed in [5] generates configurations that meet the requested level of service availability. The issues in this approach are twofold: 1) it considers deploying the configuration on a given number of physical hosts specified as input by a system designer. This may not be minimal because the designer may be aware of neither the effect of entity collocation nor the resource limitations. 2) The proposed availability estimate method considers only the availability of the components providing the service. The availability of the underlying infrastructure and the interferences between collocated entities are not taken into account.

VI. CONCLUSION

In this paper, we proposed to use AMF as middleware to manage the availability of the services provided by the VNFs. To achieve this, we proposed an approach for generating AMF configurations for VNFs. In this approach, we mapped the concepts in the NFV domain to the concepts in the AMF domain and we designed a VNFC by grouping one or more AMF component types to provide the service type of the VNFC. We determined the number of AMF entities and the number of VNFCIs with the goal that the requested availability should be met and the VNFCIs of a VNFC should be deployed using minimum number of physical hosts.

For the deployment of VNFs in the NFVI, the information about their deployment flavors, e.g. the number of VNFCIs, their placement and the VM flavor is required. This information is reflected in the AMF configurations generated for AMF managed VNFs and can be extracted to design the VNF configuration/profile. More specifically, the calculated number of VMs represents the required number of VNFCIs and each VM group denotes a VM anti-affinity group for the NFVI. In the process of generating the AMF configuration we also select a VM flavor for the VNFC. Finally, the calculated number of physical hosts determines the minimum required to deploy the VNFC for the requested service capacity.

ACKNOWLEDGMENT

This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson.

REFERENCES

[1] Network Functions Virtualization (NFV); Architectural Framework, ETSI GS NFV 002 V1.2.1 (2014-12).

[2] Network Functions Virtualization (NFV); Virtual Network Functions Architecture ETSI GS NFV-SWA 001 V1.1.1, 2014.

[3] Service Availability™ Forum Home Page, http://saforum.org/

[4] Service Availability™ Forum, Application Interface Specification AMF, SAI-AIS-AMF-B.04.01.

[5] P. Pourali, M. Toeroe, F. Khendek, "Pattern Based Configuration Generation for Highly Available COTS Components Based Systems", Information and Software Technology (IST), Elsevier, Vol. 74, pp. 143-159, 2016.

[6] Network Functions Virtualization (NFV); Management and Orchestration ETSI GS NFV-MAN 001 V1.1.1 (2014-12).

[7] M. Toeroe and F. Tam, "Service Availability: principles and practice", Wiley, 2012.

[8] Service Availability Forum™ Application Interface Specification; The Software Management Framework: Basic Concepts Explained.

[9] Network Functions Virtualization (NFV); Management and Orchestration; VNF Packaging Specification, ETSI GS NFV-IFA 011 V2.1.1 (2016-10)

[10] J. Heo, X. Zhu, P. Padala, Z. Wang, "Memory Overbooking and Dynamic Control of Xen Virtual Machines in Consolidated Environments", Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, 2009.

[11] D. Jayasinghe et al., "Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-aware Virtual Machine Placement", Proceedings of the IEEE International Conference on Service Computing, 2011.

[12] E. Bin et al., "Guaranteeing High Availability Goals for Virtual Machine Placement", Proceedings of the 31st International Conference on Distributed Computing Systems, 2013.

[13] M. Jammal, A. Kanso, A. Shami,"High Availability-Aware Optimization Digest for Applications Deployment in Cloud", Proceedings of the IEEE ICC, Services and Multimedia Applications Symposium, 2015.

[14] F. Machida, M .Kawato and Y. Maeno, "Redundant Virtual Machine Placement for Fault-tolerant Consolidated Server Clusters", Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2010.

[15] A. Abouzamazem, P. Ezhilchelvan, "Efficient Inter-Cloud Replication for High Availability Services", Proceedings of the IEEE International Conference on Cloud Engineering, 2013.

[16] M. E. Frincu, C. Cracium,"Multi-objective Meta-heuristics for Scheduling Applications with High-Availability Requirements and Cost Constraints in Multi-Cloud Environments", Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC), 2011.