Applying a Service-Based Architecture design style to Network Functions Virtualization

Bruno Chatras
Orange Labs Networks
ORANGE
Châtillon, France
bruno.chatras@orange.com

***Abstract*— This article presents a proposed an evolution of the Network Functions Virtualization (NFV) architectural framework towards a Service-Based Architecture (SBA), where management and orchestration services can be dynamically registered and discovered.**

Keywords—NFV, orchestration, service-based architectures

I. INTRODUCTION

After "network slicing", the term Service-Based Architecture (SBA) is about to become the next telecom buzzword following 3GPP's decision to apply an SBA approach for designing the control plane of the 5G core network [1]. SBA is an architectural style that places emphasis on the services provided by individual architectural components rather than on the relationships between pre-defined pairs of architectural components. SBA is expected to enable flexible and rapid development and deployment of 5G services, as it becomes possible for a new architectural component to connect to existing components without introducing specific new reference points. Network Functions Virtualization (NFV) is one of the key technologies underpinning the transformation of Telecom networks. NFV is regarded as a powerful means to simplify deployment and operation of network services, which will also apply to the deployment and operation of 5G network slices, for instance. At the heart of an NFV system is a set of management and orchestration functions, which itself could benefit from an SBA approach to enable flexible development and deployment of orchestration and management services.

Section II provides a short reminder about the NFV architectural framework. Section III discusses the properties of the SBA design style. Section IV describes several transformation steps that can be applied to the NFV architectural framework to leverage the SBA properties. It should be observed that the application of an SBA approach to the design of the NFV network services themselves is outside the scope of this paper.

II. THE NFV ARCHITECTURAL FRAMEWORK

NFV refers to a transformation of the telecom industry, where network functions traditionally hosted on bespoke dedicated servers are moved to pools of standard industry servers. NFV is a paradigm shift in network management, towards a cloud model with automated deployment and management capabilities. The NFV architectural framework [2] developed by the European Telecommunications Standards Institute (ETSI) identifies the key architectural components of an NFV system. Virtualized Network Functions (VNF) are deployed and executed on a distributed cloud infrastructure known as the NFV infrastructure (NFVI). The deployment, execution, and operation of VNFs and network services (NS) in an NFVI are steered by a management and orchestration (NFV-MANO) sub-system.

The NFV-MANO sub-system comprises three functional blocks. The NFV orchestrator (NFVO) is the entry point for other operations support systems (OSS) and business support systems (BSS) deployed by network operators. The NFVO's main responsibility is the management of the lifecycle of NS instances and the enforcement of the operator's resource management policies. The management of the lifecycle of VNF instances constituting an NS instance is delegated by the NFVO to one more or VNF managers (VNFMs). Both the NFVO and the VNFMs use the services exposed by one or more virtualized infrastructure managers (VIM) for allocating NFVI compute, storage and network resources to the objects they manage.

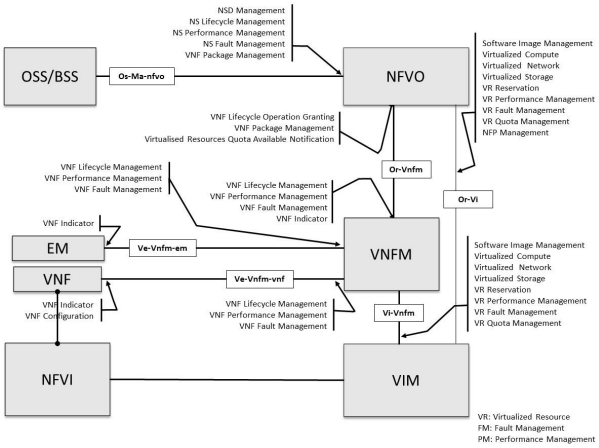

Fig. 1. NFV architectural framework

Fig. 1 depicts the ETSI architectural framework and identifies the main reference points between architectural components. ETSI provides REST API specifications for three

of these reference points: Os-Ma-nfvo between the NFVO and the OSS, Or-Vnfm between an NFVO and a VNFM, Ve-Vnfm between a VNFM and a VNF or an associated element manager (EM). There are no ETSI standards that describe the APIs exposed by a VIM on the reference point to an NFVO (Or-Vi) and the reference point to a VNFM (Vi-Vnfm). However, de-facto industry standards (e.g. OpenStack APIs) are commonly used. Fig 1 also shows the list of APIs exposed by the main functional blocks and thus the services they provide. The specifications of these APIs are currently tied to the reference point concept, a reference point representing the association between two MANO functional blocks. ETSI publishes a set of deliverables known as Group Specification (GS), each covering the scope of one reference point. For example, ETSI GS NFV-SOL 005 [3] specifies all APIs exposed by an NFVO towards the OSS.

III. Service-based architectures

There is no formal definition of what SBA really means. In the technical literature, the definition actually varies according to the authors. In some cases, SBA is used as a portmanteau word encompassing Service Oriented Architectures (SOA), Resource-Oriented Architectures (ROA), Microservices architectures and other Component-Based Software Engineering (CBSE) variants. In other cases it is regarded as a way of combining the best of both SOA (service registration and discovery) and ROA (RESTful design). Sometimes, SBA is also pitched as a middle ground between SOA and Microservices [4].

When it comes to applying the SBA style to network standards, the most noticeable difference with a conventional approach is the way interactions between functional blocks are specified. As already alluded to when introducing the NFV architectural framework, a conventional approach is centered on the specification of information flows between specific pairs of functional blocks. With SBA, every functional block provides one or more services that can be consumed by any other functional block and the information flows for consuming these services are specified independently of the actual consumer functional blocks. With an SBA design, the focus is on the services provided by the functional blocks rather than on specifying reference points between them. All functional blocks can communicate with each other as if they were connected to a shared Enterprise Service Bus (ESB) as shown in Fig. 2. One functional block can use the services provided by the other functional blocks, like in software development a main program can call external functions through an API. By analogy, functional blocks in an SBA are said to communicate through APIs, which in practical terms often means they communicate using the Hypertext Transfer Protocol (HTTP) according to a REST pattern.

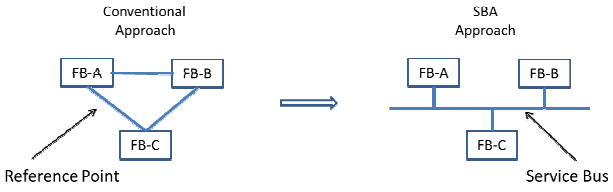

Fig. 2. From conventional architecture design to SBA

Two additional characteristics are typically associated to an SBA design and are leveraged by 3GPP as well: dynamic service registration and discovery, and the use of a common data storage service. Dynamic registration and discovery relies on a registry where all service instances available in a functional block are registered and can be discovered by other service instances in the same or different functional blocks (See Fig. 3). When a functional block instance is deployed, the services it provides and the mechanisms to invoke them are registered. Similarly if a functional block instance is removed from the network, the corresponding entries are deleted from the registry. A functional block that requires a particular service can then discover and select a functional block instance that provides this service.

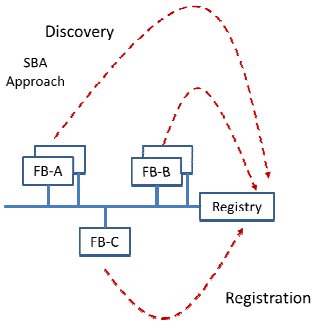

Fig. 3. SBA Registration and Discovery

SBA, as a design style, does not require a common data storage approach but goes well with it. Some functional blocks on the "service bus" can expose a data storage service that other services can use to store any kind of data, including state information. Separation between processing and data means that all data used by the services are stored in logically centralized data repositories, which in some way makes these services stateless, although they might store state information in these external repositories. This is intended to increase resilience of the overall system and facilitate data sharing across multiple services.

The "service bus" interconnecting the functional blocks and the services they provide is an evocative expression, used in analogy with a computer hardware bus. SBA does not mandate a particular type of communication bus. In a rather primitive form, the communication services provided by the bus can be limited to basic IP routing (e.g. when the communication bus is implemented as a layer 3 VPN overlaid on the network of a data center). In such cases a functional block willing to consume a service must have the processing logic to discover and select a functional block instance providing this service. This typically involves retrieving a list of candidate instances and selecting one of them according to a load balancing algorithm and querying the Domain Name Service (DNS) to determine its IP address. More advanced forms of the service bus can reduce complexity at the client side, by providing application-layer message routing or by distributing messages according to a publish/subscribe pattern. The aforementioned registry functionality is then embedded together with the communication service, in the service bus functionality; thereby enabling client functional blocks to offload the selection of a target service instance and the determination of

its IP address. Advanced communication buses may even provide fast failover functionality and message transformation capabilities to enable connecting non-compatible clients and servers.

IV. SBA APPLIED TO NFV

NFV-MANO, the subsystem of the NFV architectural framework in charge of management and orchestration functions, is likely to be a good candidate to undergo an SBA transformation. NFV-MANO functional blocks provide management and orchestration services, each of which is already exposed through a dedicated REST API/interface. Moreover, the granularity of an NFV-MANO service sounds compatible with the spirit of an SBA design. For example, an NFVO provides services such as VNF package management or network service lifecycle management (LCM), network service performance management, etc. Several incremental steps can be envisioned to transform NFV-MANO into an SBA.

A. *Initial Step*

The first step towards SBA would merely be a re-documentation exercise, consisting in decoupling the specification of APIs from the concept of reference point. In other words, APIs would be specified from the point of view of the API producer only. The scope of an ETSI GS would no longer be a reference point but a single API/service or a set of API/services produced by the same functional block. This would not prevent these specifications to include appropriate provisions to describe consumer-dependent specifics (e.g. a while the VNF lifecycle management API exposed by a VNFM can be consumed by an NFVO or a VNF, a VNF cannot use the API to create itself). Although not part of NFV-MANO, the Software-Defined Networking (SDN) controllers deployed in an NFVI could also be integrated into the SBA approach. This would enable any NFV-MANO functional block (i.e. not just the VIM), the OSS and some VNFs to access the services they provide to steer the traffic according to application-specific needs. In line with the SBA design style, the full set of functional blocks would appear as if they were all connected to a single service bus (See Fig. 4). As a side note, it should be observed that while SBA enables any functional block to communicate with any other one, there is no implication that any functional block is authorized to invoke any service provided by any other functional block. The API framework for NFV-MANO already incorporates role-based authorization mechanism, where every client is associated to a role verified at authentication time. This role determines what a particular client is allowed to do.

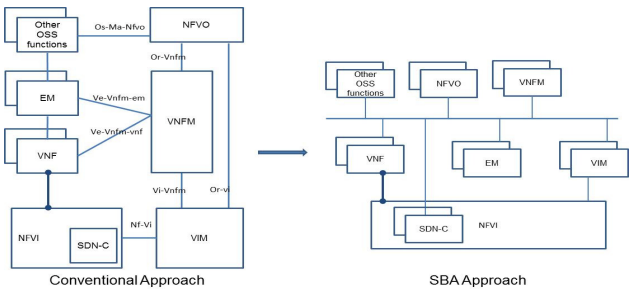

Fig. 4. An SBA view of the NFV Architectural Framework

This transformation would not bring any new management feature to NFV but would be a first step towards a fully-fledged SBA approach, as well as a means to clarify to the industry that ETSI NFV API specifications can adapt to various architectural changes. For example, an SBA approach would facilitate extending the NFV architectural framework with a security orchestrator [5] that would be able to interact with all other functional blocks via the "service bus".

B. *Dynamic Registration and Discovery*

Another step towards an SBA approach consists in enabling dynamic registration and discovery of instances of NFV-MANO functional blocks and of the services they provide. As already alluded to, this would require a *registry* to be added to the management service bus (See Fig. 5). New NFV-MANO functional block instances would be registered once created and deregistered before being taken out of service. Various communication modes can be envisioned with the registry, including query/response and subscribe/notify. NFV-MANO and other OSS functional block instances would then be able to discover other instances they need to interact with.

Registry entries would typically contain a description of the APIs / services exposed by these functional blocks, including for each service a description of the means to invoke them (e.g. an HTTP URI in case of a REST API), a list of supported versions and where it makes sense a list of supported features. For example, by interacting with the registry, an NFVO would then be able to select a VNFM that provides the services and the features it needs to manage a particular VNF, or to select a VNFM that supports the direct or indirect mode of communication with the VIM, depending on its own preferences.

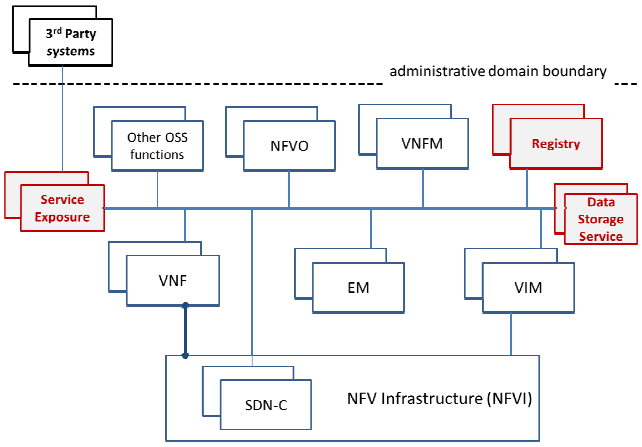

Fig. 5. Dynamic Discovery of NFV-MANO functional blocks

A variant of the above design would consist in offloading the discovery of network functions and services to the service bus. For example, with such an approach, an NFVO would no longer select the VNFM instances where to send VNF instantiation requests but would send these requests to an abstract "service type" address and let the service bus select the appropriate VNFM instance and even translate the request into a language understood by this VNFM.

C. Towards a common data storage approach

Another property typically associated to SBA is to make service implementations dataless and stateless, by moving all data handled by these services in an external high-availability database that can potentially be accessed by all services. Applying this principle to NFV-MANO would imply extending the functional architecture with a MANO data storage service functional block as shown in Fig. 5. This functional block would host VNF and NS catalogues (i.e. collections of NS and VNF deployment templates), as well as run-time information about VNF instances and NS instances. All or part of the NFVI resources catalogue could also be hosted in this repository. This transformation step, combined with the previous one, would simplify NFVO / VNFM fast failover. It would thus increase the availability of the NFV-MANO subsystem, which is crucial when considering commercial NFV deployments, as a failure of any NFV-MANO component will have huge network-wide consequences [6]. Furthermore, a side benefit of this evolution is to make the NFV inventory independent from the NFVO, directly accessible by any other functional block, thereby offloading the NFVO of its proxy tasks.

D. Service Exposure

Making the services accessible by a 3rd party is a property that is often expected from a service-based architecture. This is typically achieved by adding a service exposure function on the service bus, acting as a gateway to external administrative domains. This function, also visible in Fig. 5, typically carries out stronger authentication and authorization procedures than those used inside the SBA domain. It can also act as an application-layer firewall, a reverse-proxy and collect API usage metrics for accounting. Such a functional block could also be added to an SBA-oriented NFV-MANO architecture, to support inter-domain management and orchestration procedures in a secure way, without exposing more information than strictly necessary about the two communicating NFV systems.

E. Towards the disappearance of functional blocks

A further and more radical transformation step would be to get rid of functional blocks, in other words the way elementary services/functions are grouped would no longer be considered a standardization matter. Existing functional blocks could still be mentioned in ETSI standards but would just represent examples of typical product packaging strategies. Fig. 6 illustrates a configuration where the NFVO and the VNFM are decomposed into a set of elementary services, each corresponding to one of the APIs currently provided by these two functional blocks. Similarly the OSS functional block could be disaggregated in multiple smaller services such as event tracking and analytics, application orchestration, slice management, policy management, etc. This would allow for more flexibility in mapping the functional architecture to software implementations. While this might be perceived as disruptive by some industry players, it should be noted that most open source communities developing NFV-MANO solutions implement the NFV-MANO functional blocks in the form of Microservices interconnected via some form of service bus. Another interesting side-effect of the disaggregation is the ability to specify, deploy, update and package elementary management and orchestration services, in various ways, depending on the actual requirements.

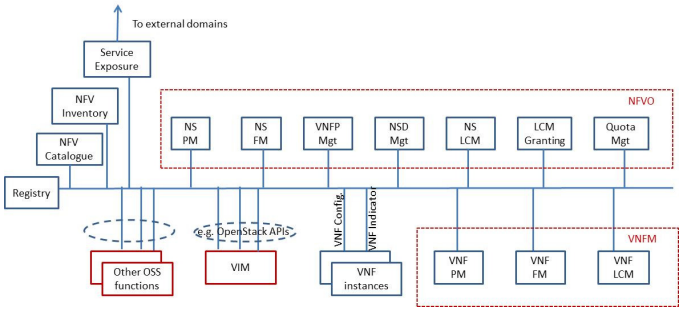

Fig. 6. SBA with disaggregated functional blocks

V. Conclusion

This paper has reviewed the transformation steps that the NFV architectural framework could undergo in order to benefit from an SBA design style. An SBA approach to NFV would increase the resilience of the NFV-MANO system itself thanks to data/state externalization and to the ability to dynamically discover fallback instances for each NFV-MANO functional blocks. This transformation would also make the NFV architectural framework more future-proof, facilitating its evolution from both a specification and implementation viewpoint. This includes allowing new pairs of existing functional blocks to communicate with each other without creating new APIs, new functional blocks to consume services exposed by existing NFV-MANO functional blocks or providing new services to existing functional blocks. Integration in the NFV-MANO architecture of new functional blocks specified by other SDOs or developed by open source communities would also be facilitated. The disaggregation of the main functional blocks would also bring more flexibility in deploying and packaging the management and orchestration services, according to real needs.

References

[1] Frank Mademann, “The 5G system architecture”, Journal of ICT Standardization, Vol.6, Combined Special Issue 1 & 2, May 2018.

[2] ETSI GS NFV 002: Network Functions Virtualization (NFV); Architectural Framework.

[3] ETSI GS NFV-SOL 005: Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point.

[4] N. Ford, “Comparing Service-based Architectures”, available on-line http://nealford.com/downloads/Comparing_Service-based_Architectures_by_Neal_Ford.pdf.

[5] Bernd Jaeger, “Security Orchestrator: Introducing a Security Orchestrator in the context of the ETSI NFV Reference Architecture”, IEEE Trustcom/BigDataSE/ISPA 2015.

[6] A.J. Gonzalez and al, “Dependability of the NFV Orchestrator: State of the Art and Research Challenges”, IEEE Communications Surveys and Tutorials, April 2018, in press.