Superfluidity: a Flexible Functional Architecture for 5G Networks

Giuseppe Bianchi[1], Erez Biton[2], Nicola Blefari-Melazzi[1], Isabel Borges[3], Luca Chiaraviglio[1], Pedro de la Cruz Ramos[4], Philip Eardley[5], Francisco Fontes[3], Michael J. McGrath[6], Lionel Natarianni[7], Dragos Niculescu[8], Carlos Parada[3], Matei Popovici[8], Vincenzo Riccobene[6], Stefano Salsano[1], Bessem Sayadi[7], John Thomson[9], Christos Tselios[10], George Tsolis[10]

1. Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Italy; 2. Nokia Israel, Israel; 3. Altice Labs, Portugal; 4. Telcaria Ideas S.L., Spain; 5. BT, UK; 6. Intel Labs Europe, Ireland; 7. Nokia Bell-Labs, France; 8. Universitatea Politehnica din Bucuresti (UPB), Bucharest, Romania; 9. OnApp Ltd., United Kingdom; 10. Citrix, Greece

***Abstract*—We propose the innovative architecture of Superfluidity, a Horizon 2020 project, co-funded by the European Union. Superfluidity targets 5G networks, by addressing key network operator challenges with a multi-pronged approach, based on the concept of a flexible, highly adaptive, *superfluid* network. Superfluidity supports rapid service deployment and migration in a heterogeneous network environment, regardless of the underlying hardware. The overall proposal offers advanced capabilities in terms of service deployment and interoperability, while at the same time guaranteeing high performance levels end-to-end.**

Keywords—5G networks, SDN, NFV, Mobile Edge Computing (MEC), CRAN, future Internet architectures.

I. INTRODUCTION

Today's networks suffer from a variety of shortcomings, including a lack of service agility, a lack of implementation agility and increasing complexity. The lack of service agility prevents us from creating new services in a rapid, flexible and tailored fashion. The lack of implementation agility means that we have to rely on rigid, cost-ineffective hardware devices with long provisioning times. Increasing complexity arises from the continuous growth and heterogeneity of network traffic, services and hardware technologies.

Several emerging trends, mainly in the context of 5G networks [1][2], are likely to exacerbate these issues: the forthcoming explosion of the Internet of Things (IOT) [3], new radio techniques such as massive 'multiple input multiple output' and beam-forming [4][5], and the desire for more flexible business models [6].

The architecture being developed by the Superfluidity project is focused on alleviating these problems. It has the following features: i) Flexibility, via an architectural decomposition of network components and network services into elementary, reusable primitives, defined in Superfluidity as Reusable Functional Blocks (RFBs); ii) Agility, via the rapid 'chaining' of these RFBs to form exactly the service required; iii) Simplicity, via virtualization of radio and network processing tasks, network functions and services (a fully cloud-based architecture); iv) Portability, via platform-independent abstractions, permitting the reuse of network functions across multiple heterogeneous types of hardware; v) High performance, via software acceleration, specialization and adaptation to hardware accelerators.

In this paper, we outline the Superfluidity architecture and describe how it builds on the well-known concepts of Network Function Virtualization (NFV) [7], Software Defined Networking (SDN) [8], Mobile Edge Computing (MEC) [9] and Cloud Radio Access Network (CRAN) [10]. We believe that a network implementing our architecture would be "superfluid": it would have the ability to instantiate new services on-the-fly, run them anywhere in the network (core, aggregation, edge) and shift them transparently to different locations; and it would enable innovative use cases in the mobile edge, empowering new business models, and reducing investment and operational costs. The use cases that are driving the design of the Superfluidity architecture are described and analyzed in [11].

The main contributions of this paper are the following ones. First, we propose a "superfluid" network, which is based on RFBs, and that iteratively allows the decomposition of network functions into smaller components. Second, we define the RFBs. Both contributions are innovative, even if we recognize that future works, targeting the experimental evaluation of the proposed architecture, are necessary to fully assess the performance and the merits of the Superfluidity architecture.

The rest of the paper is organized as follows. We describe the main building blocks of Superfluidity in Section II. The architectural framework of Superfluidity is reported in Section III. Section IV highlights the design principles and the requirements. The overall Superfluidity architecture is described in Section VI. Finally, Section VI summarizes our work.

II. BUILDING BLOCKS

Superfluidity leverages: i) SDN and NFV technologies, ii) the latest advancements in CRAN, and iii) MEC technologies. The following sections describe these building blocks.

A. NFV plus SDN

SDN [8] and NFV [7] are significant technology evolutions that are key to realizing 5G networks. The primary focus of SDN is to decouple the control plane from the data plane,

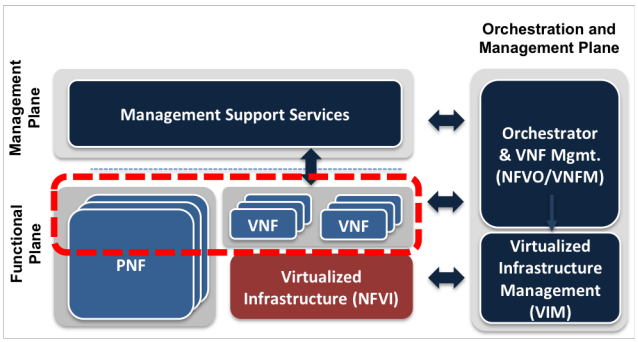

Figure 1 ETSI NFV Architecture (simplified) from [12]. PNF = Physical Network Function, VNF = Virtual Network Function, NFVO = Network Function Virtualization Orchestrator, VNFM = Virtual Network Function Manager

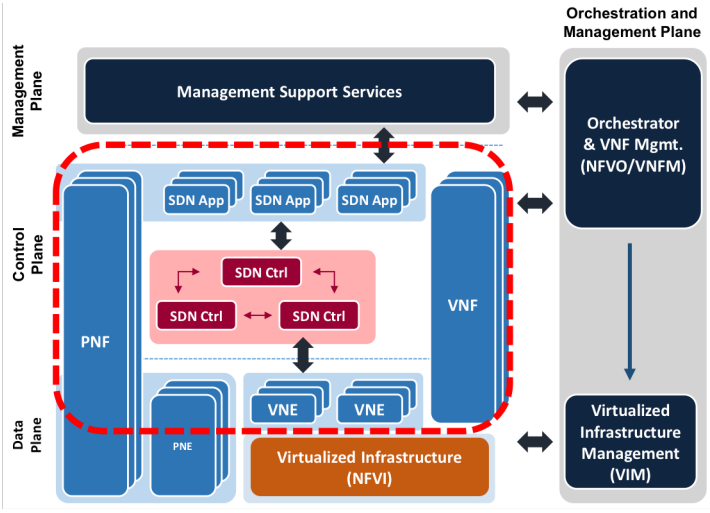

Figure 2 Combined NFV and SDN view [13]

allowing operators to simplify service and networking provisioning. NFV enables the "cloudification" of Network Functions (NFs), which may be implemented either on dedicated or commercial-off-the-shelf (COTS) hardware.

A NF could model either a traditional physical entity (like a router or an e-NodeB) or a logical function resulting from the decomposition of functionalities belonging to one or more layers of the protocol stack (from the physical layer up to application layer). Traditionally, NFV and SDN have been developed as two separate technologies. However, they may offer a significant value if combined in the same architecture. In particular, the NFV paradigm supports the deployment of network functions on demand, giving the opportunity to place them on the most suitable elements and to use the appropriate amount of resources (see Figure 1). On the other hand, these tasks require SDN to configure the network suitably, to make the network programmable and to define function chaining. Significant functional synergy exists between NFV and SDN, simply by taking the ETSI NFV architecture and introducing the SDN paradigm over it as shown in Figure 2. By doing so, NFVs splits the control plane (SDN App) from the data plane (VNE – Virtual Network Element). In the middle, the SDN Controller is introduced as an abstraction layer. Legacy VNF and SDN components can be already supported.

B. Cloud Radio Access Network

The main idea behind Cloud-RAN (CRAN) [10] is to pool the Baseband Units (BBUs) from multiple base stations into a centralized BBU pool for statistical multiplexing gain. A minimal set of critical functions remain at the radio head (RRH, Remote Radio Head), whose main function is frequency shifting. With CRAN, it is then possible to have a very tight coordination between cells and to maximize the radio capacity in bps/MHz/cell. Additionally, by leaving only the RRH on-site with a compact power supply, CRAN facilitates antenna site engineering and provides footprint reduction, as well as shorter installation times and lower rental and energy costs.

C. Mobile Edge Computing

A key enabler for low-latency scenarios of 5G networks is the concept of Mobile Edge Computing (MEC) [9], whose main idea is to deploy computing capabilities near to end users. MEC allows supporting Radio Access Network (RAN) processing and third party applications. This technology brings a set of advantages: i) ultra-low latency, ii) high bandwidth, iii) real-time access to radio network information and iv) location awareness. As a result, operators can open the radio network edge to a third party, allowing them to rapidly deploy innovative applications and services towards mobile subscribers, enterprises and other vertical segments. One of the goals of Superfluidity is to integrate MEC in the overall architecture such that the MEC platform can rely on the same physical resources of an Extended-NFVI.

III. Architectural Framework for Superfluidity

The Superfluidity project aims to design a unified, high performance and distributed cloud platform for radio and network functions support, as well as their migration. In our vision, CRAN, MEC and cloud technologies are integrated, by adopting an architectural paradigm able to create the glue that can unify heterogeneous equipment and processing into one dynamically optimized, superfluid, network. Figure 3 depicts a high-level view of the overall architectural framework. The top layer of the Figure includes the different components involved (CRAN, MEC, virtual core and Data Centers (DC)), while in the bottom layer the different types of physical DCs are shown (namely Cell-site, Local, Regional and Central). This classification is somehow arbitrary and the infrastructure of different operators can be structured in different ways. The next layer down is a traditional Operational Support System (OSS), whose main goal is to deal with all the components in order to create services for end-users.

The underlying Extended-NFVI, located at the bottom of the proposed architecture, represents an evolution of the ETSI NFVI concept. The current NFVI focuses on supporting VMs

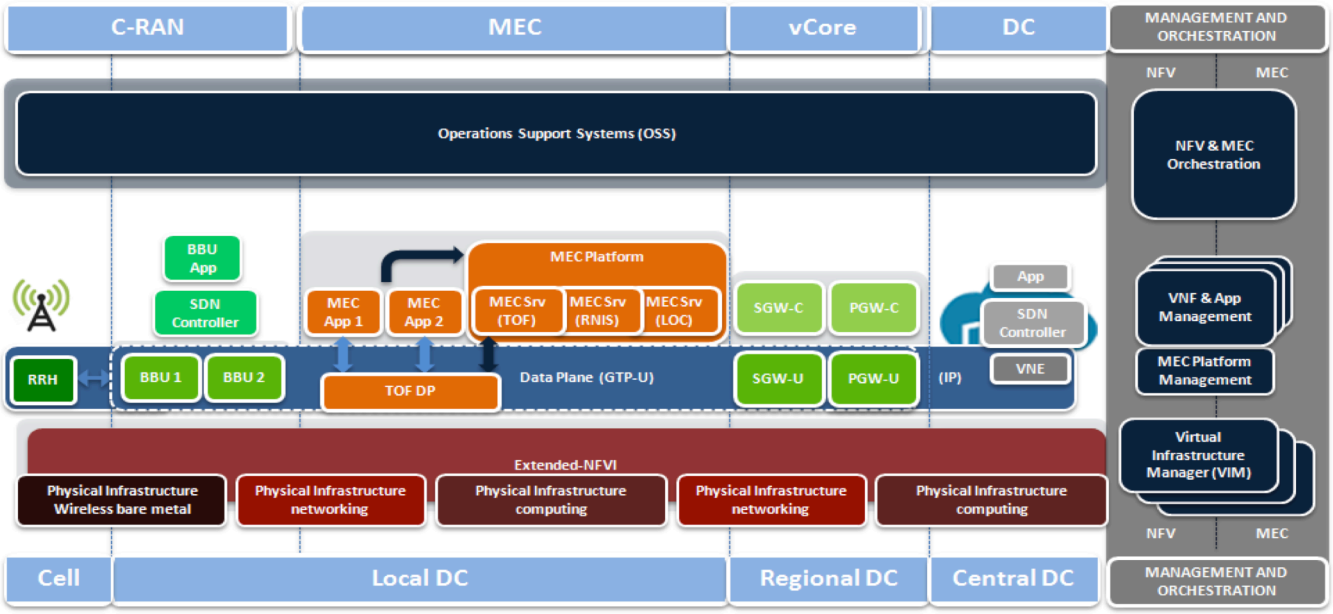

Figure 3: Architectural framework for Superfluidity with an example mapping into the physical Data Centers

or containers to run VNFs; the E-NFVI also considers heterogeneous execution environments that we will present in Section V. This extended-NFVI is common to all components, simplifying resource management and allows agile (superfluid) orchestration of services. The mapping of the components into the different physical DCs (Cell, Local, Regional) shown in Figure 3 and described hereafter is the one considered for the Superfluidity testbed implementation. However, we stress that the dynamicity of the architecture, and the concept of an Extended-NFVI, allows support for different solutions, derived by considering the various trade-offs between performances and efficient utilization of resources.

Starting from the left, the CRAN component is split into two blocks, corresponding to the RRH and the BBU components. The RRH is placed in the cell-site, while the BBU is located in a specialized local DC, adapted for telecommunications. It is assumed that the local DCs will control a small number of cell-sites and that they are geographically distributed, but in general proximity to the cell sites. The BBU functionalities will be virtualized and thus they will benefit from the centralization to scale in/out, according to the load and to be fully re-programmable under a holistic NFV/SDN controller. Figure 3 also shows the MEC components. Specifically, the white paper on MEC [21] suggests that placing servers in aggregation and radio access networks, where currently only base stations and radio network controllers operate, will increase network efficiency, along with several additional benefits. In our architecture the MEC component is deployed in the same location (local DC) as the CRAN, given the fact that it utilizes the exact same infrastructure with the latter or other VNFs. This endogenous characteristic of MEC significantly increases efficiency and ease of use. The next component shown in Figure 3 is the virtual Core (vCore), comprising the central nodes of a cutting-edge mobile network. The vCore runs on the common E-NFVI, usually located in regional DCs. In this way, both agility and fluidity of the overall architecture are improved, especially when live nodes need to be migrated and/or scaled. The Figure shows the expected evolution of mobile core networks towards the SDN model, where the control plane and data plane components are completely separated, with the former fully controlling the latter on data processing tasks. In particular, the components shown are the ones resulting from splitting the 4G/LTE Core elements. The DC component corresponds to the traditional datacenter segment, where a large number of services are deployed. These services are located at central points and deal with significant compute/storage/network resources. Beyond traditional services, central DCs also implement NFV and SDN technologies, in order to control the network components inside the DC. For this reason, a generic VNE (Virtual Network Element), a SDN controller and a generic App element are included in the architecture.

The last part of the architecture is the common management and orchestration vertical layer (right part of the Figure), which is responsible for providing the system intelligence. It includes a NFV-like set of functions, which supports an integrated view of the overall architecture, including networks, services and DCs. In this way, it is possible to take advantage of a common extended-NFVI and achieve an end-user centric view of the ecosystem. This layer is responsible for resource management over the different DCs illustrated at the bottom of Figure 3, thus building a federated environment. Moreover, it orchestrates VNFs to create complex services, taking the best decisions for services to be deployed, while considering customer needs.

TABLE I: SUPERFLUIDITY'S INNOVATIONS AND GOALS

Research Area		Superfluidity Goals
Cloud Networking	NFV Virtualized Environment	- Achieve 10-40 Gbps throughput with virtualized and software based packet processing. - Achieve tens of ms. (or lower) service instantiation. - Achieve massive consolidation and migration of services.
	Software Radio and DSP Virtualization	- Unify and abstract the protocol stack and the hardware platform to instantiate multiple vendor protocols. - Open radio platform and enable third parties to design the protocol stack. - Design portable runtime dataflow engines, enabling simultaneous seamless execution of multiple protocols.
	High Perf. Software-Based Packet Processing	- Achieve high performance with commodity hardware. - Increase the level of flexibility by going beyond the paradigms of proprietary, embedded systems. - Enable agile network function deployment.
Network Service Decomposition and Programmability		- Introduce program abstractions specifically targeted to 5G functions. - Combine block-based composition abstractions (such as those exploited in Click routers [16], or emerging in the ETSI NFV work on service chaining) with event-driven programming paradigms such as basic match/action based approaches or more powerful stateful abstractions based on extended finite state machines.
Cloud RAN and Mobile Edge Computing		- Enable modular "hot" replacing of eNB functions (such as scheduling) and allow migration of such functions between edge clouds and the antenna subsystem, so as to balance algorithmic complexity with front-haul capacity. - Enable the migration of non-RAN functions (like local caching and CDN) between the Remote Radio Head and the edge cloud, to maximize their performance.
Automated Security and Correctness		- Provide a pre-deployment checking system to ensure that virtualized network services do not negatively affect the network nor other tenants; the system has to be both scalable and stateful, able to model most types of services. - Implement a post-deployment system that will learn the behaviour of traffic and detect any anomalies, thus providing a further security mechanism in cases where the checking system does not have information about the processing performed by a network function, or when static analysis is inaccurate.

Finally, Table I summarizes the Superfluidity innovative contributions beyond the current state of the art.

IV. DESIGN PRINCIPLES AND REQUIREMENTS

In this section, we highlight the main principles behind the design of the Superfluidity architecture. In Section V we will discuss how we put these principles in practice, by detailing the vision of the Superfluidity architecture.

Recursion: The Superfluidity architecture needs to be *recursive*, natively handling layering and partitioning. The concept of layers can be described as follows. Today, global network operators can act as virtual operators in those countries where they buy network access from other operators. Similarly, we expect that some virtual network functions will actually be delivered by a (or several) virtual function(s) run in a different layer. Potentially, there could be several levels of hierarchy. An arbitrary number of layers can exist, which can be within a single operator. Regarding partitioning, the concept is that there may be several successive, separately operated networks on the end-to-end path. An important aspect of recursion is that the architecture is the same at each level and in each partition – there is nothing special about being at 'level 2' in 'regional network C', say.

Scalability: A proper layering is also important for the scalability of the system. Events may need to be handled "locally" in a layer in order to avoid higher layers being swamped with too much unnecessary information.

Separation of state and processing: The basic idea is to separate the actual in-line processing of the network function from its internal state. For example, a NAT function would read the private address and then perform a look up, in a separate data store, to find the associated public address. Such separation should help with resilience against failures and seamless scaling. Essentially, this is about reapplying the approach of cloud-scale applications to virtualized network functions. The main challenge is how to achieve good performance despite the decoupled state [14].

Support for minimal Virtual Machines: The key principle enabling resource and service abstraction is virtualization, which allows the creation of virtual machines on top of physical ones. In this context, it is useful to create lightweight virtual platforms, e.g., only retain the features that are mandatory for each VM. In this way, it is possible to make very efficient use of resources, allowing thousands of mini VMs to run on a single physical host. The instantiation time of these VMs can also be kept very low, allowing the deployment of the VMs on demand.

Support for extended finite state machines: The application of abstract finite state machines as "platform agnostic programming language" for network processing tasks has been recently proposed in [15]. This work has specifically introduced a network node architecture, which extends the stateless OpenFlow match/action abstraction, and permits to support the dynamic execution of Mealy Machines (a subset of the more general class of eXtended Finite State Machines - XFSMs). This architecture introduces a control logic in the switches (which are included in the physical infrastructure networking blocks in Fig.3), offloading the decisions based on local states from the controllers.

On-the-fly monitoring: The superfluid network will be highly dynamic in how and where it instantiates VNFs and VMs, with many individual and chained lightweight functions.

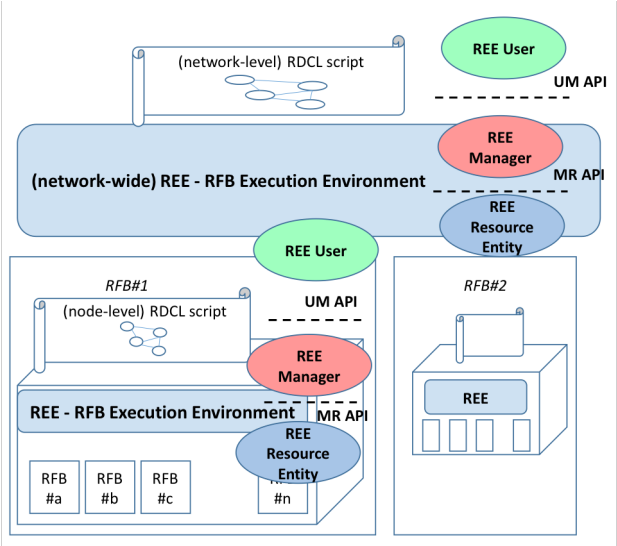

Figure 4: Superfluidity Architecture

Thus, we will need suitable measurements such that the virtualization adapts appropriately, without unnecessary data resolution and diversity – one can imagine a naive approach leading to an n^2 scalability problem, where much of the network and processing resources are expended on measuring the lack of those resources. This suggests that the key is flexibility – monitoring should be installed "on the fly", as and when needed – not to mention, what and where needed - and under the control of the 'management and orchestration' functionality at that layer and network. Even in scenarios that require more advanced monitoring capabilities (e.g., when network and service elements related to the overall Quality of Experience (QoE) of users need to be tracked or analyzed), the superfluid network should be able to facilitate this type of demand, via deploying specialized virtual probes, in an ad-hoc manner. Information regarding real-time network conditions can be extracted, further enhancing the orchestration and management abilities of the proposed architecture.

V. Superfluidity Architecture

The vision of the Superfluidity project is to move from the current architectural approaches based on monolithic network components/entities and their interfaces, to an approach where components can be "constructed" via the programmatic composition of elementary "building blocks". The allocation and deployment of the building blocks over the underlying infrastructure should be highly dynamical, ideally allowing a continuous real-time optimization. The decomposition of high-level monolithic functions into reusable components is based on the concept of *Reusable Functional Blocks* (*RFB*s). A RFB is a logical entity that performs a set of functionality and has a set of logical input/output ports. In general, a RFB can hold state information, so that the processing of information coming in its logical ports can depend on such state information. RFBs can be composed in graphs to provide services or to form other RFBs (therefore a Reusable

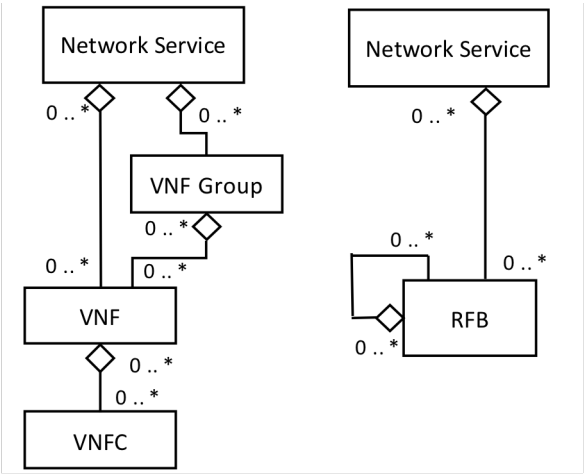

Figure 5: Class diagram for the current approach (left side) and the proposed one (right side).

Functional Block can be composed of other RFBs). RFBs need to be characterized and described in a formal manner. Additionally, we need platform-agnostic node-level and network-level "programs" describing how the RFBs interact, communicate, and connect to each other so as to give rise to specific (macroscopic, and formerly monolithic) node components, network functions and services. A language that supports the description and the interaction of RFBs is referred to as *RFB Description and Composition Language* (*RDCL*). The heterogeneous computational and execution environments, supporting the execution (and deployment) of the RDCL scripts and the relevant coordination of the signal/radio/packet/flow/network processing primitives are referred to as *RFB Execution Environments* (*REE*). Figure 4 shows the relationship between the RFB, RDCL and REE concepts. The figure also shows that the model (with technical differences) is recursively applied at different levels. At each level, we identify a *REE User* and a *REE Manager*. The REE User requests the deployment/execution of a service / service component described using a RDCL script to the REE manager. The REE Manager is in charge of deploying/executing the RCDL script using the resources within its REE. Within an REE, the REE Manager interacts with the REE Resources Entities that are required to support the realization of the RCDL script. In Figure 4, two types of APIs are shown. The User-Manager (UM) API is used by the REE User that wants to deploy a service or a component into a REE. The Manager-Resource (MR) API is the interface used by the REE Manager to interact with the resources in its REE.

A. Reusable Functional Blocks (RFBs)

The RFB decomposition concept is applied to different heterogeneous environments. An RFB may be analogous to a traditional VNF or VNFC, implemented as a fully-fledged VM running on a hypervisor or in an OS container. An RFB can correspond to a small footprint Unikernel VM running in a specialized hypervisor. In the latter case, the execution environment is the hypervisor specialized in supporting

lightweight VMs. RFBs can also be modules or components of special purpose execution environments, like extended finite state machines based on OpenFlow for packet processing [15], software routers [16], or radio signal processing chains [18].

In general, an RFB may hold state information on which the overall information processing is based upon. It also contains a set of characteristic properties, with the RFB's execution environment being the most imperative one. A logical RFB can have more than one possible execution environment meaning that it can be realized by using a variety of technologies and frameworks. RFBs are characterized by their resource requirements (i.e., storage, processing), their load limitation (i.e., maximum number of packets per second or the number of different flows) and most importantly, the set of logical ports they support (i.e., data plane ports, control plane ports, management plane ports). All types of logical ports facilitate inbound and outbound information flows to the RFB, with the majority of RFBs being designed to support seamless port operation under load. RFBs characterization may be augmented by using a formal description of their behavior.

Figure 5 highlights the difference between the current ETSI model and the proposed approach. In the ETSI model, there is a fixed hierarchy between VNF Groups (defined in [12], but not considered in [19]), VNFs and VNF Components. VNF Components cannot be further decomposed.[1] In the proposed recursive model, all elements are RFBs and the decomposition can be iterated an arbitrary number of times.

In order to ensure that the RFBs are properly combined, the consistency between the information exchanged over the ports needs to be validated. Therefore, appropriate modelling of the information is required. Logical ports are mapped to port instances when an RFB is functionally deployed. Logical ports can be combined using the concept of *aggregated* logical ports. For example, an IP router working with the OSPF protocol can be represented as having/offering two types of logical ports: data plane ports for routing of IP packets; OSPF control plane ports for discovering OSFP adjacent entities and exchanging routing information with them. The two logical ports are combined in an *aggregated* logical port, which supports the exchange of IP packets; i.e., both data plane and OSPF control plane packets.

B. RFB Description and Composition Language

A key aspect to enhancing portability is the ability to describe a complex and potentially stateful network function as the combination of RFBs. Specifically, our goal is to obtain a platform-agnostic formal description of how such RFBs should be invoked, e.g. in what order, with which input data, and how such composition may possibly depend on higher level "states", and change based on such states.

[1] Actually, the framework discussed in [9] introduces the concepts of VNF decomposition and composition in a general way, leaving their detailed specification for further study. The practical constraints on the decomposition of VNF Components have been introduced in [15].

While the idea is simple, its actual specification is challenging. Different languages may be more appropriate to different network contexts (e.g., for node-level programmable switch platforms opposed to network-wide NFV frameworks). Currently, to the best of our knowledge, there is not a clear-cut candidate standing out. Indeed, the typical approach, used in block-based node platforms (such as the Click router [16], or even in Software Defined Radio platforms) of formally modelling a composition of blocks as a Direct Acyclic Graph of such blocks may be insufficient, as it does not permit the programmer to introduce the notion of "application level states" and hence dynamically change (adapt) the composition to a mutated stateful context. Languages in the If-This-Then-That family, recently introduced in completely different contexts (such as web services or Internet of Things scenarios) may find application also in the networking context, given their resemblance with the matching functionality frequently used in forwarding tasks, although, again, stateful applications may require extensions. It is worth noting that the Superfluidity project is specifically addressing novel languages formalized in terms of eXtended Finite State Machines (XFSM).

Some of the features that VNF, Virtual Machine and Compute Host Descriptors should have for the appropriate deployment of VNFs on Virtual Machines over Compute Hosts in a telco data center are described in [20].

C. RFB Execution Environment

The RFB Execution environment is responsible for deploying/executing a predefined composition. The role of the REE is to instantiate a desired RDCL script instance, control its execution, maintain and update the application-level states, trigger reconfigurations, and so on. In traditional network-wide NFV scenarios, an orchestrator generally fulfils this role. On the contrary, with VNFs implemented in software within a container, or VM, this role is fulfilled by a software entity running in the VM itself. Conversely, the case of high-speed bare metal switches, where configuration and per-flow state maintenance must occur at wireline speed, it is more problematic. As a matter of fact, in most architectures this role may not be clearly identified, and having it "generically" delegated to the overlying software/control stack may not be the most performant or effective solution, and may lead to performance bottlenecks and slow-path operation. The REE could potentially be seen as a generalization and enhancement of the NFVI (NFV Infrastructure). In the current model the infrastructure (NFVI) provides resources and an external orchestrator coordinates them (but it only instantiates VMs and connects them according to the graph that describes the network services). In the envisaged model, the enhanced infrastructure REE provides the means to execute arbitrarily complex RDCL scripts operating at different levels.

D. Mapping in Working Platforms

We now provide different representative examples on how the RFB concept may be mapped to different working platforms.

CRAN – Today, the network is a network of elements (e.g., an e-NodeB) hosting a set of functions. However, using virtualization technology to address 5G requirements, it makes sense to re-examine this functional split across set of RFBs to introduce the required 5G flexibility. In addition, the same functionality, such as authentication/authorization, is currently implemented in different network elements (i.e., packet data network gateways and mobility management entities) to ensure vendor compatibility. Through the decomposition, each functionality (authentication, authorization, radio resource management, turbo decoding, fast Fourier transform, etc.) will be dedicated to an RFB, thus allowing a modular design of 5G. Updating any RFB is now possible without interfering with other RFBs.

Mobile Edge Computing – By introducing the notion of RFB in MEC, it becomes easier to: i) scrutinize constituent elements of the aforementioned components, ii) distinguish those that can be virtualized, implemented and re-utilized in the most efficient way, and iii) provision a network capable of handling its resources with a higher degree of granularity.

Click and ClickOS - Click [16] is a software architecture that decomposes the functionality of a packet-forwarding node into elements that are interconnected to implement arbitrarily complex functions. Click elements operate on packets (e.g. at IP or Ethernet level) by performing inspections and/or modifications of the different fields. A large number (hundreds) of built-in Click elements are available, including for example Packet Classifiers, Rate Monitor, Bandwidth Shapers, and Header Rewriters. The interconnection graph of Click elements is called a configuration, which is described by a declarative language. The language allows the definition of compound elements composed of other elements. In our vision, a single Click element represents an RFB and a configuration that combines Click elements represents again (composed) RFBs. In [17], it is shown how a Click configuration can be executed in minimalistic virtual machines that run in a specialized hypervisor called ClickOS. ClickOS Virtual Machines can be seen as RFBs and can be composed to implement the desired packet processing services.

eXtended Finite State Machines - Superfluidity plans to further extend the initial work carried out in [15] for XFSMs. The goal is to devise a programmable network node architecture, which supports "full" XFSMs and allows the deployment of more complex network functions such as load balancing, stateful traffic filtering, monitoring and traffic classification tasks.

VI. Summary

We have described the general architecture of the Superfluidity project, whose main goal is to design and to implement a converged network architecture being location, hardware and time-independent. Superfluidity will push the boundaries of what is currently possible with virtualized, software-based packet processing. Additionally, we have detailed the main idea behind Reusable Functional Blocks.

Acknowledgement

This work was performed in the context of the project Superfluidity, which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°. 671566.

References

[1] A. Gupta, and R.K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies," in *IEEE Access*, vol.3, 2015

[2] C. X. Wang, *et al.* "Cellular architecture and key technologies for 5G wireless communication networks," *IEEE Communications Magazine*, vol.52, no.2, pp.122-130, 2014.

[3] L. Atzori, A. Iera, and G. Morabito. "The internet of things: A survey," *Elsevier Com. Net.,* vol.54, no. 15 pp 2787-2805, 2010.

[4] F. Boccardi, *et al.* "Five disruptive technology directions for 5G," *IEEE Comm. Magazine*, vol. 52, n.2, pp. 74-80, 2014.

[5] J. G. Andrews, *et al.,* "What will 5G be?" *IEEE Journ. Sel. Areas in Comm.* vol. 32, n. 6, pp. 1065-1082, 2014.

[6] P. Agyapong, *et al.*, "Design considerations for a 5G network architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, 2014.

[7] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51 no.11, pp. 24-31 2013.

[8] N. McKeown, "Software-defined networking," Keynote talk at *IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009.

[9] N. Fernando, S.W. Loke, and W. Rahayu, "Mobile cloud computing: A survey." *Future Generation Computer Systems* vol. 29, no.1 pp. 84-106, 2013.

[10] A. Checko, *et al.*, "Cloud RAN for mobile networks—a technology overview" *IEEE Communications Surveys & Tutorials*, vol. 17, no.1, pp. 405-426, 2015.

[11] P. Eardley (ed.), *et al.*, "Use cases, technical and business requirements", deliverable 2.1 of Superfluidity EU project, available on line at http://superfluidity.eu

[12] ETSI ISG NFV: Network Functions Virtualisation (NFV); Architectural Framework. ETSI GS NFV 002 V1.2.1. Dec 2014.

[13] P. Neves, *et al.*, "The SELFNET Approach for Autonomic Management in an NFV/SDN Networking Paradigm," *International Journal of Distributed Sensor Networks*, 2016.

[14] M. Kablan, *et al*., "Stateless network functions," *ACM HotMiddlebox*, London, UK, August 2015.

[15] G. Bianchi, *et al*., "OpenState: programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 44-5, 2014

[16] J E. Kohler, *et al.,* "The Click modular router," *ACM Transactions on Computer Systems (TOCS),* vol. 18, no. 3, 2000

[17] J. Martins *et al.*, "ClickOS and the Art of Network Function Virtualization", USENIX NSDI '14, Seattle, USA, 2014.

[18] M. Bansal, *et al.,* "Openradio: a programmable wireless dataplane," *ACM HotSDN*, Helsinki, Finland, August 2012.

[19] ETSI ISG NFV: Network Functions Virtualisation (NFV): Virtual Network Functions Architecture. ETSI GS NFV-SWA 001 V1.1.1. December 2014.

[20] ETSI ISG NFV: Network Functions Virtualisation (NFV): NFV Performance & Portability Best Practises. ETSI GS NFV-PER 001 V1.1.2. December 2014.

[21] ETSI Portal, "Mobile-Edge Computing – Introductory Technical White Paper", September 2014, Available online on https://portal.etsi.org