Container Network Functions: Bringing NFV to the Network Edge

Richard Cziva, *Student Member, IEEE,* and Dimitrios P. Pezaros, *Senior Member, IEEE*

***Abstract*—In order to cope with the increasing network utilization driven by new mobile clients, and to satisfy demand for new network services and performance guarantees, telecommunication service providers (TSPs or telcos) are exploiting virtualization over their network by implementing network services in Virtual Machines (VMs), decoupled from legacy hardware-accelerated appliances. This effort, known as Network Function Virtualization (NFV) reduces OPEX and provides new business opportunities.**

At the same time, next generation mobile, enterprise, and Internet of Things (IoT) networks are introducing the concept of computing capabilities being pushed at the network edge, in close proximity of the users. However, the heavy footprint of today's NFV platforms prevents them from operating at the network edge. In this article, we identify the opportunities of virtualization at the network edge and present Glasgow Network Functions (GNF), a container-based NFV platform that runs and orchestrates lightweight container vNFs, saving core network utilization and providing lower latency. Finally, we demonstrate three useful examples of the platform: IoT DDoS remediation, on-demand troubleshooting for telco networks, and supporting roaming of network functions.

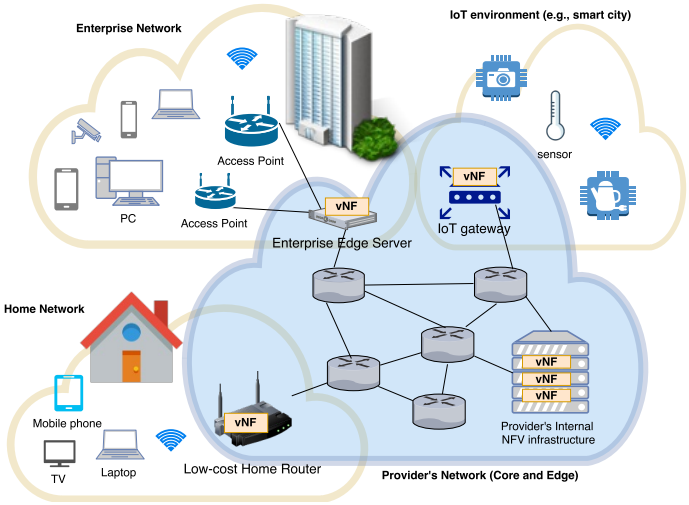


Fig. 1: Edge network examples

I. Introduction

Data consumption is growing exponentially in today's communication networks. This irreversible trend is driven by the increase of end-users and the wide-spread penetration of new mobile devices (e.g., smartphones, wearables, sensors and more). In addition, mobile data consumption is also accelerated by the increased capabilities of the mobile clients (e.g., higher resolution screens and HD cameras) and the user desire for high-speed, always-on, multimedia-oriented connectivity. In numbers, it has been estimated that connected devices will exceed 50 billion, generating zettabytes (1 billion terabytes) of traffic yearly by 2020.

At the same time, the telecommunication service provider (TSP) market is becoming competitive with the rise of many over-the-top service providers lowering subscription fees for users. Moreover, today's TSPs often experience poor resource utilization, tight coupling with specific hardware, and lack of flexible control interfaces which fail to support diverse mobile applications and services. As a result, TSPs have started to lose existing and new revenue, while suffering increased capital and operational expenditure that cannot be balanced by increasing subscription costs [1].

In order to cope with the aforementioned challenges, service providers have started to softwarise their network infrastructure. By virtualizing traditional network services (e.g., firewalls, caches, proxies, intrusion detectors, WAN accelerators, etc.), providers can save operational and capital expenses, and satisfy user demands for customized and rapidly evolving services. This transformation, referred to as Network Function Virtualization (NFV), transforms how operators architect their network to decouple network functionality from physical locations for faster and flexible network service provisioning [1]. NFV has gained significant attention since its first appearance in 2012, resulting in many, albeit still preliminary deployments at the provider's data centers.

While NFV is gaining attention, a new, fifth generation mobile architecture (5G) is being designed to support the increased user demand mentioned above [2]. As a key design objective, 5G mobile networks will utilise Mobile (or Multi-Access) Edge Computing (MEC), an IT service environment with cloud-computing capabilities at the edge of the home, enterprise or IoT network, within close proximity to the mobile subscribers [3], as shown in Figure 1. While there have been a few proof of concept (PoC) implementations of MEC (including, e.g., a video streaming deployment at the Wembley Arena[1]), these PoCs do not present a generic NFV architecture nor do they support today's customer edge devices (e.g., home routers).

In this article, we present Glasgow Network Functions (GNF), an NFV platform that brings NFV and edge computing together by using generic, lightweight Linux containers to host vNFs in a distributed, heterogeneous edge infrastructure. We present three useful applications and show that by utilising the

[1]https://blog.networks.nokia.com/mobile-networks/2015/12/04/ee-mobile-edge-computing-ready-rock-wembley-stadium/ (Accessed on: 05/03/2017)

Customer Device	Released	Architecture	CPU	Memory
Residential CPE Home Routers				
Virgin SuperHub 3 (Arris TG2492S)	2015	Intel Atom	2x1.4 GHz	2x256 MB
Google Fiber Network Box GFRG110	2012	ARM v5	1.6 GHz	not known
Orange Livebox 4	2016	Cortex A9	1 GHz	1 Gb
Commodity Wireless Routers				
TP-LINK Archer C9 home router	2016	ARM v7	2x1 GHz	128 MB
Ubiquiti EdgeRouter Lite 3	2014	Cavium MIPS	2x500 MHz	512 MB
Netgear R7500 Smart Wifi Router	2014	Qualcomm Atheros	2x1.4 GHz	384 MB
IoT Edge Gateways				
Dell Edge Gateway 5000	2016	Intel Atom	1.33 GHz	2GB
NEXCOM CPS 200 Insdustrial IoT Edge Gateway	2016	Intel Celeron	4x2.0 GHz	4GB
HPE Edgeline EL4000	2016	Intel Xeon	4x3.0 GHz	up to 64GB

TABLE I: Example edge device specifications

network edge (e.g., home, enterprise, IoT edge), providers can alleviate their unnecessary core network utilization (which can correspond to savings of millions of dollars per year), perform better troubleshooting on their network, and provide location-transparent services to their users.

II. Opportunities at the Network Edge

An edge device provides the entry point to an enterprise or service provider network that is generally faster and more efficient. Edge devices include wireless routers and switches, mobile access devices (e.g., base stations) and even IoT gateways that connect IoT devices to the Internet. As edge devices are located close to the users, services running at the edge provide higher forwarding performance (high throughput, low latency) than running services remotely, and therefore save the utilization of the WAN infrastructure.

A. Edge Device Evolution

In recent years, edge devices have become smarter and their capabilities have increased to run advanced network services, such as Quality of Service (QoS) differentiation, parental control filters, bandwidth reservations, and other multi-service functions. In Table I, a few popular edge devices are presented alongside their released date, architecture, CPU and memory parameters. The list includes large-scale residential customer premise equipment (CPE) from the UK (Virgin), US (Google Fiber) and France (Orange). In addition, we have added a few low-cost commodity home routers to this list along with three IoT gateway devices from HP Enterprise, Dell, and NEXCOM. As it can be observed from Table I, recent CPE devices and home routers are equipped with powerful computing capabilities (e.g., CPUs up to 1.6 GHz) and sizeable RAM (up to 1GB) to run a Linux-based operating system (e.g., OpenWRT or DD-WRT), and some lightweight network functionality. As demonstrated in our previous work [4], even a commodity TP-Link home router with 560 MHz CPU and 128MB of RAM can be used to run multiple vNFs using Linux containers (our demo showed rate limiting, content filtering, and firewall vNFs). Apart from the "low-cost" edge devices like home routers and residental CPE, some vendors have also introduced IoT gateways with high-end CPUs and up to 64 GB of RAM to accommodate new services such as intelligent analytics at the edge of the network. We envision all of these devices (residential CPEs, home routers and IoT gateways) along with other in-network NFV servers to be part of a distributed NFV infrastructure.

B. Related NFV platforms

While the network edge has many benefits, traditional NFV platforms have been built on top of commodity servers, mainly exploiting Virtual Machines (VM)s (using technologies such as, e.g., XEN or KVM) for vNFs. Table II presents a summary of the features supported by some existing solutions that more closely relate to the scope of our work. The information presented reflects the public information available at the time of writing.

Cloud4NFV [5] is a platform that promises to deliver a novel service to end customers by building on top of cloud, SDN and WAN technologies. Although we share a similar vision with Cloud4NFV in providing end-to-end service management with function chaining and traffic steering, we advocate the use of containers for vNFs, support roaming vNFs, and exploit the capabilities of low-cost edge devices distributed at provider's scale. The UNIFY [6] and T-NOVA [7] research projects share a similar vision of unifying the cloud and provider networks by implementing a "Network Functions as a Service" system. The OPNFV Linux foundation project is the most popular open source NFV platform with support and deployments from numerous vendors and large-scale providers. While all these platforms have made important contributions to the field, none of them have presented a container-based, edge-centric and mobility-focused NFV system so far.

III. Container Network Functions

Recently, new, lightweight virtualization technologies have been proposed for NFV, including containers, specialized VMs (unikernels) and minimalistic distribution of general-purpose VMs. These lightweight technologies could typically avoid the hardware requirements and overheads associated with hypervisors and VMs.

	GNF	**Cloud4NFV** [5]	**UNIFY** [6]	**T-NOVA** [7]	**OPNFV**
Virtualization Technology	Container	VM	VM	VM	VM
End-to-End Service Mgmt	Yes	Yes	Yes	Yes	Yes
Distributed infrastructure	Yes	Yes	Yes	Yes	Yes
Traffic Steering	Yes	Yes	Yes	No	Yes
Runs on the network edge	Yes	No	No	No	No
SFC support	Yes	Yes	Yes	No	Yes
Roaming vNFs	Yes	No	No	No	No

TABLE II: Summary of existing approaches

A. NF virtualization alternatives

As their main advantage, traditional VMs (used by, e.g., the Cloud4NFV [5] platform) allow vNF users to specify their operating system for each individual network function, while specialized VMs (used by the ClickOS [8] system) and containers (used by GNF) need to run on a specific platform. In case of specialized VMs, vNFs are compiled to a binary that can only be executed on a purpose-built hypervisor. While this approach provides high performance, depending on a custom hypervisor limits deployability and, moreover, specialized VMs restrict vNFs that need to be implemented on a specific software environment (i.e., Click in terms of ClickOs). We believe that containers are a good compromise between specialized and commodity VMs as they allow generic software to be used for vNFs. At the same time, they incur significantly lower overhead than traditional VMs and can be deployed in any Linux environment (available from commodity routers to high-end servers) with similar performances to the host machine. Similar to specialized VMs, containers also allow much higher network function-to-host density and smaller footprint at the cost of reduced isolation. Using containers, commodity compute devices (or public cloud VMs) are able to host up to hundreds of vNFs as shown in [9], [10].

B. Performance of container vNFs

In Figure 2, we highlight some basic characteristics of container vNFs that we measured on a commodity Intel i7 server with 16GB of memory. While these measurements highlight performance characteristics most relevant to our framework and use-cases, we refer interested readers to a more comprehensive evaluation of container-based vNFs reported in [10] [11].

1) Delay: Keeping the delay introduced by vNFs low is important in order to implement transparent services and therefore it is a key benchmark for vNF technologies. In Figure 2 (a), we express the delay introduced by different virtualization platforms through showing idle round-trip-time (RTT). While ClickOS reaches slightly lower delay than containers, ClickOS is built on top of a modified, specialized hypervisor that optimises packet forwarding performance. On the other hand, container-based functions use unmodified containers on a standard Linux kernel, hence allowing deployment on devices that do not support hardware virtualization (e.g., all the CPE devices and home routers). As it is also shown, other VM-based technologies such as KVM or XEN VMs result in a much higher delay, which is mainly attributed to the packet copy from the hypervisor to the VMs.

2) Instantiation time: In order to provide high flexibility for placement and vNF migration, instantiation time is crucial. Figure 2 (b) shows the time required to create, start, and stop containers versus creating XEN VMs. As shown, 50 container vNFs can be created and started in 10 seconds, while it takes more than 40 seconds just to create the same amount of XEN VMs that have not even booted up. Clearly, traditional VM technologies are not suitable for highly multiplexed, highly mobile vNFs, since roaming clients would require new vNFs to be set up and ready to forward traffic in a matter of seconds.

3) Memory requirements: Since we are designing vNFs for devices with ca. 512 MB of memory (as shown in Table I), it is important to compare memory requirements of containers with other virtualization technologies. Here, we have chosen to compare only with ClickOS specialized VMs, since traditional VMs would consume memory in the order of 100s of MBs per VM (depending on the installed OS and the statically assigned memory). As it is highlighted in Figure 2 (c), the idle memory requirement for one container is about 2.21MB, which linearly scales to only 221MB with 100 idle containers. As it is also shown, ClickOS requires more than twice the amount of memory per vNF.

IV. Glasgow Network Functions

Glasgow Network Functions (GNF)[2] [12], [9], [4] is a container-based NFV platform designed for next generation networks. It exploits lightweight, container vNFs deployed as close to the users as possible by using a programmable network edge. GNF has the following main characteristics:

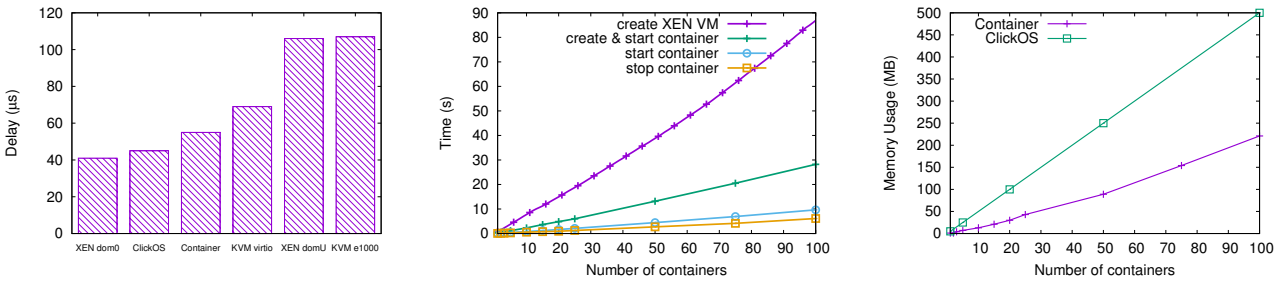
Container-based: vNFs are encapsulated in lightweight Linux containers to provide fast instantiation time, platform-independence, high throughput and low resource utilization.

Minimal footprint: GNF vNFs runs at very low-cost (e.g., taking only a few MBs of memory), allowing its deployment on commodity and low-end devices that do not support hardware-accelerated virtualization.

Support for vNF roaming: With its small footprint and encapsulated functions, GNF vNFs seamlessly follow users between cells, providing a consistent and location-transparent service.

End-to-end transparent traffic steering: Providers can

[2]https://netlab.dcs.gla.ac.uk/projects/glasgow-network-functions (Accessed on: 05/03/2017)



(a) Idle ping delays (b) Create, start and stop times (c) Idle memory consumption of container NFs

Fig. 2: Performance evaluation of container NFs.

attach and remove vNF chains transparently without adversely impacting the flow of traffic.

Figure 3 provides a high level overview of the proposed system. As shown, the overall architecture is organized in four planes: infrastructure plane (consisting of the edge devices and the central NFV infrastructure where vNFs can be hosted), the virtual infrastructure management plane (VIM), orchestration plane, and a high-level service plane. We provide more details below on three planes where GNF resides: the service, orchestration and VIM planes.

A. Service plane

The service plane provides high-level administrator access to GNF. It allows providers to either directly call the GNF Manager API or use the User Interface to manage vNF chains.

The GNF *User Interface (UI)* gives a graphical representation of all the connected edge devices as well as all the connected User Equipment (UE). The UI operates by calling the REST API of the Manager, and gives the ability to specify vNF chains and assign them to selected traffic of UE. Traffic to be forwarded through vNF chains can be selected with OpenFlow 1.3 match properties [13]. For instance, an operator can specify an intrusion detection vNF to be assigned to all HTTP (port 80) traffic from a particular mobile phone (identified by its MAC address).

Apart from creating vNF chains and assigning them to mobile clients, the UI also displays notifications sent from the vNFs. As an example, a notification can tell the provider if an intrusion has been detected by any of the intrusion detection vNFs. The notifications received from the vNFs are linked to the mobile clients that help catching users performing malicious activity. Notifications can be acknowledged, deleted, and muted for a specific vNF by the operator.

B. Orchestration plane

The orchestration plane is implemented in the *GNF Manager* that has network-wide knowledge of all vNF locations and usage statistics from all managed devices. The Manager provides a set of REST APIs to start, stop and migrate container vNFs, and keeps a live HTTP connection with all the Agents to retrieve health statistics used by the orchestration algorithms. The Manager corresponds to the NFV Orchestrator (NFVO) component of the ETSI MANO architecture [1]. In our proof-of-concept implementation, Agents send connection events, WiFi signal statistics (signal strength, packet counters), and CPU and memory utilization of the device read from Linux kernel statistics. Also, the Manager stores notifications sent from the vNFs (relayed through the Agent) and provides this information to the User Interface.

As the network-wide location of all vNFs and temporal load statistics from all edge devices and the central NFV infrastructure are available at the Manager, an orchestration algorithm is used to allocate new vNFs to optimal locations. When a new vNF is requested, GNF generally tries to host the vNF as close as possible to the user to save the most in overall network utilization. If no edge device in close proximity is suitable, GNF hosts the vNF in the central NFV infrastructure. When a user migrates between edge devices and has vNFs associated or the utilization of one of the edge devices changes (drops down or increases by 15%), GNF runs the orchestration algorithms again for the corresponding vNF chains (for the vNF chains of the user or the vNF chains associated with the edge device) to optimise placement.

C. Virtual Infrastructure Management plane

This plane of the architecture handles the network connectivity between edge devices (and a central NFV infrastructure) and the management (start, stop, migrate, remove, upgrade) of vNFs running on these devices. The two main components performing these actions are the Agent and the Network Controller detailed here alongside with details on the traffic classification used in GNF.

1) Agent: The *Agent* is a lightweight daemon running on the edge devices and at the central NFV infrastructure managed by the provider. It is responsible for the instantiation of the vNFs, and for reporting periodically the state of the device to the Manager. The Agent corresponds to the VNF Manager (VNFM) and Virtual Infrastructure Manager (VIM) components of the ETSI MANO architecture [1].

When a new vNF is requested by a user, the Manager notifies the most suitable Agent which retrieves (if not already available locally) the NF from a central repository and starts the vNF in a container. If the Agent is hosted on an edge device that handles user connections, it also listens and reports all client connections to the Manager by subscribing for instance

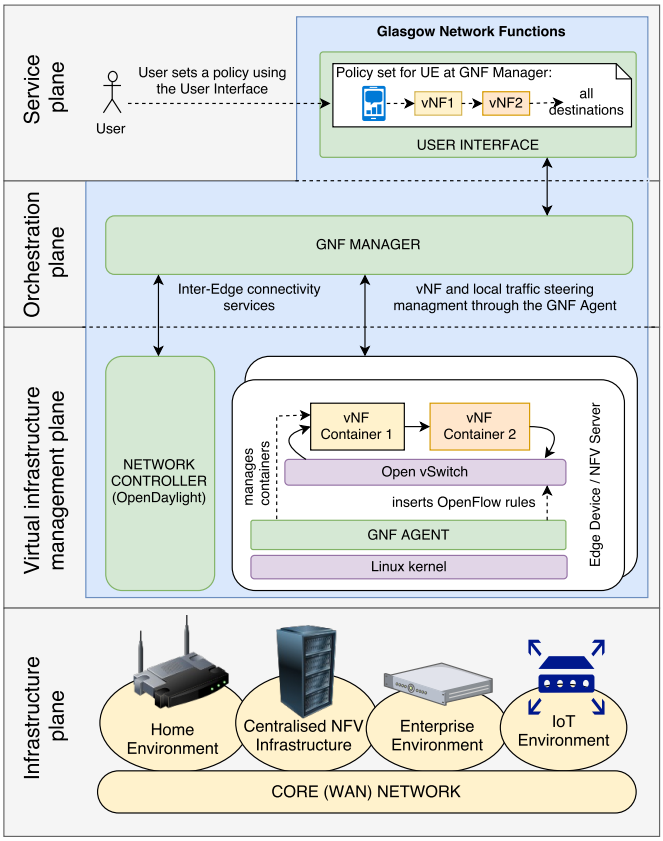


Fig. 3: The GNF platform - an overview

to HostAPD (Host Access Point Daemon) events. In case a client leaves the edge (e.g., user roams to an other edge device or disconnects from the network), the Manager gets notified to either stop, re-locate or leave the vNF running at the same edge. When a client is connected, the Manager is notified to decide whether vNFs need to be started for this client or not.

On top of the lifecycle management of vNFs, the Agent is responsible for setting up the containers' virtual interfaces inside the hosting devices. In GNF, all container vNFs are connected to a local software switch (Open vSwitch) by two virtual Ethernet pairs, where one interface pair is used to receive traffic at the vNF and the other is used to send traffic from the vNF. The connection between edge devices is done by the Network Controller.

2) Network Controller: The *Network Controller* is an SDN controller that manages transparent traffic redirection between edge devices and the central NFV infrastructure that spans across the provider's network. This component is built on top of OpenDaylight, an open-source, carrier-grade SDN platform that gained acceptance from telco providers. Our OpenDaylight module uses OpenFlow to manage traffic redirection by inserting/deleting flow entries that do not modify the original packets. We have implemented a transparent, hop-by-hop redirection (that does not break existing connections) through the provider's network where flow entries are matching on input ports and forwarding packets on output ports, however, tunneling techniques can also be used to manage traffic between distributed Open vSwitch instances.

3) Traffic classification: Classification is required to match and forward packets to appropriate vNFs. To support fine-grained yet standard and high-performance classification, GNF relies on OpenFlow 1.3 classifiers that allow packets to be matched on properties such as input port, Ethernet, MPLS, ARP, IP, TCP and UDP headers. Traffic classification can also exploit flow priorities provided by OpenFlow, meaning that operators can set overlapping classifiers with different priorities.

V. Use Cases

In this section, we present three example use cases of the proposed GNF platform.

A. IoT DDoS mitigation

The Internet of Things (IoT) is a proposed development where everyday objects run software equipped with network connectivity that allows them to send and receive data. Recently, there has been an increase of such devices in billions of households and in many "smart cities" installations. Example devices of this architecture include security cameras, lightbulbs, smart TVs, weather sensors.

Recently, a historically large number of distributed denial-of-service (DDoS) attacks have been built on exploiting vulnerabilities of insecure routers, IP cameras, digital video recorders and other unprotected devices. This malware, dubbed "Mirai", spread to vulnerable devices by continuously scanning the Internet for IoT systems protected by factory default or hard-coded usernames and passwords. In one of the recent attacks, an Akamai-hosted website peaked at an unprecedented 665 Gbps (and 143m pps), and resulted in the website taken offline due to the financial implications of the extensive network utilization[3]. Another similar, unseen IoT DDoS attack on 21st October 2016 has caused widespread disruption of legitimate Internet activity since insecured IoT devices directed large amount of bogus traffic to DNS services[4].

We advocate that such distributed attacks from IoT devices can be efficiently mitigated by providers with a distributed NFV platform (similar to the one proposed in this article) that utilises the network edge as the first point of controlled entry to the provider's network. As a GNF vNF can be deployed on generic home or IoT gateways (also called as "capillary gateways"), malicious traffic can be blocked in a matter of seconds by creating a new iptables-based GNF vNF and setting up DROP rules on the selected traffic. Blocking traffic at the customer edge is not only easy even after the attack is launched, but it also avoids unnecessary core network utilization that costs millions of dollars to serve. Moreover, edge vNFs can reduce the complexity of securing new applications and devices in the future by automating proactive security configuration in-the-network.

[3]https://blogs.akamai.com/2016/10/620-gbps-attack-post-mortem.html (Accessed on: 05/03/2017)

[4]https://www.theguardian.com/technology/2016/oct/21/ddos-attack-dyn-internet-denial-service (Accessed on: 05/03/2017)

B. Distributed, on-demand measurement and troubleshooting

Current telecommunication networks face the difficult task of maintaining an increasingly complex network and at the same time introducing new technologies and services while keeping expenditure low. According to recent studies, configuration of network access control is one of the most complex and error-prone network management tasks that are hard to identify (unless a user complains) and require highly skilled engineers to fix [14]. As these misconfigurations become the main source of network unreachability and vulnerability, providers seek ways to perform customer-centric and automated troubleshooting of their network.

Through using a platform like GNF, operators can install small vNFs at different points in the network (e.g., customer edge or VNF servers at the core) that perform basic troubleshooting actions using simple tools like *ping*, *traceroute* or *tcpdump*, that are lightweight and are available in a Linux kernel. While performing similar actions today takes long manual setup effort and involvement of an engineer, GNF can allow collecting troubleshooting data (alarms, routing tables, configuration files, etc.) from multiple points in the network in an automated or on-demand fashion. This can reduce operational expenses and result in a faster problem identification and mitigation. For proof of concept, we have implemented network monitoring vNFs for this use-case that can be found in our GitHub repository[5].

C. Roaming Network Functions

5G cellular systems are expected to deploy and overlap different types of cells, such as small/spot cells that utilise high frequency (5 GHz or above) to support high capacity transmission with limited spectrum sharing. While these small cells offer high performance, they increase roaming between cells. Hence, to efficiently support users with customized network services, we advocate that network services should also migrate between cells, following the UE.

As shown in Figure 4, GNF allows vNF chains to be associated with a particular UE. In the presented scenario, a simple chain with 2 vNFs is assigned to any traffic leaving the UE. As shown, once the UE is connected, these services can be co-located to a nearest edge device, called Home Router 1. In case of roaming between cells, the GNF Manager recomputes the allocation of vNFs and initiates migration if a more optimal placement can be reached. In our example case, one vNF is placed to the edge device closer to the user, while one vNF has been migrated to a central NFV infrastructure (since the closest edge device is considered overloaded by the GNF Manager). This migration scenario has been demonstrated in our previous work [4] with rate limiters, firewalls and parental filter vNFs.

VI. Discussion

While containers provide many benefits for NFV, there are technology-related challenges to be considered when choosing containers as a platform for vNFs.

[5]https://github.com/UofG-netlab/gnf-dockerfiles (Accessed on: 05/03/17)

A. vNFs for the network edge

As edge devices have relatively low capabilities compared to traditional NFV servers, some of today's vNFs cannot be run on the edge. However, according to recent research [15], many virtual appliances in service provider data centers are simple packet or application firewalls and gateways that can be implemented on these resource-constrained devices. Also, it is important to note that next generation, container-based vNFs will be tailored to a single or a small group of clients (providing customized services), while traditional vNFs handle aggregate traffic from many clients.

B. Security and isolation

Containers typically offer weaker isolation between co-located instances than traditional VMs. While this has many benefits on the performance and agility of the vNFs, it can potentially result in interference between vNFs if deployed without proper resource guarantees, as analyzed in detail by ETSI [11]. However, deploying with OS-level security measures (such as, e.g., using SELinux with access control security policies support, and using AppArmor to set per-program restricted access profiles) containers can be mature enough for production environments. Recently, security improvements have also focused on minimal host OS distributions for reducing the attack surface while executing host management tools in isolated management containers [11].

C. Management framework

Containers for the network edge introduce additional management and orchestration challenges, since many small vNFs can potentially replace the few large vNFs that we see in NFV frameworks today (also known as the micro-services architecture). Moreover, vNFs will need to be managed over a distributed, heterogeneous infrastructure that the edge forms, which further increases the complexity of placement and orchestration algorithms.

VII. Conclusion

Most NFV platforms have been targeted towards exploiting traditional or specialized VMs for hosting vNFs typically found in remote, over-provisioned data centers. However, as a programmable network edge is gaining traction with the rise of the next generation mobile networks (5G), there is a need for lightweight NFV technologies that can exploit the benefits the edge offers (e.g., localized, high-throughput, low-latency network connectivity). In order to bring NFV to the network edge, we have proposed the Glasgow Network Functions (GNF), an NFV platform built on top of standard Linux containers that are lightweight enough to run on a variety of edge devices. By running GNF vNFs on the edge of next-generation enterprise, mobile, and IoT networks, service providers have the ability to run customized, high-performance network services while reducing the increasing cost of core network management and operations.

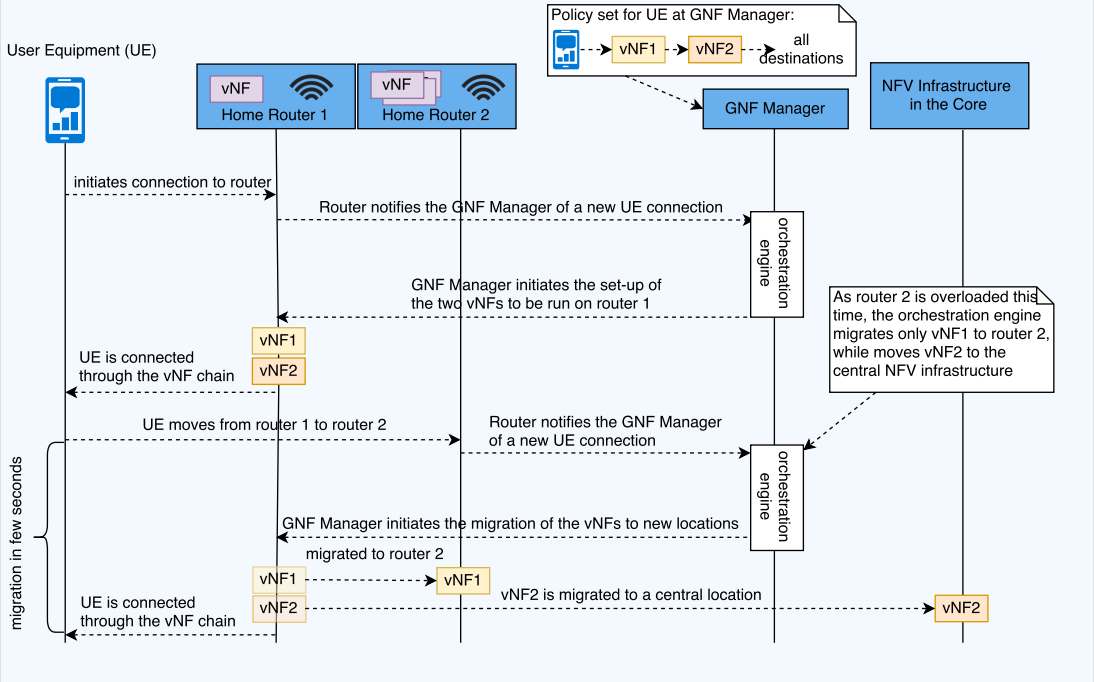


Fig. 4: vNF migration timeline

ACKNOWLEDGMENTS

The authors would like to thank Simon Jouët for his contributions to earlier versions of GNF. The work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/L026015/1, EP/N033957/1, EP/P004024/1, and EP/L005255/1, and by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication services protecting end-user applications from disaster-based failures.

REFERENCES

[1] M. Chiosi, et al., Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action, ETSI White paper (2012).
[2] S. Abdelwahab, B. Hamdaoui, M. Guizani, T. Znati, Network function virtualization in 5G, IEEE Communications Magazine 54 (4) (2016) 84–91.
[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile edge computing - a key technology towards 5g, ETSI White Paper 11 (2015).
[4] R. Cziva, S. Jouet, D. P. Pezaros, Roaming Edge vNFs using Glasgow Network Functions, in: Proc. of ACM SIGCOMM, 2016, pp. 601–602.
[5] J. Soares, M. Dias, J. Carapinha, B. Parreira, S. Sargento, Cloud4nfv: A platform for virtual network functions, in: Proc. of IEEE CloudNet, 2014, pp. 288–293.
[6] A. Császár, W. John, M. Kind, C. Meirosu, G. Pongrácz, D. Staessens, A. Takács, F.-J. Westphal, Unifying cloud and carrier network: Eu fp7 project unify, in: Proc. of IEEE/ACM UCC, 2013, pp. 452–457.
[7] G. Xilouris, M. A. Kourtis, M. J. McGrath, V. Riccobene, G. Petralia, E. Markakis, E. Palis, A. Georgios, G. Gardikis, J. F. Riera, A. Ramos, J. Bonnet, T-NOVA: Network functions as-a-service over virtualised infrastructures, in: Proc. of IEEE NFV-SDN, 2015, pp. 13–14.
[8] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, F. Huici, ClickOS and the art of network function virtualization, in: Proc. of USENIX NSDI, 2014, pp. 459–473.
[9] R. Cziva, S. Jouet, D. P. Pezaros, GNFC: Towards Network Function Cloudification, in: Proc. of 2015 IEEE NFV-SDN, 2015, pp. 142–148.
[10] A. Ghanwani, D. Krishnaswamy, R. R. Krishnan, P. Willis, N. Sriram, A. Chaudhary, F. Huici, An Analysis of Lightweight Virtualization Technologies for NFV, Internet-Draft draft-natarajan-nfvrg-containers-for-nfv-03 (Accessed on: 06/03/2017), Internet Engineering Task Force (Jul. 2016).
[11] ETSI, DGS/NFV-EVE004. Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework (2016).
[12] R. Cziva, S. Jouet, K. J. S. White, D. P. Pezaros, Container-based network function virtualization for software-defined networks, in: Proc. of IEEE ISCC, 2015, pp. 415–420.
[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling Innovation in Campus Networks, ACM SIGCOMM Computer Communication Review 38 (2) (2008) 69–74.
[14] R. Alimi, Y. Wang, Y. R. Yang, Shadow configuration as a network management primitive, ACM SIGCOMM Computer Communication Review 38 (4) (2008) 111–122.
[15] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, V. Sekar, Making middleboxes someone else's problem: network processing as a cloud service, ACM SIGCOMM Computer Communication Review 42 (4) (2012) 13–24.

Richard Cziva [M] (r.cziva.1@research.gla.ac.uk) received a B.Sc. degree in Computer Engineering from the Budapest University of Technology and Economics, Hungary, in 2013. He is a final-year PhD student at the School of Computing Science, University of Glasgow. His research focuses on the development and orchestration of lightweight, container-based NFV frameworks. He has worked with wide-area network providers such as NORDUnet and REANNZ, won numerous travel grants, and received two best paper awards.

Dimitrios P. Pezaros [SM] (dimitrios.pezaros@glasgow.ac.uk) is Senior Lecturer (Associate Professor) and director of the Networked Systems Research Laboratory (netlab) at the School of Computing Science, University of Glasgow. His research focuses on the resilient and efficient operation of future virtualized networked infrastructures through the exploitation of programmable technologies such as SDN and NFV. He is a Chartered Engineer, and holds B.Sc. (2000) and Ph.D. (2005) degrees in Computer Science from Lancaster University, UK.