Roaming Edge vNFs using Glasgow Network Functions

Richard Cziva, Simon Jouet, Dimitrios P Pezaros
Networked Systems Research Laboratory, University of Glasgow, UK
richard.cziva@glasgow.ac.uk, simon.jouet@glasgow.ac.uk, dimitrios.pezaros@glasgow.ac.uk

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Management

1. INTRODUCTION

The significant growth in mobile data consumption driven by the exponential penetration of smart devices running resource-intensive applications poses serious challenges to the infrastructure and its dynamic (re)configuration.

It is predicted that 50 billion connected devices will be exchanging Zettabytes of traffic between them by 2020. However, more than 70% of mobile data consumption is expected to be highly localized, occurring indoors in homes, offices, shopping centers, and other public places [4]. The 5G mobile network architecture is being designed to support the emerging requirements mentioned above. As shown in Fig. 1, 5G networks will utilize smaller and denser network cells that combine various access technologies (e.g., licensed LTE or millimeter-wave), and will introduce compute nodes at the "network edge" to offload the core network and provide customized services to users at low latency and high throughput [1]. Such nodes at the network edge will be diverse, ranging from low-cost customer equipment to high-end servers. Low-overhead and high-density Network Function Virtualization (NFV) frameworks can therefore leverage resources at the edge by dynamically allocating network services such as firewalls, caches, rate limiters, etc., and remove the need for dedicated hardware which would be prohibitively expensive if not impossible to deploy over such large-scale distributed setup.

Using commodity hosting platforms, network services can be placed on demand to optimized locations, reducing capital and operational expenditure and resulting in a more efficient network-wide resource utilization. However, current network edge services and NFV frameworks (e.g., OpenEdge or ClickOS) either rely on specialised hypervisors or utilise commodity x86 servers using resource-hungry Virtual Machines, preventing their use in future wide-area and 5G networks where high network function density and mobility is paramount.

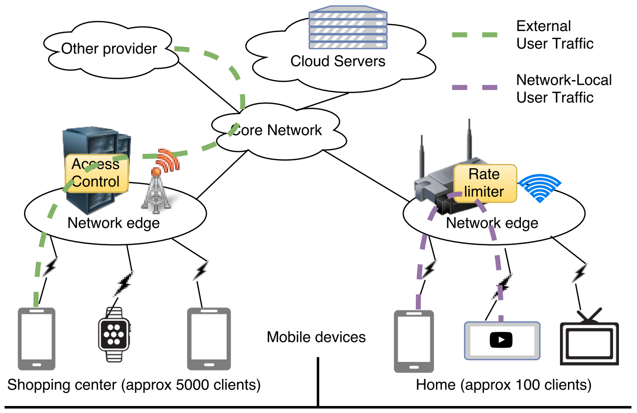

Figure 1: High-level view of 5G networks with example network functions at the edge.

In this demo, we showcase Glasgow Network Functions (GNF) [3], a virtualization framework for next generation mobile networks that exploits lightweight virtual network functions (NFs or vNFs) deployed at the edge. GNF has the following main characteristics:

Minimal footprint: GNF has been designed to run at very low-cost allowing its deployment on commodity and low-end devices.

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

Function roaming: With its small footprint and encapsulated functions, GNF seamlessly moves the NFs when the user roams between cells, providing consistent and location-transparent service.

Transparent traffic handling: Providers can transparently attach and remove NFs to the clients without adversely impacting the flow of traffic.

SIGCOMM'16, August 22–26, 2016, Florianopolis, Brazil.
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4193-6/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2934872.2959067

2. CONTAINER-BASED NFV

While different software-based NF implementations such as ClickOS, FlowOS or Vyatta exist, they run on customized platforms or hypervisors preventing widely used applications and tools to be deployed without modification. GNF uses Linux containers that provide a lightweight equivalent to virtual machines, allowing each container to use the host OS kernel to isolate processes, network routing tables, and their associated resources. This approach does not require each isolated function to run on a separate OS image, hence allowing a 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 [2]) are now able to host up to hundreds of NFs.

The minimal cost of starting and stopping containers as well as the single package encapsulation allows for NFs to roam alongside the client. As the client roams between cells, an equivalent function can be started on the newly assigned cell and removed from the previous cell.

3. DESIGN AND IMPLEMENTATION

GNF consists of three well-defined components: the Manager, the Agent, and the User Interface (UI).

The *Manager*, allows single or chain of NFs to be associated with a subset of a selected client's traffic. This is achieved by providing a set of APIs to control the state of NFs' containers across all stations and keeping a connection with all the Agents in the network. The Manager is also responsible for continuously monitoring the health and resource utilization from the GNF stations, allowing the provider to detect resource-hotspots and therefore the part of the infrastructure that should be upgraded. Using the same API, individual NFs can relay notifications through their local Agent to the Manager, informing the provider about events that should be reviewed such as an unexpected or inconsistent NF state or expected but anomalous events such as an intrusion attempt or detected malware.

A GNF *Agent* is a lightweight daemon running on the stations managed by the provider. It is responsible for the instantiation of the NFs on the hosting platform (e.g., edge router, access point, Internet gateway), notifying the Manager of clients' (dis)connection and reporting periodically the state of the device. When a NF is requested by the provider, the Manager notifies the closest Agent that retrieves (if not already hosted locally) the NF from a central repository and starts it in a container. Apart from starting and stopping NFs, the Agent is responsible for setting up the containers' local virtual interfaces. All containers are connected to the local software switch by two virtual Ethernet pairs (for ingress/egress traffic, respectively).

The *UI* provides the overall management interface for the system through a direct connection to the Manager's API. Using a simple interface, the entire network health, status, and notifications can be monitored, including the number of online stations, connected clients, enabled NFs, and current processing and network resource consumption. New NFs can be attached in seconds or removed from clients as well as scheduled to be enabled only during specific time periods.

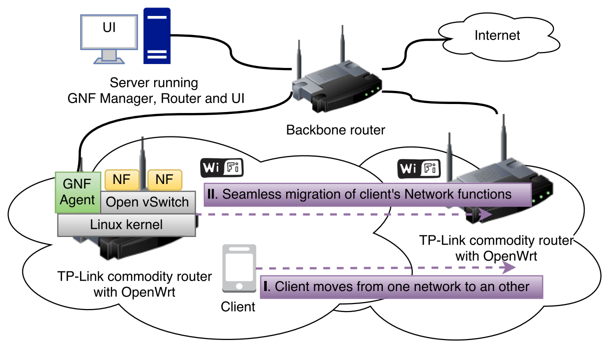
UI
Internet
Server running
GNF Manager, Router and UI
Backbone router
GNF Agent
NF
NF
Open vSwitch
Linux kernel
II. Seamless migration of client's Network functions
TP-Link commodity router with OpenWrt
TP-Link commodity router with OpenWrt
Client
I. Client moves from one network to an other

Figure 2: Architecture and demo of GNF. When a client roams between networks, associated NFs seamlessly migrate with it.

4. MOBILITY USE-CASE

As a proof of concept demonstration, we are presenting the migration of multiple lightweight NFs attached to mobile clients (smartphones) roaming between wireless networks. Throughout the demo, our UI will be used to both show real-time statistics (network traffic, CPU load, memory usage) as well as to assign (deploy) NFs to wireless clients. NFs will be hosted on off-the-shelf home routers (TP-Link WDR3600) with OpenWRT compiled to support the containers and all of the Linux utilities (such as *ip*, *tc*, *nfqueue*) our NFs use. A high-level overview of the demo setup is shown in Fig. 2. The available GNF NFs include an iptables-based packet firewall, a HTTP filter and a DNS load balancer[1].

A video presenting the mobility use-case and the UI can be viewed at: https://youtu.be/R7PimtddIXo.

5. REFERENCES

[1] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart. Networks and devices for the 5g era. *Communications Magazine, IEEE*, 52(2):90–96, 2014.

[2] R. Cziva, S. Jouet, and D. P. Pezaros. Gnfc: Towards network function cloudification. In *IEEE NFV-SDN*, 2016.

[3] R. Cziva, S. Jouet, K. J. S. White, and D. P. Pezaros. Container-based Network Function Virtualization for Software-Defined Networks. In *IEEE ISCC*, 2015.

[4] Qualcomm. 1000x: More Small Cells whitepaper, 2014.

[1]All GNF NFs can be found at: https://github.com/glanf/