Chapter 5

Networking Solutions for Integrated Heterogeneous Wireless Ecosystem

Roman Florea, Aleksandr Ometov, Adam Surak,
Sergey Andreev and Yevgeni Koucheryavy

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/67084

Abstract

As wireless communications technology is steadily evolving to improve the offered connectivity levels, additional research on emerging network architectures is becoming timely to understand the applicability of both traditional and novel networking solutions. This chapter concentrates on the utilization of cloud computing techniques to construct feasible system prototypes and demonstrators within the rapidly maturing heterogeneous wireless ecosystem. Our first solution facilitates cooperative radio resource management in heterogeneous networks. The second solution enables assisted direct connectivity between proximate users. The contents of the chapter outline our corresponding research and development efforts as well as summarize the major experiences and lessons learned.

Keywords: traffic offloading, heterogeneous network, SDN, LTE, WiFi, 5G

1. Introduction

1.1. Motivation

This chapter describes application of cloud computing techniques to improve technology research process with examples in the area of modern wireless networks and in various environments built around these networks. Cloud computing toolbox proved to be very useful in augmenting research in domains of heterogeneous networks (HetNets) [1] with prototypes, testbeds and demonstrators. These research domains include centralized radio resource management in emerging cellular network architectures, network assistance role in device-to-device (D2D) communications [2] and study of prospective services in these networks.

Applying virtualization and programmability principles to the research infrastructure resulted in a flexible environment providing the researchers with tools to easily orchestrate various scenarios and network layouts. Virtualization platform allowed to quickly set up new architecture entities while programmable and scriptable networking allowed to build overlay networks interconnecting the entities. Using these tools facilitated the processes of designing and assessing future network architectures demonstrating certain connectivity and functionality interesting for the research.

First use case demonstrated in this chapter elaborates on designing a platform that would mimic a device connected through a heterogeneous network, allowing researchers to experiment with traffic flow optimization in an environment close to the envisioned next-generation network architecture. Prototype solution and testbed were designed building on software defined network principles of automation, abstraction and software-based flow switching and were implemented using overlay networks and virtual network functions.

Second demonstrated use case is within D2D communications research, where the task was to design architecture demonstrating feasibility of traffic offloading from infrastructure network onto direct links. Prototype was implemented with automated routing control in overlay network.

Resulting research environment supports the envisioned use of cloud computing in production networks with such technology trends as Heterogeneous Cloud Radio Access Network and Virtual Evolved Packet Core, as well as encourages researchers to adopt cloud tools in their work.

1.2. Potential solutions

Fortunately, the evolution of information technology (IT) and communications is not all about challenges. Over the last decade, the development of technology in various directions such as cloud computing, virtualization, high speed fiber optics communications, efficient signal processing, advances in data center IT architectures, etc., has provided researchers and engineers with powerful tools for elaborating future-proof solutions to modern challenging demands.

HetNets are modern solution for service providers, brought to address emerging connectivity demands by hierarchically adding smaller and smaller cells [3]. The resulting setup allows user device to interact with the infrastructure via multiple radio access technologies (RATs). While more and more user devices are being equipped with multiple radio transceivers, this architecture allows mobile network operators to significantly improve network capacity by efficiently utilizing spectra of these radio technologies, thus offering higher quality of experience to their customers.

Thorough study of integration of new RATs and understanding of required intelligence sharing between user equipment (UE) and infrastructure, for efficient use of these technologies, are envisioned to be fundamental enablers for future 5G networks. A confirmed example of benefits brought by HetNet architectures could be performance improvements in an unlicensed-band network, like WiFi, by leveraging centralized control from designated entity in mobile network core, like 3GPP LTE [4].

http://dx.doi.org/10.5772/67084

The agility and efficiency of modern software development processes have made software elaboration, operation and evaluation principles very attractive to other technology fields. Software-defined networking (SDN) paradigm comprises a set of automation and abstraction concepts aimed to bring networking closer to its ultimate goal of interconnecting users and applications in the most efficient way, by providing entities that use the network with tools to shape network services to their needs [5]. Within the scope of this work, SDN vision aligns well with HetNet architectures in desire to optimize resource allocation through centralized decisions, based on end-to-end view of traffic flows, and in attempt to involve both user and network sides of communication into optimization process, while allowing UE to efficiently use all the available communication technologies. While the term SDN is relatively new and there is a lot of debate around SDN concepts going among researchers, engineers and equipment vendors, the principles of network control automation, abstraction from underlying network functionality and software-based flow switching laid foundation for prototyping within this research work.

Another networking solution proved useful in research and development of new architectures is the concept of *overlay* networks [6]. Building another logical network layer on top of existing transport network is useful in highly dynamic environments like data centers, or in cases, where major changes to *underlay* transport network are very expensive, like in large scale operator networks. Within this work, configuring tunnels over different provider networks and different technologies allowed to achieve desired connectivity without modifying underlying network protocols, which otherwise would require open access to a live cellular installation. The drawback is increased overhead in the network, but it does not interfere with research objectives and is acceptable for the scope of this work.

1.3. Overview of tools

- **Open VPN** is a network tunneling software. Open VPN is widespread due to its openness and availability for a variety of platforms. Particularly, interesting for this project is its capability to dynamically deploy and execute custom applications or scripts on connecting client, based on triggered events, and capability to run as remote gateway interconnecting multiple private networks.
- **Open vSwitch** is a multilayer software switch. Important feature for the project is its ability to expose forwarding functions to remote entities for programmatic extension and control via protocols like OpenFlow.
- **VMware vSphere** Easy to use and well-documented virtualization solution. Used to abstract and share physical server resources to multiple virtual machines. Important feature is its robustness.
- **Docker** is software packaging solution, allowing to isolate an application at OS kernel level together with all required dependencies, files and network interfaces in a standardized unit container. Used for persistent and automated deployment and runtime across various environments.

– **General Routing Encapsulation** (GRE) is simple tunneling protocol used to build overlay networks on top of IP networks. Important feature is its capability to encapsulate various protocols including Ethernet.
– **Network Functions Virtualization** (NFV) is a concept complementary to SDN. NFV suggests new way to implement and operate network functions, by abstracting them from hardware appliances and using virtualization techniques to package these functions as virtual machines running on generic hardware.

1.4. Goals and structure of this work

This work addresses several prototyping solutions produced with the utilization of cloud computing techniques. These resolve several important challenges in the streamlined delivery of connectivity with higher service experience over the multi-radio heterogeneous deployments to a variety of modern handheld and wearable mobile devices. The two major goals of the work are as follows:

1. Developing a framework for the centralized radio resource management in multi-radio heterogeneous networks with highly configurable and integrative methods.
2. Implementing network-assisted direct communications mechanisms over converged LTE and WiFi radio access infrastructure with flexible and dynamic performance.

Sections 2 and 3 cover the summarized work within the research on the enabling technologies, namely, Heterogeneous Networks and Device-to-Device Communications. The subsequent section describes the challenges faced during system design, the limitations of the current architecture and the proposed solutions to resolve these issues.

2. Centralized radio resource management in HetNets

2.1. Introduction and motivation

5G communications ecosystem targets to reach higher rates never challenged before, and to keep up with these rates, modern heterogeneous network architectures have already developed improved ways to integrate various RATs. Seeking new vectors for evolution, emerging paradigm of heterogeneous cloud radio access network (H-CRAN) merges RAT integration with advanced cloud infrastructures [7, 8]. This approach enables improved management on the network-wide scale, allowing to implement cross-cell radio resource allocation in a coordinated way. Recent research addressed the gaps in theoretical performance analysis and provided assessment and mathematical methodology for *real-time* optimization of cooperative radio resource management in H-CRAN [4]. Resulting algorithms allow to balance between throughput and fairness metrics in a flexible way, as well as might align with network operator's development plans. Also, this approach demonstrated some advantages over state-of-the-art multi-radio resource allocation schemes.

http://dx.doi.org/10.5772/67084

2.2. Realistic testbed implementations

The next phase of our research is to design the considered algorithms in a practical testbed, to then address their expected operation. In this subsection, we account for realistic demands while developing the proposed methodology. To this matter, it is required to integrate radio resource management solutions into the stack of protocols for contemporary LTE and WiFi systems.

2.2.1. Implemented R&S demonstrator

Recently, our research group at Tampere University of Technology (TUT) has completed underlying HetNet testbed configuration, based on LTE eNodeB emulator, CMW500, by Rohde and Schwarz[1]. In this experiment, CMW500 was readily set up for a scenario of offloading traffic from LTE to WiFi, to assess capabilities of CMW500 equipment for 5G research. Integration example demonstrated a simple handover scheme for WiFi-LTE data offloading case, where WiFi signal strength was main deciding factor [9]. CMW500 was configured with LTE FDD cell signaling on frequency band 7, downlink 2645 MHz, uplink 2525 MHz, total bandwidth = 20 MHz. WiFi AP and CMW500 were bridged at the same server, which was also configured to lease IP addresses to UE via DHCP. An RF Shield Box model CMW-Z10 was used to control WiFi link quality. The shield box was connected to CMW500 unit via RF1 COM port and served as antenna for cellular link. This way, UE placed into open shield box connected to gateway server via both LTE and WiFi and was able to use the link with a better quality. Closing shield box lid introduced over 80 dB of digital attenuation for WiFi, thus triggering UE to switch its data transmission to LTE. When the lid was open again, WiFi link quality became better than that of LTE, and traffic was offloaded back to WiFi. CMW500 capabilities to emulate LTE provider's network allowed to quickly setup a testbed for improved offloading logic design, for the purposes of 5G research. Further research targeted intelligent data offloading, by employing advanced LTE link parameters. **Figure 1** demonstrates a short video presenting the setup. The video is available at http://winter-group.net/rohde-schwarz-tutorial/.

2.2.2. H-CRAN architecture prototype

Current prototype implements a multi-radio scenario, where a cellular network is coupled with a WiFi access point, thus providing UE with a possibility to seamlessly alternate both RATs, or to effectively employ two of them simultaneously. Clearly, development of the considered technique demands access to the side of the cellular BS. It is also required that the UE is capable of communicating the relevant control information on the appropriate radio channels. However, development kits for contemporary mobile platforms that are available on the market offer only limited support for controlling the respective data flows and interfaces.

[1]Video presentation of the experiment with R&S CMW500: http://winter-group.net/rohde-schwarz-tutorial/.

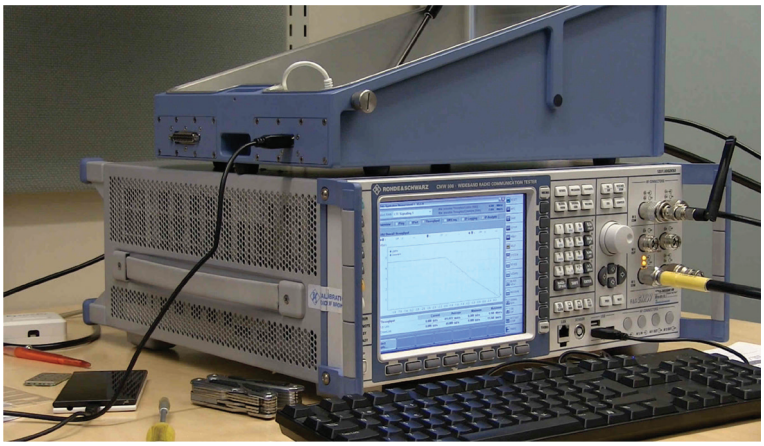

Figure 1. Setup with Rohde and Schwartz CMW500.

In this framework, the UE was connected to a traditional cellular system and a stand-alone AP of WiFi, both offering Internet connectivity with their corresponding ISPs. Simulating an integrated network for the two considered radio links, the UE starts two VPN tunnels to cellular and WiFi radio interfaces, respectively, while terminating at the aggregating node (the latter may, e.g., be placed onto the Internet). The said aggregator, establishing two links, mimics a packet gateway module residing in the LTE system.

Aggregating both radio links at the side of the UE, the Open vSwitch (OVS)[2] was utilized. Accordingly, vSwitch daemon, running on the UE, maintains links that correspond to WiFi and cellular connections in addition to a virtual local interface, which becomes a binding point as far as the outgoing traffic is concerned. The latter is generated by the applications with the help of *sockets* API. Here, the OVS specification requires that all of the connections in the virtual switch are capable of working with Ethernet headers. Note that it may not be the case in a cellular system exposed to the network as a point-to-point RmNet interface.

However, the discussed aspect might be resolved by upgrading vSwitch to enable the type of interfaces at hand, or using a dedicated module for skipping the processing of the Ethernet headers with an offset. Our testbed follows the lines of adding an extra tunneling layer on top of the current VPN with a Generic Routing Encapsulation (GRE) tap tunnel (known as Ethernet GRE or EGRE as well) because it offers implementation simplicity. For the sake of consistency, the GRE tap layer has also been implemented for WiFi VPN. The respective stack of the protocols for the UE is summarized in **Figure 2**. A similar approach is followed at the aggregating node on the side of the operator (simulated). Correspondingly, on its highest level of abstraction, our considered architecture comprises two switches connected by two redundant links. The core topology of our system is displayed in **Figure 3**.

[2]Open vSwitch website: http://openvswitch.org/.

http://dx.doi.org/10.5772/67084

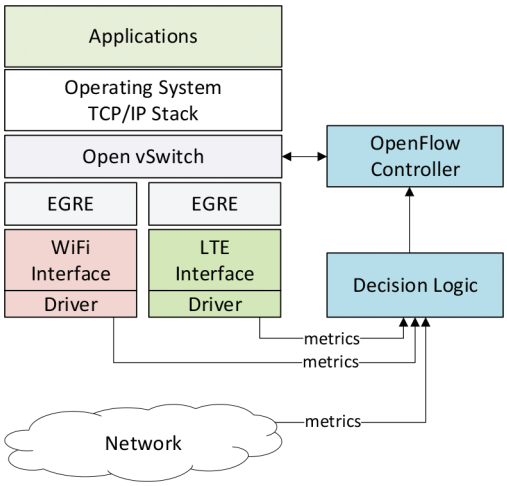

Figure 2. Proposed testbed topology for multi-RAT HetNets.

Our considered design allows for developing a controller that is capable of maintaining the forwarding table of vSwitch by relying on the OpenFlow protocol[3], and hence deliver the required levels of per-flow resource allocation (named here traffic steering) based on dynamic demands. A stand-alone module aims to collect information regarding the state of the underlying physical links from the involved RATs, such as received signal strength indicator (RSSI) and radio signal strength. The controller uses these data to efficiently assign new flows to either of outgoing interfaces. Coming back to the analytical algorithm implementation, this means that it is possible to transfer implemented resource allocation logic (e.g., relative fairness scheme) into the OpenFlow controller and specify control channel interface (which is LTE in this case).

2.2.3. Infrastructure for the prototype

The network side entities were installed as virtual machines (VMs) running on virtualization platform, built with vSphere[4] from VMWare. This approach allowed to deploy and interconnect architecture elements in a highly flexible and dynamic fashion.

For the research scenario, WiFi access point was connected to the Internet through university network and configured as a router between wireless and wired sides. UE was connecting to both WiFi and LTE networks and had Internet connectivity through either network. Using specific static routes, the UE was contacting particular VPN gateways over certain technology, that is, IP address of the gateway terminating LTE side VPN was routed via LTE network, with similar behavior for WiFi side. On the network side, traffic to the VPN aggregator node

[3]OpenFlow website: https://www.opennetworking.org/.
[4]VMWare vSphere solution: https://www.vmware.com/products/vsphere/.

was chained by the platform through an entry node switch that was splitting traffic destined to LTE VPN and WiFi VPN, and then passed through WAN link emulator node. Details of such service chaining are illustrated in subsection 2.3.

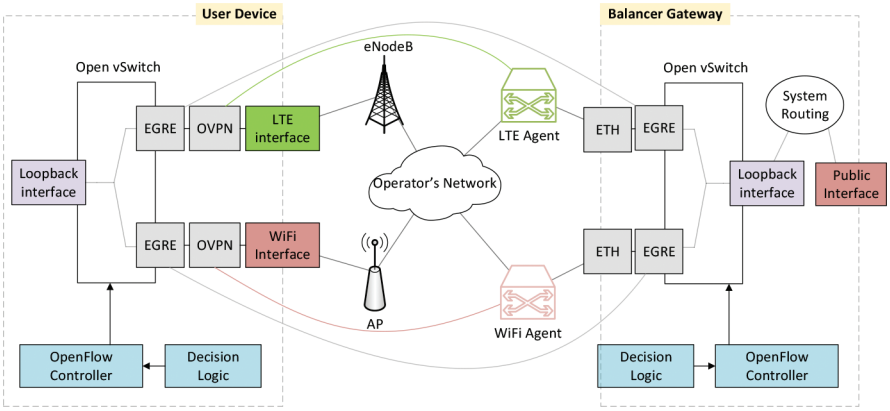

Figure 3. Overall architecture of prototype implementation.

2.3. Technical details

Most important features of current prototype implementation are as follows: software-based switching, network control automation and *overlay* network over regular provider networks. The following section summarizes how flow-based switching was implemented with Open vSwitch software, and network control was performed with OpenFlow controllers. Next, design of VPN and processes of measuring and influencing link conditions are outlined. Significant work was carried out to prepare UE to handle these tools—the modifications to the UE and procedures to enable it for research scenarios are covered as well. Finally, architecture is summarized with the workflow of a research scenario.

2.3.1. Toolchains and custom software on user equipment

An appropriate UE platform for the purposes of implementing our testbed has been the SailfishOS[5] running on Jolla mobile phones[6]. It combines *Linux* and *MER* software[7] as well as enables a powerful software development kit (SDK). The platform in question delivers all of the necessary tools to construct customized kernel modules, together with the generic GNU\ Linux software. Therefore, by installing OVS, GRE and virtual Ethernet (VETH) modules, in addition to OVS database and userspace tools, was rather straightforward. The software in SailfishOS is packaged with RPM Package Manager (RPM) (originally Red Hat Package

[5]Sailfish OS website: https://sailfishos.org/.
[6]Jolla phone website: http://jolla.com/.
[7]MER project website: http://merproject.org/.

http://dx.doi.org/10.5772/67084

Manager, now a recursive acronym), which provides powerful tools to integrate third-party software and to install additional components.

Some of the tools used in prototype require integration at OS kernel level, which means that UE kernel has to be built with corresponding functionality enabled. Customization of MER-based Sailfish OS is performed through Platform SDKs provided by the project. These platform SDK tools include compilers, Scratchbox2 cross-compilation toolkit, MIC image creator, Zypper package manager and other instruments to make development easier. In this setup, SDKs are run as an image in a *chroot* environment; however, SDK can also be used as a dedicated virtual machine. SDK was installed on development machine from a *rootfs tarball* containing essential tools for MER platform development. The goal of using platform tools was to compile GRE, Open vSwtich, and VETH kernel modules for SailfishOS running on UE. As a basis for the new kernel configuration, authors extracted configuration file from running system of a Jolla device. Configuration file was edited to include appropriate modules and was used for the build process, resulting in corresponding kernel object (.ko) files compiled. These module files were transferred to the UE and loaded into running kernel as part of prototype implementation. Another feature of the Platform SDK that was useful for us is that provided *cross-compiler toolchain* can be used to build custom userspace software, which is otherwise not available in official repositories. Using this *toolchain*, authors were able to compile userspace part of OVS software.

2.3.2. OpenFlow software switch

OVS is a multilayer virtual switch, designed to target at multi-server virtualization deployments. It provides a lot of important features for highly dynamic environments at large scales; however, for this prototype, the most important feature of OVS is that it adheres to the emerging SDN paradigm and thus supports forwarding based on flow tables and can use OpenFlow as a method of exporting remote access to control traffic forwarding. These features enable flexible placement of traffic control entity (at UE itself, on network side, or combination of both), and per flow control over traffic forwarding, meaning that certain sessions can be selectively placed on WiFi or LTE interfaces in a dynamic way. OVS consists of kernel side module, responsible for actual packet forwarding in the *data-path*, and userspace daemon, interacting with network state database and exposing control functions to external entities.

OpenFlow-based SDN vision implies that forwarding tables of a device should be maintained by a separate entity—the controller. This controller is responsible for making forwarding decisions and programming them into the flow tables along the data path, using Open Flow protocol. The solution in this installation was based on POX controller—a networking software platform written in Python[8]. Controller assumes the management of UE's forwarding plane consisting of 3 interfaces—WiFi interface, LTE interface and internal system interface. Main task for controller is to run one of the optimization algorithms supplied by researchers. Algorithm implementation receives metrics fed from the software, performs measurements of network conditions on both radio interfaces, and reallocates incoming and outgoing flows accordingly.

[8]POX controller: https://openflow.stanford.edu/display/ONL/POX+Wiki.

2.3.3. Network side

In the absence of access to an operational cellular network installation, researchers had to simulate joint control of WiFi and LTE networks for UE using VPN tunnels. This testbed used Open VPN owing to its simplicity and wide platform adoption. Clients would connect to dedicated VPN anchors through WiFi and LTE links, this way allowing the laboratory network to control setup of tunnel interfaces in a similar way as real operator network would control setup of physical links.

The authentication in VPN network is based on Rivest-Shamir-Adleman (RSA) certificates, as opposed to pre-shared key-based authentication. Using EasyRSA tools, authors have created certificate hierarchy where every user device would receive its own certificate to authenticate in the VPN network, and, based on information in that certificate, network would uniquely identify the user and configure it with predefined features.

Network-wise, VPN anchor was configured for *subnet* layout. Unlike in *point-to-point* layout, where a dedicated subnet is allocated for every client-server link, in *subnet* layout, all clients share common address space which slightly facilitates routing on the server side for prototype purposes. During start-up process, VPN agent places policy-based routing (PBR) rules into routing subsystem, to selectively route traffic from VPN network to the gateway facing the Internet to provide external connectivity.

To separate operation domains of WiFi and LTE side VPN servers, without the need to run them on different virtual or physical nodes, both processes are executed on the same VM, isolated in dedicated Docker[9] containers. Docker is a set of tools that provide convenient way to manage Linux container (LXC) environments. Every container running *openvpn* process has in its namespace a dedicated host network interface, facing the client, and a VETH link bridged with host network. This way, *openvpn* container acts as a router between host side network, leading to internal operator network, and client network behind VPN network. Detailed network layout of the VPN server is presented in **Figure 4**. Container mounts into its internal filesystem a directory with corresponding configuration files and certificates (e.g., LTE directory for container running LTE side) and executes *openvpn* process according to these configuration files.

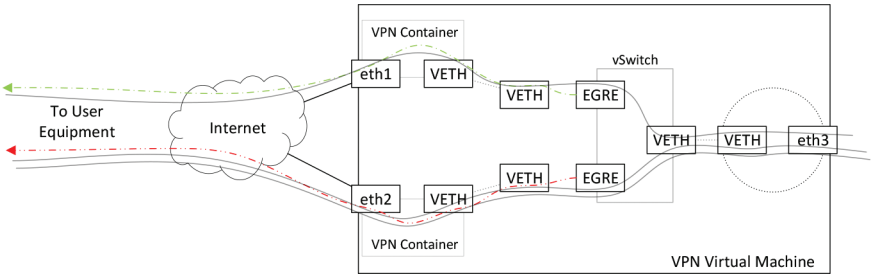

Figure 4. Detailed network architecture of VPN server.

[9]Docker platform: https://docs.docker.com.

http://dx.doi.org/10.5772/67084

Most of commercial-off-the-shelf (COTS) WiFi access points are shipped with proprietary, closed source operating systems, and are built with network chips run by closed drivers from hardware vendors. This way, access point becomes a "black box", exposing enough control to set up the network, but prohibiting any new functionality on top of the hardware. For an access point to be useful in such research, both operating system and wireless interface firmware must be open, to allow modifications and give access to internal variables. A good example of such device is a dual band WiFi router from Linksys model WRT1900AC. For this research, the device was installed with OpenWRT[10] system, custom compiled to include WiFi drivers as modules, rather than built into the kernel, making it easier to dynamically add changes into driver's code. One of the most important changes to the operating system was to enable Linux *debugfs* filesystem that exposes multiple system values, like status of numerous transmit queues and current transmission rates.

This way, AP was prepared to send measurements of interest to the process implementing forwarding optimization logic, where the latter would combine WiFi state data with LTE state data, or any other information found useful by researchers, like prioritization of users and RANs, and would yield new forwarding rules for UE side and operator side.

To simulate various network conditions, all traffic to the emulated aggregator node on network side was routed through a dedicated VM running WANem—a wide area network emulator[11]. This allowed us to impose various WAN characteristics common to LTE networks, like network delay, packet loss, packet re-ordering and jitter to the links that are in fact running in research laboratory LAN. Such service chaining of traffic dedicated to one VM through another VM is not natively supported by basic feature set of available vSphere installation, so several safety policies had to be disabled, and in such cases, certain broadcast and unknown traffic should be suppressed by virtual switch, and greater caution should be paid when routing traffic, to avoid forwarding loops. Network scheme of traffic flows in the setup with WAN emulator node is demonstrated in **Figure 5**.

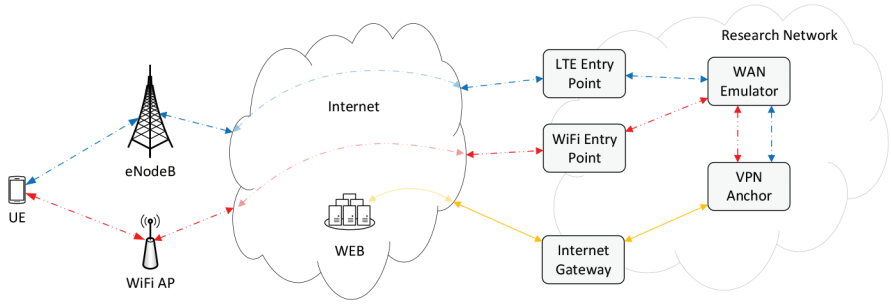

Figure 5. Setup with WAN emulation node.

[10]OpenWRT system: https://wiki.openwrt.org/doc/start.
[11]WANEM emulator: http://wanem.sourceforge.net.

2.3.4. Architecture summary and workflow

Network side of the deployment consists of a number of VMs serving as VPN anchor, WAN Emulation and Internet gateway. When VPN anchor VM boots up, it performs the following steps

1. starts VPN processes isolated in containers;
2. moves dedicated network interfaces to corresponding container namespaces;
3. creates a VETH pair for each container;
4. moves one end of every VETH pair to corresponding container namespace;
5. runs GRE tap tunnel to client over second end of VETH pair when client connects;
6. adds GRE tap interface to OVS;
7. creates another VETH pair for host networking;
8. adds one end of host VETH pair to vSwitch;
9. sets second end for routing to internal operator network.

When UE side and network side virtual forwarding planes are ready, both devices start measuring link qualities. Depending on coordination scheme scenario, controller software is started on either side and both virtual switches connect to it. Measurements from UE, AP and operator side are also sent to the controller. This brings the setup to final operational state, where controller would use all available metrics, to run resource optimization algorithm provided by researchers, and would re-allocate traffic sessions across available links accordingly.

2.4. Some numerical results

In this subsection, we provide a comparison of prospective H-CRAN deployments with full UE-centric network control methods as well as network-assisted multi-radio integration architectures that have a lower degree of manageability. Focusing on the illustrative example of the UE-centric solution, we analyze the max-usage control scheme. Here, each UE attempts to occupy all of the available radio resources on all possible RANs that it may connect to. Such a strategy is greedy, but it does not impose tight signaling requirements on neither the network nor the UE, as well as performs reasonably well from the individual user throughput perspective. Most importantly, this option is very easy to implement.

The second scenario that we have taken into account is a network-assisted resource allocation scheme. Basically, it exploits the concept of cell range expansion. The main idea behind it is to manage the efficient thresholds of association to WiFi and LTE cells for those UEs that reside under macro coverage. The approach in question allowed for incorporating the load variation effects across the individual RANs by effectively increasing/deceasing the UE numbers that are connected to the respective cells. At the same time, a separate decision-making entity implemented at eNB together with the corresponding signaling to supply UEs with the relevant assistance information is required when utilizing this method.

http://dx.doi.org/10.5772/67084

In this implementation, we force the UE to prefer a single RAN at a time [4]. Such a decision was made mainly due to its practical feasibility, since otherwise said technique would lose to the UE-centric approach by assigning the maximum radio resources to the users with better connectivity conditions, while potentially not allowing the users with worse channel to even attempt the WiFi/small cells. In a nutshell, the description of the considered algorithm is the following: first, the user connects on the WiFi layer, but should the resultant signal turn out to be poor, the UE will then probe the pico connectivity layer and, if there remains no other appropriate opportunities, the user will then be handled at the macro layer.

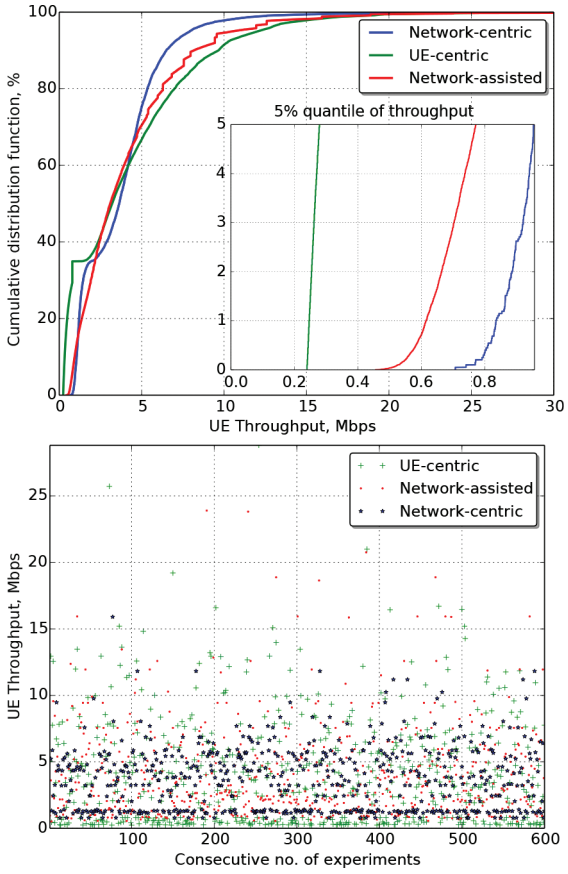

Figure 6. Per-UE throughput CDF.

In addition, we take into account a network-centric scheme made available by the H-CRAN that centrally manages the cross-RAT radio resources subject to the actual connection quality of the user. In this case, both network-assisted and network-centric techniques offer more

attractive performance in the H-CRAN scenarios. In addition, the centralized solution offers better control flexibility.

We analyze all the three scenarios from the per-UE throughput point of view and the results are given in **Figure 6**, top subplot. The threshold for the network-assisted scheme for both LTE and WiFi small cells has been set to the maximum sensitivity level. Concerning the network-centric solution, resource management method takes as an input the vector of spectral efficiencies for each user first without any modifications. As a result of this default configuration, all three solutions demonstrate comparable results in terms of the average throughput, but the network-centric mechanism performs much better at the cell edge.

Next, we have a closer look at the results for the per-UE throughput given in the bottom subplot of **Figure 6**. It is clearly visible that the UE-centric scheme is characterized by lower fairness as compared to the network-centric/assisted methods based on 5%-quantile sub-figure embedded into **Figure 6**, top right subplot.

The last analyzed scenario is the network-assisted case. Here, the previously discussed effect is not visible as the UEs have only one active connection at all times. At the same time, we may observe another interesting effect: step-wise behavior may be seen in the upper part of the curve. It could be explained based on the difference between the UEs with very good WiFi link qualities—various numbers of UEs associate with different APs causing variations in throughput if the number of connected UEs is low and the SNR of each user is high.

Summarizing, the centralized resource optimization and control technique is suitable for real-time resource allocation across multiple RANs in H-CRAN environment. It brings high level of flexibility and adjustable balance between the overall H-CRAN system throughput and fairness of the resulting allocations.

3. Network assisted device-to-device communications

3.1. Introduction and motivation

The increase in mobile data traffic pushes network operators to seek ways to relieve congestion on their infrastructures. A natural way to mitigate the shortage of available radio resources is to deploy an increasing number of various sized BSs; however, such approach is costly and faces some practical challenges. An alternative strategy would be to leverage the capability of modern UEs to establish simultaneous connections via different radio links, and to enable traffic offloading from cellular network onto D2D connections in unlicensed bands like WiFi [10]. The problem with WiFi is that there is no fast and efficient method of service or device discovery built into the protocol, as well it lacks functionality to efficiently manage multiple D2D links [11]. Recent research [12] demonstrates that these limitations can be solved by a certain amount of network assistance made available to D2D communications.

The offloading principles, discussed in Section 2, are not limited to be used only in network operator's infrastructure. Similar techniques could be utilized to improve network performance, by offloading communications between proximate users from infrastructure to direct link between users, or D2D links. This section covers research and prototyping activities performed

http://dx.doi.org/10.5772/67084

in exploring improvements to user traffic offloading from infrastructure network onto direct D2D links brought by assistance from cellular network.

3.2. Implementing traffic offloading prototype

Within proof-of-concept implementation, authors of this work were responsible for making required changes to UE operating system, elaborating offloading algorithm based on IP routing, setting up and maintaining infrastructure to run prototype components and implementing UE side of the prototype [12]. Next subsections elaborate on implementation of these prototype components.

3.2.1. Android networking subsystem

Android, as a Linux-based system[12], allows to have simultaneous connections over more than one radio interface. Once the LTE and the WiFi links are activated, the UE operates on two interconnected networks, while having a single default gateway to transfer its traffic outside these systems. In the considered case, it may be feasible to connect to another peer on the WFD channel only when the IP header's address of the destination node is the WFD address of the said peer, whereas the address of the source is that of the originating UE. Also, WFD link uses private address range that is not reachable through anything else than WFD link; once the link is disconnected, the peer becomes unreachable. For the suggested D2D architecture, a device would be required to be capable of reaching the public IP address of its peer on the LTE interface via the WFD connection.

As a major target of our constructed solution, we aimed at its design transparency for the currently available applications. Accordingly, we contribute to the user adoption levels of the proposed technical implementations. Along these lines, any changes at the physical layer may not be appropriate, as these are tightly coupled with radio interface driver. The same applies to link layer—implementing new functionality, there would make the solution vendor-specific. Since existing applications heavily rely on current transport layer protocols, modifications in transport layer are also not considered. On the other hand, system offers various tools to make changes in IP layer (or network layer). This would allow to leave underlying radio interfaces as-is and would enable application developers to use same routines to obtain network access as before. IP addresses are in a way bound to the physical interfaces, but selection process is made without direct interaction with interfaces. This would allow us to create an interface independent solution without modifications of upper layers.

The default configuration of an Android system allows to have routes with multiple gateways, but one of them is inserted into routing table with lower cost than the others. This way no load-balancing is performed and only one route is used. In case of WiFi link and LTE link, or cellular link in general, the route through LTE gateway is preferred for the Internet connectivity because WiFi link and, especially, WFD do not guarantee Internet connectivity at all. Therefore, changing cost of default route might cause unreachability of the Internet for all applications in the mobile device.

[12]Android OS: http://www.openhandsetalliance.com/android_overview.html.

3.2.2. Traffic offloading based on routing

The proposed solution is based on allowing mobile device to route IP traffic as usual, and then inject more specific routes for particular peers into routing table. As Android system is based on Linux kernel, it is possible to enable routing functions by modifying the value of system variable *net.ipv4.conf.all.forwarding* from 0 to 1. After this change, Android mobile device gains capabilities of a generic router, known from computer networks. This way, mobile device can forward packets from one interface to another. An interesting point to note here is that it allows to send IP packets with source address being LTE interface public IP address and destination address being peer's WFD private IP address, and IP layer of Android system will send them through WiFi interface.

There are, however, several issues with such approach to link selection that need to be solved. The first issue arises with application use of network sockets. When an application needs network connectivity, it requests the service from operating system using sockets API, and operating system chooses to use one of the available IP addresses as source address. Source IP address, source transport layer port, destination IP and destination port must remain the same throughout communication, which cannot be guaranteed in case session is bound to WiFi interface IP addresses, because interface can be disabled at any moment. Another issue is with routing private networks through operator's infrastructure—if a session is bound to WFD's private IPs, it cannot be switched to LTE interface because the operator's infrastructure prohibits routing packets to this private IP range.

As a workaround for prototype demonstration, a set of OpenVPN tunnels has been elaborated from devices to an anchor point in the network infrastructure, thus allowing routing of private networks through operator's core. In a real world deployment, however, this would not be necessary, as operator could set up internal routing according to D2D link networks, issued by D2D server. Also, the system creates an *overlay* GRE tunnel bound to loopback interfaces on the devices. Loopback interface is meant to be a constant anchor point for applications, and only *overlay* tunnel endpoints will be rerouted, ensuring that session IP addresses remain the same all the time, regardless of *underlay* interfaces used, and thus providing service continuity.

Since only communication with one particular peer needs to be offloaded, a route can be inserted into routing table stating that only the peer's loopback IP address is reachable through peer's WFD private IP address. Insertion is performed by the command iprouteaddPEER_LOOPBACK_IP/32viaPEER_WFD_IP. Once this command is accepted by IP routing layer of Android, all traffic with destination IP address PEER_LOOPBACK_IP will be forwarded to WFD interface and this way *overlay* tunnel traffic will be sent over WFD.

The insertion and removal is performed by management application that is running in the background. Both actions are invoked upon a corresponding command from D2D server, but route removal can additionally rely on local channel quality measurements. Within this implementation, the only channel quality metric considered is the RSSI. When reaching a particular threshold in RSSI, route can be inserted or removed, thus selecting D2D or infrastructure channels, and reaching even higher threshold can trigger complete link disband.

http://dx.doi.org/10.5772/67084

This injection of specific routes into routing table does not have to be performed symmetrically on both devices participating in D2D offloading. Also, route injection scheme is not limited to be used only with a single peer.

3.2.3. Infrastructure for the prototype

The service relies on three parts: (i) Client side phone application; (ii) Content register service; and (iii) D2D server, as it is shown in **Figure 7**.

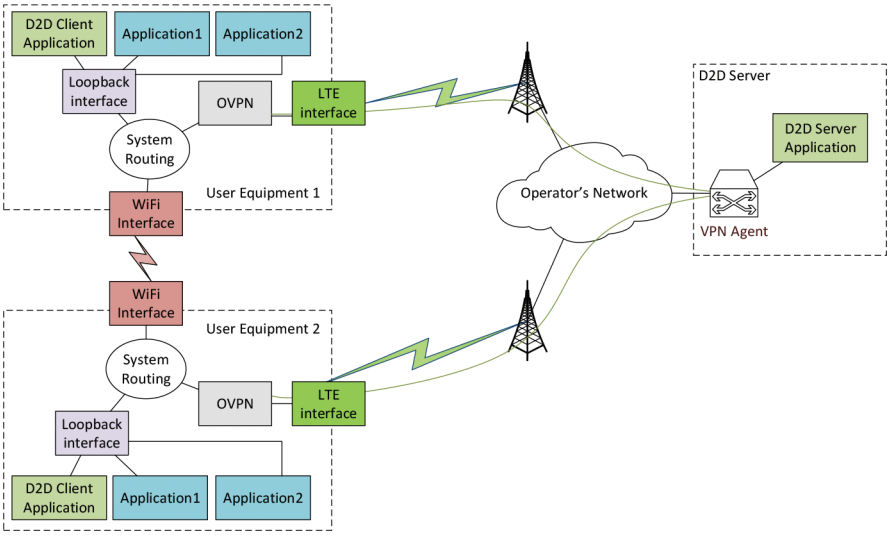

Figure 7. Prototype setup for D2D architecture.

The content register is assumed to be any platform providing data sharing services to its users e.g., social networks like Facebook, Google+, etc., with a difference that in this case only metadata about content is shared in form of a hyper-reference, instead of actual data [13]. D2D server, on its turn, is designed to be run by network service provider. It is worth noting that besides suggested architecture layout, the service is flexible enough for both entities to be run by a single authority or even on the same host.

To emulate integration of the service with providers, the two applications at the side of the server have been designed within the cloud infrastructure of the research group. Accordingly, a pair of virtual machines have been deployed to assume the roles of the Content Register and the D2D server, respectively. Here, the Content register component has been developed in PHP and accommodated by the web server running Apache. Hence, our constructed application is a generic webpage, which offers any pre-registered UE an opportunity of posting its willingness to share certain information, or of identifying the desired locations of thus shared information.

User posts only information required to access data, rather than its location, that is, sharing protocol and port number, while IP address that locates the data is handled by D2D server. The part that glues service sharing and service discovery is the introduced in the URI protocol identifier, which is returned by the Content register module—"d2d://". The UEs are made capable of interpreting the protocol scheme in question in terms of a request to initiate the Client application. The latter is able to interact with the D2D server to convert the username of the peer node that serves information into an appropriate IP address. In turn, the D2D server has been realized in Python in a form of a standalone application by utilizing HTTPS as a means of transport for the control messages. The system considers that the mobile data connection of a user is active all the time throughout the use of the service. Our proposed solution indicates that the content register as well as the D2D services are easy to integrate seamlessly into the current web service infrastructure.

End user devices used in service demonstration were Sony Xperia ZL phones provided by Sony Mobile. Android, being an open source mobile platform, provides needed flexibility in configuration and available tools to fulfill requirements demanded from user devices. One of the main features user devices should have is the ability for both mobile data link and WiFi link to be up simultaneously. Due to energy consumption constraints, most of consumer devices on the market restrict their network connectivity to use only one of the available connections at a time. To enable both interfaces in the system, authors had to bypass native Android service controlling WiFi and interact with WiFi driver directly. As Android system to some extent is derived from GNU\Linux, it easily gives us needed tools—*wpa_supplicant* interface controlled via *wpa_cli* utility. Another issue is that in order to integrate with these utilities Android has to be compiled with *userdebug* feature enabled, and stock firmware provided by Sony for their devices does not have this property. Therefore, another option to consider was to build the system based on source code from Android Open Source Project, using proprietary binary drivers released to public by the vendor. However, resulting system at that time lacked radio drivers, needed for intended operation of mobile data connection. Final choice for the user device platform was popular aftermarket firmware Cyanogenmod[13], based on Android and maintained by FreeXperia group.

Another requirement for the mobile system of the end user is to be able to receive the incoming links on a mobile data connection. Accounting for the fact that the network operators utilize private IPv4 address pools to distribute across user devices as well as provide Internet access by means of a NAT, using the services that run on the user devices externally from the outside of the local connection may not be feasible with such a setup. One of the possible solutions to overcome this issue would be using IPv6 addresses, but local service providers at the time being did not provide IPv6 connectivity options to the extent suitable for practical demonstration of D2D service. Another tested option for demonstration was encapsulating mobile data link of both communicating devices inside a VPN tunnel to a common VPN server, thus moving both devices into the same IP subnet. Also, discussing the issue with local service provider—TeliaSonera Finland Oyj—researches were offered a certain access point name that would provide user devices with a publicly routable IPv4 address.

[13]Cyanogenmod distribution: https://wiki.cyanogenmod.org/w/Main_Page.

http://dx.doi.org/10.5772/67084

3.3. Prototype implementation: technical details

This subsection covers deeper technical details of implementation for service architecture prototype described above. As mentioned before, the whole system comprises three entities—UE, D2D server and Content register. Besides that, prototype infrastructure contains a node serving as anchor point for VPN layer, and a node serving as gateway for Internet connectivity.

3.3.1. User equipment

User side of the prototype was implemented in two layers—main network assisted offloading logic, implemented as native Android application, and functional layer performing all interactions with Android OS implemented in Bash shell scripts.

Bash scripts implement functions like inserting/withdrawing routes, enabling WFD links and establishing tunnels to the infrastructure. Also, they provide an abstraction layer to D2D Client application, while client application in turn establishes control channel with D2D server and interacts with the server using HTTPS to execute offloading algorithm steps.

The workflow of suggested solution starts with running D2D client Android application. The first phase for application is to invoke initialization *shell* script to bring UE to a state, where it would be ready to interact with infrastructure and D2D server. As mentioned in subsection 3.2, system WiFi service will not allow to have both radio interfaces up simultaneously, so the initialization script will stop this service and will run a new copy of *wpa_supplicant*, with configuration files and parameters needed to prepare WiFi *p2p* interface. Also, initialization script loads necessary additional kernel modules. The last phase of initialization script is to start OpenVPN tunnel, and this phase creates a blocking dependency, where initialization script in order to complete needs OpenVPN tunnel to successfully establish connection with infrastructure. This is resolved by implementing a *trap* in the initialization script, so that it can safely *sleep* after starting OpenVPN tunnel, and be resumed by *openvpn* process after tunnel is up. Finally, initialization script will mark its completion in a process *pid* file, so that other system parts can later check if initialization has already been performed.

After *init* phase is performed, mobile device has established a control channel to D2D server via the tunnel and is ready to setup WiFi direct connections, when instructed to do so. D2D Client application regains control, registers itself to D2D Server and UE becomes available for offloading. Depending on the role of UE as publisher or consumer, D2D Client application is invoked either manually or by a link from Content register (see subsection 3.3), respectively. Since components responsible for the network-assisted offloading features are completely decoupled from application layer, in case user has published some content, they must ensure that the application that is actually serving it (e.g., a file server, game server or multimedia streaming application) is running and is awaiting incoming connections. In case user followed the link from Content Register, D2D Client application subscribes to server with data from the link. At this stage, network knows that one user is ready to serve content and the other one is willing to receive it with offloading features enabled.

The next step to prepare devices for traffic offloading is to start *loopback* interfaces and *overlay* GRE tunnels. When both peers public LTE (or OpenVPN in case of this implementation) IP addresses are known, D2D server instructs both devices to start a *loopback* interface, to add routes to peer's *loopback* interface over cellular network and to bring up a GRE tunnel bound to those *loopback* interfaces. These functions to operate *loopbacks* and GRE are implemented in corresponding *shell* scripts, as well. *Overlay* tunnel endpoints will be rerouted to switch traffic from one radio network to another providing service continuity as described in subsection 3.2. Addressing for the *loopback* networks and *overlay* network is carried out by coupling predefined prefixes with host portion of device's LTE IP address.

When D2D server detects that two users, who have previously subscribed to be assisted in D2D communication, are now in appropriate conditions to use offloading, server will generate a group name and a pre-shared key for WFD and will instruct one device to start WFD by invoking a Bash *shell* script implementing start, join, leave, or remove functions. The script, in its turn, will use *wpa_cli* tool to interact with *wpa_supplicant* process started at initialization phase. As in case with *loopback* and *overlay* network addressing, for this prototype implementation, when WFD link is established, it is configured in a stateless manner with IP addresses from 10.1.1.0/30 range with 0.1 assigned to group owner and 0.2 to WFD client.

At this point, both devices have two radio interfaces up and are ready to offload traffic, by rerouting *overlay* tunnel endpoints through either of these interfaces. When instructed by the network, D2D Client application will insert or withdraw routes by invoking corresponding *shell* scripts. D2D server keeps track of UE location, and when the distance and RSSI metrics reach predefined levels, server will instruct devices to use WFD, to fallback to using cellular network while keeping WiFi radio on or to completely shutdown WiFi interface.

Upon users' request D2D server can de-register them and stop following their proximity.

3.3.2. D2D server

Server is running Python application listening for HTTPS requests on port 8099. All the relevant data from clients are passed as HTTP parameters and server is running an internal database to store registered users and their associated data like public IP addresses and current location. In present prototype implementation, UE will periodically poll the server about status changes and server will reply with corresponding instructions according to current network state.

3.3.3. Content register

Content register is a PHP application, running embedded database to authenticate users and store metadata about their shared content. When users wish to post some service available for access and ready to be offloaded to D2D link when appropriate, they log in to Content register and publish access protocol and port number for the service, e.g., http/80. When another user sees this intent to share, it can request shared content, and Content Register will present the metadata in form of a URI with d2d://scheme and embedded information about peer username, target D2D server, shared service protocol and port *d2d://amie@d2d.winter.rd.tut.fi@simhost.winter.rd.tut.fi/webcam:8080*.

http://dx.doi.org/10.5772/67084

Web browser in UE is setup to recognize d2d://URI scheme as bound to D2D Client local Android application. Hence, following the link given by Content register will start D2D Client application that will establish control channel to its D2D server and will register its intent to engage into D2D connection with the named peer, when network conditions are favorable to do so. Optionally, user may choose to start using shared content straight away and be offloaded when possible, or to wait before content is available via direct link.

3.3.4. Utility nodes

All server side components are running in virtual environment, implemented with vSphere from VMWare. Both D2D server and Content register are running on dedicated virtual machines. To terminate OpenVPN connections, another dedicated VM was installed. The VPN VM is selectively routing traffic from VPN connections, either to other peers using *openvpn* "subnet" type of configuration, or to the Internet gateway using Policy Based Routing. The Internet gateway is another VM, running Brocade Vyatta[14] software router and performing NAT to provide Internet connectivity to internal networks.

3.4. Some numerical results

In this subsection, some results on the performance evaluation of the discussed network-assisted D2D system are given. These comprise the measurements executed on the real deployment (discussed above), as to facilitate the future D2D implementations. The primary goals of said evaluation are as follows.

– To indicate the bottlenecks that could potentially hinder the adoption of the D2D connectivity by the future wireless technology.

– To establish the appropriate performance bounds and limitations for the D2D technology and outline which services could be most suited for direct communications in contemporary and near-future markets.

In order to provide the best available accuracy, the considered D2D protocol was split into individual messages and we further performed our measurements on a significantly large sample set [13]. First, we measured latency between the client device equipped with a USB LTE dongle and the D2D served deployed behind the UGW. The distance between the client and the eNodeB has been fixed to 10 m for the entire time of measurements. We utilized Iperf tool to assess the maximum throughput of a particular radio access technology. Later, it has also been used to create a preset load on both uplink and downlink channels, thus emulating the required measurement conditions.

The measured network latencies are displayed in **Table 1**. We may notice that if the overall loading on the LTE network does not exceed 90%—the round-trip times between the client and the D2D server range from 13 to 25 ms. Hence, cell load of up to 90% does not have any evident effect on the D2D signaling procedure.

[14]Vyatta Router: http://vyos.net/wiki/Main_Page.

Cell load	Idle	50%	90%	99%
Measured RTT (ms)	13.00	25.00	25.00	1248.00
UL latency (ms)	10.40	16.67	16.67	832.00
DL latency (ms)	7.00	8.33	8.33	416.00
Stage 2 (ms)	27.00	41.67	41.67	2080.00
Stage 3 (ms)	33.00	58.33	58.33	2912.00
Stage 4—path switch (ms)	7.00	8.33	8.33	416.00
Full procedure (ms)	67.00	108.33	108.33	5408.00

Table 1. Network latency measurements for different cell loads.

If the cell load is increased to the extreme conditions of 99%, that is, the input UE queues are overfilled, the resulting values range on the order of seconds. Hence, without the appropriate QoS support in LTE, the obtained latency would dramatically influence the proximity detection, as well as the overall user adoption of the considered D2D technology.

Further, we analyzed the delay values for different types of cell load, as it is shown in **Figure 8**. While these values remain negligibly small for up to 90% loading, the saturated UE scenario yields a major increase in the observed delay. In this case, it becomes hardly acceptable and ranges in seconds. In this extreme case, the implementation of QoS on the UE side is crucial to prioritize the D2D signaling messages. While there is some additional delay introduced, in almost all of the cases, the connection will be established on time for the users to reliably benefit from it before they come into physical contact, even when walking towards each other.

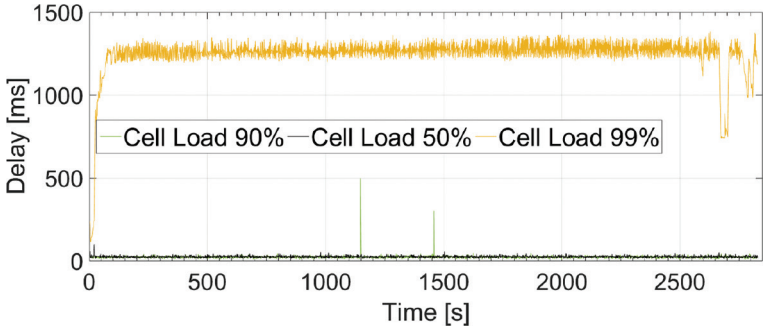

Figure 8. Network latency for alternative cell loads.

http://dx.doi.org/10.5772/67084

4. Summary of this work

4.1. Key lessons learned

This subsection highlights the challenges faced in the system design, the limitations of the current architecture and the network deployment choices.

Regarding the centralized radio resource management in **HetNets**, most of today's UE devices have both cellular and WiFi interfaces on board. At the same time, the need to reduce energy consumption has led equipment manufacturers to impose a limitation on the mobile UE's operating system to have at most one radio interface active at a time (except for the Bluetooth). Consequently, the emerging demand for improved wireless connectivity has convinced major equipment vendors to assist the developers in access and effective utilization of the heterogeneous wireless connectivity. A promising example of a research-friendly UE platform is Jolla phones running Sailfish OS. It was verified that the devices utilized in our trial have a flexible and open architecture, augmenting modern hardware with capable developer tools to allow for intended modifications on system level and thus achieve the desired degrees of connectivity.

As the next point, we were looking for the feasible options to integrate cellular and WLAN RATs to advance the vision of H-CRAN. In reality, an integration of a WiFi access point into the operator's network would require access to an open cellular BS, which is hard to acquire in the research environment today. Hence, we decided to encapsulate the corresponding radio links into separate OpenVPN tunnels utilizing two independent wireless technologies simultaneously.

Our prototype architecture operates as an advanced SDN solution and specifically employs the OpenFlow protocol to manage effectively the UE connectivity. The operational radio channels on a particular device are considered to be connected to a common forwarding plane on the smartphone that is mimicked by utilizing the Open vSwitch software. At the same time, actually occurring forwarding decisions are made by the controller software based on a set of active measurements of the radio link conditions. In our trial, the controller is deployed on the device side but it could be easily moved to the operator's environment.

From the network assisted **D2D** communications perspective, the proposed solution follows the general D2D implementation guidelines considered in the standards. However, it has encountered several deployment issues, which have forced deviations from the reference descriptions. The existing architectural principles in the LTE system prevent from implementing the cellular assistance mechanisms in the manner consistent with the engineering expectations. Therefore, it was necessary to implement a number of careful steps to provide a working system based on the contemporary technology.

The resulting trial implementation is developed to offer the required packet-switched data services together with important options, including VoIP over the integrated LTE and WiFi deployment. The EPC has been dimensioned to allow for high data rate applications with the relevant QoS provisioning. For the calls (both voice and video), the switching function-

ality is realized by utilizing the high capacity IP Multimedia Subsystem together with its key components. This is carried out for mobile as well as fixed users and enables connectivity with outside telephone/teleconferencing networks.

The ultimate target of the LTE test network implementation is to leverage a holistic and customizable mobile system allowing for prompt implementation and demonstration of the emerging D2D communications technology. We therefore efficiently utilized this important tool to confirm that LTE-assisted WiFi-Direct paradigm has already matured sufficiently to be implemented in the commercial-grade integrated LTE/WiFi networks.

4.2. Conclusions

This subsection concludes the chapter with a review of presented networking solutions for integrated heterogeneous wireless ecosystem. Wireless networks are constantly evolving in offered connectivity levels, thus strongly consolidating in our lives as a necessity. More and more devices are joining the network requesting continuous high quality service, which brings unprecedented challenges to network design in upcoming 5G era. Transformation of mobile user experience requires complex changes in both network infrastructure and device operation, where user experience is optimized taking into account surrounding network context.

Advances in modern cloud technologies offer a set of enablers for next generation networks, consolidating management and resource allocation under a single abstraction level exposed to automation and orchestration frameworks for live optimizations. Primary focus of this work was to support current research on novel integrated multi-radio network architectures, by elaborating a set of prototypes and testbeds.

Solutions in Section 2 support research performed on optimization problems in cooperative radio resource management in H-CRAN.

- Implementation is using COTS equipment making said resource management available already today and demonstrating that no hardware changes are required.
- Prototype allows user device to be connected to both LTE and WiFi at the same time. Setting battery life issues aside, this feature significantly extends user connectivity.
- Resource allocation is controlled using multidimensional view on the network, employing nonconventional parameters like cross RAT loading.
- System efficiently utilizes available radio connections by properly assigning traffic flows, e.g., streaming high resolution video over WiFi while downloading important software update over LTE.
- Deployment is flexible and loosely coupled to enable portability, automation and scalability of the solution.

Solution in Section 3 enables direct communication between proximate users to offload traffic from network infrastructure and create new proximate services.

- Implemented using COTS equipment making the solution available without additional modifications to hardware.

http://dx.doi.org/10.5772/67084

– Allows to leverage peer proximity to offer richer set of user applications and services.

– Increases user awareness about proximate peers, services and content. With this feature, the user is not limited to only consume the services offered by the network provider.

– Network assistance ensures efficient use of short range radio interfaces by facilitating discovery and authentication functions. This significantly relaxes energy constraints currently seen in technologies like WiFi Direct.

– Deployment using conventional web mechanisms enables prompt integration of new entities and connectivity solutions.

This chapter demonstrated application of novel and traditional networking concepts in implementing prototypes and demonstrators for emerging wireless network architectures and ecosystems. Resulting implementations supported research in centralized radio resource management and network assisted traffic offloading. Proposed approach demonstrated efficient prototyping for research scenarios using automation and abstraction concepts, decoupling network functions from hardware and modifying open source components. Use of modern software packaging and delivery techniques, like hardware level virtualization, OS kernel level virtualization and automated configuration, significantly accelerated implementation process and made resulting architecture components highly reusable for future projects.

Author details

Roman Florea[1], Aleksandr Ometov[1], Adam Surak[2], Sergey Andreev[1]* and Yevgeni Koucheryavy[1]

*Address all correspondence to: sergey.andreev@tut.fi

1 Tampere University of Technology, Tampere, Finland

2 Algolia, France

References

[1] M. Peng, Y. Li, J. Jiang, J. Li, and C. Wang, "Heterogeneous cloud radio access networks: A new perspective for enhancing spectral and energy efficiencies," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 126–135, 2014.

[2] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. Miklós, and Z. Turányi, "Design aspects of network assisted device-to-device communications," *IEEE Communications Magazine*, vol. 50, pp. 170–177, 2012.

[3] V. Jungnickel, K. Manolakis, W. Zirwas, B. Panzner, V. Braun, M. Lossow, M. Sternad, R. Apelfrojd, and T. Svensson, "The role of small cells, coordinated multipoint, and massive MIMO in 5G," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 44–51, 2014.

[4] M. Gerasimenko, D. Moltchanov, R. Florea, S. Andreev, Y. Koucheryavy, N. Himayat, S.-p. Yeh, and S. Talwar, "Cooperative radio resource management in heterogeneous cloud radio access networks," *IEEE Access*, vol. 3, pp. 397–406, 2015.

[5] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[6] Y. S. Soh, T. Q. Quek, M. Kountouris, and H. Shin, "Energy efficient heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 5, pp. 840–850, 2013.

[7] M. Peng, Y. Li, J. Jiang, J. Li, and C. Wang, "Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 126–135, 2014.

[8] Y. Zhou and W. Yu, "Optimized backhaul compression for uplink cloud radio access network," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1295–1307, 2014.

[9] A. Orsino, M. Gapeyenko, L. Militano, D. Moltchanov, S. Andreev, Y. Koucheryavy, and G. Araniti, "Assisted handover based on device-to-device communications in 3GPP LTE systems," in *Proceedings of IEEE Globecom Workshops (GC Wkshps)*, San Diego, CA, USA, pp. 1–6, IEEE, 2015.

[10] G. Fodor, S. Parkvall, S. Sorrentino, P. Wallentin, Q. Lu, and N. Brahmi, "Device-to-device communications for national security and public safety," *IEEE Access*, vol. 2, pp. 1510–1520, 2014.

[11] Y. Cai, F. R. Yu, C. Liang, B. Sun, and Q. Yan, "Software defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7349-7360, Sept. 2016.

[12] A. Pyattaev, O. Galinina, K. Johnsson, A. Surak, R. Florea, S. Andreev, and Y. Koucheryavy, "Network-Assisted D2D Over WiFi Direct," in *Smart Device to Smart Device Communication*, Switzerland, pp. 165–218, Springer, 2014.

[13] S. Andreev, J. Hosek, T. Olsson, K. Johnsson, A. Pyattaev, A. Ometov, E. Olshannikova, M. Gerasimenko, P. Masek, Y. Koucheryavy, et al., "A unifying perspective on proximity-based cellular-assisted mobile social networking," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 108–116, 2016.