Integrating Mobile OpenFlow Based Network Architecture with Legacy Infrastructure

Martin Nagy, Ivan Kotuliak, Jan Skalny, Martin Kalcok, Tibor Hirjak

Integrating Mobile OpenFlow Based Network Architecture with Legacy Infrastructure

Martin Nagy, Ivan Kotuliak, Jan Skalny, Martin Kalcok, Tibor Hirjak

Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Bratislava, Slovakia
{martinko.nagy, hirjak.tibor, martin.kalcok}@gmail.com
ivan.kotuliak@stuba.sk, jan@skalny.sk

Abstract. UnifyCore is a concept of SDN centric, OpenFlow based and access agnostic network architecture, which changes the way networks are being built today. It is designed in a way, so present access technologies can be easily integrated in it. It provides set of architectural components and rules, which help to easily decouple components of the access technology and put their functionalities into UnifyCore building blocks. This simplifies the overall network architecture and allows the use of common transport core for all access technologies. First proof of concept built on UnifyCore is the GPRS network, which is a challenge for SDN, since it does not have split user and control plane transport. In this paper we introduce and explain features that allow fully SDN UnifyCore to be integrated with existing legacy network infrastructure (switches/routers).

Keywords: 3GPP networks, GPRS, SDN, Software Defined Networking, NFV, Network Functions Virtualization, OpenFlow, signaling and user data separation, wireless networks, cellular networks, PCU-ng, PCUng, ePCU, vGSN, ReST, MAC tunneling, Ethernet tunneling, ICMP topology discovery, ARP APN search.

1 Introduction

One of the drivers behind software defined networking (SDN) trend was the inflexibility of existing networking approaches and industry that limited the space for innovation. On the other hand, researchers also struggled with black box networking approaches and architectures, which limited the experimental capabilities of existing network equipment. Since then, SDN spread through wired networks and it is making its way, together with network functions virtualization (NFV), to the network operator world, where most of the industry struggles with network equipment, which is often hard to integrate with existing infrastructure that does not provide open interfaces, so complicated work arounds need to be done.

In UnifyCore architecture we are trying to address the heterogeneity and inflexibility of the network infrastructure, which causes complicated network management, control, new service deployment and orchestration. This is the case mainly with large network operators, who provide services over multiple technologies

such as multiple wireless technologies (GPRS/UMTS/LTE), xDSL and optical at the same time. Customers naturally expect same look and feel of the service regardless of the technology being used. With standard networking approaches, this hard and often expensive to reach.

Our UnifyCore approach offers joint control by using open APIs on the central SDN controller and access network control elements (access managers). By using this approach, network operators can easily orchestrate and have better control of the network.

The paper is structured as follows. First two sections give an overview of the foundation of mobile networks and SDN. Next, state of the art in the area of mobile software defined networks is briefly introduced. Rest of the paper focuses on the UnifyCore architecture and its features. Last section concludes the paper.

2 Mobile Networks Basics

As general packet radio service (GPRS) was the first network technology we integrated into UnifyCore, we will first introduce some essential concepts of this network. In this paper we focus only on the packet switched part of the network, therefore we won't explain procedures and nodes of the circuit switched part of the network.

GPRS network consists of the radio access network (RAN) and the core network (CN). In RAN, base transceiver station (BTS) and base station controller (BSC) are located. BTS is a device which handles the radio interface. It is responsible for modulation/demodulation, error checking and correction and communicates with BSC on one side and mobile station (MS) on the other side. In BSC, all logic of the radio access network is located. Multiple BTSs are controller by a single BSC. BSC connects the RAN to the core network, more precisely to the serving GPRS support node (SGSN). This node is responsible for mobility management, session management, authentication and ciphering in the GPRS network. Further to the core network, SGSN connects to the gateway GPRS support node (GGSN). As the name implies, this node is a gateway from the mobile network to the external networks such as Internet or corporate intranet/VPN.

A basic call flow in mobile network includes two main procedures. Fist attach procedure is executed. During this procedure the mobile station is authenticated and gets connected to the network. At this point, mobile station does not have any IP connectivity. Circuit switched calls and SMSs are available (attach both to circuit switched and packet switched part of the network is assumed). In order to communicate for example with the Internet, second procedure called PDP context activation has to be executed. In this procedure, the mobile station specifies the service, which is requested by filling up the access point name information element (APN). If the procedure succeeds, the network assigns an IP address to the mobile station and transfer of the data across the network is possible.

Further details about GPRS and other mobile networks such as universal mobile telecommunications system (UMTS) and long term evolution (LTE) technologies can be found in respective 3GPP standards or books [1,2,3].

3 Software Defined Networking

As mentioned before, the key driver behind SDN was situation in network industry, that mainly used black boxes from different vendors, which provided only CLI or SNMP for management and integration, and there was no standard APIs providing full control over the network appliance. This situation made integration of network infrastructure of different vendors very difficult and expensive. Such integration complicated network automation and integration processes. It also led to a vendor lock-ins in some cases. From the research point of view, black boxes provide little to no space for experiments, so SDN was introduce to challenge these limitations.

SDN brings separation of user and control plane of the network appliance. By doing this, each plane can evolve separately and can be optimized for its needs. Moreover as these two functions formerly residing in the same box are split by SDN, need for a communication protocol or API between these two planes was evident. Most successful SDN approach is probably the OpenFlow protocol.

3.1 OpenFlow

OpenFlow, as the name induces, builds on the idea of network flows. A flow in the network is specified by n-tuple of protocol header fields. Different set of protocol headers and header fields are supported in each version of the protocol. OpenFlow network is composed of OpenFlow controller which communicates with OpenFlow switches or forwarders in other words.

Forwarder is composed by set of flow tables, where flow entries can be written and by which packets are processed. In each flow entry, selected protocol header fields – match fields are specified, and set of actions to be performed after match are associated with it. Flow entries are installed by the SDN controller at any time. When a packet is received by the OpenFlow forwarder, its header fields are compared against flow entries and in case of match actions and instruction are executed. This way, any new networking approach is dependent only at the logic in the controller, since the OpenFlow protocol and OpenFlow switch capabilities are standardized and at atomic level (network flow) [4].

There are many more SDN related approaches, both academic – I2RS [5], ForCES [6], PCEP [7] and vendor specific – OnePK [8], but these have little relevance to our work, moreover OpenFlow is the leader on the market and the academia.

4 Related Work

Most of the present work focusing on mobile SDN is addressing different kinds of mobile gateway nodes decomposition or network functions placement [9] [10]. Then there are approaches, which address network architectures in general and bring new

use cases and functionalities, which are enabled by OpenFlow [11]. Some of telco vendors address mobile SDN with their specific approaches such OpenFlow's mobile counterpart MobileFlow [12] or extend standard OpenFlow protocol with mobile specific features [13]. Third part of the SDN mobile related research is the SDN based/controlled RAN [14] [15].

The vast majority of the papers focus on the same technology – LTE. However GPRS, on which the UnifyCore demo is based, is the dominant technology for the M2M services, thanks to its maturity and simple radio interface that enables low terminal price that is crucial for massive M2M deployment. Finally, GPRS is expected to continue to provide such services and an umbrella fallback network for next one or two decades.

5 UnifyCore – Novel Core Network Architecture

UnifyCore architecture was developed with backwards compatibility and SDN focus in mind. It is aimed to provide mobile services and features of packet core (GPRS/UMTS/LTE), but can be also used as a transport core platform for aggregation of traffic from different access technologies and provide umbrella control and automation platform.

In UnifyCore architecture, the access technology specific protocols are terminated as close to the border between access network and core network as possible. The idea behind this is to use a common transport core, which is not complicated by various access technologies. Different access technologies such as GPRS, UMTS and LTE or WiFi are controlled by dedicated control elements called access managers. These nodes understand the signaling protocols used by the access network and terminals and provide necessary operations such as mobility/session management and signaling. As we mentioned before, common transport core is independent of access technologies connected to it, thus is controlled by a logically separate element – SDN controller. Core control SDN controller and access managers communicate via ReSTful API.

Traffic in the different access networks is usually encapsulated to various access specific protocols, moreover some technologies combine control and user data in a single stream of messages (for example GPRS, as shown later in the evaluation part). For separation of user and control plane data, UnifyCore uses OpenFlow enabled border forwarders called adaptors. Some of the access network protocols are not compatible with present OpenFlow match rules, so we use OpenFlow extensions to support such protocols. These extensions have to be supported on both access managers and border forwarders (adaptors), however they do not have to be supported in core, as it is access agnostic and based just on Ethernet tunneling [16]. This further emphasized the aim for simple common core.

From the mobile networks architecture, UnifyCore borrows the APN concept. As mentioned in the second section, in 3GPP mobile network, the APN is associated with GGSN (P-GW in LTE) interface and signifies a service offered at that point. We use the concept of APN, but as we do not have a GGSN or P-GW in our architecture, our APNs may be located at any border forwarder.

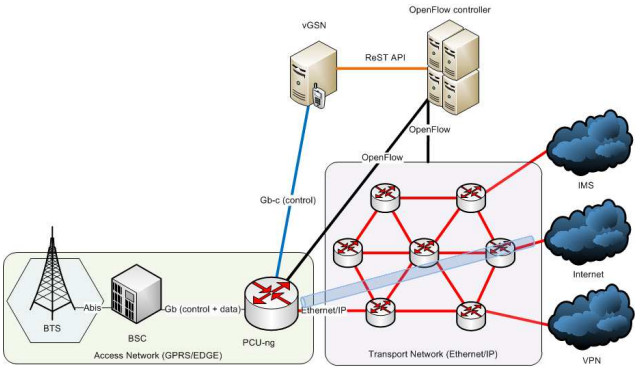

Fig. 4. High level UnifyCore architecture

Further details on the philosophy and architecture can be found in previous paper on the topic [16]. In this paper, we focus on the backwards compatibility enablers of the UnifyCore – mainly ICMP topology discovery and ARP APN search.

5.2 ICMP Topology Discovery

In order to setup a MAC (Ethernet) tunnel, few procedures have to be executed. These procedures include ICMP topology discovery (executed when a new OpenFlow forwarder joins UnifyCore topology) and ARP discovery for localization of traffic egress and ingress points (APNs). For the topology discovery UnifyCore uses its own topology discovery method based on the ICMP protocol.

Process works in two phases. First phase includes bootstrap of OpenFlow enabled forwarders. When a forwarder joins UnifyCore controller, it is asked to clear its whole configuration. Next a new OpenFlow rule is installed to first flow table (table 0 in our case). This rule forwards all ICMP echo requests with given destination IP to the controller. Together with this topology discovery flow rule, a rule for ARP discovery is installed as well (explained in separate section of the paper). If this new node is an adaptor type, extra rules are installed. These rules ensure adaptation of access network user traffic for core network and routing of control plane messages to the access network manager.

Second phase is the topology discovery itself and starts when controller constructs ICMP echo request with encoded source forwarder ID (datapath ID) and source port ID in the payload of ICMP message and injects them to all ports of newly joined forwarder. As these packets reach the adjacent forwarders (these forwarders joined network before), they are matched with the ICMP discovery rule and are forwarded back to the controller. Controller examines the message that has been just forwarded to it and extracts the source forwarder ID and source port from the OpenFlow header and originator forwarder ID and port from the ICMP payload. From this information, controller is able to construct a view of topology.

ICMP topology discovery method, same as MAC tunneling, is compatible with standard featureless L2 switches. If a L2 switch (or group of switches) connects two

forwarders, incoming ICMP echo will be flooded to all ports of the switch and finally will reach some adjacent OpenFlow forwarder, which will send the ICMP echo to the controller. Controller will examine the content of the OpenFlow header and payload and update the topology accordingly. In this case, the L2 switch connecting two OpenFlow forwarders is considered to be a direct link between the sending and receiving forwarder. However this L2 switch does not break the UnifyCore concept and capabilities in any way. If there are more interconnected switches between two forwarders, the ICMP echo may be received by the controller multiple times and multiple connections may be discovered (Fig. 6).

5.4 ARP Search – Ingress and Egress Point (APN) Discovery

As mentioned before, the APN represents the ingress and egress point of the UnifyCore domain.

At the very start, UnifyCore controller looks at its configuration file and finds all the APNs (ingress and egress points) it is serving. Each APN name from the configuration file gets resolved by the DNS lookups to an IP address. Next, when a forwarder joins the network, together with ICMP topology discovery ARP search process is executed at this forwarder.

The process has several phases. It starts with the deletion of the whole flow table configuration (as mentioned earlier). This first step is common for ARP search and ICMP topology discovery. Next, rules for ARP search are installed on this new forwarder (together with ICMP discovery rules as mentioned before). First flow rule installed is the redirection of all ARP replies to the controller. At this rule we match Ethernet type 806 and ARP operation 2. Action for this flow rule is to forward ARP replies to controller, where it could be further processed. It has to be noted, that from the definition, OpenFlow forwarders do not feature ARP logic, therefore ARP message processing has to be done in the controller (or non-standard OpenFlow extensions have to be used).

Next, the controller sends an ARP request from each port of the forwarder. For each APN in the database (from the configuration file) controller sends one ARP request per forwarder port (target IP address of the APN). Following this approach, we expect, that at the APN location (adjacent network domain) there is a non-SDN capable edge router, thus we use a standard “legacy” ARP procedure. If a SDN capable domain was behind the UnifyCore domain, we could have used some SDN inter-domain signaling. This approach further improves UnifyCore compatibility with existing legacy networks.

It has to be noted, that since ARP is a LAN protocol, the source IP address has to be from the same subnet, as is the APN. Moreover different APNs can have and normally they have different IP addresses from different subnets, therefore controller choses addresses from these domains.

When these ARP request are sent out through all ports of newly added forwarder, they are captured and processed by the adjacent forwarders or an edge router serving the APN. In case of adjacent OpenFlow forwarder nothing happens and no ARP reply is generated (because OpenFlow forwarders do not process ARP the way standard routers do). In case of edge router with given IP address (serving the APN controller

was looking for), the router generates ARP response, which will be received by the given forwarder and sent to the controller (ARP reply rule matched). Controller processes the message and extracts the forwarder ID and port ID from OpenFlow header. This way, the controller discovers APNs, their location in the network topology and can construct tunnels for user traffic transport (Fig. 6).

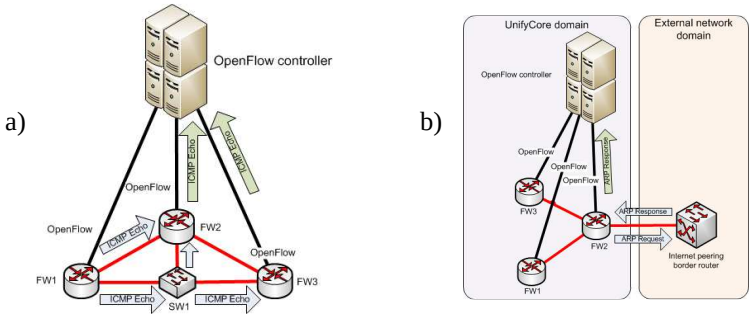

Fig. 6. ICMP topology discovery (a) and ARP search (b).

When a new ingress or egress point (APN) location is found, controller starts tunnel setup between all already discovered APNs and this newly discovered one. First a shortest path algorithm is executed, which returns a set of forwarders and ports which should be used along the way from one APN to another. If an ingress point is an adaptor, first flow table is left for the traffic adaptation rules, which are set on a user basis. Rules in this table will strip off the access specific headers and forward packets to second table, which is the MAC tunnel table. Here the destination MAC address of the Ethernet frame is set and the frame itself is forwarded to tunnel by assigned interface. Next forwarders along the way perform very similar task. They match the destination MAC address and forward the frame to respective port (given by the OpenFlow rule). The very last forwarder in the way may change the destination MAC address to match the MAC address of the egress point (APN). This is the case only when more MAC tunnels are established to the same egress point (APN), for example for different QoS classes or tunnels from different source. In case of single tunnel towards this given APN, destination MAC address corresponds to the MAC address of the router in the adjacent domain

In the opposite direction (downlink), border router serving given APN in the adjacent domain could search for MAC address of an IP address present in the access network. As mentioned before, forwarders do not have the capability to respond to ARP request, so this is forwarded to the controller by an OpenFlow rule. Controller responds with the MAC address of the tunnel belonging to given end device in the access network. After receiving the requested MAC address, the edge router sends packet to the edge forwarder. This forwarder examines the destination MAC address and forwards it to a given port, specified by the OpenFlow rule. Next forwarders in the way to the access network forward the Ethernet frame in a similar manner. The access edge forwarder (adaptor) finally appends the access specific protocol headers, and in case there are more tunnels towards this endpoint, sets the destination MAC address to the MAC address of the first network node in the access network.

It has to be noted, that uplink and downlink tunnels have different tunnel IDs (MAC addresses), and so traffic can be routed in an asymmetrical manner.

Presence of tunnels even before any need for data transfer further enhances the session setup time. In comparison with for example GPRS or UMTS, where the tunnels are created in a dynamic manner based on mobile station requests. Tunnel setup by exchange of signaling messages between SGSN and GGSN takes naturally more time, than proactive tunnel setup at UnifyCore start.

7 Evaluation

For our initial UnifyCore proof of concept, we chose a rather specific use case – GPRS over UnifyCore. As mentioned before, most of the mobile oriented SDN research papers deal with LTE or UMTS. However, both technologies share the split user plane and control plane approach, thus introduction of SDN to such system is rather trivial.

Our work focuses on GPRS, which is basically the oldest packet based 3GPP network. Despite its age, it is still being heavily used around the globe. Moreover development on this technology continues and for example release 13 GPRS/EDGE terminals and networks bring further enhancements for the M2M use cases [17]. This indicates that even now, GPRS is highly relevant network technology and integration of GPRS and SDN is an interesting topic.

We implemented the UnifyCore GPRS architecture in the following way. We removed SGSN and GGSN from the architecture and split their logic between SDN controller and GPRS access manager called vGSN (virtual GPRS Support Node). Session management (tunnel management) functions are centralized in the SDN controller and GPRS signaling (mobility management, signaling and authentication) is performed on the vGSN. This function split is following the UnifyCore concept introduced in one of previous paper [16]. In user plane we use a GPRS adaptor (GPRS enabled OpenFlow forwarder), which first splits GPRS message stream into user plane data and signaling messages. Next the signaling is sent to vGSN and user plane data is adapted to pure Ethernet (MAC tunneling). In the downlink direction, the GPRS adaptor encapsulates the pure Ethernet data into GPRS protocols and sends it to GPRS radio access network (Fig. 7). We named the GPRS adaptor ePCU or PCU-ng. This stands for enhanced PCU or PCU for next generation networks.

For the evaluation, we implemented the whole solution over the open-source software. As a controller base, Ryu controller framework was used. In the controller, OpenFlow extensions were added, in order to enable controller to command the ePCU (GPRS protocol stack extensions). As a forwarder, ofsoftswitch13 was chosen. GPRS protocol stack extensions were implemented here as well. The GPRS access manager module is based on open-source code from a hacker community, which is focusing on security holes in mobile networks – osmocom. Snippets of source code of two projects – osmo-sgsn and openGGSN were combined in order to build our vGSN. In the GPRS access network sysmoBTS hardware was used. This base station is compatible with osmo-sgsn and compliant to standard 3GPP Gb interface signaling.

The setup was verified using off-the-shelf mobile phones of different types – from smart phones to feature phones. During tests, terminals were not aware of any changes in the core network, which was basically one of our most important goals and GPRS data transfer was functional in both directions.

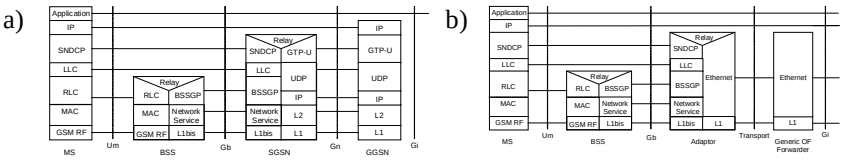

Fig. 7. GPRS protocol stacks (user plane) in standard 3GPP architecture [1] (a) and in UnifyCore based architecture (b).

6 Conclusion

The transformation from classical network architectures to SDN based is inevitable. However, very similar to IPv4 to IPv6 transition, for a certain time, classical networks and SDN networks will coexist. First in the form of SDN islands inside classical network sea, next the situation will be just the opposite. Finally, SDN will become the dominant networking technology.

For this transition period, UnifyCore features set of approaches such as ICMP discovery and ARP search, which enable it to integrate with standard router/switch based transport architecture. These methods not only allow UnifyCore to communicate with existing adjacent infrastructure, but also allow operators to protect past investments in the existing hardware with which is UnifyCore fully compatible.

Both ARP APN discovery and ICMP topology discovery mechanisms might seem redundant, but it has to be noted, that pure OpenFlow forwarders do not support standard features of switches or routers such as ARP message processing or Ethernet broadcast forwarding/flooding. Processing of such messages has to be set by the controller by OpenFlow match rules, actions and instructions.

From the 3GPP mobile network point of view, the UnifyCore easily integrates with standard 3GPP networks – end to end by Gb and Gi interfaces. Our prototype proves that even complicated Gb interface (without user data and signaling separation) is easy to integrate into UnifyCore with a flexible SDN approach.

At the time being we are starting with performance evaluation of the key features of the GPRS prototype. As mentioned before, functional validation was already done with real mobile phones, however such setup was unable to generate traffic load.

Performance of MAC tunneling implemented over user space forwarder application (ofsoftswitch13) is being evaluated using common iPerf2 and iPerf3 tools. For the GPRS related parts (signaling and user data separation, GPRS encapsulation/decapsulation) we are not aware of any free open-source performance measurement tools. Therefore, commercial tools such as Spirent LandSlide [18] or

Ixia EPC test [19] need to be used, or new tool for such evaluation has to be implemented from scratch.

Acknowledgments. This work is a result of the Research and Development Operational Program for the projects Support of Center of Excellence for Smart Technologies, Systems and Services, ITMS 26240120005 and for the projects Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 26240120029, co-funded by ERDF.

References

1. 3GPP: 23.060 rel. 13.1.0 – General Packet Radio Service (GPRS); Service description; Stage 2, 2014.
2. 3GPP: 23.002 rel 13.1.0 – Network architecture, 2014.
3. Ahtiainen, A., Kaaranen, H., Laitinen, L., Naghian, S., Niemi, V.:UMTS networks: architecture, mobility and services. John Wiley &Sons Ltd, 2005. ISBN 0-470-01103-3.
4. Open Networking Foundation: OpenFlow Switch Specification 1.4.0, https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, 2013
5. IETF: Interface to the Routing System, http://datatracker.ietf.org/wg/i2rs/charter/, 2015
6. IETF: Forwarding and Control Element Separation workgroup, https://datatracker.ietf.org/wg/forces/documents/, 2015.
7. Le Roux, J., Vasseur, J.: RFC 5440 – Path Computation Element Communication Protocol, 2009.
8. Cisco: Cisco's One Platform Kit (onePK), http://www.cisco.com/c/en/us/products/ios-nx-os-software/onepk.html, 2015.
9. Basta, A., Kellererm W., Hoffmann, M., Morper, J.,H., Hoffmann, K.: Applying NFV and SDN to LTE mobile core gateways, the functions placement problem, In Proceedings of the 4th workshop on All things cellular: operations, applications, 2014.
10. Hampel, G.; Steiner, M.; Tian Bu: Applying Software-Defined Networking to the telecom domain, Computer Communications Workshops (INFOCOM WKSHPS), 2013.
11. Jin, X., Li, E.,L., Vanbever, L., Rexford, J.: SoftCell: scalable and flexible cellular core network architecture. In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, 2013.
12. Pentikousis, K., Yan, W., Weihua, H.: Mobileflow: Toward software defined mobile networks. In Communications Magazine, IEEE , vol.51, no.7, 2013.
13. Kempf, J., Johansson, B., Pettersson, S., Luning, H., Nilsson, T.: Moving the mobile evolved packet core to the cloud. In: WiMob, 2012.
14. Yang, M., Yong, L., Jin, D., Su, L., Ma, S., Zeng, L.: OpenRAN: A software-defined RAN architecture via virtualization. In SIGCOMM Comput. Commun., 2013.
15. Gudipati, A., Perry, D., Li, E., L., Katti, S.: SoftRAN: software defined radio access network. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13), 2013.
16. Nagy, M., Kotuliak, I.: Utilizing OpenFlow, SDN and NFV in GPRS Core Network, In. Testbeds and Research Infrastructure: Development of Networks and Communities - 9th International ICST Conference, TridentCom 2014, Guangzhou, 2014.
17. Nokia: Nokia LTE M2M , Optimizing LTE for the Internet of Things, 2014.
18. Spirent: Landslide, http://www.spirent.com/Ethernet_Testing/Software/Landslide, 2015.
19. Ixia: EPC test, http://ixiacom.com/solutions/wireless/epc-test, 2015.