Mobile Core Traffic Balancing by OpenFlow Switching System

Ebrahim Ghazisaeedi
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
eghazisaeedi@sce.carleton.ca

Rahim Tafazolli
Centre for Communication Systems Research
University of Surrey, Guildford, UK
r.tafazolli@surrey.ac.uk

***Abstract*—Nowadays, one of the main problems of mobile operators is to handle network traffic more resourcefully. Techniques, which increase the traffic capacity over the mobile core with minimum costs, play a fundamental role. Following advent of smart phones, the most of the traffic bandwidth is being used for connecting to external IP networks, like the Internet. Consequently, this overloaded traffic causes congestion over the mobile core network and degrade the quality of service. In this regard, we suggest a process, by benefiting from recently revealed switching system (OpenFlow), in order to balance the traffic over the mobile core network and use resources more efficiently. In this paper, we imply a technique to decrease the core network load using this switching system and provide a better quality of service for specific services over 3G and Evolved Packet Core (EPC). Mobile Operators with the proposed solution may save millions of dollars per year.**

Keywords-component; OpenFlow, Traffic Balancing, LTE, EPC, UMTS, Mobile Core Network

I. INTRODUCTION

The telecommunication industry is in a period of radical change with the advent of mobile broadband radio access and the convergence of Internet and mobile services [1]. IP and packet-switched technology are soon expected to be the base for data and even voice services on both the Internet and mobile communications networks [1]. Today, LTE network has been revealed as an end-to-end IP based network which the both radio access and the core network are based on IP.

Now, the mobile operators provide lots of new services to their customers. These kinds of new services enable customers not only use Internet on their handheld devices, but also connect their laptops to the Internet network through the mobile network coverage. Depending on the location, the increase in mobile data traffic ranges from 300 to 800 percent a year in 2008 [3]. The high demand of bandwidth for increasing number of mobile subscribers as well as overloaded traffic which is result of Internet consumption, leads to congestion in mobile core networks. The congestion degrade the quality of mobile network's service and make the operator and customers unsatisfied with the system. This became a big challenge for the mobile operators. The easiest solution for this difficulty is to increase the number of core network elements to handle this amount of traffic, while it costs a lot for the providers and make it more difficult to manage the whole network. Moreover, some other solutions have been proposed which require reconfiguring in some part of the core network that is not easy to be implemented.

In this paper, we suggest a process which benefits from adapted version of OpenFlow switching system. In this regard, we try to balance the traffic by rerouting the Internet traffic over mobile core network.

Since the large portion of overloaded traffic is coming from the Internet connections and 47.71% of the Internet usage is for HTTP applications [2], we suggest re-routing a portion of the Internet traffic, at the core network, directly to the Internet network. Besides, as there are lots of different traffic flows in mobile core for the same applications, we try to aggregate them using OpenFlow switching system. Moreover, as the applications like VoIP which use Internet, need less latency, we route them through shorter circuit switching paths. Furthermore, we propose different options to enable OpenFlow switches detect the HTTP application flows. In consequence, we anticipate decreasing the traffic load of mobile core network traffic, which caused by Internet usage. As it was expected this scheme is solving the mentioned challenge without adding any main infrastructure. Besides, it is a cost effective approach and also it is not required to reconfigure the network elements.

In the next sections, we reassess the OpenFlow switching system. Moreover, we propose a plan using OpenFlow for mobile networks to increase the quality of service by balancing the core network load. Finally, we evaluate our suggested technique in defined scenarios with OPNET simulator.

II. OPENFLOW SWITCHING SYSTEM

OpenFlow is a new switching system which has recently proposed by Stanford University. OpenFlow is based on a packet switch, with an internal flow table and a standardized interface to add/edit/remove flow table entries. It makes good use of the fact that most recent network switching elements benefits from tables that run at line-rate and map incoming traffic to outgoing ports. While each vendor's device is designed differently, OpenFlow uses a common set of functions implemented in all switches.

OpenFlow provides the ability to program the flow table in different switches using an open protocol. The data path in an OpenFow switch is consisting of a flow table and action which associated with each flow entry. While in normal packet switching network elements each packet is switched

978-1-4799-2851-4/13/$31.00 ©2013IEEE

individually, all packets in a flow are switched the same way, making the flow the fundamental unit of manipulation within the switch.

An OpenFlow switch or router consists of at least three units:

- A flow table, with an action associated with each flow entry, to tell the switch how to process the flow.
- A secure channel that connects the switch to a remote controller.
- The open protocol which provides and open and standard way for controller to communicate with a switch [4].

Each flow table in OpenFlow switches contains flow entries. Each flow entry includes a *Match field, counters,* and a set of *Actions*. The three basic actions are listed below.

- Forward this flow's packets to a given port (or ports).
- Encapsulate and forward this flow's packets to a controller. Typically used for the first packet in a new flow.
- Drop this flow's packets. Can be used for security, to curb denial of service attacks, or to reduce spurious broadcast discovery traffic from end-hosts [4].

III. PROPOSED ARCHITECTURE

Nowadays, most of the mobile operators are providing Internet services for their customers. As in new network architectures like UMTS and LTE, networks could provide a high speed radio access for the customers, not only mobile handhelds are able to use Internet connection on the road, but also laptops are also enabled to use broadband services through the mobile networks. Nevertheless, the increasing number of subscribers who are using Internet overloads the traffic on the mobile core network. Due to capacity limits of core network elements, the quality of service will be degraded and the congestion rate over the packet core network will be increased. In this regard, operators have two options. Firstly, operators could cope with the challenge by increasing the number of core network elements. For instance, an UMTS operator should increase the number of SGSN gateways as well as GGSN routers to deal with the overloaded traffic, and prevent the congestion. However, this routine costs a lot and it is becoming more difficult to manage the whole network. Secondly, 3GPP has released 3GPP Direct Tunnel. Despite the previous approach the Direct Tunnel focus on data path. Direct Tunnel removes the data path between RNCs and SGSNs, and forwards them directly to GGSN. However, this architecture has two disadvantages. First, it is mandatory to reconfigure the network elements, at least GGSN. Secondly, although they bypassed the SGSN and this increase the traffic capacity, but it is still necessary to increase the number of GGSNs.

Nevertheless, in this paper, we investigate a novel process which does not require expanding the infrastructure and reconfiguring the network parts. The operators with our novel method may save million of dollars a year.

As mentioned before, since the most part of overloaded traffic is for Internet consumptions and 47.7% of the Internet usage is for HTTP applications [2], we intend to reroute the Internet traffic, specifically HTTP traffic of subscribers directly to external IP network. In this regard, we need a router, which is able to separate the Internet traffic, aggregate all of the Internet flows, and route them directly to external IP networks.

Hence, we imply to use two OpenFlow switches. The first one will be implemented on the data path from RNC to SGSN in UMTS network. The second one will be placed on the data traffic from eNodeBs to Serv GW for LTE/EPC networks.

These switches just reroute the HTTP traffic. Therefore, they will not affect the critical mobile network tasks like mobility management, location update, authentication, etc, still taken by SGSN/GGSN in UMTS and MME/Serv GW/PDN GW in EPC. Thus, the switches forward all the data, except HTTP sessions, as a simple forwarder to SGSN or Serv GW. When the HTTP sessions are passing the switch they will be detected and rerouted directly to Internet.

Therefore, to separate the HTTP data, aggregate them, and reroute them directly to Internet we proposing to use OpenFlow switching system. OpenFlow switches are able to easily identify HTTP data by checking the headers. OpenFlow will detect the HTTP sessions using destination TCP port which is 80 for HTTP applications. Besides, the data, except HTTP sessions, will be forwarded to their normal connection points, immediately. Moreover, this switching system, using central controller, helps us to utilize the traffic dynamically over the network.

Our proposed architecture is shown in Figure 1. Our suggested switches will be connected to an Internet edge router. The OpenFlow controller controls all the OpenFlow enabled switches concurrently. Thus, the controller is able to detect the congestion over the network. In case of link failure in one of our suggested links, the controller immediately updates the flow tables in switches and forwards the packets normally to SGSN or Serv GW. In the latter case, the OpenFlow switches act as a simple bridge.

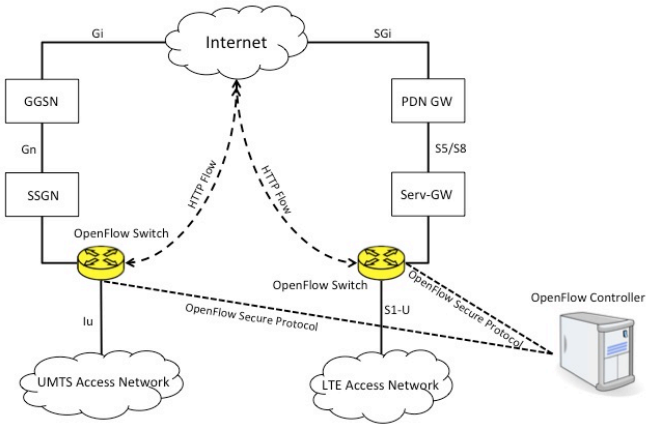

Figure 1. Our Proposed Architecture

The OpenFlow switch needs to identify the TCP header. However, the data over Iu or S1-U interfaces are already encapsulated using GTP-U. GTP-U adds GTP and UDP/IP header to the original packet. We propose three options to solve the problem.

1) Option 1

It is required to de-capsulate packets over Iu or S1-U interfaces before giving them to OpenFlow switch, while we are saving the tunnel header. We assume the links as unidirectional, and propose different systems for outbound and inbound traffic.

For outbound traffic, as it shown in Figure 2, firstly, we decapsulate the packet and save the GTP/UDP/IP header fields. Besides, some of the vital information of packet will be saved in a table. In this regard, each switch will make a table. The table as is shown in Figure 2, will contain Source IP address, Source UDP port, and TEID. This information will be used later for the inbound traffic to encapsulate the data and put them back into the tunnel. Each tunnel is unique with its TEID, and UDP port number.

Secondly, the packet will be forwarded to OpenFlow switch. The OpenFlow switch may redirect the packet to core network, SGSN or Serv GW, since the traffic may not come from HTTP applications. In this case, we reattach the saved header to the packet. Nevertheless, OpenFlow switch may redirect the packet to the Internet gateway, since the traffic is from HTTP applications. In this case, the de-capsulated packet will be routed normally to the Internet.

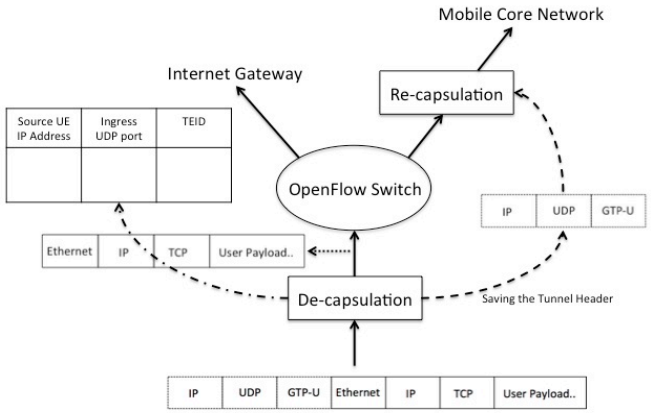

Figure 2. Option 1-Outbound Process

For inbound traffic, the requested data will come back to the OpenFlow switch. The data may arrive to the OpenFlow switch from the core network. So, the packets already have the tunnel header, and OpenFlow switch simply forward them to RNC or eNodeB. However, the packet may arrive directly from the Internet port. These packets should get back to the tunnel which is over the Iu or S1-U interfaces. So, we need to reconstruct the GTP-U header and encapsulate the packets using the saved table information, before giving them to OpenFlow switch. We just need to know the TEID of tunnel and UDP port number of ingress end-point. As mentioned in outbound traffic process each switch makes a table based on Source IP address (UE source IP address), Source UDP port (Ingress Point UDP port number) and TEID. The incoming packet has a destination IP address which is destination UE IP address. Therefore, we are able to easily match the destination IP address over the table and find the egress UDP port number and TEID. By having these values, the header will reconstructed and the packet will be encapsulated. The encapsulated packet will be forwarded to OpenFlow Switch and OpenFlow switch simply forward it back to the tunnel. OpenFlow does not need to identify the packet header, and easily forward all incoming packets back to the tunnel. This process is shown in Figure 3.

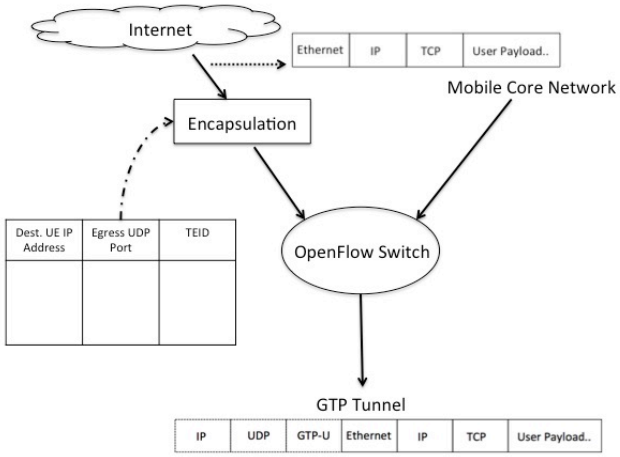

Figure 3. Option 2-Inbound Process

It is required to mention that the saved table elements are the same for inbound and outbound while for outbound we log the source IP address of packet which will be destination UE IP address for inbound traffic. The same story will happen for UDP port. For outbound traffic, we log the ingress UDP port number while it will be changed to egress UDP port number for the inbound traffic.

By this method we do not need to reconfigure any network elements, even OpenFlow switch standard. However, we may experience more delay as the penalty we are paying for the method. The delay for different suggested options will be reviewed in IV.B.

2) Option 2

Secondly, we suggest updating the OpenFlow switch standard. OpenFlow switch, as discussed in II, is able to identify VLAN tags and MPLS header formats. This means OpenFlow will ignore the VLAN header and will check the payload to find and match the header fields. By updating the OpenFlow switch to enable it to identify the GTP encapsulated data, it is not needed to de-capsulate data stream before getting to the switch, while OpenFlow is able to easily identify the HTTP traffic by checking the packet payload and probing the TCP header field.

Using this approach, we still do not need to reconfigure any network element, except OpenFlow switch. This method will experience less delay in compare of previous one.

3) Option 3

Finally, to enable OpenFlow switch to identify HTTP traffic, we could add a tag or label for HTTP packets at the RNC or eNodeB side. The RNC or eNodeB will identify the

HTTP packets by probing the TCP header and will add a pre-defined tag after GTP header. The OpenFlow switch is able to identify the HTTP traffic by probing the tag, RNC or eNodeB has added to the packets.

This approach requires reconfiguration of the standards over the RNC which is less desired for us and the mobile operators. However, this may load less latency over the network.

IV. SIMULATION

In this section, we try to evaluate our proposed method using OPNET ver16.0. The suggested architecture will be simulated whereas the scope area is covered by LTE access network. We try to assess our advised process for HTTP traffic while measuring the core network traffic. Thus, it will be possible to compare the traffic load over the core network with/without our planned technique, and evaluate different proposed options.

In this regard, we use the topology which is shown in Figure 4. The topology has a 7-cell LTE network whereas each cell has an eNodeB at the centre. Four mobile nodes are covered by one eNodeB. The eNodeBs are linked to an OpenFlow switch through 100BaseT cables. The switch is already connected to Serv GW as well as directly to the Internet. Besides, the HTTP traffic over the network will be generated with the following specifications:

- HTTP ver. 1.1.
- Maximum Connections = 2.
- Maximum Idle period = constant, 10 seconds.
- Page Inter-arrival Time = exponential, mean=5 second.
- Object Size = constant, 10000 bytes.
- Pages per Server = exponential, Mean=10.
- Type of Service = Best Effort

Moreover, in each cell, the arrival process has uniform distribution with 5 seconds mean.

To evaluate the implied process precisely, two scenarios are simulated over the main topology.

A. First Scenario

By using our suggested architecture, it is expected to get a lower level of traffic over the mobile core network. This is the main objective of our proposed method. Thus, to assess this impact of the technique on the Evolved Packet Core, we simulated the topology of Figure 4, while all the eNodeBs are connected to our optimized OpenFlow switch. The switch is programmed to function as our suggested procedure in III. EPC traffic, received/sent data rate, is measured to evaluate the objectives.

We measured total number of broadcast, multicast and unicast IP datagrams received/sent per second by the EPC from the network across all the IP interfaces in the EPC. The Cumulative Distribution Function (CDF) of gained results is plotted in Figure 5 for IP.Received data rate and in Figure 6 for IP.Sent data rate.

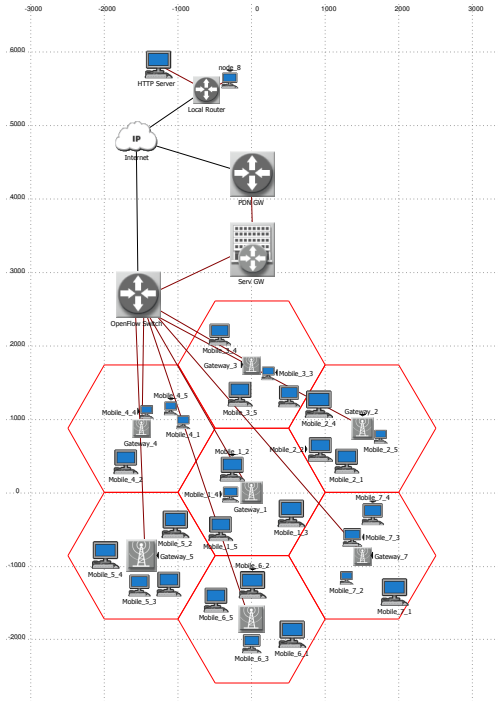

Figure 4. Simulation Topology

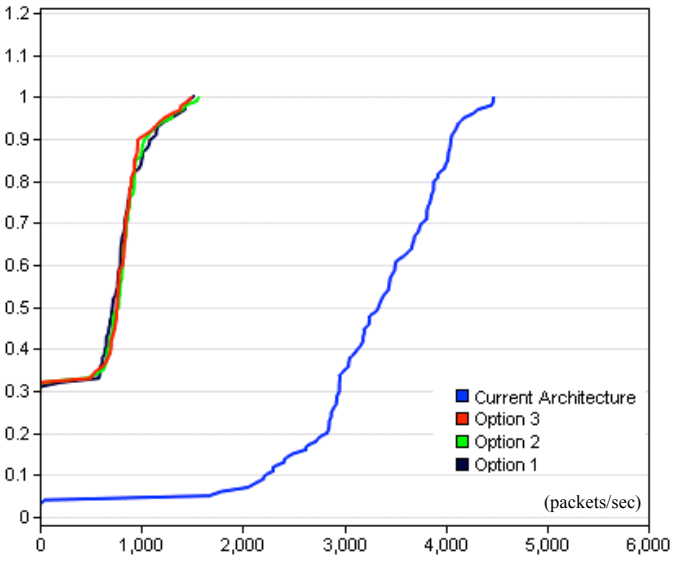

Figure 5. CDF of EPC Received Traffic

CDF of received IP traffic by EPC is plotted for different suggested options. Firstly, the topology is simulated in current architecture of mobile network, without using our suggested method. In this mode, OpenFlow switch is removed from the topology. This curve is drawn in blue color. Secondly, CDF of the EPC received IP traffic is plotted while our suggested method is used and the OpenFlow switch is added to the network topology. The result for different options is plotted.

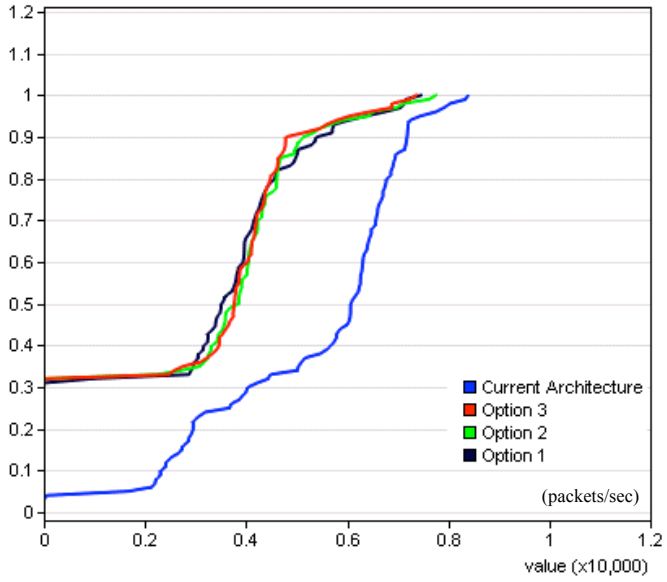

Figure 6. CDF of EPC Sent Traffic

Comparing the CDF curves, current architecture curve is in far right side of the other curves, while the other option curves give almost the same traffic load over the EPC. This means by using our proposed architecture the probability of higher inbound traffic rates over the EPC, has been decreased. Hence, our advised technique decreases the inbound traffic of EPC. This proves our expected objective of the implied scheme.

The same analysis could be used for the Figure 6. In this figure, CDF of outbound EPC traffic is plotted. Comparing the CDF curves shows the higher outbound traffic rates, by using our implied method, are less probable than when network is working in current architecture. This means our recommended architecture decrease the outbound traffic of Evolved Packet Core.

Thus, the proposed process with any of the identification options helps to decrease the load of Evolved Packet Core. This is the main achievement of the project. All in all, this scenario proves that our implied technique in III will degrade the overloaded traffic came from the Internet consumption, on the EPC.

B. Second Scenario

In previous scenario, we have showed our new technique decreases the overloaded traffic of mobile packet core. However, in III, we have suggested three options. Each option tries to imply a way to enable OpenFlow switch identifying HTTP traffic. Each option has its advantages and disadvantages. In this scenario, we try to give a good comparison between these three options by using simulation.

The same topology, which was shown in Figure 4, is simulated. However, all proposed options are examined using OPNET simulator.

HTTP traffic delay is measured over the network for each option. The Delay (in seconds) of packets received by the TCP layer in HTTP server is calculated for all connections. It is measured from the time an application data packet is sent from the source TCP layer to the time it is completely received by the TCP layer in the HTTP server node. The CDF of traffic delay for all of the options is plotted in Figure 7. The blue curve stands for current architecture of mobile network while the proposed technique is not used. The red curve stands for when the suggested method is used with Option 3. The green graph shows the result for option 2, and the dark blue curve stands for option 1.

Analyzing the CDF of TCP delay graph, for different options, shows that option 1 packets experience more delays over the network in comparison to the other options. This was expected since option 1 packets need more processing than the other options. For this option, all the outbound traffic needs to be de-capsulated and in some cases inbound traffic needs to be encapsulated. This increases delay time.

Besides, the option 3 has more delay than option 2. However, option 2 has less delay than option 3, even half of the option 2 curve has lower values than current architecture of network. This shows if eNodeBs identifies the HTTP traffic and add a tag to the packets, takes more time than when OpenFlow identifies the HTTP traffic and reroute it. Nonetheless, as the traffic of option 2 will be routed directly to the Internet, packets may experience less delay than the current architecture of network, due to shorter path.

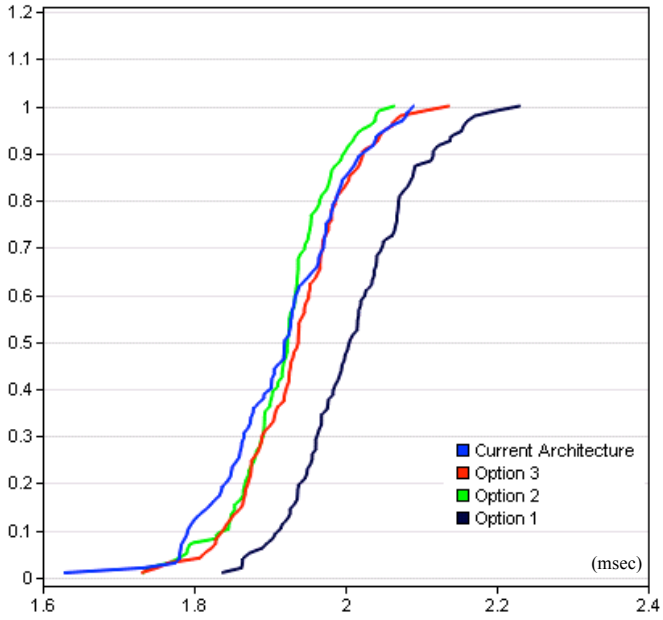

Figure 7. CDF of TCP Delay for different suggested options

Furthermore, CDF of IP processing delay is plotted in Figure 8. We measure the delay experienced by an IP datagram though the IP layer (i.e, the delay from the time when the packets arrives at the IP layer to the time it is dispatched). This delay includes queuing and processing delay.

The diagram shows that option 1 packets experience more IP processing delay than the other options. While option 2, 3 and even in current architecture of network has almost the same IP processing delay.

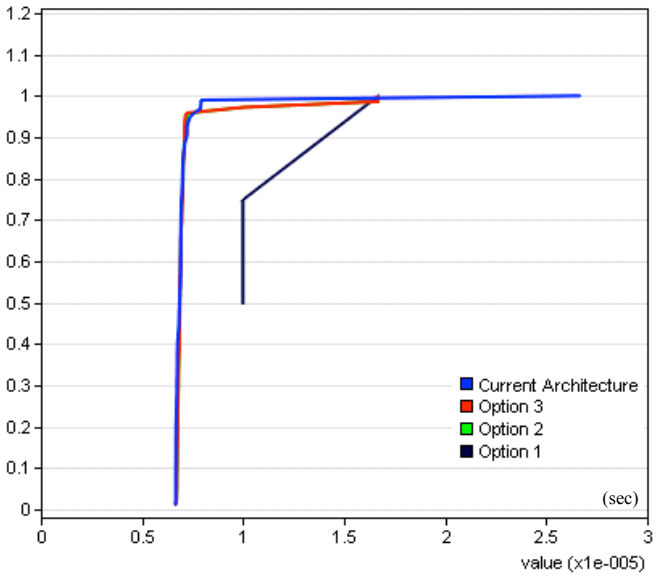

Figure 8. CDF of IP Processing Delay for different suggested options

In conclusion, option 1 objective is to enable OpenFlow switch to identify HTTP traffic while it is not forcing operator to reconfigure any network element. Nevertheless, the penalty that we should pay is delay. Option 2 and 3 brings almost the same delays. Conversely, option 3 requires eNodeB reconfiguration, while option 2 just need OpenFlow switch matching system reconfiguration.

V. CONCLUSION

Nowadays, most of the mobile operators provide Internet access for their customers. This loads a huge traffic over the mobile core network. Since, all of the network elements are bandwidth limited, lots of the packets will be dropped due to the low capacity or congestion. Hence, the service quality will be degraded and users will be unhappy.

Nevertheless, in this paper, we have proposed an approach, which is not expensive and does not necessitate operators, or vendors, to reconfigure the network parts.

The most portion of overloaded traffic is coming from Internet consumption. In addition, 47.7% of the Internet consumption over the mobile networks is for HTTP applications [2]. So, we suggested using recently revealed switching system, OpenFlow, route the HTTP traffic directly to the Internet, before reaching to the core network. The approach is not interfering with the main tasks of the network, like mobility management. Besides, it is not essential to reconfigure the network elements. The suggested technique is expected to decrease the traffic load over the mobile core network of UMTS and LTE/EPC. Moreover, as OpenFlow need to identify the HTTP traffic over the GTP-U tunnel, three different identification processes has been proposed. .

VI. FUTURE WORK

We implied a novel way using OpenFlow switching system for the mobile networks. We proved that our suggested architecture decreases the traffic over packet core network. In the proposed method, the HTTP traffic is routed directly to the Internet. The method effect for the EPC has been reviewed, and showed a worthy consequence. However, when HTTP traffic is routed, directly, to the Internet before reaching to the core network, the operators may have some concerns for the billing, authentication and security issues. Hence, as the first future plan, we need a signaling system between our suggested switch and HSS, in UMTS network, and MME, in LTE/EPC network (Figure 9). It is suggested to use statistics table of OpenFlow that saves all statistics of packets in forwarding and processing concerns. The plan may enable operators to have billing and authentication services for the HTTP applications traffic as well. Besides, the signaling system should provide QoS and mobility management services for the traffic that routed directly to the Internet.

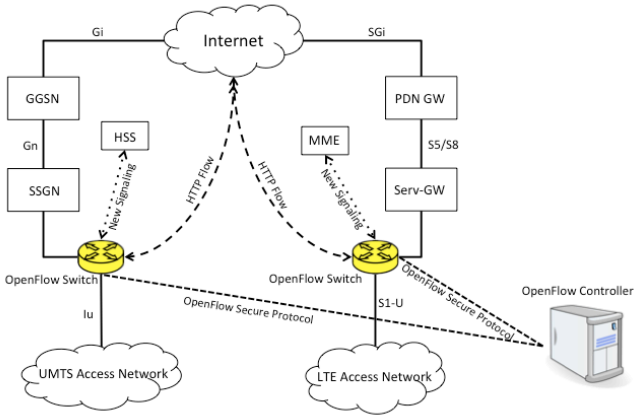

Figure 9. Future signalling system architecture

REFERENCES

[1] Olsson, Magnus; "*SAE and The Evolved Packet Core*". Academic Press. First Edition, 2009. ISBN: 978-0-12-374826-3

[2] Kitahara,T.;Riikonen,A.;Hammainen,H.;"*Characterizing traffic generated with laptop computers and mobile handsets in GPRS/UMTS core networks*". Local Computer Networks (LCN), 2010 IEEE 35th Conference on . 2010 , Page(s): 1046 - 1053

[3] Cisco Systems, Inc.; " *The Role of 3GPP's Evolved Packet Core in the Build-outof the Mobile Internet*". Cisco White Paper. February, 2009.

[4] N. McKeown, et. al., "*OpenFlow: Enabling Innovation in Campus Networks*", SIGCOMM CCR, Vol. 38, Issue 2, March 2008.

[5] Saurav Das, Guru Parulkar, Nick McKeown, "*Packet and Circuit Network Convergence with OpenFlow*". Optical Fibber Communication Conference 2010. San Diego. March 2010.

[6] "*OpenFlow Switch Specifications*", Version 1.1.0 (Implemented). Wire Protocol 0x02. 28 February 2011.

[7] "*Stoke Mobile Data Offload*", White Paper, Stoke Inc. November 2009.

[8] Cisco Systems, Inc; "*Overview of GSM, GPRS, and UMTS, Global Systems for Mobile Communications*". Cisco White Paper. 2002

[9] Flavio Muratore; "*UMTS Mobile Communications for the Future*". JOHN WILEY & SONS, LTD. 2001. ISBN: 0-471-49829-7

[10] S. Das, G. Parulkar, N. McKeown, "*Unifying Packet and Circuit Switched Networks*", IEEE Global Communications Conference (GLOBECOM '09) [C], Honolulu, Hawaii, 2009, pp. 1 – 6.

[11] 3GPP Technical Specification Group Services and System Aspects; Vocabulary for 3GPP Specifications; Release 10; 3GPP TR 21.905 V10.3.0. 2011-03.

[12] 3GPP Technical Specification Group Core Network and Terminals; General Packet Radio Systems (GPRS) Tunnelling Protocol; User Plane (GTPv1-U); Release 10; 3GPP TS 29.281 V10.2.0. 2011-06.