SMORE: Software-Defined Networking Mobile Offloading Architecture

Junguk Cho
junguk.cho@utah.edu

Binh Nguyen
binh@cs.utah.edu

Arijit Banerjee
arijit@cs.utah.edu

Robert Ricci
ricci@cs.utah.edu

Jacobus Van der Merwe
kobus@cs.utah.edu

Kirk Webb
kwebb@cs.utah.edu

School of Computing
University of Utah
50 S. Central Campus Drive
Salt Lake City, UT - 84112

ABSTRACT

We present our Software-defined network Mobile Offloading aRchitecturE (SMORE). SMORE realizes traffic offloading in mobile networks without requiring any changes to the functionality of existing mobile network nodes. At the same time, it is fully aware of mobile network functionality, including mobility.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]:
Wireless communication

Keyword

mobile network; service offloading; software defined network (SDN)

1. INTRODUCTION

Unprecedented growth in data traffic on mobile networks [15] is driven by the popularity of smartphones and tablets, advances in bandwidth available on mobile networks and the availability of a myriad of mobile applications and services. Despite these undeniable successes, the delay experienced across core mobile networks is still problematic for many applications that need near-realtime communication. Online gaming applications, for example, have stringent quality of service (QoS) demands and latency can negatively interfere with game play [7, 11]. A key architectural reason for delay remaining relatively high in the mobile core is the fact that the gateway nodes (Packet Data Network Gateway (PGW) nodes in LTE/EPC nomenclature), are deployed in a highly centralized fashion [3]. This results in inefficient hierarchical routing and high delay—packets travel for a significant distance in the mobile core network before even reaching the Internet [9].

Approaches to reduce end-to-end delay involve various offloading strategies [17, 10], including offloading to data centers inside the mobile provider core network footprint [5]. Despite the potential benefits of these approaches, they are inherently difficult to deploy in real networks—because of the fundamental complexity of mobile architectures, approaches that require significant changes to those architectures are very hard to realize in practice.

In this paper we propose an alternative approach to *realizing* offloading in a mobile network: our Software defined network Mobile Offloading aRchitecturE (SMORE). Like previous offloading approaches [5], we propose to deploy offloading data centers within the core of the mobile network and to selectively redirect traffic to these data centers. The defining characteristic of SMORE is that this offloading is accomplished *without* requiring modifications to the functionality of the core network proper. That is, without modifications to the functionality of network elements in the core mobile network. SMORE is realized by deploying an SDN infrastructure at aggregation points in the mobile core network. This SDN infrastructure allows two functions critical to the functioning of SMORE. First, it enables *monitoring* of the mobile network control plane through a lightweight monitoring component. The SMORE monitor continuously extracts information from the mobile network control plane and makes this information available to the SMORE controller. Second, based on this information and the service logic executed by the SMORE controller, the SDN infrastructure is used to transparently realize the offloading of selected traffic to offloading data centers. We argue that the ability to deploy these changes without modifying functional components of the mobile core is critical to making the deployment of new functionality such as offloading practical.

This paper makes the following contributions:

- We develop the SMORE architecture to realize offloading in an LTE/EPC mobile network. Specifically, we show how offloading for selected traffic of subscribed users can be realized without modifying the standard LTE/EPC interactions, even when devices are mobile.
- We present a prototype realization of our architecture using an LTE/EPC testbed and an Open vSwitch (OVS) based SDN. Specifically, our OVS enhancements allow transparent encapsulation and decapsulation of offloaded traffic.

2. BACKGROUND

In this section, we briefly describe the LTE/EPC mobile network architecture, typical deployment and user plane (data plane) protocol stack.

AllThingsCellular'14, August 22, 2014, Chicago, IL, USA.
Copyright 2014 ACM 978-1-4503-2990-3/14/08 ...$15.00.
http://dx.doi.org/10.1145/2627585.2627595.

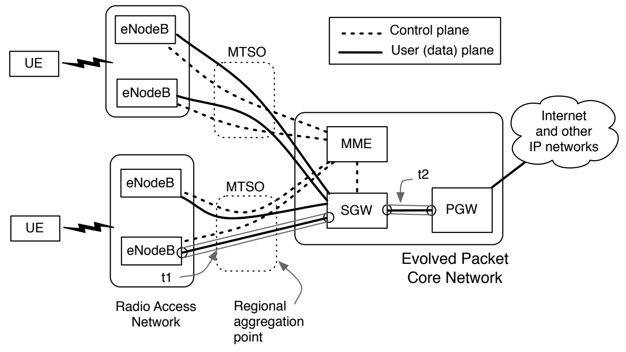


Figure 1: LTE/EPC architecture

Network architecture. As shown in Figure 1, the LTE/EPC architecture consists of two main components, the radio access network (RAN) and the evolved packet core (EPC) network. The RAN consists of eNodeBs (access points) which connect to User Equipment (UE), like cellphones, through a radio link and sends packets received from the UE to Serving Gateway (SGW) in the core network. In addition, it manages radio resources and supports intra-LTE mobility. The EPC consists of the Mobility Management Entity (MME), Serving Gateway (SGW), and Packet Data Network Gateway (PGW). The MME performs UE registration, authentication, and mobility management. It is also responsible for the tracking and paging of UEs in idle-mode. It does not participate in packet forwarding. The SGW and the PGW are responsible for routing/forwarding the data packets from all UEs to and from the external network. Besides packet forwarding, the SGW works as an anchor point in case of handover between eNodeBs and reserves buffers for paging functionality. The PGW performs IP address allocation for UEs, packet filtering and supporting Quality of Service (QoS) according to the users' data plan. When a UE attaches to the mobile network, control messages are exchanged between the eNodeB, MME and SGW. A successful attach procedure will result in tunnels being established between the eNodeB and the SGW and between the SGW and the PGW, $t1$ and $t2$ in Figure 1. These tunnels realize the user plane (data plane) which the UE uses to send and receive packets.

Typical deployment. Figure 1 also depicts typical deployment information that is relevant to our approach. EPC components (MME, SGW and PGW) are typically deployed in a small number of centralized locations (or central offices), e.g., on the order of ten in the US [21, 3]. This means that each such centralized location serves a large geographic area, with thousands of eNodeBs. LTE/EPC is a packet-based architecture, which means that there exists a packet "transport network", i.e., a regular IP network, in between the eNodeBs and the centralized EPC locations. For efficiency, connectivity from a set of eNodeBs gets aggregated at a regional aggregation point. As shown in Figure 1 these aggregation points are called (or co-located with) Mobile Telephone Switching Offices (MTSOs) and there are typically an order of magnitude more MTSOs than centralized EPC locations in a typical deployment. As such, MTSOs are an attractive deployment location for offloading data centers [5].

User plane protocol stack. The tunnels depicted in Figure 1 are realized with the GPRS Tunneling Protocol (GTP). The mobile network uses GTP for data transfer as well as for supporting per user mobility and quality of service. Each tunnel can be distinguished by a unique Tunnel EndPoint Identifier (TEID). When a packet is encapsulated, the TEID field in the GTP-U packet header is set to the value expected by the tunnel destination. As we explain later, obtaining the TEID and other information from the control plane and being able to use that to transparently encapsulate and decapsulate GTP data packets is crucial to the functioning of SMORE.

3. SMORE ARCHITECTURE

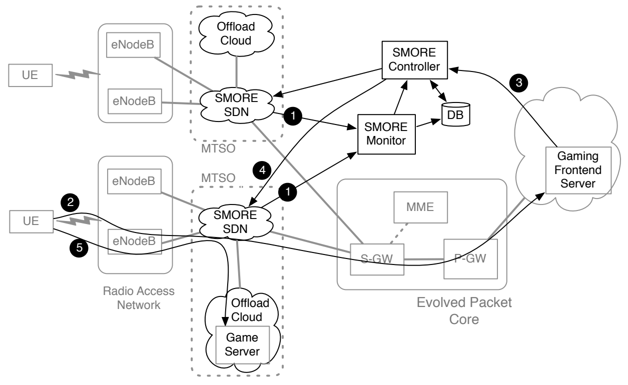


Figure 2: SMORE architecture

The SMORE architecture is depicted in Figure 2 in the context of an LTE/EPC mobile network. The service goal of SMORE is to reduce end-to-end delay for selected traffic by offloading such traffic to *offloading cloud* infrastructures close to mobile devices. Regional aggregation points in the form of MTSOs represent an ideal target for deploying these offloading infrastructures for two reasons. First, based on current centralized mobile network deployments, all traffic passes through MTSOs en-route to the Internet. Further, the MTSO locations in a typical deployment are at a sweet spot, both in terms of geographical coverage (to reduce delay) and the number of locations (filling the space between approximately ten central office locations and thousands of eNodeB locations). As shown in Figure 2, SMORE assumes that the offloading cloud platform and the SMORE *SDN substrate* are deployed at MTSO locations. The SDN substrate enables offloading to the offload cloud and also allows the SMORE *monitor* to snoop on the LTE/EPC control plane. The SMORE monitor continuously monitors the control plane, extracting relevant information. The monitor enters information into a database and also provides triggers to the SMORE *controller* about UE attach and mobility events. The SMORE controller interacts with the SMORE SDN substrate to effect offloading when needed and provides a front-end that Internet based service providers can interact with to register for offloading services. Note that in all cases service providers using the SMORE service will only be able to offload traffic associated with their own services.

On-demand use case. The interactions between these components are best explained with a typical use case, using the numbered steps from Figure 2. The SMORE SDN and monitor allow the control plane to be continuously monitored (#1 in figure). To simplify exposition, we assume an Internet-based service provider, e.g., a gaming provider, who wants to make use of the SMORE service. Note that this generalizes to any Internet service provider that wants to make use of the SMORE offloading service. End-users of the game service access the Internet-based gaming frontend server via the normal data path after attaching to the mobile network (#2). Realizing that it wants to offload the user in question to an offload instance, logic in the gaming frontend server requests offloading via the interface provided by the SMORE controller (#3). The SMORE

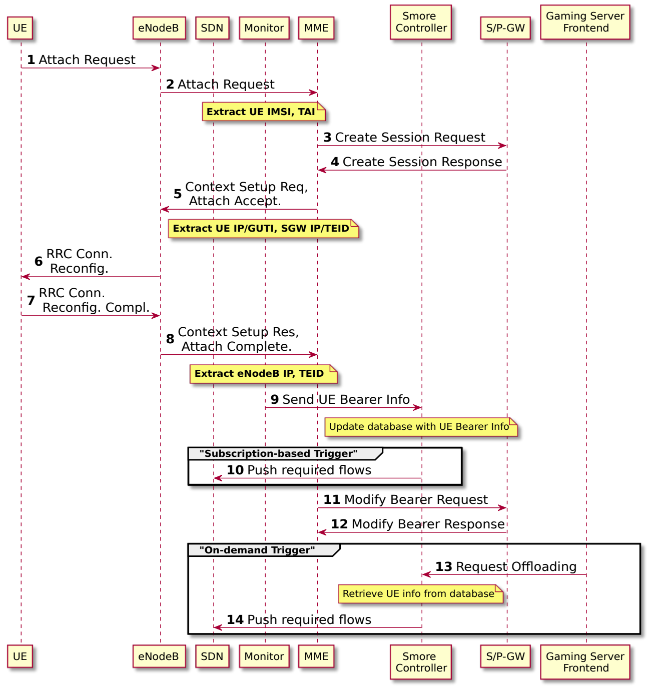


Figure 3: Attach call flow

controller consults the SMORE database to determine the current location of the UE in question, and proceeds to configure the SDN substrate in the appropriate MTSO location to set up offloading of the desired subset of traffic (#4). When the UE connects to the game server, this traffic will be transparently redirected to the offload server in the MTSO cloud (#5).

Subscription use case. Service providers (or end users) can also opt to make use of a subscription model for offloading. Once subscribed, information about the subscriber UE is maintained in the SMORE database. In this case, every time a subscriber UE connects to the network, the offloading path is constructed without the need for step #3 in the previous scenario. When the UE attaches, the SMORE monitor detects this event and signals the SMORE controller, which immediately installs the appropriate rules in the SMORE SDN.

3.1 Interaction with standard LTE/EPC

We now consider in more detail how SMORE can realize its functionality without requiring any modifications to LTE/EPC network elements. We specifically consider how the SMORE monitor extracts information from the LTE/EPC control plane during UE attach and handoff procedures. The information obtained from these procedures triggers the installation of rules in the SMORE SDN to realize offloading in a transparent manner.

Attach. Figure 3 shows how the SMORE monitor snoops on the control plane interactions between the eNodeB and the MME during UE attach and subsequently helps trigger the pushing of required offloading policies in the SMORE SDN via the SMORE controller. Specifically, during the UE attach procedure, the SMORE monitor extracts the UE's IMSI (International Mobile Subscriber Identity) and TAI (Tracking Area Identifier) from the 'attach request message' sent to the MME via the eNodeB (#2). When the MME responds with the 'attach accept' message (#5), the monitor extracts the UE's IP address, SGW's IP address, SGW's TEID, and UE's GUTI (Globally Unique Temporary ID). Finally, the monitor obtains the eNodeB's IP address and eNodeB's TEID when the eNodeB responds with the 'context setup response' message (#8). At this point, the SMORE monitor has all the information required to trigger offloading and sends it to the SMORE controller (#9) which, in turn, updates its database. In case of a subscription-based use case, at this point the SMORE controller pushes the required flows to the SMORE SDN (#10), which enables offloading of game server specific traffic to the game server hosted in the in-network cloud platform. For the on-demand use case, when the SMORE controller receives the offloading trigger from the gaming frontend server (#13), it retrieves the UEs bearer information from the database and pushes the required flows to the SMORE SDN (#14) to enable offloading.

Handover. Figure 4 shows the steps required to enable offloading in presence of user mobility. Specifically we consider X2-based handover between eNodeBs without MME and SGW relocation. (This is the common handover case in LTE/EPC.) As seen in the figure, when the source eNodeB decides to handover the UE (based on measurement reports) to the target eNodeB, it sends a handover request (#2) to the target which, in turn, does the necessary resource allocation and sends back a handover acknowledgement message (#3) to the source. The target also sends a path switch request message (#4) to the MME that contains the target eNodeB information (IP address and TEID), which will subsequently be used by the SGW to forward downlink packets. The SMORE monitor extracts this bearer information, and sends it to the SMORE controller (#5), which proactively constructs a new rule to be pushed to the SMORE SDN to enable processing of downlink packets corresponding to the target eNodeB. Note that all the other UE bearer information remains unchanged (including the SGW identifiers) and the SMORE controller obtains the other required information from its database. When the SGW updates the user bearer to forward downlink packets to the target eNodeB instead of the source eNodeB, the MME sends back a path switch acknowledgement message (#10) to the target indicating the success of the path switch request. When the

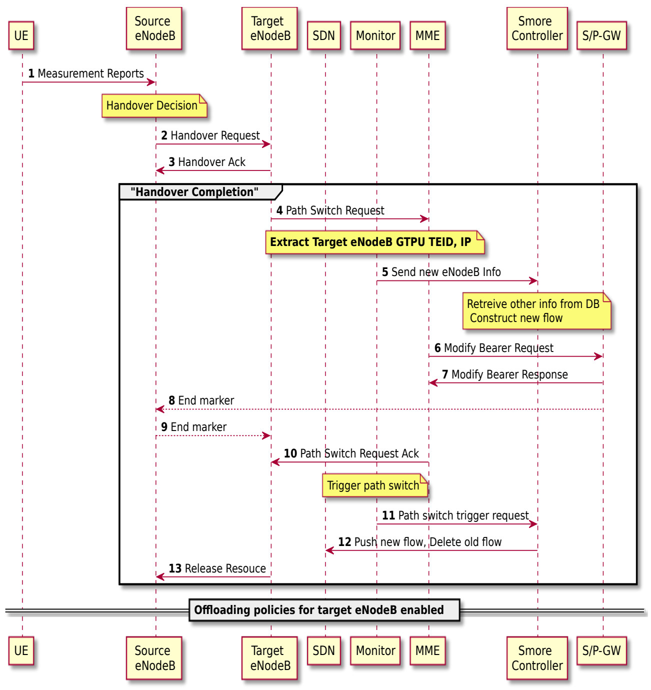


Figure 4: Handover call flow

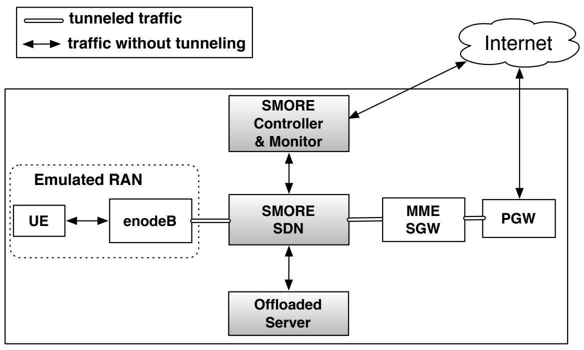


Figure 5: Testbed setup

SMORE monitor sees this message, it sends a path switch trigger request (#11) to the SMORE controller, which in turn, pushes the previously constructed rule to the SMORE SDN and deletes the previous rule (#12). Subsequently, the SDN substrate routes (after proper encapsulation) all downlink packets from the cloud based game server to the target eNodeB.

4. IMPLEMENTATION

We developed a prototype implementation of the SMORE architecture and deployed it in a local LTE/EPC testbed based on the OpenEPC LTE/EPC architecture implementation [8]. As shown in Figure 5 the RAN part of our current setup is based on an emulated implementation from OpenEPC. Each component shown in the figure is realized as a physical machine instance in the Emulab [20] facility. Our current implementation does not support the handover mechanism described in the design because our LTE/EPC testbed does not currently have handover support.

Below we consider the prototype implementations of each SMORE component in more detail.

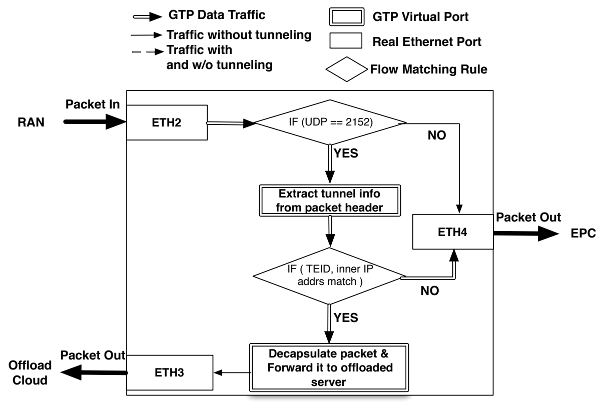
SDN. To implement SMORE SDN, we used Open vSwitch (OVS) 2.0 which supports the OpenFlow 1.3 standard. Tunneling in OVS is implemented using a virtual port (vport) abstraction which simplifies header manipulation [14]. As shown in Figure 6, we added three vports to realize offloading, GTP decapsulation, and GTP encapsulation. By default the SMORE SDN component works as a standard layer 2 switch. It forwards packets from the eNodeB to the SGW/MME and vice versa. However, when it receives a traffic offload request from the SMORE controller, flow rules are setup to divert UE connections to alternate destinations.

Packet processing in the uplink direction is shown in Figure 6(a). The SMORE SDN first checks whether the incoming packet is GTP data traffic (whether it has 2152 as UDP port number). If so, the offloading vport extracts the SGW TEID in the GTP header and the source and destination IP addresses from the inner IP layer (i.e., the UE source and destination IP). If this information matches a SMORE OVS table entry, then the packet is destined for an offloading server and is sent to the GTP decapsulation vport. Otherwise, the packet is sent along the default path to the SGW. When the GTP decapsulation vport receives a packet, it removes the GTP header, replaces the destination MAC address, and forwards the packet. The replacement MAC address is that of the next hop along the route to the offload destination.

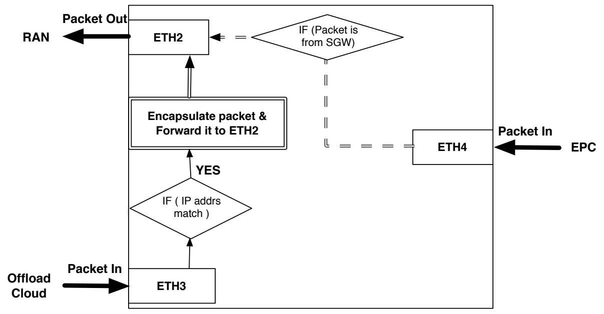
Downlink packet processing is shown in Figure 6(b). Packets arriving from the SGW/MME are forwarded toward the eNodeB as per normal. However, if the packet is from an offloading server, the SMORE SDN uses the IP addresses (src: offloaded server IP, dst : UE IP) to find the correct encapsulation information for this exchange. The packet is forwarded to the GTP encapsulation vport along with the associated tunnel information (outer IP addresses (src: SGW IP, dst: eNodeB IP) and GTP (eNodeB TEID) identifiers). It is also necessary to replace the destination MAC address with the next hop's MAC address (in the direction of the UE). With encapsulation complete, the GTP packet is forwarded on toward the eNodeB.

Monitor. We implemented the SMORE monitor using tshark, a terminal-based version of wireshark. The monitor listens to the interface between the eNodeB and the MME and processes all messages that are exchanged to detect bearer set-up and tear-down events. Specifically the monitor implements the information extraction and triggers as described in the *attach* description in Section 3.1.

Controller. We implemented a prototype of SMORE controller using the Ryu OpenFlow controller. The SMORE controller has a simple database storing registered offloading server IP addresses and IMSIs of UEs that subscribed to offload services. We extended the Ryu API to support GTP flow modification control messages. The SMORE controller modifies the flow table of the OVS using the extended API of Ryu controller. The SMORE controller compares the bearer setup information provided by the SMORE monitor with its database of registered UEs and offloading service addresses to decide whether to push offloading flows for a specific UE to the OVS flow table.


GTP Data Traffic
Traffic without tunneling
Traffic with and w/o tunneling
GTP Virtual Port
Real Ethernet Port
Flow Matching Rule
Packet In
RAN
ETH2
IF (UDP == 2152)
NO
YES
Extract tunnel info from packet header
Packet Out
ETH4
EPC
IF (TEID, inner IP addrs match)
NO
YES
Decapsulate packet & Forward it to offloaded server
Packet Out
Offload Cloud
ETH3

(a) Handling uplink packet

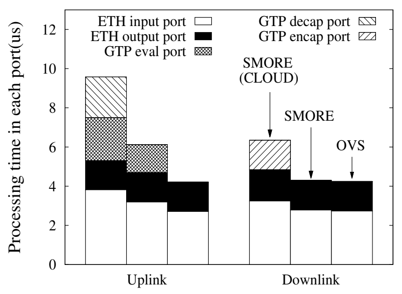


(b) Handling downlink packet

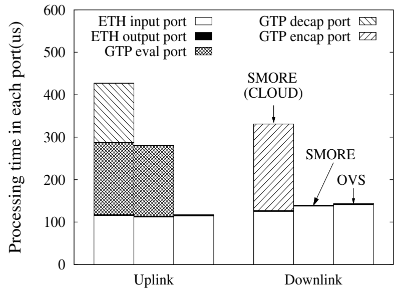
Figure 6: Handling uplink and downlink packet in OVS

5. EVALUATION

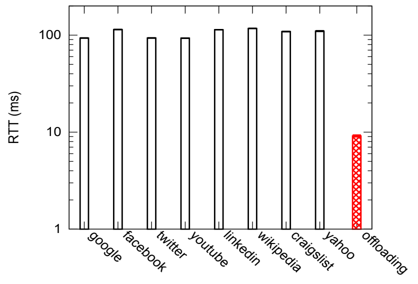
We measured the SMORE SDN processing time, the responsiveness and latency of the SMORE monitor and controller, and illustrated the potential for latency reductions with our approach.



(a) Breaking down processing time of each port in datapath



(b) Breaking down processing time of each port in OpenFlow table



(c) RTT of different Internet services and the offloading server

Figure 7: Evaluation results

Our evaluation was performed using machines with a single 3 GHz CPU and 2 GB RAM.

5.1 SMORE SDN

OVS has two tables, namely, the OpenFlow flow table in userspace and a simple flow table called datapath in the kernel. The data path table is essentially a cache for active flows. The userspace table is consulted when no match is found in the datapath, typically for the first packet of a flow. We evaluated both the datapath and Openflow table processing of our SDN implementation. We measured the processing time for each port and vport in Figure 6 using the *getnstimeofday()* kernel function.

Datapath processing. A breakdown of the processing time when a flow entry exists in the data path table is shown in Figure 7(a). SMORE (Cloud) means the path between the RAN and Offload cloud. SMORE is the path between the RAN and the EPC when SMORE rules are installed. OVS is the same path when no SMORE rules are installed. Compared to (native) OVS, SMORE (Cloud) and SMORE increased uplink processing by 5.366us and 1.911us and downlink processing by 2.105us and 0.058us respectively. SMORE shows slightly longer processing time in ETH input than OVS because more complex matching rules are used and there are more flow entries in the datapath.

Openflow table processing. Figure 7(b) shows a similar breakdown for the case where a flow entry does not exist in datapath but in only in the OpenFlow flow table. As expected, the processing times are much longer because of the userspace processing involved, however, this is only incurred on the first packet of a flow.

In both cases the overhead is modest compared to the end-to-end delay in LTE.

5.2 SMORE monitor and controller

SMORE controller micro-benchmark. We measured the time the SMORE controller takes to install offloading flows into the OVS when it receives a trigger. (I.e., a trigger for either on-demand or subscription use cases.) The average processing time is 4.677ms.

SMORE monitor micro-benchmark. We generated a series of synthetic UE attaching events to measure the time taken to detect an UE attach event. The average detection time is 2.973ms (not including packet reading time incurred by tshark). Our current packet capture realization based on tshark is quite inefficient, contributing an additional 850ms.

Subscription use case evaluation. We evaluated the responsiveness of SMORE for the subscription use case described in Section 3. Since an attached UE might immediately start to interact with the offloading server, this case demands fast response from the SMORE monitor and controller.

We measured the elapsed time (t_{res}) between the moment when the 'attach complete' message (#8 in Figure 3) arrives at the OVS and when the offloading flows are successfully installed (in the OVS) by the SMORE monitor and SMORE controller. We measured t_{res} for 10 different UE attach events and found the mean value for t_{res} to be 1.056s with a standard deviation of 18ms. This is an acceptable delay as it is of the same order of magnitude as application startup delay on a mobile device.

5.3 RTT improvement

We measured the RTT between a UE and the top websites in the U.S [2]. We derived a synthetic delay from previous measurement works as the delay of the cellular core network [6, 12, 18]. Specifically, the core network delay is calculated as: *core network's delay = UE-server's delay - PGW-server's delay - radio's delay*. The one-way delay from UE to server was set to 35ms as reported in [12]. The PGW-server delay was set to 8ms as a typical "regional" Internet delay reported in [18]. The delay on the radio link was 4ms one-way [6]. In total, the synthetic delay of the core network (from OVS node to PGW) was set as 23ms. The delay between the OVS node and the offloading server was set to a typical "local" link and it was almost negligible, i.e, less than 1ms.

We ping each server 50 times and reported the average RTT in figure 7(c). As expected and shown in the figure, the delay when using the offloading server is significantly smaller than using the servers in the Internet.

6. RELATED WORK

The issues associated with current hierarchical routing approach in the cellular networks have been identified in works like [9, 22]. To alleviate the burden of the core network and enable efficient routing, the 3GPP standards body has proposed Local IP Access (LIPA) and Selected IP Traffic Offload (SIPTO) - for offloading mobile traffic to a local network using femtocells and, to the Internet via regional gateways closer to the user locations, respectively [1]. However, these methods require significant changes in the current standard and expensive additional infrastructure for their realization. Our aim, in this work, is to enable efficient offloading without requiring any changes to the existing standard. Other works have proposed opportunistic offloading of delay tolerant cellular traffic using alternative access technologies like WiFi [4, 10]. Our work is focused on efficient offloading strategies in cellular networks specially for traffic with strict delay requirements (e.g., real time gaming).

We draw inspiration from research works exploring the possibility of leveraging SDN to enable more flexibility in the core mobile network [13, 16, 14] and using cloud resources in user proximity to enable low latency applications [19]. Our work is grounded in the reality that it is not economically feasible for the mobile network operators to partake in a major overhaul of the existing infrastructure built on standards compliant closed-source vendor specific equipment. Our proposed architecture can be deployed in strategic locations to co-exist with the current infrastructure and, enables much needed flexibility in the mobile network using emerging technologies like SDN and cloud, without requiring any changes in the existing architecture.

Our work is strongly inspired by MOCA [5]. MOCA proposed an SDN-based architecture for cellular networks to offload selected traffic to a cloud-based local SGW, PGW and an offloading server inside the core network. In MOCA architecture, although modifications in the current network were intentionally minimized, the MME still played a central role and needed to be modified. With SMORE we designed and prototyped similar functionality without requiring modification to *any* LTE/EPC network element.

7. CONCLUSION

We leveraged SDN to introduce an offloading architecture for cellular networks. The architecture allows traffic to be intercepted and rerouted to offloading servers located inside the cellular core. The SMORE approach allows us to realize offloading with no modifications to the current network architecture.

Acknowledgements: This research was supported in part by NSF award CNS-1305384.

8. REFERENCES

[1] Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO). `http://www.3gpp.org/ftp/Specs/html-info/23829.htm`.

[2] ALEXA(2014). Alexa: Top sites in united states. `"http://www.alexa.com/topsites/countries/US"`.

[3] BALAKRISHNAN, M., MOHOMED, I., AND RAMASUBRAMANIAN, V. Where's that phone?: geolocating IP addresses on 3G networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference* (2009), ACM, pp. 294–300.

[4] BALASUBRAMANIAN, A., MAHAJAN, R., AND VENKATARAMANI, A. Augmenting mobile 3G using WiFi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services* (2010), ACM, pp. 209–222.

[5] BANERJEE, A., CHEN, X., ERMAN, J., GOPALAKRISHNAN, V., LEE, S., AND VAN DER MERWE, J. MOCA: a lightweight mobile cloud offloading architecture. In *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture* (2013), ACM, pp. 11–16.

[6] BLAJIĆ, T., NOGULIĆ, D., AND DRUŽIJANIĆ, M. Latency Improvements in 3G Long Term Evolution. *Mipro CTI, svibanj* (2006).

[7] CHEN, K.-T., HUANG, P., AND LEI, C.-L. Effect of Network Quality on Player Departure Behavior in Online Games. *Parallel and Distributed Systems, IEEE Transactions on 20*, 5 (2009), 593–606.

[8] CORICI, M., MAGEDANZ, T., VINGARZAN, D., AND WEIK, P. Prototyping mobile broadband applications with the open evolved packet core. In *Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference on* (Oct 2010).

[9] DONG, W., GE, Z., AND LEE, S. 3G meets the internet: understanding the performance of hierarchical routing in 3G networks. In *Proceedings of the 23rd International Teletraffic Congress* (2011), ITCP, pp. 15–22.

[10] HAN, B., HUI, P., KUMAR, V. A., MARATHE, M. V., SHAO, J., AND SRINIVASAN, A. Mobile data offloading through opportunistic communications and social participation. *Mobile Computing, IEEE Transactions on 11*, 5 (2012), 821–834.

[11] HENDERSON, T. *The effects of relative delay in networked games*. PhD thesis, University of London, 2003.

[12] HUANG, J., QIAN, F., GUO, Y., ZHOU, Y., XU, Q., MAO, Z. M., SEN, S., AND SPATSCHECK, O. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *Proceedings of ACM SIGCOMM* (2013).

[13] JIN, X., LI, L. E., VANBEVER, L., AND REXFORD, J. Softcell: Scalable and flexible cellular core network architecture. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies* (2013), ACM, pp. 163–174.

[14] KEMPF, J., JOHANSSON, B., PETTERSSON, S., LUNING, H., AND NILSSON, T. Moving the mobile evolved packet core to the cloud. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on* (2012), IEEE, pp. 784–791.

[15] KEVIN FITCHARD. Data now 85% of mobile traffic but 39% of revenue: What gives? `http://gigaom.com/2012/03/20/data-now-85-of-mobile-traffic-but-39-of-revenue-what-gives/`, March 2012.

[16] LI, L. E., MAO, Z. M., AND REXFORD, J. Toward software-defined cellular networks. In *Software Defined Networking (EWSDN), 2012 European Workshop on* (2012), IEEE, pp. 7–12.

[17] MA, L., AND LI, W. Traffic Offload Mechanism in EPC Based on Bearer Type. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on* (2011), pp. 1–4.

[18] NYGREN, E., SITARAMAN, R. K., AND SUN, J. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review 44*, 3 (2010), 2–19.

[19] SATYANARAYANAN, M., BAHL, P., CACERES, R., AND DAVIES, N. The Case for VM-based Cloudlets in Mobile Computing. In *IEEE Pervasive Computing* (November 2009).

[20] WHITE, B., LEPREAU, J., STOLLER, L., RICCI, R., GURUPRASAD, S., NEWBOLD, M., HIBLER, M., BARB, C., AND JOGLEKAR, A. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation* (Boston, MA, Dec. 2002), USENIX Association, pp. 255–270.

[21] XU, Q., HUANG, J., WANG, Z., QIAN, F., GERBER, A., AND MAO, Z. M. Cellular data network infrastructure characterization and implication on mobile content placement. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems* (2011), ACM, pp. 317–328.

[22] XU, Q., HUANG, J., WANG, Z., QIAN, F., GERBER, A., AND MAO, Z. M. Cellular data network infrastructure characterization and implication on mobile content placement. In *SIGMETRICS* (2011).