P2P Offloading in Mobile Networks using SDN

Ryan Saunders
ryan.saunders@utah.edu

Junguk Cho
junguk.cho@utah.edu

Arijit Banerjee
arijit@cs.utah.edu

Frederico Rocha
fred@cs.utah.edu

Jacobus Van der Merwe
kobus@cs.utah.edu

School of Computing,
University of Utah

ABSTRACT

The peer-to-peer (P2P) architecture and the mobile network architecture have conflicting designs. P2P can take advantage of peers being in close proximity to decrease latency. However, the mobile network is hierarchical as routing is directed through centralized gateways. Because of network state needed to establish connections in the mobile network, user equipment's traffic needs to travel up the hierarchy before being redirected back down to a nearby peering device. The inherent delay and the additional network state strip P2P applications of their primary advantages over client-server applications. We have developed an SDN architecture to offload and redirect peering traffic before reaching the core of the mobile network. Our implementation allows for any P2P communication independent of the mobile provider and the peering application. We have evaluated our design and demonstrate that there is a decrease in latency by approximately a factor of two with our method compared to a standard P2P procedure between smartphones in the same mobile network.

CCS Concepts

•**Networks** → *Network architectures; Mobile networks;* Network design principles;

Keywords

P2P; Mobile network; SDN; Openflow

1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SOSR '16, March 14-15, 2016, Santa Clara, CA, USA
© 2016 ACM. ISBN 978-1-4503-4211-7/16/03…$15.00
DOI: http://dx.doi.org/10.1145/2890955.2890963

P2P applications have become widely adopted. Video and voice communication, gaming, file sharing, and multimedia sharing commonly use P2P technology to provide services between peers [28, 27]. The primary benefit of P2P communication is that a connection between peers in close proximity would perform with lower latency than if a centralized server was required to relay such communication. While it is difficult to provide exact statistics about the percentage of P2P traffic due to the difficulty to identify P2P traffic [15, 14], these applications contribute about 50 percent of consumer traffic and are forecasted to only increase [4].

However, the current mobile network does not differentiate P2P and regular internet traffic, such that it does not utilize the P2P architecture to provide low latency connections to peers in close proximity. Hence, the current P2P applications, e.g., Skype, use the mobile network as a communication bit-pipe in an over-the-top (OTT) manner. Traffic from one user equipment (UE) to another UE in close proximity follows a sub-optimal route that leads through a large amount of mobile network infrastructure. This is due to the hierarchical routing strategy employed by the mobile network, in which UE traffic is directed through network core gateways deployed at a few centralized locations. This sub-optimal route results in high latency, even if two mobile devices are only a few miles apart from each other. Although latency is generally an issue on the mobile networks [19] and latency highly depends on the provider [10], we contend that focusing on improving the P2P experience is fundamental since many P2P applications expect low latency for quality user experience. For example, applications like Skype use P2P for voice and video messaging [27]. Skype is an ideal mobile application since it takes full advantage of mobile device capabilities such as voice and video capabilities, but its performance suffers from high latency on the mobile network. Improving the P2P performance on the mobile network will not only improve the existing P2P applications, but will also lay the foundation to enable new P2P applications.

Even so, modifying the current mobile networks to

support P2P applications would be an extremely difficult and costly affair. Mobile networks employ complex standardized protocols to enable communication and maintain significant per device network state across various core network entities, including routing tunnel information, to provide connectivity even in presence of device mobility.

In contrast, a benefit of P2P is that it allows communication between peers without the need of infrastructure support. By utilizing the capabilities of Software Defined Networking (SDN), we can enable P2P in a mobile network with a modest amount of network state, without requiring any change in the current network operations. Towards this end, we propose our architecture: a Software-defined Architecture for P2P Offloading in Mobile networks (SAPOM) which can be deployed at regional aggregation points close to the peering UE locations. With this method of offloading, the traffic between two mobile peers in close proximity follows a more direct route via the regional aggregation point, without traversing all the way to the centralized gateways in the core mobile network. This technique significantly reduces the latency of such a P2P connection since regional aggregation points are usually close to the UEs that route to them. Furthermore, our method also reduces the load on the centralized core mobile network entities by offloading the heavy traffic usually generated by P2P applications, thus, improving the overall network performance. We make the following contributions:

1. We designed and implemented SAPOM, an architecture that successfully identifies and offloads P2P traffic before reaching the core mobile network.
2. We demonstrate SAPOM works during radio active mode, and after a UE is paged during idle mode.
3. We evaluate SAPOM in an OpenEPC-based [6] LTE/EPC test bed with real UEs and a real eNodeB while running Skype, a popular P2P application. Our results show that SAPOM significantly improves the latency between peering devices, and also reduces the traffic load on the core mobile network.

2. BACKGROUND

2.1 Mobile Network Architecture

Figure 1 shows the general architecture of the mobile network. The Evolved Packet Core (EPC) of the mobile network connects UEs, such as cellphones, to the Internet. This is achieved by UEs wirelessly connecting to an access point (eNodeB) on the radio access network (RAN). Clusters of eNodeBs traditionally aggregate at regional aggregation points, located at a Mobile Telephone Switching Office (MTSO). Within a mobile network these regional aggregation points continue the

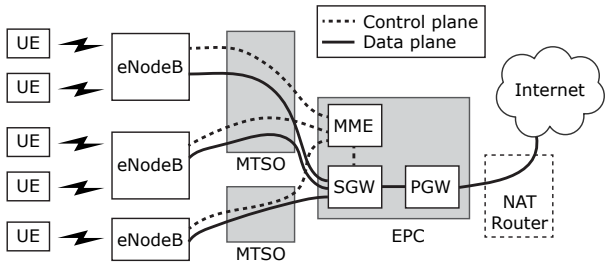


Figure 1: Mobile Network Architecture including UE, eNodeB, MTSO, and EPC.

connection to the Serving Gateway (SGW) in the EPC. The EPC connects the UEs to the internet and other IP networks. As this hierarchical structure suggests, for each centralized EPC there are significantly more MTSO locations, and for each MTSO there are significantly more eNodeBs [12]. Conclusively, the mobile network makes the distance for data-packets to travel to reach the IP network a much longer traversal than that of a typical home network.

Controlling the designation of SGW to UE is done using control-packets between the eNodeB and Mobility Management Entity (MME). The MME identifies and authenticates UEs as they attach to the mobile network. When a UE becomes inactive, it detaches from the radio bearer and enters the idle state [24]. While in this mode, the UE does not send or receive traffic from the IP network because it has no established bearer.

When an incoming packet is destined for an idle UE, the MME starts the paging procedure to activate the UE and reestablish the UE's data connection to the EPC. Once active, a UE will be under a different radio bearer and possibly a different eNodeB.

2.2 NAT in Mobile Networks

Many mobile service providers deploy middleboxes like Network Address Translation (NAT) enabled routers between the IP network and EPC to reduce the number of public IP addresses allocated and to prevent unsolicited external connections with their clients' devices [26]. Due to these constraints, two devices with allocated private IP address behind the NAT router cannot easily establish P2P connections. As a workaround, mobile peer-to-peer applications often use NAT traversal or hole-punching methods to establish communication between peers behind a NAT [13] enabled router.

For example Skype, one of the most popular P2P applications [3], uses NAT Traversal techniques [9, 29] to support P2P connections behind the NAT router.

2.3 Motivation

In the current mobile networks, a peer-to-peer connection, even between two geographically close UEs, extend through the EPC to the NAT router deployed at a centralized location. Such sub-optimal routing fails to leverage the benefit of P2P and degrades the

quality of experience of latency-sensitive P2P applications like video streaming, audio streaming, and gaming, even when using high bandwidth 4G LTE connections [10, 11]. Because of poor latency, many developers may choose not to make use of P2P on mobile applications [22], even given the benefits that mobile devices offer such as portability and accessibility.

Additionally, P2P applications, like multiplayer games, file sharing applications, and Voice over IP (VoIP) typically generate a heavy amount of traffic [5]. P2P is ideal for such applications because a major portion of the traffic can be diverted from a centralized server, thus significantly decreasing the server load. Although NAT traversal on a mobile network allows to alleviate the load on the P2P application server, the core mobile network entities, i.e., the centralized gateways, still suffer from congestion due to traffic heavy P2P applications. Network congestion can also cause bufferbloat, increasing the latency of the connection [11]. By offloading mobile P2P traffic before reaching the EPC, mobile network providers can significantly reduce the load on the core network.

We argue that there is a need for a mobile network architecture that efficiently accommodates P2P applications to reduce latency as well as core network load. In order to seamlessly integrate with the current mobile networks and still improve quality of service for P2P applications, such an architecture must meet the following criteria:

1. The architecture must seamlessly work with current mobile architecture, NAT router domains, and mobile applications,
2. The architecture should not affect an application,s performance negatively, and connections other than P2P should not be impacted,
3. Realizing P2P connectivity should be possible, even if a UE has entered an idle state.

SAPOM utilizes SDN priciples to achieve the above goals in the current mobile network.

3. ARCHITECTURE

3.1 SAPOM workflow

The basic concept of SAPOM is to insert an SDN switching fabric and controller into the mobile network to offload P2P traffic without modifying the underlying network architecture. This is achieved by inserting an SDN switching fabric at the MTSO, the aggregation point through which eNodeBs are connected to the EPC. A depiction of our architecture is shown in Figure 2. By deploying our architecture at the MTSO, we ensure less propagation delay for P2P connections for any peering UEs under the same MTSO. On the other hand, if offloading occurred at the eNodeB, two UEs only a few miles apart will not be able to realize the

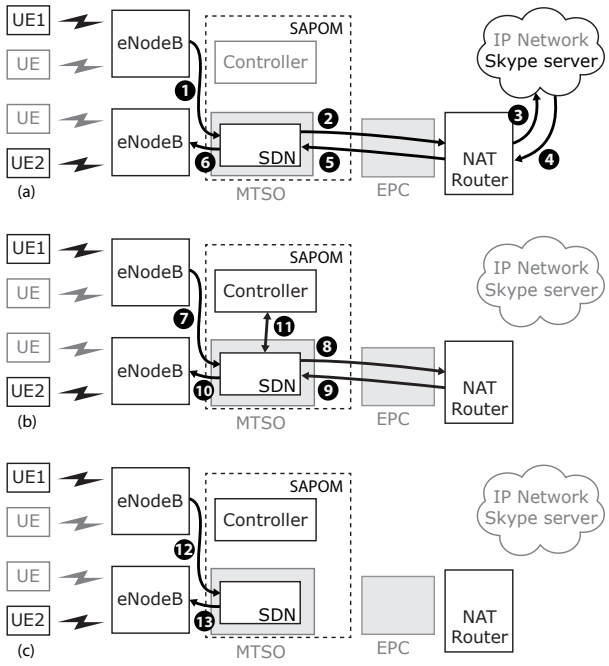


Figure 2: SAPOM positioned at an MTSO in the mobile network. Figure shows the P2P connection establishment procedure with SAPOM offloading enabled.

benefit of our system. Similarly, offloading at the centralized gateways, latency improvements wouldn't be nearly as significant.

A P2P connection begins by one peer requesting a connection with another peer. Under the NAT router, an outside server usually helps perform NAT traversal for two peers under the same NAT domain. In Figure 2(a), UE1 begins a Skype-based P2P connection with UE2 by contacting a Skype server dedicated for helping with NAT traversal (1-3 in Figure 2(a)). The Skype server begins the NAT traversal process and the private IP address of the connecting UEs are shared (4-6). During this phase, SAPOM forwards the traffic via the default route (1,5). Once NAT traversal is completed, each UE is aware of the other's assigned IP address under the NAT domain, and a P2P connection is established between the two UEs (7-10 in Figure 2(b)). The connection currently traverses the EPC until the packets exit the GPRS Tunneling Protocol (GTP) tunnel and are decapsulated (8). The packet is redirected to the MTSO after being encapsulated in a GTP packet (9). Preconfigured flow rules added to the SAPOM SDN switching fabric will forward a copy of these initial packets to the controller for creation of flow rules for this particular connection (11). We present the details about the handling of these flow rules in Section 3.2. Once the flow rules for this connection are added to the SDN switching fabric, SAPOM will intercept the P2P traffic (12 in Figure 2(c)) preventing these packets from continuing to the EPC, and instead redirect the

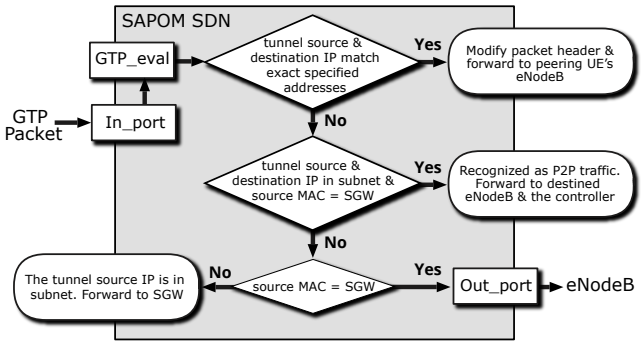


Figure 3: Handling P2P packets in SAPOM SDN

packets to the eNodeB associated with the destination UE (13).

3.2 Flow rules at SDN switch and controller

Relevant information about UE-EPC interaction, such as the UE's private IP address and information regarding the eNodeB associated with the UE, can be obtained from evaluating the GTP packet headers, the tunneling protocol used to route packets in the EPC. When a P2P connection is requested, this information is provided to the controller to create OpenFlow [18] match rules for the SDN switch. These OpenFlow rules identify traffic that needs to be offloaded, or redirected, while all other traffic are forwarded as expected. This is similar to the standard SDN model in that the controller logic is triggered by data plane activity. However, our SDN approach differs from the standard model in that the logic only optimizes P2P interaction. All other forms of traffic flows are not impacted by our architecture.

A flowchart of our P2P offloading rules is shown in Figure 3. A SAPOM SDN switch first checks whether or not an incoming packet (from either an eNodeB or the SGW interface) is a GTP data packet. All GTP packets are forwarded to a GTP evaluation port. At the evaluation port, an incoming packet is first checked against all flow rules corresponding to existing P2P connections. In case a match is found, the packet is forwarded to the corresponding peering UE's eNodeB with necessary header modifications. Otherwise, the packet is compared to the next rule to determine if it belongs to a new P2P connection. If the inner source and destination IP addresses of the packet matches the subnet of the NAT domain, and the source MAC address matches to that of the SGW, the packet corresponds to a new P2P connection, and is forwarded to the eNodeB interface and also to the controller. If the packet matches neither of the previous rules, it is forwarded to either the destined eNodeB or to the SGW based on the source MAC address.

The controller is aware of the network's private IP subnet domain under the NAT router, and sets up the initial flow rules that require this subnet mask. If a packet is received at the controller, a new P2P connection was requested between two UEs under the subnet. From this packet, the controller extracts the eNodeB IP address, MAC address, and GTP tunnel identifier (TEID) associated with the destination UE; the IP address and MAC address of the SGW; and the IP addresses of the source and destination UEs. With this information, a flow rule is constructed and sent to the switch to modify and redirect subsequent packets from this new P2P connection.

3.3 P2P with Idle Mode

SAPOM supports P2P offloading even if a user device has entered idle mode. The offloading mechanism of SAPOM is decoupled from the underlying mobile network control protocols and states. When one of the peering UEs requests a connection with an idle peer, the mobile network performs a paging procedure [24] to activate the idle UE and reestablish the associated radio bearers. After paging has established the radio bearer for the UE, it is able to receive the P2P connection request and begin the connection. At this point, the procedure to offload the P2P connection are identical to those found in Section 3.1. This is a significant advantage of SAPOM architecture since it seamlessly integrates with the existing architectures and protocols, without requiring a monitor to continuously sniff mobile network specific events and transactions.

4. IMPLEMENTATION

For our prototype implementation, we used Open vSwitch (OVS) [20] and Ryu [2] for SAPOM SDN and controller respectively. To support the functionality described in Section 3, we extended OVS and Ryu to handle GTP packets, since the current OpenFlow standard does not support the GTP protocol.

SAPOM Controller. The SAPOM controller was implemented using the Ryu OpenFlow controller. However, Ryu is not equipped with a default GTP data packet parsing library or GTP flow modification library. To extract GTP header information and inner packet information from GTP data packets, we modified the UDP Ryu library and added GTP data packet parsing functionality using *struct* [23]. We also extended the Ryu API to support flow modification for GTP packets. These modifications enable the SAPOM controller to insert flow rules that match GTP TEIDs, prefix inner IP addresses, and exact inner IP addresses, and to rewrite GTP TEIDs based on matching rules. These modifications are instrumental to offload packets to the destination eNodeB in a P2P connection.

When SAPOM is enabled, the controller inserts the initial flow rules as specified in Section 3.2. Priority are set on the flow rules to ensure correct logic flow. Additionally, since the size of flow tables is limited, a timeout is set on the flow rules that match specific P2P connections.

SAPOM SDN. SAPOM SDN is configured through the initial flow rules from the controller to offload P2P traffic. Other forms of traffic are forwarded as expected on a standard L2 switch. To handle GTP data packets in SAPOM SDN, we use a virtual port (vport) abstraction, which simplifies header manipulation [16] in the OVS. SAPOM SDN has two designated virtual ports. The first virtual port acts as the GTP evaluation port described in Section 3.2. It is used to extract the inner IP addresses and GTP TEID from GTP data packets. The second virtual port is used to offload P2P traffic by rewriting the outer IP addresses, MAC addresses, and GTP TEID from GTP data packets. This port forwards modified packets to the destined eNodeB. SAPOM SDN also uses physical ports to interface with the eNodeBs and the SGW.

5. EVALUATION

We evaluate our prototype in the PhantomNet [8] testbed. We use two Nexus 5 machines as UEs, and a real IpAccess small cell as the eNodeB. For the EPC, we use OpenEPC-based components as provided by PhantomNet. We add our SDN components between the eNodeB and the EPC. To evaluate scenarios with different load conditions, we generate background traffic on the EPC using emulated UEs and an eNodeB that bypasses SAPOM to the EPC. In our evaluation, we configure a 25 ms delay between SAPOM and the EPC. This number is derived from average mobile network round trip time (RTT) [12]. The actual delay in a mobile network largely depends on the capabilities of the network, the distance between key components, and the service provider [10, 11, 19].

To validate our design, we evaluate the SAPOM components and end-to-end latency reductions with our approach.

5.1 SAPOM Controller

We measure the processing time of the SAPOM controller in installing flows. The average processing time for the controller to parse the incoming packet is 1.36 ms and the average processing time to install the necessary flow rules into the OVS is 2.54 ms. Only two packets are forwarded to the controller for processing for each P2P connection.

5.2 RTT Improvement

To evaluate RTT improvements, we compare the RTT between two devices with SAPOM and without SAPOM; packets are redirected from SAPOM SDN and from the NAT router in the EPC respectively.

Using ping between two UEs connected within our testbed, one UE sends ping request packets to the other at one second intervals for 300 seconds with its private IP address. Figure 4 shows our experiment result. The first RTT is similar with and without SAPOM since with SAPOM enabled this packet is forwarded to

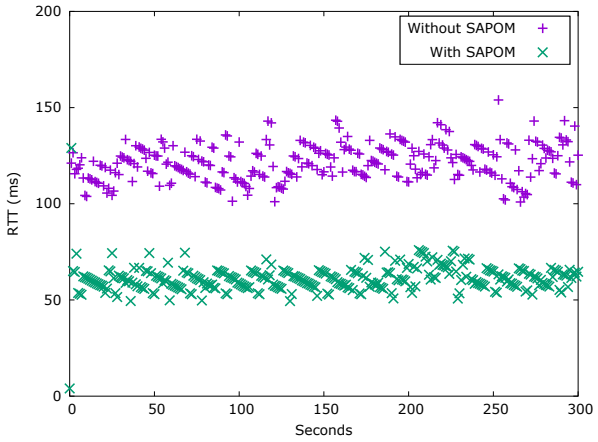


Figure 4: Latency of ping test between two Nexus 5 devices under same eNodeB without SAPOM and with SAPOM

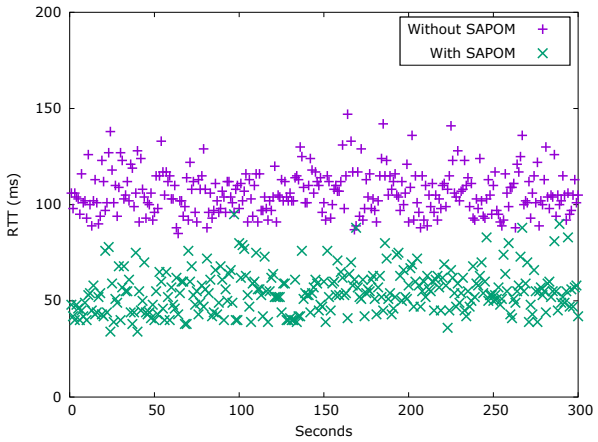


Figure 5: Latency of Skype audio and video message call test between two Nexus 5 devices under same eNodeB without SAPOM and with SAPOM

the SAPOM controller to create flow rules for offloading, while also being forwarded to the NAT router for NAT traversal. After inserting offloading rules into the SAPOM SDN, the end-to-end RTT is 61.1 ms on average. Whereas, without SAPOM the average end-to-end RTT is 121.4 ms. The evaluation reveals an anomalous downward pattern in RTT during the ping test. It is our guess that this is caused by a modulation and coding scheme (MCS) between the UE and eNodeB according to traffic load. Increasing the traffic by introducing high background traffic removes this anomaly. However, this altered the average RTT of our ping tests therefore such evaluation results are omitted.

To evaluate a more realistic case, we start a Skype video call between two Nexus 5 phones. Similar to the ping test, one UE calls the other with audio and video activated. End-to-end RTT is recorded for 300 seconds with Skype's in-app "technical call info" feature.

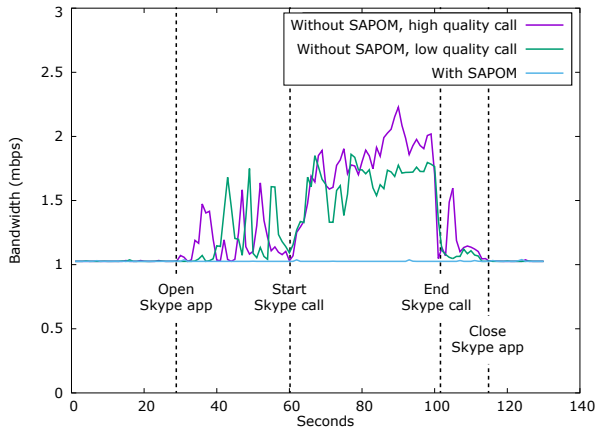


Figure 6: Reduction of EPC load with SAPOM

Figure 5 shows the result of our experiment. With SAPOM, the end-to-end RTT is 54.7 ms and without SAPOM, the same is 106.8 ms.[1]

Our tests show about a 50 percent lower RTT using our solution. We also execute the ping and the Skype test when one device is in idle mode and verify that both cases work after the paging procedure.

5.3 EPC Load Reduction

To evaluate how much load SAPOM reduces of the EPC network for P2P applications, we measure the bandwidth with Skype applications use. Since there are two options to adjust video quality in Skype applications, we test both low quality video and high quality video cases. In this experiment, we generate 1 Mbps traffic from the emulated UEs and run the Skpye application on both Nexus 5 devices. We then measure the total bandwidth in the EPC.

Figure 6 shows the result of our experiment. With SAPOM, the bandwidth usage is 1 Mbps since only the traffic from the emulated UEs goes through EPC and the Skype traffic is redirected from the SAPOM SDN. The difference between the with SAPOM and the without SAPOM case is the reduction of the EPC traffic load with SAPOM. As the number of UEs using P2P applications increases, the bandwidth SAPOM saves increases improving the performance of other applications on the network.

6. RELATED WORK

Recently, several proposals have been made to support device-to-device (D2D) communication within the mobile architecture [21, 25]. In D2D, a device uses its LTE air interface to identify and communicate with other devices over a direct link [17]. However, D2D only supports communication between devices in very close proximity and does not replace P2P. D2D is also still in its research phase and requires communicating devices to be D2D capable, whereas SAPOM works seamlessly with existing devices and protocols. Our work is conceptually similar to *local breakout* mechanisms proposed by the standards body [1]. However, unlike these works, SAPOM does not require additional expensive offloading infrastructure and significant modification in standard protocols.

Our work is largely motivated by the SMORE architecture [12] which supports SDN-based offloading in a mobile network. However, our work is significantly different from SMORE in the following aspects: i) it does not require a monitor to sniff control messages leading to a cleaner, lightweight design, ii) it does not need to store device state, and iii) it offloads without the need of a subscription service. SAPOM is also completely decoupled from the underlying mobile network state, and works seamlessly with mobility related functions such as paging and handover.

7. DISCUSSION

To deploy SAPOM in real mobile network, it would require several features. SAPOM would need a charging mechanism for offloaded P2P traffic that avoids the Policy and Charging Enforcement Function (PCEF) in PGWs. Since offloaded P2P traffic only passes logical ports in SAMPO SDN switches, we can add interfaces (gz/gy) to talk to the charging components (e.g., OCS and OFCS) in the mobile network for only P2P traffic by extending functions of SAMPO SDN. SAPOM would also need to allow deep packet inspection for lawful interception of traffic [7]. Both features are feasible and will be addressed in future work. We also consider how SAPOM would fit within the 5G standards. Although there are many uncertainties regarding 5G, our work would supplement the expected SDN addition to the 5G deployment. Lastly, we expect to explore other P2P offloading possibilities for different mobile network topologies in the future. Specifically, we expect to enable P2P offloading between multiple MTSO locations by utilizing inter-MTSO SDN switching fabric and a distributed controller framework. However, we are not concerned with extending SAPOM across multiple providers. While it is true that providers want standards to communicate across themselves, they also want services that differentiate themselves, such as those found in SAPOM.

8. CONCLUSION

We have presented SAPOM, an SDN-based P2P offloading architecture that improves P2P connections latency by offloading traffic close to connected mobile peers, and also reduces core network load. In the future, we expect to enable several core mobile network features, as well as exploring P2P offloading potential in other network topologies.

[1]In some tests without SAPOM, Skype would avoid a P2P connection and route through a dedicated supernode. In this case, we measured an average RTT of 220.1 ms.

9. REFERENCES

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

[2] Ryu controller, a component-based software-defined networking framework. http://osrg.github.io/ryu/.

[3] Skype internal data. http://advertising.microsoft.com/en/skype, 2013.

[4] Cisco visual networking index: Forecast and methodology, 2014-2019. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html, 2015.

[5] How much bandwidth does Skype need? https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need, 2015.

[6] OpenEPC. http://www.openepc.com/, 2015.

[7] Balbas, J.-J., Rommer, S., and Stenfelt, J. Policy and charging control in the evolved packet system. *Communications Magazine, IEEE 47*, 2 (February 2009), 68–74.

[8] Banerjee, A., Cho, J., Eide, E., Duerig, J., Nguyen, B., Ricci, R., Van der Merwe, J., Webb, K., and Wong, G. Phantomnet: Research infrastructure for mobile networking, cloud computing and software-defined networking. *GetMobile: Mobile Computing and Communications 19*, 2 (2015), 28–33.

[9] Baset, S., and Schulzrinne, H. An analysis of the skype peer-to-peer internet telephony protocol. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings* (April 2006), pp. 1–11.

[10] Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R., and Towsley, D. A measurement-based study of multipath tcp performance over wireless networks. In *Proceedings of the 2013 Conference on Internet Measurement Conference* (New York, NY, USA, 2013), IMC '13, ACM, pp. 455–468.

[11] Chen, Y.-C., and Towsley, D. On bufferbloat and delay analysis of multipath tcp in wireless networks. In *IFIP Networking 2014* (2014), IFIP.

[12] Cho, J., Nguyen, B., Banerjee, A., Ricci, R., Van der Merwe, J., and Webb, K. SMORE: Software-defined networking mobile offloading architecture. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges* (New York, NY, USA, 2014), AllThingsCellular '14, ACM, pp. 21–26.

[13] Ford, B., Srisuresh, P., and Kegel, D. Peer-to-peer communication across network address translators. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2005), ATEC '05, USENIX Association, pp. 13–13.

[14] Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., and Faloutsos, M. Is p2p dying or just hiding? [p2p traffic measurement]. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE* (Nov 2004), vol. 3, pp. 1532–1538 Vol.3.

[15] Karagiannis, T., Broido, A., Faloutsos, M., and claffy, K. Transport layer identification of p2p traffic. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA, 2004), IMC '04, ACM, pp. 121–134.

[16] 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.

[17] Lin, X., Andrews, J. G., Ghosh, A., and Ratasuk, R. An overview on 3gpp device-to-device proximity services. *CoRR abs/1310.0116* (2013).

[18] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev. 38*, 2 (March 2008).

[19] Nguyen, B., Banerjee, A., Gopalakrishnan, V., Kasera, S., Lee, S., Shaikh, A., and Van der Merwe, J. Towards understanding tcp performance on lte/epc mobile networks. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges* (New York, NY, USA, 2014), AllThingsCellular '14, ACM, pp. 41–46.

[20] OpenVswitch. http://openvswitch.org/.

[21] Osseiran, A., Doppler, K., Ribeiro, C., Xiao, M., Skoglund, M., and Manssour, J. Advances in device-to-device communications and network coding for imt-advanced. *ICT-MobileSummit* (2009).

[22] Patro, A., Rayanchu, S., Griepentrog, M., Ma, Y., and Banerjee, S. Capturing mobile experience in the wild: A tale of two apps. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies* (New York, NY, USA, 2013), CoNEXT '13, ACM, pp. 199–210.

[23] Python Tutorial. struct - Interpret strings as packed binary data. docs.python.org/2/library/struct.html.

[24] Rao, V. S., and Gajula, R. Protocol signaling procedures in lte. *White Paper, Radisys Corporation* (2011).

[25] Tehrani, M., Uysal, M., and Yanikomeroglu, H. Device-to-device communication in 5g cellular networks: challenges, solutions, and future directions. *Communications Magazine, IEEE 52*, 5 (May 2014), 86–92.

[26] Wang, Z., Qian, Z., Xu, Q., Mao, Z., and Zhang, M. An untold story of middleboxes in cellular networks. *SIGCOMM Comput. Commun. Rev. 41*, 4 (Aug. 2011), 374–385.

[27] Xu, Y., Yu, C., Li, J., and Liu, Y. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference* (New York, NY, USA, 2012), IMC '12, ACM, pp. 371–384.

[28] Yahyavi, A., and Kemme, B. Peer-to-peer architectures for massively multiplayer online games: A survey. *ACM Computing Surveys (CSUR) 46*, 1 (2013), 9.

[29] Zhang, D., Zheng, C., Zhang, H., and Yu, H. Identification and analysis of skype peer-to-peer traffic. In *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services* (Washington, DC, USA, 2010), ICIW '10, IEEE Computer Society, pp. 200–206.