MobiScud: A Fast Moving Personal Cloud in the Mobile Network*

Kaiqiang Wang
kaiqiang.wang@utah.edu

Minwei Shen
Minwei.Shen@utah.edu

Junguk Cho
junguk.cho@utah.edu

Arijit Banerjee
arijit@cs.utah.edu

Jacobus Van der Merwe
kobus@cs.utah.edu

Kirk Webb
kwebb@cs.utah.edu

School of Computing, University of Utah

ABSTRACT

We present MobiScud, an evolutionary mobile network architecture that integrates cloud and SDN technologies into a standard mobile network in backwards compatible fashion. MobiScud enables personalized virtual machines to seamlessly "follow" mobile network users as they move around.

CCS Concepts

•**Networks** → **Network architectures; Cloud computing; Mobile networks;**

Keywords

Mobile Network; Cloud Computing; Service Offloading; Software Define Networking (SDN).

1. INTRODUCTION

Recent years have seen phenomenal growth in wireless communications with the advent of smartphones and various other wireless devices that have transformed our lifestyle and communication patterns. Further, with the rapid adoption of wearables and Internet of Things (IoT), new devices and ideas are emerging at an unimaginable speed, dramatically changing the way we interact with cyber space (e.g., the Google Glass, smart wristbands, the iWatch etc.).

Imagine what amazing things we can do with them: You wear a pair of Google glasses. It continuously does facial recognition for you. Whenever you meet someone you recently met, it reminds you of his name and when you two met the last time. More magically, maybe your Google glasses can analyze the other person's subtle facial expressions and body gestures, telling you whether he is lying or not. Another example is a smart wristband that carries a lot of sensors. It collects and records statistics of your body, like your heart rate, blood pressure and body fat. With sufficient data, it performs data mining and analysis, giving you suggestions on your diet, the amount of sleep you should have and recommended types and amount of exercise. To quote the CloudLet paper [10], we are entering an "entirely new world in which mobile computing seamlessly augments users' cognitive abilities via compute-intensive capabilities such as speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality, planning, and decision making." We echo the CloudLet vision that to realize such computationally demanding applications with strict latency requirements, we need an infrastructure that allows a mobile user to instantiate and interact with a resource-rich customized virtual compute instance, i.e., a personalized cloud [10, 11].

However, the current cellular network architecture, typically composed of monolithic hardware boxes deployed in a few centralized operator locations, is ill-suited to deal with the phenomenal growth of mobile devices and demanding application requirements [13]. Hence, operators are gradually moving towards highly distributed mobile network architectures [9], leveraging emerging technological trends like network function virtualization (NFV) and software defined networking (SDN), to meet the requirements (e.g., scalability, flexibility, low latency) of future applications [9]. Additional constraints like backward compatibility and standards compliance render the job of realizing a CloudLet-like architecture in the current cellular network even more challenging.

Aligned with these trends and challenges, we propose MobiScud, an evolutionary mobile network architec-

*Scud clouds are low clouds that often move faster than their associated storm clouds.

AllThingsCellular'15, August 17-21, 2015, London, UK
 ISBN 978-1-4503-3538-6/15/08…$15.00
DOI: http://dx.doi.org/10.1145/2785971.2785979

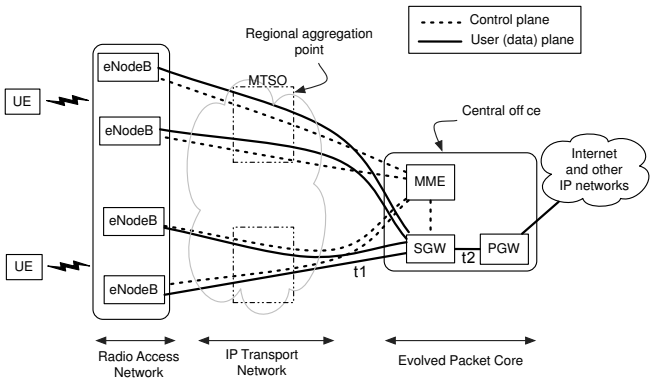

Figure 1: LTE/EPC Architecture

ture that integrates cloud and SDN technologies into a standard mobile network in a backwards compatible fashion. MobiScud assumes the presence of a highly distributed cloud platform in, or close to, the RAN (Radio Access Network) [9], where user-specific customized private VMs are instantiated, and leverages SDN capabilities to offload low-latency, compute-intensive applications to the private VM instances [10, 11]. Furthermore, MobiScud monitors the control plane message exchanges between the user and the mobile network, and migrates the private VM to the nearest cloud location as the user moves (hands over) from one base station to another. MobiScud leverages existing live VM migration techniques and dynamically updates SDN flow rules to ensure that ongoing connections are not disrupted due to user mobility.

Our contributions are summarized as follows,

•We design MobiScud, a practical cloud computing architecture for mobile (LTE/EPC) networks that integrates cloud and SDN technologies into a standard mobile network in backwards compatible fashion.

•We explore how this architecture can support low latency services/applications in a distributed mobile network. Specifically, we use the example of a personal VM [10, 11] moving with the user to show how future applications can benefit from a close interaction between the cloud and the mobile network.

•We show the feasibility of MobiScud using a prototype implementation in PhantomNet, an LTE/EPC testbed [7].

2. BACKGROUND AND MOTIVATION

Current LTE/EPC Architecture. We briefly describe the LTE/EPC mobile network architecture, typical deployment and user-plane (data plane) protocol stack. 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. 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. The SGW and the PGW are responsible for routing/forwarding the data packets from all UEs to and from the external network. 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, (e.g., $t1$ and $t2$ in Figure 1). These tunnels realize the data path (user plane) which the UE uses to send and receive packets.

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 [13]. 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 exist 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 regional aggregation points. As shown in Figure 1 these aggregation points are called (or co-located with) mobile telephone switching offices (MTSOs) and there are an order of magnitude more MTSOs than centralized EPC locations in a typical deployment. Finally, since eNodeBs realize the RAN functionality in mobile networks, the eNodeB footprint is highly distributed with hundreds of thousands of cell sites for a typical provider [1].

Motivation. MobiScud is strongly motivated by the CloudLets [10] work. CloudLets envisions the presence of a private virtual compute instance in a cloud, located very close to a mobile user, to enable offloading of compute-intensive applications with extremely low latency requirements, (e.g., facial recognition, speech translation, augmented reality etc.). With the advent of next-generation devices like Google Glass, the iWatch and autonomous cars, the vision of CloudLet is now an imminent reality. Unfortunately, the centralized nature of the current mobile network architecture is ill suited to support the requirements of a CloudLet-like architecture. First, all user traffic is currently routed (tunneled) to centralized operator gateways before exiting to the Internet and cloud instances, leading to path inflation which results in unpredictable and long delays [14]. Second, applications like cognitive augmentation and personal healthcare require exchange of extremely sensitive and private information between the mobile devices and the cloud. As others have argued, sending such data over untrusted networks to a far away third-party cloud exposes them to unnecessary privacy related attacks [11]. Last, the current cellular network is

composed of costly, closed-source, standards compliant vendor specific equipment. It is difficult to add customized functionality to these elements to support new application requirements.

We designed MobiScud to address these limitations and challenges. We take inspiration from our earlier work on the SMORE architecture [4] which allows offloading of selected traffic to an in-network cloud platform using SDN capabilities, but does not require any change to the existing standards-based interactions. MobiScud follows the same practical design principle of backward compatibility.

Finally, we observe that mobile operators are gradually moving towards a highly distributed mobile network architecture and leveraging emerging technological trends, like network function virtualization (NFV) and software defined networking (SDN), to meet the requirements of the future applications [9]. Our work is aligned with these current trends and we assume the presence of a highly distributed cloud platform in or close to the RAN, where a user-specific customized private VM is instantiated. We leverage SDN capabilities to offload low-latency, compute-intensive applications to the private VM instance.

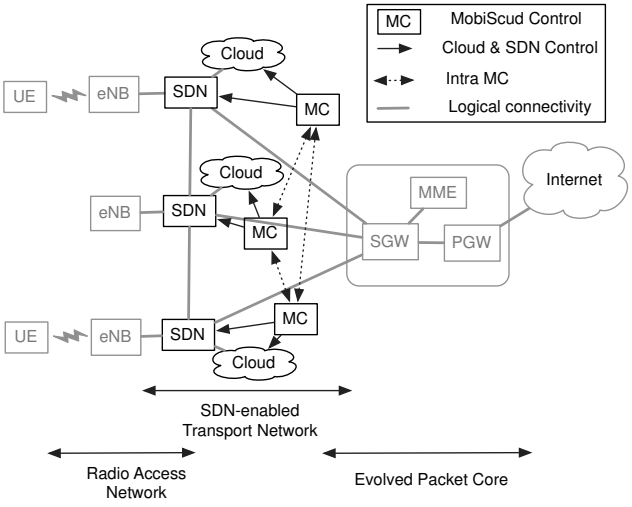

Figure 2: MobiScud Architecture

3. ARCHITECTURE

The MobiScud architecture is depicted in Figure 2. The figure also attempts to depict the fact that, as mobile operators adopt NFV and SDN technologies [2], we can expect the distinction between RAN, transport and core networks to become less well defined as functionality can be more flexibly realized on virtualized platforms. Specifically, for our work on MobiScud, we assume the presence of an operator cloud platform at each eNodeB location.

The MobiScud control (MC) function interfaces with the mobile network, the operator cloud and the SDN

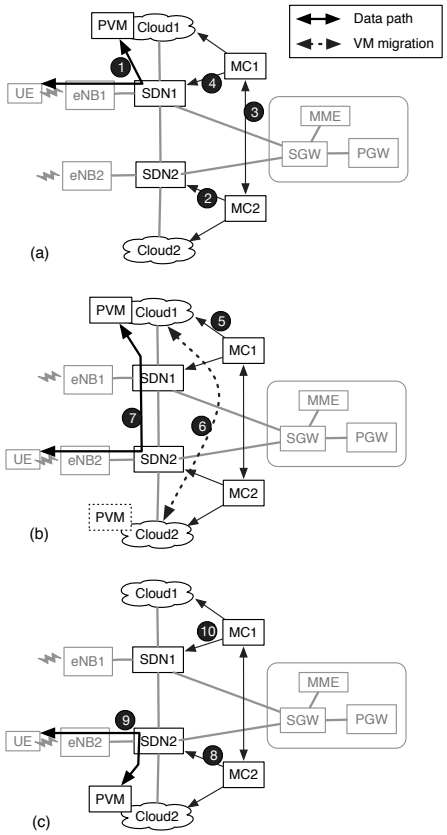

Figure 3: MobiScud workflow

substrate to selectively offload traffic to a private VM hosted in the cloud for low latency user applications. The MobiScud control function consists of two logical entities. First, the *monitor* is responsible for monitoring the control plane signaling messages exchanged between mobile network elements and extracting the necessary data to allow MobiScud to be aware of mobile network activity (e.g., connection and handover requests). Second, the *controller* uses this data extracted by the monitor and orchestrates between the SDN substrate and the provider cloud to realize MobiScud functionality. Specifically, the controller uses the extracted data plane parameters to construct relevant flow rules (routing, GTP encapsulation and decapsulation) for the SDN substrate to enable application-specific offloading, and further interacts with the cloud platform to instantiate and migrate the private VM instance across the operator cloud as the user moves from one location to another. We assume connectivity between the distributed SDN and cloud instances, providing connectivity between MobiScud controllers and enabling the migration of personalized VMs between cloud instances. Note that MobiScud allows the flexibility of offloading only selected application traffic that can leverage the benefit of a closely located private VM. All other user traffic continues to flow through the default path via the centralized gateways.

Figure 3 depicts a workflow of MobiScud's opera-

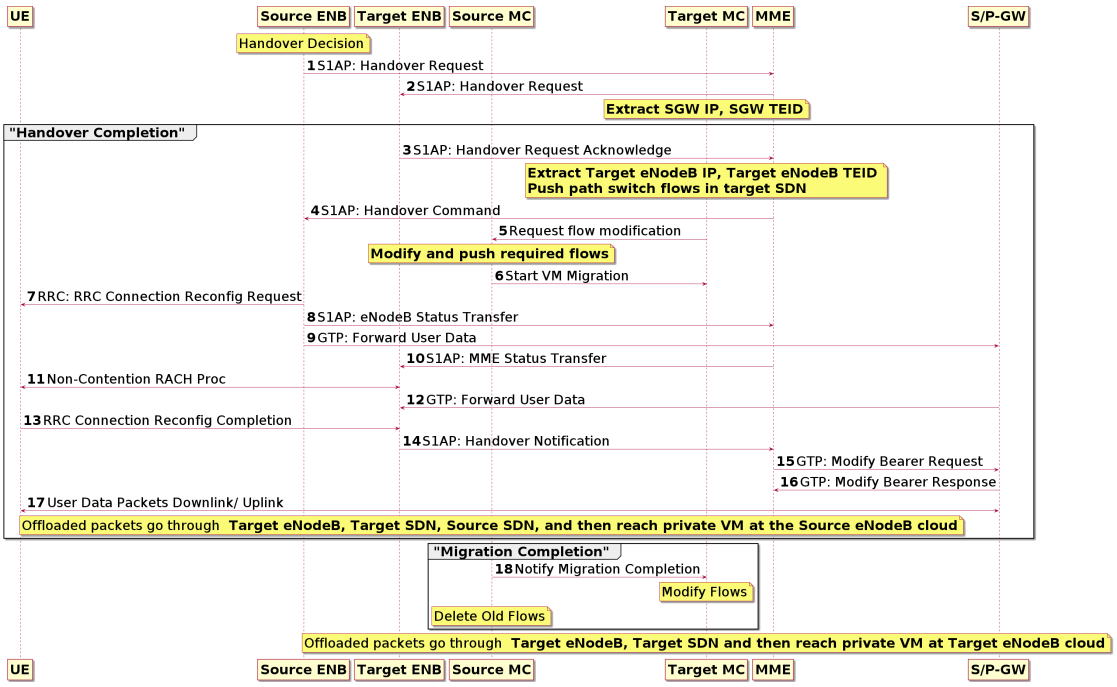

Figure 4: Combined handover, SDN and personal VM migration signaling

tion. Figure 3(a), shows the initial state when a UE connects to the network. A private VM (PVM) instance is created for it in the operator cloud co-located with the corresponding eNodeB (eNB1, Cloud1). The MobiScud monitor function monitors the control plane messages exchanged for the attach procedure between the eNodeB and the MME, and extracts the necessary data and control plane parameters like GTP tunnel IDs, S1AP session identifiers etc., for offloading. The MobiScud controller uses this information to construct flow rules for the SDN substrate (SDN1) that decapsulates uplink GTP-U packets (from the eNodeB to the SGW) belonging to the offloaded application and routes the same to the private VM instance. Similarly, the downlink packets from the private VM instance are encapsulated as GTP-U packets with relevant tunnel-IDs before they are routed back to the eNodeBs. (#1 in Figure 3.) This encapsulation/decapsulation allows the offloading function to be transparent to the eNodeBs and does not require any change to the standard interaction between the eNodeB and the core network [4].

As the UE moves towards eNB2, signaling messages are exchanged between the source eNodeB (eNB1) and the target eNobeB (eNB2) via the MME. The MobiScud monitor function watches for handover events in the control plane message exchanges. In this case MC2 will detect the imminent handover to eNB2 and will trigger the PVM migration actions. MC2 extracts the GTP tunnel IDs for the SGW and the target eNodeB (eNB2) from the S1AP Handover Request message and Handover Request Acknowledgement message respectively. The target MC (MC2) then adds flow rules in the target SDN (SDN2) to enable path switch (#2) and also sends a flow modification request to the source MC (MC1) (#3). The source MC (MC1), in turn, performs the necessary flow rule modifications in the source SDN (SDN1) (#4), to ensure that offloaded traffic from the target eNodeB can still reach the private VM located at the source cloud (Cloud1) once the handover is completed. At this point, the source MC (MC1) also proactively starts live VM migration (#5 in Figure 3(b)), to migrate the personal user VM from the source cloud (Cloud1) to the target cloud (Cloud 2) (#6). Once the handover completes, data between the UE and the PVM will flow via eNB2, SDN2, SDN1 to the PVM which is still active in Cloud1 (#7). Once VM migration is completed, the target MC (MC2) is notified and it changes the flow rules in SDN2 (#8 in Figure 3(c)), so that the user traffic is directly offloaded to the new PVM location in Cloud2 (#9). At the same time, the source MC (MC1) deletes the old flow rules from the source SDN (SDN1) (#10), thereby disconnecting the indirect path between the target eNodeB and the source cloud.

Figure 4 shows a ladder diagram depicting this interaction between handover signaling and MobiScud signaling related to SDN and PVM migration in slightly more detail.

4. EVALUATION

We have prototyped the MobiScud architecture in the PhantomNet testbed [7]. We have implemented all components as described in Section 3. For PVM migration we used standard Xen live migration [6]. For our prototype we further simplified the VM migration pro-

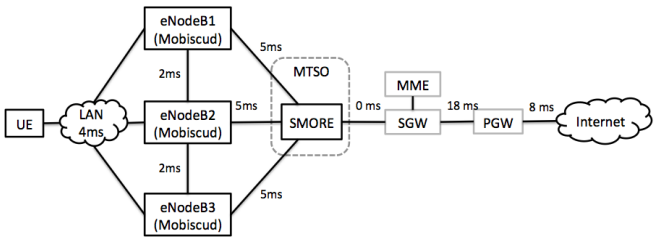

Figure 5: Evaluation setup in PhanomNet

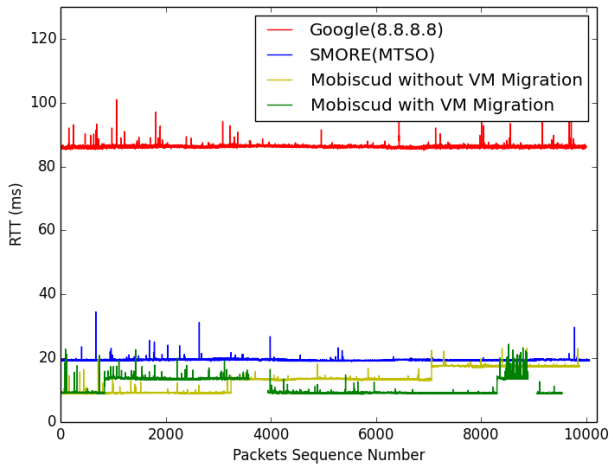

Figure 6: RTT of different services

cess by making use of network-based storage (iSCSI) for the VM and the storage is not migrated. Since solutions to migrate storage together with the VM memory state exist [3, 12], this is not a fundamental limitation.

Our evaluation setup is depicted in Figure 5. For our evaluation we used a simple emulated RAN as well as an emulated UE available in PhantomNet. This mode is sufficient for our purposes as our focus is on the integration of SDN and cloud technologies on the mobile core side of the eNodeBs. As shown in the figure we added delay nodes in the evaluation setup to represent realistic delay between the various network elements. Specifically, the delay between the UE and each of the eNodeBs is 4ms, between eNodeB and MTSO is 5ms, between SGW and PGW is 18 ms and between PGW and Internet is 8ms. Additionally, we set 2ms delay between eNodeBs and assume the MTSO and SGW/MME are co-located (0 ms delay).

To perform our evaluation we compared the round-trip-time (RTT) of MobiScud with that of accessing a website in the Internet, as well as with an offloading solution where the cloud platform is deployed at the MTSO location, i.e., the solution presented in SMORE [4]. To show the value of the moving PVM solution in MobiScud, we also compare this with the case where the VM is not migrated. In our evaluation, the UE is initially attached to eNodeB1. We then trigger an S1 handover procedure from eNodeB1 to eNodeB2. After a short time we similarly trigger a handover from eNodeB2 to eNodeB3. We send `ping` packets from the UE to a network based server in each case to determine the RTT.

RTT Comparison: The RTT results for our evaluation is shown in Figure 6. To allow for easy comparison we plot the results of all four experiments in one figure.

The red line in Figure 6 is the RTT between the UE and an Internet-based server (Google's public DNS). The average RTT in this case is 85.7 ms and S1-Interface handover does not impact the RTT value. The blue line in Figure 6 is the RTT between the UE and SMORE offloading, with the cloud server located at the MTSO. The average RTT is 19.3 ms and the S1-Interface handover again has no impact on the RTT. The yellow line in Figure 6 is the RTT between the UE and the PVM in the MobiScud setup for the case where no VM migration is triggered. I.e., in this case the PVM is allocated at the cloud associated with eNodeB1 and it remains there while the UE move to eNodeB2 and eNodeB3 respectively. As can be expected, the RTT increases with a small step after each S1 handover. The first handover happens at packet sequence number 3300 and the average RTT increases from 9.05 ms to 13.4 ms. The second handover occurs at sequence number 7200 and the average RTT increases from 13.4 ms to 17.6 ms. Finally the green line is the RTT between the UE and the PVM for the case where a handover triggers VM migration. After the first handover, at around sequence number 600, the average RTT initially increases from 9.03 ms to 13.66 ms. Once VM migration completes, at sequence number 4000, the PVM is located at eNodeB2 and the average RTT reduces back to 9.06ms. Similarly, after the second handover the RTT increases to 13.99 ms and returns to 9.1 ms once the PVM is successfully migrated.

Figure 6 also shows packet loss due to the final step of the VM migration process after each migration [6]. (The green line is interrupted just before the RTT reduces.) This period of VM "downtime" is 2.87 s and 1.44 s respectively for the two migrations. This downtime is the result of the (unoptimized) default Xen migration used in our setup. Finally, the figure shows the migration time during the first handover to be approximately 14 s, while the second migration took almost 7 s. The difference between the respective downtimes and migration times for the two handovers appears to be a function of the performance and location of physical nodes allocated as cloud nodes in the PhantomNet experiment.

While the downtime related to VM live migration can be optimized [12], the downtime due to migration does present a potential tradeoff point in our approach. Depending on factors such as the application's tolerance for loss versus latency, the nature of user mobility and cloud and network conditions, the system may adapt the frequency of VM migration. We plan to explore these tradeoffs as part of our future work.

5. RELATED WORK

Our work is motivated by, and related to, industry trends towards more distributed mobile network architectures [9] and the use of network function virtualization (NFV) and software defined networking (SDN) [2]. MobiScud borrows from our own earlier work on the SMORE architecture [4]. Architecturally, however, MobiScud takes a much more extreme position in assuming a highly distributed cloud platform close to (or co-located) with the RAN footprint, to enable very low latency applications.

From a service perspective, MobiScud's personal VM approach was motivated by the Cloudlet work [10] and also more recent work that used personal VMs in the context of online-social-networks to store user personal data [11]. However, unlike our work in MobiScud, these previous efforts did not address the networking aspects of such an approach, effectively treating the network as a black box.

Our use of live VM migration is related to a fairly well explored space [6, 12], and indeed we use this component unmodified in MobiScud. However, to our knowledge, MobiScud is the first attempt to apply this technology in the context of mobile networks.

Our work is also related to various cloud offloading efforts [5, 8]. These efforts largely focused on how and when to perform offloading, and again largely treated the network as a black box.

6. CONCLUSION

We presented the MobiScud architecture. To ensure that low latency between the mobile device and the cloud platform is maintained as users move around, MobiScud makes use of live virtual machine migration to move a personal VM associated with each user in concert with mobile network handovers associated with the user. We prototyped and evaluated the MobiScud architecture in an LTE/EPC testbed. As part of our future work we plan to explore the appropriate "distributedness" and scalability tradeoffs of cloud platforms in mobile networks.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1305384.

7. REFERENCES

[1] Annual Wireless Industry Survey. http://www.ctia.org/your-wireless-life/how-wireless-works/annual-wireless-industry-survey.

[2] AT&T Domain 2.0 Vision White Paper. http://www.att.com/Common/about_us/pdf/AT&T%20Domain%202.0%20Vision%20White%20Paper.pdf.

[3] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual execution environments*, pages 169–179. ACM, 2007.

[4] J. Cho, B. Nguyen, A. Banerjee, R. Ricci, J. Van der Merwe, and K. Webb. Smore: Software-defined networking mobile offloading architecture. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*, 2014.

[5] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, 2005.

[7] Flux Group, University of Utah. PhantomNet. https://www.phantomnet.org/.

[8] M. S. Gordon, D. A. Jamshidi, S. A. Mahlke, Z. M. Mao, and X. Chen. Comet: Code offload by migrating execution transparently. In *OSDI*, pages 93–106, 2012.

[9] A. Lemke. Why distribution is important in nfv. http://www2.alcatel-lucent.com/techzine/distribution-important-nfv/.

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 2009.

[11] A. Shakimov, H. Lim, R. Caceres, L. Cox, K. Li, D. Liu, and A. Varshavsky. Vis-a-vis: Privacy-preserving online social networking via virtual individual servers. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, Jan 2011.

[12] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. van der Merwe. Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines. *SIGPLAN Not.*, 46(7):121–132, Mar. 2011.

[13] Q. Xu, J. Huang, Z. Wang, F. Qian, A. Gerber, and Z. M. Mao. 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.

[14] K. Zarifis, T. Flach, S. Nori, D. Choffnes, R. Govindan, E. Katz-Bassett, Z. M. Mao, and M. Welsh. Diagnosing path inflation of mobile client traffic. In *Passive and Active Measurement*, 2014.