2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing

Follow-Me Cloud: An OpenFlow-based Implementation

Tarik Taleb, Peer Hasselmeyer, and Faisal Ghias Mir
NEC Europe, Heidelberg, Germany
[tarik.taleb,peer, faisal.mir]@neclab.eu

***Abstract*— Services are increasingly provided by cloud computing systems. Such services are often accessed while on the move. If the distance between the user and the service is getting large, quality-of-experience, in particular for interactive services, deteriorates. To counter this, the provisioning location of the service should be as close to the user as possible. This paper describes the concept of follow-me cloud, according to which services are migrating in unison with the user's movements. The key components required for such integrated user mobility/service mobility management are introduced and mappings to an OpenFlow-based implementation are described.**

I. INTRODUCTION

MOBILE operators are facing the challenging task of accommodating huge mobile traffic volumes, far beyond their original network capacities. Operators are thus investigating cost-effective methods for accommodating such huge mobile network traffic with minimal investment into their infrastructure. Most important solutions pertain to Selective IP Traffic Offload (SIPTO) as close to the Radio Access Network (RAN) as possible [1]. The key enabler of efficient SIPTO is to place data gateways close to the RAN, essentially leading to a relatively decentralized mobile network deployment [2].

Looking at the other end of the service chain, a strong trend towards using cloud technology to provide services can be observed. Clouds are a preferred method for service provisioning as with the cloud model, creators of services do not need to care about setting up and managing the service provisioning hardware and software, but they can rather focus on their main business of producing compelling applications and services for their users.

Along with their great success and promising market, cloud providers are moving towards distributed data center architectures by building increasing numbers of regional data centers [3][4]. The reasons for this trend are manifold and include different regulatory regimes, resilience, load balancing, and response times. The combined result is an enhanced Quality of Experience for users by ensuring short average response times, low error rates, and short downtimes.

Ideally, traffic offload close to the RAN and distributed cloud data centers should go hand-in-hand. Indeed, traffic between an application and its back-end service shall be extracted from the backhaul network at the earliest possible point and shall be sent to the data center closest to the break-out point. Unfortunately, the back-end service may not be necessarily hosted at the data center closest to the user. Even if the service is initially provided from the optimal data center; "optimality" changes over time with the movements of the user and the load situation on the data center and the network. For operators to gain the most benefits from traffic offload without impacting the QoE of their customers, mobile services must migrate together with user movements, changing workloads, and potentially other criteria.

This paper deals with how to improve quality of experience by introducing a new user/service mobility management scheme called "follow-me cloud". Follow-me cloud allows services to migrate in unison with users' movements. Services are therefore always provided from data center locations that are optimal for the current locations of the users and the current conditions of the network. As a by-product, bandwidth demand on the back-bone of the mobile network is reduced as traffic is kept locally as much as possible. Another advantage of the follow-me cloud technology is that migration of services is seamless and transparent to users. On-going sessions between users and services are not interrupted and connections do not need to be reestablished, even if users and/or servers (i.e., hosting services) change location. The paper describes the components needed to enable the follow-me cloud capability, in particular the detection of user movements, the decision logic for migrating services and the method for making migration seamless. An OpenFlow-based implementation is described. The problems that were encountered and their solutions are detailed. A solution that achieves the objectives of the follow-me cloud concept without the usage of any software defined networking (SDN) technologies (e.g., OpenFlow) is described in [19].

The remainder of this paper is structured as follows. Section II gives an overview on some related research work. The proposed follow-me cloud concept is described in Section III. An OpenFlow-based implementation of the follow-me cloud concept is described and evaluated in Section IV. The paper concludes in Section V.

II. RELATED WORK

Movement of services in the form of code, data, state, and virtual machines is a well-investigated topic with a large body of research work. This paper does not propose new migration technologies; it rather builds on existing technologies in the area of virtual machine migration.

In the context of the Evolved Packet System (EPS) [5], its richness of accesses has led to different interesting 3GPP study items whereby a User Equipment (UE) is allowed to have simultaneous accesses to different networks using different access technologies [6]. In [7], the 3GPP System Architecture group investigated different possibilities for dynamic IP flow mobility between 3GPP and non-3GPP accesses. The study proposed allowing a UE, equipped with multiple network interfaces, to establish multiple PDN connections to different Access Point Names (APNs) via different access systems and to selectively transfer PDN connections between the accesses with the restriction that multiple PDN connections to the same APN shall be kept in one access. In [8], a solution is proposed enabling a UE to know how and when to establish a new optimized PDN connection for launching new IP sessions to a particular APN, without compromising the on-going (old) PDN connections to the same APN. In [9], a solution is proposed to support SIPTO. In this solution, the user plane of a mobile network is assumed to be decentralized and the objective is to enable a per-flow offload of certain IP traffic as near to the edge of the operator network as possible. This is

978-0-7695-5046-6/13 $26.00 © 2013 IEEE
DOI 10.1109/GreenCom-iThings-CPSCom.2013.59

achieved with the involvement of the mobile network's Domain Name Server (DNS) that informs a UE of the gateway to connect to for establishing a particular flow and that is upon making a DNS resolution request. Most of these flow mobility and session mobility mechanisms work under the assumption that IP addresses of users do not change when the session is active, or do change but when the UEs are in idle mode so users will not notice any service disruption.

Generally speaking, migration of an IP service, due to movement of the receiving user followed by change in his IP address, would result in the breakdown of the session and the need to reestablish a new one. This is intuitively due to the fact that IP addresses are in practice used for identifying both an endpoint and a network location. This overloading causes sessions to break when the location changes, but sessions continue. Session identifiers should therefore be separated from location identifiers. Methods for such separation have been devised before. DNS realizes such a separation, but it was not designed to provide constant updates of current location. It is rather used only once at session establishment time. The Locator/Identifier Separation Protocol (LISP) [10] makes such separation explicit, but does not natively support endpoint mobility. Serval [11] caters for user and service mobility and provides identifier/location separation by introducing an additional layer in the networking stack. It makes use of service identifiers which require changes to applications using the system. To avoid the breakdown of an IP session between two peers when the IP address of any of the two peers changes during the course of a session, Network Address Translation (NAT) can be also used. In the context of mobile networks, the support of NATing would require changes to nodes of the mobile network operator and also many operators are not in favor of NAT mainly with the foreseen expansion of IPv6.

In the Host Identity Protocol (HIP) [12], "Host" and "Location" identifications are separated. The location is bound to an IP address which can be changing. In short, IP address is used for routing packets to current location. However, the host identification is created by public/private key infrastructure and HIP associations should be maintained by end-points. The HIP associations are used for preserving transport connections upon movement. That is done by a special HIP control message "HIP re-address". Further, it may also involve a server where end-points can update their location information that can later be retrieved by clients. In comparison to HIP, the present work identifies an IP session with the help of OpenFlow rules, whose scalability represents the main challenge. Indeed, there are various dimensions for scalability, including the number of flows, the flow set-up rate, number of packets and the bandwidth of the control channel. Some ideas have been proposed to deal with this issue. DevoFlow [13] reduces the number of control packets by moving some of the flow creation work from controllers to switches. In [14], the scalability of OpenFlow rules in a follow-me cloud scenario is assessed and an approach to distribute control plane functions is proposed to enhance the system scalability. In [19], the authors describe how the objectives of the follow-me cloud concept can be achieved with no usage of any SDN technologies, consequently avoiding any associated scalability issue. Changes to 3GPP standards, including those relevant to the nodes and interfaces of the EPS architecture or the underlying protocols, are also avoided.

Regarding the placement of services depending on user location, a plethora of research work has been conducted in the recent literature. In particular, the demonstrator described in [15] shows how services can be placed according to information retrieved from an ALTO (Application-Layer Traffic Optimization) network server. This work can be used to find optimal service locations, but it is orthogonal to the migration hiding mechanism described herein.

III. FOLLOW-ME CLOUD CONCEPT

The basic idea behind the follow-me cloud concept is that services, provided by a cloud, are following users throughout their journey. As soon as a user moves and thereby changes his attachment point to the network, the optimal data center for providing the services being received by the user is determined. If the optimal data center is different from the currently used one, a decision is made for or against moving the service to the optimal location. As a result, the services follow the user throughout his movements. The follow-me cloud concept can be realized by different technologies. These technologies depend on the envisioned scenario and the underlying environment.

In order to realize the follow-me cloud concept, a number of functions need to be realized. From the description above, the main functions can be derived directly: *i*) detection of user movements, *ii*) selection of optimal service location, and *iii*) service migration. All three functions need to be present, independently of the underlying technologies.

A. Movement Detection

Detection of movement is used as a trigger for the following steps in the follow-me process. Most technologies have some inherent means of detecting changes of location. This is due to the fact that a (large enough) change in location is followed by a change in the network attachment point. Such change can be detected either directly or indirectly. A direct observation can be done by looking at the network attachment point. Indirect movement detection can be done by looking at the identifier used by the user equipment (UE) for transmitting data (e.g., IP address). As such identifiers are usually location dependent, a change of this identifier commonly signifies a change in location.

B. Location Selection

As soon as movement of the user has been detected, the optimal service provisioning location has to be calculated. In general, we assume the existence of a number of data centers that can provide the service a user is currently accessing. Among those data centers, the "best" one needs to be selected; and that is in terms of network-related parameters (e.g., latency and available bandwidth), parameters affecting Quality of Service (QoS) such as server utilization and server throughput,

and business-related parameters (e.g., network/server/storage costs, bulk discounts, and preferred providers). If the optimal location is deemed the same as the current location of service provisioning, nothing further needs to be done. In case the determined optimal location differs from the current location, service migration needs to be considered. To decide whether migration is appropriate or not, the costs of shifting the service from its current location to the new, optimal location need to be taken into account. The costs primarily consist of bandwidth costs needed for transferring the service. Many services consist of multiple cooperating pieces. For example, a remote desktop application may consist of the running operating system (OS), the OS image, and the user's stored data. All three parts can be located in different places. When migrating the desktop application, all three parts might be transferred to the destination. Alternatively, only parts of the application might be moved, for example, only the running OS and its image, while the user's data stays at its original location. Therefore, in the case of distributed services, a decision on which parts of the application to move needs to be made. Also, there might be different optimal locations for different parts of the application, e.g. data centers with cheap, large storage for data; and those with strong processing capabilities for calculations.

C. *Service Migration*

Once it has been decided to change the location of service provisioning to a different data center, the service (or parts of it) needs to be moved. Service movement in clouds is possible in multiple ways. A number of basic approaches can be distinguished, depending on whether a software-as-a-service (SaaS), platform-as-a-service (PaaS), or infrastructure-as-a-service (IaaS) model is used for providing a particular service. Services provided by a SaaS system can be moved to a different fulfillment place by sending the state and associated data to the destination location. As the software providing the services is available in all SaaS fulfillment locations, it is enough to only make service state and data available to the software at the destination location. To migrate services provided on top of a PaaS system, service state and data need to be transferred just as in the SaaS case. But as the cloud provider only provides the platform, not the service code itself, the code realizing the service needs to be transferred to the destination data center as well. In an IaaS environment, services are provided by virtual machines (VMs). A VM encapsulates code, data, and state of the service it provides. For service migration, the virtual machine with all three constituents needs to be migrated. Common hypervisors support such migration, usually in both a cold migration mode (with the service being unavailable during migration) and a live migration mode (with service access being possible while the VM is moving). Migration is similar to services on top of PaaS systems, but the complete OS and supporting platform need to be shipped to the destination as part of the migrating VM.

Although service migration is possible in all three models, it suffers from the same problem in all three cases: different data centers own different ranges of IP addresses. As soon as a service moves to a different data center, it also changes its IP address. As a result, all connections to its clients break down and need to be reestablished. Service provisioning is therefore not seamless when service migration happens. The remainder of this paper describes how service migration can be enabled without disrupting the service due to changes in the IP addresses of end-users.

IV. OpenFlow-based Implementation

The Follow-Me Cloud concept and the functions described in the previous section can be realized with different technologies. They can be also realized with no usage of any SDN technologies [19]. In order to show the breadth of possible technologies and the particular problems that arise in those domains, an OpenFlow-based implementation is described hereunder. For a Markov chain-based analytical model of the FMC concept, the interested reader is referred to [20].

A. *Experimental Setup*

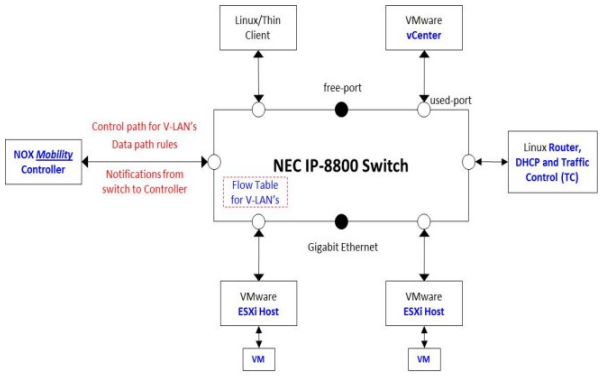

Fig. 1. OpenFlow-based follow-me cloud setup.

Fig. 1 shows the overall experimental setup which consists of data centers hosting VMs, client network based on WLANs, routers and a NOX based follow-me cloud (FMC) controller that are all connected to ports of an NEC IP-8800 OpenFlow switch. For the sake of simplicity, each datacenter in the cloud is modeled by a VMWare ESXi hypervisor. Each ESXi host is equipped with two 1Gbps network cards for forwarding the management and OpenFlow traffic over the network. A virtual network topology is defined inside the ESXi host by two vSwitches (soft-switch) where each physical NIC is connected with each soft-switch instance. The ESXi host manages the VM resources that run the standard Windows XP OS. Further, each VM is configured with two virtual NICs (vNIC) that are connected with the virtual network through the soft-switches. One vNIC carries the management traffic [16] and the other NIC carries the OpenFlow traffic. The storage space is shared between the two datacenters and is accessed by the standard iSCSI protocol. The datacenters are remotely managed by the VMWare vCenter software. Further, the client network consists of two WLANs. Given the client and data center networks, a router entity is used for correctly forwarding traffic among different network segments. For the sake of simplicity, the router acts as the first hop for traffic originating from client and data center networks. Further, the Linux router runs DHCP servers and Linux Traffic Control (TC) for controlling the path

characteristics (e.g., delay and congestion) between the two network segments. From the physical OpenFlow switch perspectives, four virtual switches (VLAN) are used for separately carrying the traffic of the two data centers and the client network. The FMC controller manages the forwarding behavior on the four VLAN's and also monitors the path characteristics between a data center and the client network and that is for resource management optimizations. For live VM migration, the VMotion® [17] cloud infrastructure technology from VMware is used. VMotion® traffic is mapped on the management network whereas all active communication between the VM and remote users are managed by the OpenFlow network.

B. The OpenFlow based NOX Controller

Fig. 2 shows the architecture of the OpenFlow-based FMC controller that has been developed in NOX. For validating the FMC concept, the controller entity is assumed to be aware of *i*) the virtual switch instances and their data path identifiers on the physical OpenFlow switch, *ii*) the VM identifiers [18] (namely the IP and MAC addresses), *iii*) the location and IP addresses of each default gateway in the test bed, *iv*) the OpenFlow switch ports identifiers at which the data center, router and client networks are connected, *v)* the IP address ranges managed by each DHCP server both for client and datacenter networks, and *vi*) the locations of distributed data centers that can either be part of the operator network or could be autonomous domains. In addition to a database, the OpenFlow-based FMC controller consists of seven components, each playing a particular role described hereunder.

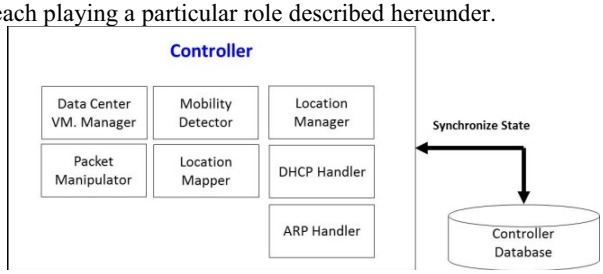

Fig. 2. The FMC controller architecture.

1) Location Manager

For correctly installing forwarding rules into the OpenFlow switch, each client and VM is linked to a home location. This is based on the IP address allocation and gateway settings configured in the VM. The configuration settings may be changed by the administrator during the service time and the database is accordingly updated. Such configuration also holds for clients in the client network. Given the home location is known to the controller, if any traffic from a particular client or VM appears on a different network than its home location, the Location Manager updates the status for that entity to be in a Visited Network/Location. Hence, the Location Manager keeps always track of current location of clients and VMs in home and visited networks.

2) Location Mapper

Given the home and visited locations of both client and VMs are known to the FMC controller, the Location Mapper module optimizes the path characteristics by selecting the appropriate data center location for the VM. Such control logic can be mapped on the geographical location of the data centers, path characteristics metrics based on average delay, load or even congestion situations between the client and datacenter networks. For evaluation, proprietary API for vCenter® is used at the controller for triggering VM migration across datacenters.

3) Mobility Detector

The actual VM migration is carried out by the cloud infrastructure software. For proof of concept, VMotion® technology is employed. However, orthogonal to the underlying technology, the FMC controller shall be able to detect when a VM has been actually moved to a new location. For evaluation, the Mobility Detector function keeps track of the flow entries installed in each OpenFlow virtual switch instance pertaining to home and visited locations. The OpenFlow rules for home locations are pro-actively installed in the switch. However for visited network, no such rule is installed. When traffic for a newly migrated VM hits the OpenFlow switch of a visited network, it must result into no match for that flow table in the switch. Afterwards, the packet should be forwarded to the controller which compares the location information with IP/MAC addresses to ascertain that VM has been indeed moved to the visited network.

4) ARP Packet Processing

VMs and clients are configured with default gateway settings. Initially, ARP (Address Resolution Protocol) caches are assumed to be empty at both endpoints and the router involved in the setup. There is no ARP specific forwarding rule installed in any of the virtual switches. Instead, each ARP packet, both request and response types, is explicitly forwarded to the FMC controller. As depicted in Fig. 3, upon reception of an ARP request from a virtual machine, the controller answers with an appropriate response. The response is constructed using the controller's knowledge of the layer-2 information of the attached end-points, including the gateway. The controller uses the PACKET_OUT OpenFlow command instructing the switch to send the ARP reply on the switch instance/port on which the original request was received. The necessary state (i.e., data path identifier, input port, source MAC and IP address) for constructing the reply message is taken from the original ARP request packet that has been forwarded to the controller. On the end-host, once the ARP reply arrives, the ARP cache is updated. In contrast, the ARP cache at the router is still stale because the controller replied on behalf of the router. Subsequently, when an IP packet arrives at the router for that particular end point, it also generates an ARP request which is again forwarded to the controller. In the current setup, the controller simply forwards all ARP requests from the router to the end points on the same subnet. However, it would also be possible to let the controller reply to these requests as in the previous case for a similar effect.

5) Packet Manipulator

Given a VM can either move from the home network to a visited network and vice versa, a key functional requirement from the controller perspectives is to preserve all ongoing user sessions while the VM is migrated. This implies that no con-

figuration change (e.g., IP address and gateway configurations) is allowed on the VM. Further, the IP address ranges managed by datacenters can be overlapping and the first hop setting may not be consistent across subnet boundaries. Hence, a Packet Manipulator module is introduced at the controller for creating a "virtual tunnel" within the visited network segment. The "virtual tunnel" operates by re-writing the IP address field within the packet IP header for each outgoing packet from the VM to outer network. The original IP header is restored for the packet when the last hop in the visited network segment is reached. The same technique is applied for all the incoming traffic to the VM. This is achieved by modifying the set of OpenFlow rules installed in the visited network.

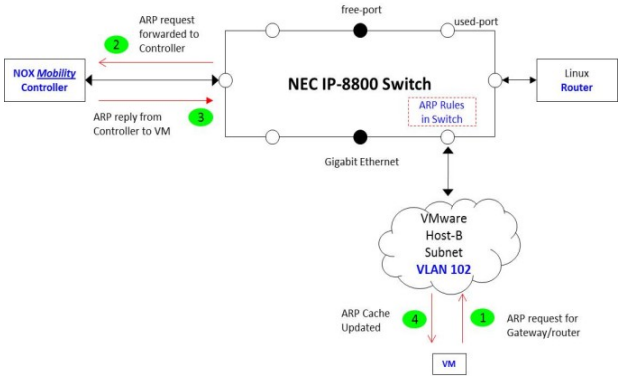

Fig. 3. Handling of ARP packets by FMC Controller.

6) DHCP Handling

The client network consists of two WLANs and the IP address ranges are managed by two DHCP (Dynamic Host Configuration Protocol) servers. The client location is dynamic and beyond operator control. However, based on client's location, the optimal location of VM can be decided. Therefore, a DHCP server is also implemented in the NOX controller and specific rules are installed in the switch such that all DHCP traffic should traverse the controller entity. The amount of DHCP traffic is small. DHCP overhead is therefore deemed to be negligible. Fig. 4 portrays the flow of messages exchanged among client, FMC controller and DHCP server till a connection is established/restored between the client and an adequate data center, based on their locations. Of particular interest, Step 10 in the figure shows that the client machine successfully acquired a new IP address and based on the current client location the decision for VM migration can be made.

C. System Execution and Performance Evaluation

In the remainder of this section, we evaluate the performance of the follow-me cloud setup, as depicted in Fig. 1. Further results based on an analytical model of FMC are available in [20]. In this setup, we configure the queue parameters for each virtual interface using the Linux Traffic Control modules on the Linux machine. Without any purposes in mind, the communication delays between a client network and its optimal data center and between a client network and its "sub-optimal" data center are set to 1ms and 50ms, respectively. Fig. 4 shows the ping latency between the client and its corresponding VM hosted in the data center and that is considering two scenarios, namely when follow-me cloud is used to enable VM migration and when it is not used. When the follow-me cloud is not used, the ping latency remains equal to 50ms. The initial 150ms high latency is mainly attributable to OpenFlow rules when the new traffic arrives at the controller. In contrast, when the follow-me cloud is used, the ping latency drops to 1ms and that is around 32 s after the start of the experiment. This is mainly due to the fact that the VM was dynamically shifted to the optimal data center following the movement of the client. It shall be noted that during the VM migration, few ping losses were noticed.

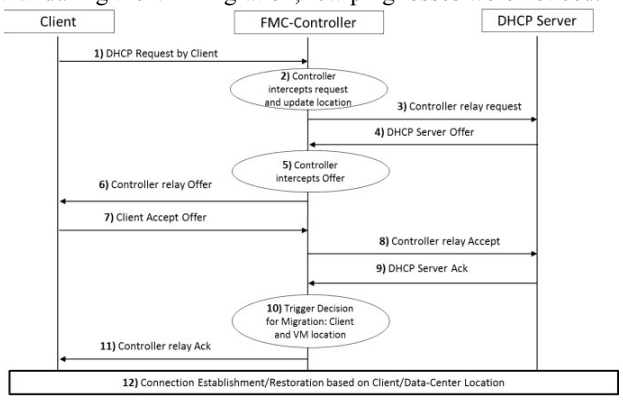

Fig. 4. Intercepting DHCP Packets for Location Mapper.

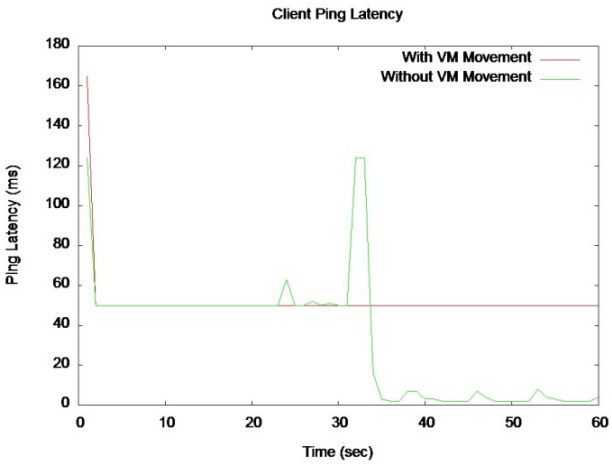

Fig. 5. Client ping latency with and without FMC.

One major use case of follow-me cloud is its implementation in mobile networks. As such networks have traditionally large user bases, scalability of follow me cloud up to many millions of users is a must. The introduced follow-me cloud implementation uses OpenFlow to enable movement of services following that of users. As every moving end-point needs certain OpenFlow rules to map between identifier and locator of the end-point, the size of the rule set depends on the number of moving end-points. With millions of users and services, the rule set is beyond the capabilities of current OpenFlow-enabled switches. But the build-up of networks from multiple switches inherently provides a distribution of end-points to switches and therefore a distribution of OpenFlow rules. The set of rules pertaining to a particular switch is therefore a fraction of the overall rule set. Our analysis shows that the number of rules per switch is within the limits of currently available hardware [14]. Together with the OpenFlow rules on individual switches, the management of the rules at the follow-me cloud con-

troller is an issue for scalability, as the controller has to manage the rules of multiple switches. For large networks, it is essential to realize a distributed controller in order to deal with the large size of the rule set. Distribution can happen across two dimensions, namely network scope and controller role. The network scope refers to assigning certain parts of a network to a particular controller. By narrowing the scope of the assigned network slice, the number of rules managed by a single controller shrinks. In addition, the follow-me cloud system distinguishes three different roles for a controller: home, foreign, and correspondent. Different sets of rules and knowledge are required for the different roles. By separating these three roles, a separate controller can be assigned to each of these roles even further reducing the number of rules managed by a single controller. A more detailed assessment of the scalability of our follow-me cloud controller can be found in [14], opening up new challenges for the community of OpenFlow researchers.

V. Conclusions and Future Work

Cloud computing and mobility have been two major trends of the last few years and are expected to grow in importance over the coming years. Together they will be an important part of the future computing and communications infrastructure. The follow-me cloud concept introduced in this paper combines the two trends and shows how they can interact and bring benefits to mobile users by allowing service access from data centers optimal to the current location of users. The article showed the feasibility and viability of the follow-me cloud, describing an OpenFlow-based implementation. The described architecture shows how mobility management systems from different domains such as wireless networks and server virtualization can work together to realize new capabilities that are not possible when looking at just a single domain.

Service composition has been a topic of discourse in the computing domain for quite some time and we expect the federation of management systems spanning multiple functional and/or administrative domains to become increasingly important for providing new and improved services in an automated fashion. This paper showed how such a federation between functional (network and compute) domains as well as administrative (different operators of data centers and network segments) domains could join efforts to provide high quality mobile services.

The paper outlines a possible implementation of the follow-me concept that is based on VMware cloud infrastructure technology. A possible extension to this work is to enable such functionality in an open source cloud IaaS platform such as Openstack that supports defacto-standard networking[1] API's for manipulating the tenant based virtualized networks. Given that VM's could be deployed across distributed data centers, another possible direction are to explore inter data center connectivity for managing resources across the data centers under an umbrella of a single logical controller entity. Similarly, for live VM migration the VM management traffic should be tunneled from the source to destination data centers either through shared storage or exploiting live block migration. For the user plane traffic, OpenFlow controllers in both domains should co-ordinate for managing the ongoing session on behalf of the migrated VM. Finally, this functionality should be exposed to higher layer applications in the form of specific API's for VM placement and migration along with user location in the network.

[1] Neutron: wiki.openstack.org/wiki/Neutron

Acknowledgment

The research work presented in this paper is conducted as part of the Mobile Cloud Networking project, funded from the European Union Seventh Framework Program under grant agreement n°[318109].

References

[1] K. Samdanis, T. Taleb, and S. Schmid, "Traffic Offload Enhancements for eUTRAN", in IEEE Communications Surveys & Tutorials journal, Vol. 11, No. 3, Aug. 2012, pp. 884-896.

[2] T. Taleb, K. Samdanis, and F. Filali, "Towards Supporting Highly Mobile Nodes in Decentralized Mobile Operator Networks," in *Proc. IEEE ICC 2012*, Ottawa, Canada, Jun. 2012.

[3] R. Miller, "AOL Gets Small with Outdoor Micro Data Centers," Data Center Knowledge, Jul. 2012.

[4] R. Miller, "Solar-Powered Micro Data Center at Rutgers," Data Center Knowledge, May 2012.

[5] 3rd Generation Partnership Project, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access," TS 23.401 (work in progress).

[6] 3rd Generation Partnership Project, "Multi Access PDN connectivity and IP flow mobility," 3GPP TR 23.861 V1.3.0, Feb. 2010.

[7] 3rd Generation Partnership Project, "IP flow mobility and seamless Wireless Local Area Network (WLAN) offload; Stage 2," 3GPP TS 23.261, Jun. 2010.

[8] T. Taleb, Y. Hadjadj-Aoul, and S. Schmid, "Geographical Location and Load based Gateway Selection for Optimal Traffic Offload in Mobile Networks," in *Proc. IFIP Networking*, Valencia, Spain, May 2011.

[9] T. Taleb, K. Samdanis, and S. Schmid, "DNS-based Solution for Operator Control of Selected IP Traffic Offload," in Proc. IEEE ICC, Kyoto, Japan, Jun. 2011.

[10] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID separation protocol (LISP)," Internet-Draft draft-ietf-lisp-13.txt, IETF Secretariat, June 2011.

[11] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, "Serval: An End-Host Stack for Service-Centric Networking," *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12)*.

[12] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," RFC 4423, May 2006, URL: http://www.ietf.org/rfc/rfc4423.txt

[13] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-Performance Networks," ACM SIGCOMM 2011, Toronto, Canada, August 2011.

[14] R. Bifulco, M. Brunner, R. Canonico, P. Hasselmeyer, and F. Mir, "Scalability of a Mobile Cloud Management System," Workshop on Mobile Cloud Computing, SIGCOM 2012, Helsinki, Finland, Apr. 2012.

[15] M. Steiner, B. Gaglianello, V. Gurbani, V. Hilt, W. D. Roome, M. Scharf, and T. Voith, "Network-Aware Service Placement in a Distributed Cloud Environment," ACM SIGCOMM 2012, Helsinki, Finland, August 2012.

[16] The Architecture of VMWare ESXi (White Paper). www.vmware.com/files/pdf/ESXi_architecture.pdf

[17] VMware VMotion: Live Migration for Virtual Machines Without Service Interruption. www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf

[18] Virtual Machine Identifier – UUID. www.vmware.com/support/ws5/doc/ws_move_uuid.html

[19] T. Taleb and A. Ksentini, "Follow Me Cloud: Interworking Federated Clouds & Distributed Mobile Networks", to appear in IEEE Network Magazine, Sep. 2013.

[20] T. Taleb and A. Ksentini, "An Analytical Model for Follow Me Cloud," in Proc. IEEE Globecom 2013, Atlanta, USA, Dec. 2013.