An Evolution of Mobile IPv6 to the Cloud

Ákos Leiter
Nokia Bell Labs
Budapest, Hungary
akos.leiter@nokia-bell-labs.com

László Bokor
Department of Networked
Systems and Service
Budapest University of
Technology and Economics
Budapest, Hungary
bokorl@hit.bme.hu

István Kispál
Nokia Bell Labs
Budapest, Hungary
Istvan.kispal@nokia-bell-labs.com

ABSTRACT

The disruptions caused by Network Function Virtualization (NFV) and Software Defined Networking (SDN) introduce severe impacts also on mobility management. In this work, we will show how IP-based mobility management, especially Mobile IPv6 (MIPv6), can be redesigned to fit into the new environment. We will present an evolutionary procedure on how MIPv6 can arrive in the cloud. At the end of the process, the unique architecture of MIPv6 can run on the top of Kubernetes and OpenStack while keeping compatibility with the existing standards. Orchestration is also an environmental change, comes with cloud, where MIPv6 should be aligned with closed-loop orchestration support.

CCS CONCEPTS

Networks -> Network Architectures

KEYWORDS

MIPv6, IP mobility management evolution, ONAP, cloud-native, Kubernetes, Openstack, MMaaS

ACM Reference format:

Akos Leiter, Laszlo Bokor and Istvan Kispal. 2020. An Evolution of Mobile IPv6 to the Cloud. In *Proceedings of 18th ACM International Symposium on Mobility Management and Wireless Access (MobiWac'20)*, Alicante, Spain ACM, NY, NY, USA. 5 pages. <https://doi.org/10.1145/3416012.3424633>

1 Introduction

Network users on the move always cause complications in any telco infrastructure as they change their Point-of-Attachments (PoA) while also requiring continuous reachability.

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.

MobiWac '20, November 16, 2020, Alicante, Spain.

© 2020 Association of Computing Machinery.

ACM ISBN 978-1-4503-8119-2/20/11...\$15.00.

DOI: <https://doi.org/10.1145/3416012.3424633>

In an IP network, their IP addresses are also modified according to the new network domain where they move during PoA changes. But a new IP address (even) for the same Mobile Node (MN) causes service disruption as its outside reachability changed. This is the place where mobility management comes into the picture. In this paper, we stick to the IP-based mobility management, whose main advantage is to keep all the operational work of mobility management on IP level (L3). There is no additional need for control messages and procedures implemented in below or upper layers, no matters where the MN arrives in the network. This also paves one of the ways for inter-technology handovers (wire-to-wireless, wireless-to-wireless) where MN can be reached continuously, independently of the used access technology.

However, IP-based mobility management procedures were mainly designed to be a software module for physical infrastructure elements like routers or gateways. With the advent of virtualization and especially with Network Function Virtualization (NFV) and Software Defined Networks (SDN), the overall telecommunication environment has changed or under reconsideration. These trends introduce new elastic networks which drive the architectural redesign of older protocols and node to fit into the latest technical paradigms. Virtual Machines (VM) and containers have become the new building blocks of telecommunication software with their orchestration (e.g., OpenStack and Kubernetes, respectively) where hardware and software components are decoupled. There are new standards (ETSI NFV MANO [7]) whose responsibilities are to be the cornerstone for the new virtualized environments. In this paper, we propose a modern architecture for IP-based mobility management, especially for Mobile IPv6 (MIPv6, [1]), how it can be placed in this new environment while still being compatible with the standardized ways of IPv6/MIPv6 packet transmission. We call the new architecture of MIPv6 as Cloud-Native MIPv6 (CN-MIPv6), which runs on the top of a Kubernetes cluster. The goal of this paper is to show how MIPv6 can evolve to be a Virtual Network Function (VNF) or a Cloud-Native Network Function (CNF). With the modification of MIPv6, we implement Mobility Management as a Service (MMaaS) as the literature calls it.

The remaining sections are organized as follows: Section 2 presents related works. The environment where MIPv6 should fit

are located at Section 3. Section 4 shows the high-level architecture of the adjusted MIPv6 while Section 5 puts the requirements to release it. The control flows of the rearchitected MIPv6 are at Section 6. Section 7 goes into the details of packet flow. Other (minor) modification are suggested at Section 8. Conclusion and future work are presented at Section 9 and 10 respectively.

2 Related works

To our best knowledge, the NFV and cloud-native transformation of MIPv6 has not been examined so far, however, from SDN point of view, there are advances. These makes sense to mention as SDN is one of the drivers of cloud transformation, just like NFV.

Xiaozhou Jia in his master theses developed and SDN-based mobility management protocol for seamless handover on IP level. This realizes the functionalities of MIPv6. *Adam Calabrigo* in his master theses[30] examines the applicability MIPv4 concepts in an SDN environment. He solves MIPv4 triangle routing problem with SDN. *Misumi Hata et.al.* [31] deals with inter-domain mobility management. Their approach solves triangle routing (CN $\leftrightarrow$ MN) problem when mobile user changes its SDN Controller (domain). *You Wang et. al*[32] also propose architecture how mobile IP functionality can look like in SDN-era. They use controllers as location tracker and route optimizer too. *Seong-Mun King et al.* [25] examine how PMIPv6[2] can be put into SDN context. The gain of this approach is to eliminate tunneling overhead while LMA is put on the controller. *Kuljaree Tantayakul et al.*[26] proposed an SDN mobility framework where the functionalities of PMIPv6 can be ensured. *Hao Yu*[27] examines vertical handover capability of PMIPv6 with SDN support. PMIPv6 elements are kept but SDN controller is responsible for tunnel handing. *Syed M. Raza et al.*[29] integrate PMIPv6 to SDN context to where PMIPv6 is not modified at all to the corresponding standards. However, Mobile Access Gateway (MAG) L2 responsibilities are maintained by SDN controller.

3 Context of Cloud-Native Mobile IPv6

3.1 Evolution to Containers with Orchestration

Decoupling software from hardware is a natural evolution in the IT industry. This trend reached the telecommunication industry too, where of course, with several exceptions, this software and hardware reorganization can be taken place. To make a software component modular and hardware agnostic, using virtual machines is a clear and straightforward choice. This brings into a new virtualization layer, and also a new kernel instance must be run for each separated application. The more application we have, the more virtual machines we need, not to mention clustered/redundant applications, where several more VMs required for the same application purpose. After a while, this complexity cannot be handled manually. Intuitively, there should be a framework to handle the management issues. This is where OpenStack comes into the picture [3]. OpenStack ensures

appropriate lifecycle management of the application with their underlying virtual machines even in a declarative way (OpenStack Heat component [4]).

But running separate VMs and their corresponding kernels for every application can be superfluous. Constrainers (e.g., Docker [5]) can be the solution. Here, every needed component of a specific application is packaged together, and the rest of the functionalities are served by the kernel of the underlying host OS. This approach mitigates the number of required kernels, but its costs being more dependent on the specific underlying OS, e.g., not possible to have independent kernel model management. But with all the drawbacks, containers have their place in IT and telecommunication infrastructures as well. The management complexity problem has also appeared with containers. Kubernetes [6] realize the functionality that a (private) cloud orchestrator (OpenStack, VMware) provides for VMs for containers. The basic building block for Kubernetes is Pods constructed by multiple Docker containers, which are coupled together as Kubernetes Services.

Figure 1 - Common architecture of cloud orchestrator and Kubernetes

Figure 1 represents the aggregated architecture of a cloud orchestrator and Kubernetes, where an NFV/CNF-ready telecommunication service should run. In this case, VMs are Kubernetes Worker Nodes. Of course, it is possible to eliminate VMs if Kubernetes Workers are bare-metal implementations. This is the context where our architecture proposal for MIPv6 fits into and details the way how MIPv6 should evolve into this direction. Using OpenStack and Kubernetes as orchestrators have other benefits to this work: the software should only deal with its core functionality. Applications should not maintain database or storage infrastructure, and either OpenStack or Kubernetes provide these.

3.2 ETSI Management and Orchestration framework (ETSI MANO)

The ETSI NFV MANO standard [7] is the framework that enables a well-defined environment where VNFs can be utilized. Mainly ETSI NFV MANO standard tells how the orchestration and management of the entire telecommunication infrastructure in a virtualized environment should be divided into roles and responsibilities. Essential elements of the standard for our work are:

- NFV Orchestrator (NFVO): central network service orchestrator;
- VNF Manager (VNFM): responsible for lifecycle management of Virtual Network Function (VNF);

- Virtual Infrastructure Manager (VIM): in our case, it's OpenStack or Kubernetes.

These have well-defined interfaces too, which are requirements for telecommunication services interoperability. Network Service Descriptor (NSD) and VNF Descriptor (VNFD) are responsible for defining elements declaratively.

At the time of writing this paper, ETSI standards do not differentiate between VNF and CNF, so we regard Kubernetes-based services as another VIM zone. However, we think that there will be differences of CNF as at least typical orchestration is needed for OpenStack and Kubernetes, depicted in Figure 1, to release a CNF. Furthermore, Open Network Automation Platform (ONAP) [8], which deals with ETSI NFM MANO implementations too, has proposals for CNF-ready orchestration at [9], but this discussion is out of the scope of this paper.

4 High-Level Design of CN-MIPv6

CN-MIPv6 separates control and data plane. Also, the data plane is divided into two parts: generalized and specialized data plane. The control plane service of CN-MIPv6 (CN-MIPv6-CPS) handles control messages related to bindings. It maintains the Binding Cache Service (CN-MIPv6-BCS), which stores the actual “positions” of MN. Furthermore, as Kubernetes Services are Network Address Translations (NAT), there should be a database for services. As mentioned earlier, this database can be grabbed from the infrastructure (Kubernetes, OpenStack); there is no need to deal with actual infrastructure maintenance from the CN-MIPv6 side.

General Data Plane Service (G-DPS) is responsible for holding the HoA of those MNs which are not involved in flow mobility. The more MN is not involved in flow mobility, the higher number of IP addresses has the G-DPS. The corresponding pods of G-DPS can be scaled out and based on the actual traffic requirements. There is no dedicated POD for MNs, and it's the proposed Kubernetes Network Service responsibility to load-balance between PODs behind the service.

When flow mobility is used, a new Specialized Data Plane Service (S-DPS) must be initialized for every MN. Flow rules are set on the MN side, and such settings also should be placed on the S-DPS side. S-DPS has only one HoA. Its configuration has flow rules as well. Depending on the flow rules, the S-DPS forward packets to the corresponding pods. The higher number of individual flow handling is required, the higher number of Pods needed. Every flow has its dedicated Pod. Of course, it is possible the scale up the number of Pods for a certain IP flow. During high constant high traffic load, it is possible to build up a service mesh around S-DPS. Level 1 S-DPS (L1-S-DPS) holds HoA, and then it forwards packets to the Level 2-S-DPS (L2-S-DPS), which is responsible only to serve one IP flow and can scale up to the required number of Pods to help traffic. With these two levels approach, flow traffic can be spread across the whole Kubernetes cluster. Figure 2 depicts the high-level design of CN-MIPv6. The yellow line represents the control plane traffic, while the green striped lines are the options for the G-DPS or S-DPS traffic path. Of course, if the two-layer is

not needed due to some reason, there is no need for it, one layer is enough. This is one of the flexibility advantages of service meshes and cloud-native design. The reason why G-DPS and S-DPS should be separated is scalability: in this manner, the number of required Pods and KNS can be decreased. G-DPS and S-DPS services are intended to be policy routing engines that manage netfilter[10] and iproute2[11] according to the needs of MNs.

Figure 2 – Two-layer high-level design of CN-MIPv6

5 Building Blocks

Kubernetes Services[12] is an abstraction layer above Kubernetes Pods to hide the effects of the mortality of Pods whose corresponding IP address changes. A Kubernetes Service IP is fixed and permanent which ensures constant reachability of a specific set of Pods. Kubernetes Services is the frontend of a Kubernetes-based application. IP address changes can be frequent as Pods and Application are typically coupled together as a Kubernetes Deployment[13] which can recreate, heal, scale, etc. PODs. As Kubernetes Services has a permanent IP address, it can be used to represent MIPv6 HoA. But MIPv6 protocol family requires more from an ordinary Kubernetes Services (stated at HLD, Section 4):

- Multiple IP addresses assigned to a service
- Flow identification

These are the new requirements that claim a new type of Kubernetes element: Kubernetes Network Service (KNS).

ORIGINAL SERVICE	KUBERNETES	KUBERNETES NETWORK SERVICE
<pre> apiVersion: v1 kind: Service metadata: name: my-service spec: selector: app: MyApp ports: - protocol: TCP port: 80 targetPort: 9376 </pre>		<pre> apiVersion: v1-kns kind: NetworkService metadata: name: HA-S-DPS-MNx spec: selector: app: HA-S-DPS-MNx HoAs: - ip: <HoA-of-MNx> flows: - flow: - srcIP: <CoA> - dstIP: <CN> - proto:<L4proto> - srcPort:<srcport> - dstPort:<dstport> </pre>

Table 1 - New requirements for Kubernetes Services

These modifications create new requirements for vehicular and telco clouds too. Implementing these changes with their

corresponding Container Network Infrastructure (CNI) is our highly prioritized future work but requires a separate paper. Table 1 proposes YAML-based example and comparison to KNS.

6 Control Flow

Control flow has two parts: when using flow mobility or not. Figure 3 represents both parts: the red part when flow mobility is not requested; the green part rectangle shows flow mobility usage.

Figure 3 - High-level control flow of CN-MIPv6 binding registration

In the case of flow mobility usage, S-DPS service is needed to be initialized for a new MN registration. Without flow mobility, only the G-DPS service should be updated, according to Section 5. Even though flow mobility is not needed, but separate S-DPS service is necessary for a specific MN, e.g., some policy reason, then the KNS can be created by a flow range, where every L4 port and TCP and UDP are covered. The MIPv6 RFC[1] has specified some reserved field in Mobility Header, so these can be used to align MIPv6 to this architecture. One of the definite modifications is that, HA has two addresses, control plane address and data plane address (HoA), so the software which runs on MNs should be updated too.

7 Low-Level design of CN-MIPv6

This section is intended to represent the detailed packet flow of the MIPv6 data plane. Figure 4 shows the two directions of packet flows; one originated from MN, the other, directed to the MN. As Kubernetes services are not, the tunnel IP addresses are changed based on the direction of the packets.

To make the representation easier, G/S-DPS are represented twice, based on the packet direction, but they are the same. In essence, packets go into the Kubernetes Services and reach the same Kubernetes Services.

8 Additional modification for CN-MIPv6

This section is intended to describe what other constraints a cloud-native application has to be appropriately aligned to the cloud infrastructure. This section is also a contribution to the Future Work, Section 10. In overall, these thought leads to the overall questions and insights of cloud-native telecom platforms [21][22][23].

CN-MIPv6 is a cluster of multiple PODs on multiple VMs. VNF has health check interfaces, so there should be a new management interface assigned to CN-MIPv6. The implementation of health check must also consist of a decision-making algorithm to decide whether CN-MIPv6 is healthy or not. This also connects the VNF Event Stream (VES) [14] implementation of VNFs. VNF is lifecycle managed by VNFM, so these lifecycle interfaces should be added to the VNF design. Network Services and VNFs are described by NSDs and VNFS respectively, so CN-MIPv6 should be a declarative description method according to the ETSI MANO NSD/VNFD standards [15]. If the number of involved MN reaches a certain level, it is beneficial to store KNS information in a separate distributed database with Kubernetes DaemonSet [16] where a DB instance can be stored on every node. Also, interfaces between CN-MIPv6 elements and databases should be defined. REST APIs should also be added to CN-MIPv6 components to make them easily configurable, which are docker images.

Figure 4 - Low-Level Design of CN-MIPv6

9 Conclusion

In this work, we presented a concept on how to put MIPv6 into a cloud-native environment. This is not just about containerize MIPv6; it's about the complete ecosystem that cloud concept introduced with its microservice basis. However, we focused on OpenStack and Kubernetes' environment; the proposed steps can be aligned to other cloud infrastructure implementations. If a cloud has Kubernetes interfaces, this overall CN-MIPv6 will be portable; public clouds such as AWS, Google Cloud, Azure have already placed Kubernetes APIs. Using Kubernetes Deployments also a step closer to have self-healing capabilities of CN-MIPv6. At the time of writing this paper, ONAP does not have well-defined CNF interfaces, but we showed that MIPv6 could be container-ready, thus can be integrated into the ONAP environment too. Being able to separate each flow and assigning individual Pods to them might be costly but paves new ways of application and network policy alignment. Furthermore, this is the bases to make MIPv6 scalable and even distributed by assigning policies where to run a specific POD. which can also increase redundancy. The only required modification from the infrastructure is to create the new Kubernetes Network Service, mentioned in Section 5, which is our priority future work.

10 Future Work

First, we will implement KNS, and then we will design CN-PMIPv6 and also corresponding protocols like NEMO[24] can be examined in cloud-native environment. Also, mip6d-ng[17] is an excellent candidate to use as a basis for CN-MIPv6. Furthermore, mip6d-ng and its redesigned “cn-mip6d-ng” are also suitable for performance comparison purposes.

ACKNOWLEDGMENTS

This work was supported by FIEK_16-1-2016-0007 (Center for University-Industry Cooperation).

REFERENCES

- [1] D. Johnson, C. Perkins, J. Arkko: RFC 3775 - Mobility Support in IPv6
- [2] S. Gundavelli, Ed et al.: RFC 5213 - Proxy Mobile IPv6
- [3] Openstack: <https://www.openstack.org/>
- [4] Openstack Heat: <https://wiki.openstack.org/wiki/Heat>
- [5] Docker: <https://www.docker.com/>
- [6] Kubernetes: <https://kubernetes.io/>
- [7] ETSI MANO architecture: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
- [8] ONAP: Open Network Automation Platform, <https://www.onap.org/>
- [9] ONAP ETSI CNF support proposals: <https://wiki.onap.org/display/DW/ETSI+CNF+Support>
- [10] Netfilter: <https://www.netfilter.org/>
- [11] IProute2: <https://mirrors.edge.kernel.org/pub/linux/utils/net/iproute2/>
- [12] Kubernetes Service: <https://kubernetes.io/docs/concepts/services-networking/service/>
- [13] Kubernetes Deployments: <https://kubernetes.io/docs/concepts/workloads/controllers/deployments/>
- [14] VNF Event Stream: <https://wiki.opnfv.org/display/PROJ/VNF+Event+Stream>
- [15] ETSI GS NFV-SOL 001
- [16] Kubernetes DaemonSet: <https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>
- [17] mip6d-ng: <http://mip6d-ng.hit.bme.hu/linux-source-codes/>
- [18] Wolfgang Kiess et al.: Ultra-reliable low latency services: 5G architecture and operational alternatives with cost analysis 2019 IEEE Wireless Communications and Networking Conference (WCNC)
- [19] Wolfgang Kiess et al.: 5G via evolved packet core slices: Costs and technology of early deployments, 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)
- [20] J.Varga, A.Hilt, J.Bíró, Cs.Rotter, G.Járó, "Reducing operational costs of ultra-reliable low latency services in 5G", Infocommunications Journal, HTE, Vol.X, No.4., pp.37-45, ISSN 2061-2079, Budapest, Hungary, December 2018
- [21] Csaba Rotter, János Illés, Gábor Nyíri, Lóránt Farkas, Gergely Csatári, Gergő Huszty: Telecom Strategies for Service Discovery in Microservice Environments, Innovation on Clouds, Internet and Network (ICIN) 2017
- [22] Duc-Hung Luong, Huu-Trung Thieu, Abdelkader Outtagarts, Yacine Ghamri-Doudane: Predictive Autoscaling Orchestration for Cloud-native Telecom Microservices, IEEE 5G World Forum, 2018
- [23] J. Fontenla-Gonzalez, C. Prez-Garrido, F. Gil-Castieira, F. J. Gonzalez-Castao, C. Giraldo-Rodriguez, "Lightweight container-based OpenEPC deployment and its evaluation", NetSoft Conference and Workshops (NetSoft) 2016 IEEE, pp. 435-440, 2016.
- [24] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert: RFC 3963 - Network Mobility (NEMO) Basic Support Protocol
- [25] S. Kim, H. Choi, P. Park, S. Min, and Y. Han, "OpenFlow-based Proxy mobile IPv6 over software defined network (SDN)," in 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), 2014, pp. 119–125.
- [26] Kuljaree Tantayakul, Riadh Dhaou, Beatrice Paillassa: Impact of SDN on Mobility Management, ICAIMA 2016
- [27] Hau Yu: Software Defined Networking in Heterogeneous Radio Access Networks, Terena Networking Conference, TNC 2014
- [28] Xiaozhou Jia: SBMP: An SDN-based Mobility Management Protocol to support seamless handover, The University of Tokyo, 2015, https://repository.dl.itc.u-tokyo.ac.jp/?action=repository_action_common_download&item_id=7321&item_no=1&attribute_id=14&file_no=1
- [29] Syed M. Raza, Dongsoo S. Kim, DongRyeol Shin, and Hyunseung Choo: Leveraging Proxy Mobile IPv6 with SDN, ACM IMCOM 2014
- [30] Adam Calabrigo: SD-MCAN: A software-defined solution for IP mobility in Campus Area Networks, Faculty of California Polytechnic State University, San Luis Obispo, <https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=3085&context=theses>
- [31] Misumi Hata, Satoru Izumi, Toru Abe, Takuo Suganuma: A Proposal of SDN Based Mobility Management in Multiple Domain Networks, Network Operations and Management Symposium (APNOMS), Asia-Pacific
- [32] You Wang and Jun Bi: Software-Defined Mobility Support in IP Networks, The Computer Journal, Volume 59, Issue 2, February 2016, Pages 159–177,