On using the Edge Application Server Discovery Function to enforce Edge Computing in 5G Networks and Beyond

Giulio Carota
EURECOM
Sophia Antipolis, France
giulio.carota@eurecom.fr

Karim Boutiba
EURECOM
Sophia Antipolis, France
karim.boutiba@eurecom.fr

Adlen Ksentini
EURECOM
Sophia Antipolis, France
adlen.ksentini@eurecom.fr

Abstract—**Multi-Access Edge Computing (MEC) is a key technology in the field of telecommunications and computing. It brings computing and storage resources closer to the edge of the network, typically at or near base stations and hence reduces the access latency of User Equipment (UE) to applications hosted at the edge. However, mobility of UEs brings challenging issues for service continuity and Service Level Agreement (SLA) fulfilment of 5G services. To solve these issues, 3GPP introduced a new Network Function (NF) called the Edge Application Server Discovery Function (EASDF) [1]. The latter aims to support session breakouts by dynamically resolving the Domain Name Service (DNS) of MEC applications to application servers closer to the UE's physical location. However, the 3GPP specifications [1] do not provide details about how the EASDF handles the UE's mobility. To fill this gap, we propose a novel design and implementation of the EASDF on the top of OpenAirInterface (OAI) open-source 5G network [2]. Simulation results show the efficiency of the EASDF in reducing the access latency during the UE's mobility with a small overhead of less than 4ms in high-load scenarios.**

I. Introduction

Multi-Access Edge Computing (MEC) is a new network architecture that brings computation and storage resources closer to User Equipment (UE), which can significantly reduce latency [3]. This is done by deploying MEC servers at the edge of the network near the base stations. MEC servers can then offload tasks from UEs, such as processing data or running applications. Mobility is primordial for many MEC applications. For example, a MEC-enabled streaming application needs to be able to move the user's streaming session between different MEC servers as the user moves around. This is because the user needs to maintain a low latency connection to the streaming server in order to have a good Quality of Experience (QoE) [4]. In MEC mobility scenarios, 5G networks must offer the ability to influence the User Plane (UP) to properly redirect and offload traffic to edge applications while meeting Quality of Service (QoS) requirements [5][6]. The 3GPP standards propose several methods to achieve this goal. Among them, the Uplink Traffic Classifier [5] allows classifying UP packets based on their headers and deciding which uplink tunnel to forward the packet to. This approach allows individual packets to be routed using their internal IP headers without relying directly on IP routing, thus taking advantage of the flexibility of the 5G UP. In Rel. 18, 3GPP introduced the EASDF [1] to support MEC session breakouts. Within the functionality of the EASDF is Domain Name Service (DNS) resolution to the UE, thus resolving to application servers closer to the UE's physical location.

In this paper, we introduce a framework that leverages the EASDF to allow users to discover the IP of the MEC application instance and integrates this architecture on top of the 5G UP to dynamically redirect traffic while preserving all UE SLA requirements by taking advantage of what the 3GPP specifications offer.

The main contributions of this work are manifolds:

- We propose a novel architecture that enables MEC mobility leveraging the EASDF.
- We implement the EASDF on top of OAI 5G open-source implementation.
- We tested the MEC mobility under high-load scenarios to show the efficiency of the EASDF in supporting session breakouts.

The rest of the paper is organized as follows: Section II introduces the required background to understand the paper's contribution and the key works in the State of The Art (SoTa). Section III introduces the proposed framework. Section IV tackles the EASDF implementation details. Finally, Section V presents the performance evaluation results.

II. Background

A. MEC

Multi-access Edge Computing (MEC) [7] is an industry initiative within the European Telecommunications Standards Institute (ETSI) that aims to bring computing capabilities closer to the network edge. This enables low-latency and high-bandwidth applications, such as augmented reality, IoT, and content delivery, by leveraging edge computing resources. ETSI MEC defines standards for the architecture, interfaces, and Application Programming Interfaces (APIs) to enable interoperability and efficient deployment of edge computing services in telecom networks.

Edge computing in 5G introduces mobility challenges due to the dynamic nature of mobile devices. Seamless handovers, low-latency adaptation, dynamic resource allocation, congestion management, secure authentication, and consistent Quality of Service (QoS) are key concerns as devices move, impacting the efficient functioning of edge computing in a mobile environment. Exploiting the capabilities offered by the 5G Network, it becomes easier to reduce the impact of the devices' mobility on edge computing performances.

B. 5G Core Network

The 5G Core Network [5] meets the needs of different use cases that require low-latency communication, high bandwidth, reliability and seamless integration with the network. In the 5G Core Network standard, 3GPP defines a cloud-native, service-based architecture for all network functions that make up the core network. The 5G Core architecture embraces the Control and User Plane Separation (CUPS) paradigm, simplifying the single component (Network Function) architecture and thus the deployment in cloud-based scenarios. Communication between different components is also simplified by the use of interfaces based on open and standard services.

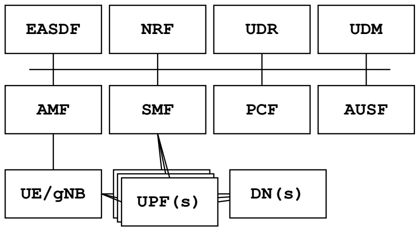


Figure 1: 3GPP 5G System Architecture

In edge computing scenarios, 5G networks must offer the ability to influence the User Plane (UP) to properly redirect and offload traffic to edge applications while meeting Quality Of Service (QoS) requirements. The 3GPP standards propose several methods to achieve this goal. Among them, the Uplink Traffic Classifier [5] allows classifying UP packets based on their headers and deciding which uplink tunnel (UL) to forward the packet to. This approach allows individual packets to be routed using their internal IP headers, without relying directly on IP routing, thus taking advantage of the flexibility of the 5G user plane. These forwarding rules can be applied dynamically via the Policy and Charging Function (PCF) when a new application is deployed or when a new compute cluster is available. The Session Management Function (SMF) will then translate these rules into Packet Detection Rules (PDRs) and Forwarding Action Rules (FARs) that will be leveraged by the UP to detect packets and decide where to forward them based on user-defined rules. However, despite the ability to influence the traffic and to handle how the packets are routed towards the application. The 5G CP or UP themselves do not offer the UEs the possibility to discover edge applications. This task can be accomplished at the application layer, using, for example the Domain Name System Servers.

C. DNS Resolution

Domain Name System (DNS) [8] resolution is a fundamental process in computer networks that translates common application names into IP addresses that are used by machines to communicate with each other. DNS resolution is a vital component of the Internet that ensures users can easily access Web sites and services using easy-to-understand names rather than having to remember IP addresses that may also change over time. The DNS system is based on a hierarchical structure of DNS servers and local caches. Because of the presence of the caches, a Time To Live (TTL) is added to DNS records to control how long a cached record remains valid. Clearly, the TTL plays a critical role, particularly in applications where the mapping between the application name and IP changes frequently. When an MEC application is deployed, it is exposed to other applications and users through the MEC platform, which allows the domain names of those applications to be discovered. In addition, a DNS server is configured during the instantiation phase to store the mapping between the application name exposed through the MEC platform and the application instance. A DNS server can support different types of records depending on the needs of the use case. Each record has a so-called TTL (Time-To-Live) because, since the DNS service relies on a DNS cache system between the client and the most authoritative server, it is important to define an expiration time for entries. The TTL impacts the propagation time of any change in DNS records and the amount of DNS traffic in the network.

D. Related works

AKAMAI CDN [9] uses DNS to point the client to the best Point of Presence (PoP); however, their work remains general and does not explain how to fully integrate this paradigm in a 5G System explain the advantages of the 5G User Plane. ETSI Enhanced DNS [10] concept still remains generic and does not give recommendations on how the full framework should be implemented. 3GPP Traffic Influence service [11] describes how to dynamically change the Packet Data Unit Session anchors while maintaining service continuity from the user perspective. However, this technical specification only figures out how to influence the UP to deliver the packets to the application but does not explain how the UE should know that it is associated with a new application instance or, more generally, how to handle the steering from L3 and higher layers perspective. Another solution, often employed in edge computing scenarios, is based on the Anycast IP. The Anycast IP is a unique and virtual IP that refers to all the instances of a specific application. With this solution, the complexity is moved towards the 5G layer since it is now up to the 5G UP to handle the association between the UE and destination instance and to eventually apply destination NAT rules to substitute the destination anycast IP with the real application IP. APIs to handle the 5G UP dynamically may be more complex to operate for a vertical and are also not often available. Finally, another solution is the DNS-based steering, but using a central DNS Server. This method

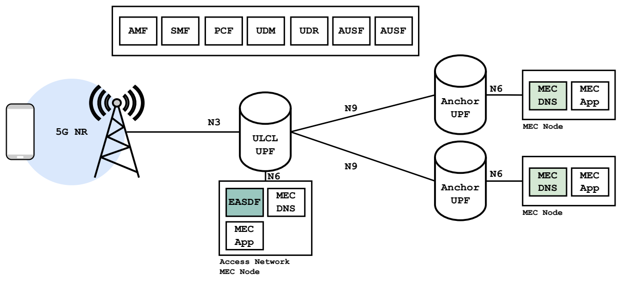


Figure 2: Dynamic Edge Traffic Steering Framework

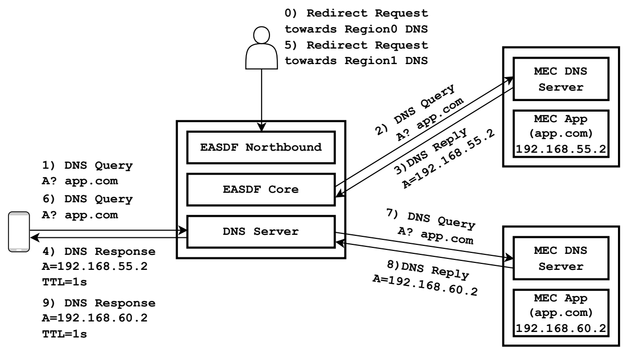
exploits a centralized framework and requires a single DNS Server to manage both the association between the UE and the associated application instance and the DNS records for each edge node. Our solution borrows the idea of DNS-based steering and exploits a decentralized knowledge of the DNS records in order to scale the infrastructure more easily.

III. Dynamic Edge Traffic Steering Framework (DETSF)

Our framework DETSF leverages a DNS-based system to allow UE to discover the IP of the application instance and integrates this architecture on top of the 5G UP to dynamically redirect traffic while preserving all UE slicing requirements by taking advantage of what the standards offer. It addresses two main issues: how the UE discovers the Edge Application IP Address and how the user plane is configured to deliver the UE traffic based on the destination IP.

A. DETSF: how the UE discovers the MEC Application?

The first problem is addressed using the Edge Application Server Discovery Function (EASDF). The EASDF follows the latest 3GPP standard in terms of interfaces and procedures [1]. This makes it easy to integrate this component with any 3GPP-compliant CN and allows verticals to influence traffic forwarding decisions using the EASDF NF. The system acts as a DNS server for UE, handling all their DNS requests and managing them according to instructions given by other NFs or third-party application functions. In our scenario, the EASDF is used to forward the DNS request to another DNS server that is selected from the MEC DNS servers of different MEC nodes. The local DNS of an MEC platform contains only the records of the applications deployed in that node. It is important to note that the DNS name of the application must be the same in the different edge nodes for the system to work. With the EASDF, it is possible to discover the IP address of an application deployed at a specific MEC node by forwarding the UE's DNS request to that node's DNS server and then sending the DNS response back to the UE. From the UE's perspective, the result is that the same DNS name can be translated into different IP addresses depending on how the request is forwarded. EASDF's forwarding rule can be dynamically updated from its northbound interface [1] to change the association between a node and an IP address. In fact, EASDF provides a northbound interface that allows rules to be installed for each user's equipment [12]. An example of the complete EASDF workflow is shown in Figure 3.



0) Redirect Request towards Region0 DNS
5) Redirect Request towards Region1 DNS
EASDF Northbound
EASDF Core
DNS Server
1) DNS Query A? app.com
6) DNS Query A? app.com
4) DNS Response A=192.168.55.2 TTL=1s
9) DNS Response A=192.168.60.2 TTL=1s
2) DNS Query A? app.com
3)DNS Reply A=192.168.55.2
7) DNS Query A? app.com
8)DNS Reply A=192.168.60.2
MEC DNS Server
MEC App (app.com) 192.168.55.2
MEC DNS Server
MEC App (app.com) 192.168.60.2

Figure 3: An example of EASDF Workflow

In this framework, the EASDF is implemented within an edge node of the access network that intercepts the DNS request near the UE without adding unnecessary latency and without impacting the performance of the UP. The implementation of this feature in the access MEC Cluster makes it possible to exploit the user and network information exposed by MEC services such as the Radio Network Information Service (RNIS) [13] and the Location API [14] to determine the best DNS forwarding rule of the EASDF (i.e., associate the UE with the best MEC application instance based on these factors). The component is integrated on top of OAI, including the Radio Access Network and the Core Network.

B. DETSF: how the UE reaches MEC Application?

The way packets are routed in the 5G UP is not as simple as it is for the Internet. Indeed, the simplicity of IP routing is replaced by a more complex packet identification and tagging system that favors traffic engineering and QoS enforcement. This means that the destination IP of the application we want to communicate with is not sufficient to determine the path of the packet and how it should be routed to reach the destination. However, thanks to UpLink-CLassifiers (ULCL), it is possible to detect packets from their IP headers and apply specific FAR. Finally, by combining ULCL with a Multi-Anchored PDU session, it is possible to have several breakout interfaces to different Data Networks (DNs) within a single PDU session. This feature is used in our framework to provide an IP-like routing scenario without losing the Traffic Engineering benefits provided by the 5G UP. In addition, the ability to dynamically distribute UP functions at the edges and dynamically modify IP-based forwarding rules to create local traffic breakout interfaces at the edges can introduce high degrees of freedom for the network. The combination of ULCL-capable UPF, the PCF NF used to manage the traffic rules, and the anchor UPFs deployed at the edge of the different DNs constitute the lower layer of the proposed framework.

Figure 4 shows a small example of how the traffic can be redirected to different DNs by using a ULCL Classifier and traffic rules enforced via the PCF. This work leverages the

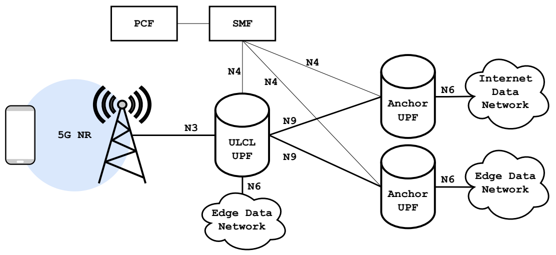


Figure 4: Traffic Offloading using ULCL

OAI 5G Core Network [15], which is a full-working, 3GPP Rel. 15 compliant implementation that offers flexibility and the possibility to add and integrate new features thanks to its open-source nature.

IV. Edge Application Server Discover Function (EASDF)

A. EASDF System design

The EASDF component implemented in this work is cloud-native and can run on any container-based system. One of the key requirements is resilience to multiple and parallel requests due to the large number of UEs that can leverage a single instance of EASDF. However, the cloud-native design allows the application to scale to cope with the limitations of the single instance. EASDF can be deployed in a distributed fashion over the access network, handling a smaller subset of UEs and also providing lower latency. The UE can access the EASDF via a distributed local breakout UPF in the access EDGE and leverage ULCL-based data plane distribution to access applications on different edge nodes.

The EASDF is mainly composed of three functional blocks:

1) 3GPP Standard Northbound Interface: The northbound interface implementation relies on the 3GPP EASDF definition [12]. It allows network functions belonging to the same Public Land Mobile Network (PLMN) or Vertical Application Functions to install DNS forwarding rules on a per-UE basis. Since the EASDF can be adopted by verticals to handle redirection of UE traffic, our implementation is CAPIF-compatible[16] and offers the ability to be discovered and exploited by third-party entities without introducing additional security concerns. This design choice increases the robustness and observability of the interface, but it also simplifies the use of the service from the vertical perspective.

2) EASDF Core Application: The core component of EASDF is used to process incoming DNS requests and forward the contents of the request to the destination DNS server, which is selected based on information from the northbound interface. The Core application can also process the response from the destination server and possibly modify the fields contained in the DNS response.

3) UDP/TCP DNS Server: The DNS server receives requests from the user equipment, which are then processed by the EASDF Core application. The requests are handled and processed in parallel to enable low-latency resolution, which can be crucial in edge computing scenarios. The data contained in and extracted from the DNS query is sent to the EASDF Core Application, which will forward the requested content to the destination server defined in the DNS context of the specific client.

B. EASDF implementation

The system implementation is based on a Python application comprising three different components, as described in the previous section, which run on different threads. Thus, the minimum number of threads required by EASDF is 3 when no clients interact with the application. Starting with the Northbound, the server was initially generated using the OpenAPI document [12] released by 3GPP that describes all the services exposed by EASDF and their different interfaces. The neasdf-dns-contexts service is one of the key services. It allows the creation of a DNS context related to a client-specific IP, slice, and PLMN. When a new DNS context is created, for each DNS request that corresponds to the triplet (UE IP, Slice ID, and PLMN ID), EASDF must apply the actions and rules described in the body of the request. Within a DNS context, a rule can be described as the following pseudo-Json:

```
"dnsRules":
  "rule1":
    "dnsRuleId": "rule-8ff938f"
      "actionList":
        "dnsFwd0":
          "applyAction": "FORWARD"
          "fwdParas":
            "dnsServerAddressInfo":
              "dnsServerAddressList":
                "ipv4Addr":"192.168.121.50"
```

The neasdf-dns-contexts service can then be exposed and published using the Common API Framework, allowing EASDF to be discovered by application functions or the vertical itself that needs to interact with it. Another key block is the Core Application, which handles client requests and processes them based on information gathered from the northbound service. For each incoming request, a new thread is created to handle it and wait for the response from the destination DNS. Implementing this function using threads is necessary, otherwise a blocking query could slow down other clients or parallel queries. The body of the DNS query is encapsulated in a new UDP packet and sent to the destination DNS, which in this use case corresponds to a local DNS server deployed in an MEC node. At this point, the thread starts waiting for the DNS response. Once the response containing the local IP of the application instance deployed in the target MEC node is received, the EASDF Core Application proceeds to process the DNS response and modify the TTL field. This implementation exploits the modification of this field to impose a low Time To Live and prevent DNS clients from storing information for long periods. This step is critical because, in high mobility scenarios, the edge of the target MEC can change very frequently, and therefore DNS information must be updated very quickly.

V. Performance Evaluation

In the balance of this section, the simulation environment and parameters will be introduced. The aim of this evaluation is to understand the limits of the implementation when facing a growing request rate from multiple clients. To analyze the impact on the EASDF implementation, three different parameters are taken into account:

1) The end-to-end latency from the client perspective between the moment the DNS Query is sent and the moment the query is received. The results are compared with the ones obtained by queering directly the target DNS Server in order to visualize which is the latency added to the system by the EASDF.
2) The CPU consumption of a EASDF instance in percentage in order to monitor how it can be scaled and optimized.
3) The failure rate of the DNS requests which corresponds to the rate of rejected requests from the server or the requests that have encountered a timeout due to the target DNS.

A. Testbed description

The testbed consists of a physical machine running Ubuntu Linux 18.04.6 LTS and disposing of an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz and 32 GB of RAM. In order to respect the promises of a cloud-native framework, the EASDF has been deployed as a docker container inside this machine, ensuring that it could have access to all the CPU cycles of the 64 Cores offered by the machine and the entire memory without any limitation. Apart from the EASDF component, we deployed aswell a dnsmasq-based DNS [17], which will serve as the target DNS for the EASDF, but also two provide measurements of latency excluding the EASDF from the chain. Finally, a Python script simulates the DNS Clients. The Python script will gradually increase the number of UEs requesting DNS translation. The script is coded in order to maintain a fixed per-UE request rate equal to 1.33 $\mathrm{req/s/UE}$. Increasing the number of UE, the final request rate will increase from 1.33 to the maximum of 1468,33 $\mathrm{req/s}$. The maximum corresponds to the moment in which the CPU usage does not grow anymore due to OS Scheduling policies. The simulation is repeated 100 times for each step resulting in an error of $10^{-2}\mathrm{ms}$

B. Results

Figure 5 and 6 show the impact of introducing the EASDF component between the UEs and the DNS server in terms of latency. The testbed does not take into account the network latency between the EASDF and the destination server.Instead, the EASDF and the target DNS Server are deployed in the same physical machine in order to measure only the latency due to the EASDF processing. Despite the rapid growth of latency with respect to the increase in request rate, we can observe that the added latency grows linearly without any exponential trend. This means that the EASDF can be exploited even at higher request rates and that the reason for the rapid growth of end-to-end latency should be found mainly in the target DNS.

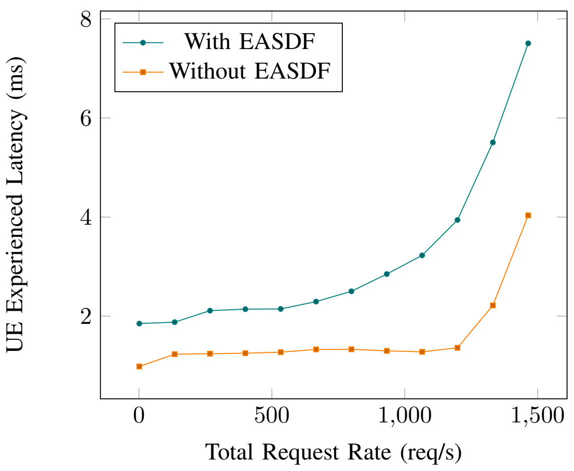


Figure 5: The average latency experienced by the UEs with respect to the total number of request per second.

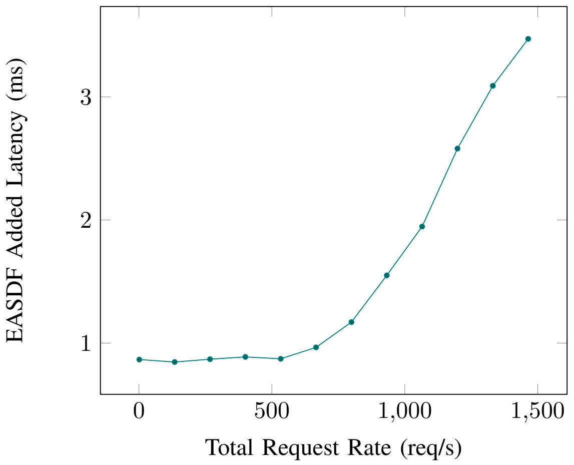


Figure 6: The contribution in terms of latency added by the EASDF

Figure 7 represents the failure rate of the DNS requests sent by the UE towards the EASDF and eventually redirected to the selected destination server. The failure counter is based on the number of requests that do not receive a reply from the EASDF. The reported reasons for a possible failure are the handling thread crashing or a timeout of the DNS Client in the UE. Despite the increased latency, the failure rate in figure 7 does not increase with the growth of the request rate showing that the system is resilient to higher request rates.

Finally, Figure 8 depicts the CPU consumption and its trend as a function of the request rate. The CPU Usage grows linearly with the number of requests per second, proving that the behaviour of the EASDF remains predictable even under higher loads.

All in all, the framework based on the EASDF introduces a highly scalable distributed DNS System that can benefit from UE network information to adapt the DNS replies based on the UE status. A framework that can scale easily since it does

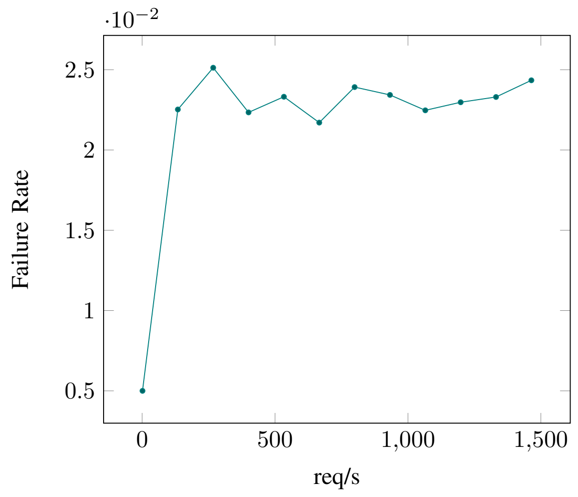


Figure 7: The rate of failed DNS requests with respect to the total request rate.

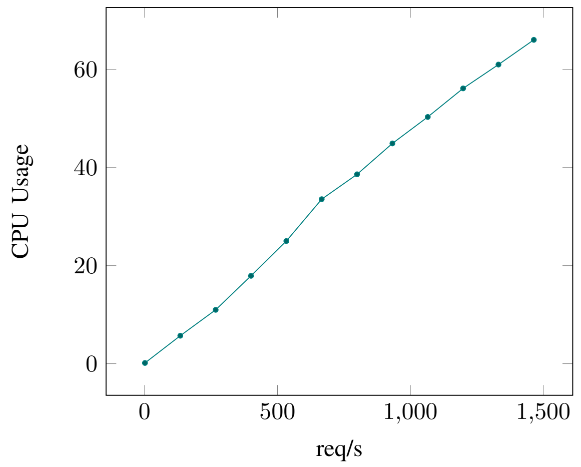


Figure 8: The impact of an EASDF instance on the CPU.

not store the DNS information, but only the forwarding rules to the real DNS. The results presented in this section show that deploying the EASDF in a live network will not impact the baseline performances of the DNS resolution system while enhancing the resolution system with tailor-made rules for each single UE.

Conclusion

This paper introduced a new architecture that exploits the 5G UP, the 3GPP EASDF and the MEC local DNS. The introduced architecture dynamically changes the association between the application domain name and the MEC application instance associated with it on a per UE basis. Future work will focus on making the EASDF smarter by dynamically computing the TTL value according to the UE's mobility pattern and the MEC load balancing requirements.

Acknowledgment

This work is partially supported by the European Union's Horizon Program under the 6GBricks project (Grant No. 101096954).

References

[1] *5G System Enhancements for Edge Computing*. TS23548. Rev. 17.2.0. 3GPP. May 2022.

[2] Florian Kaltenberger et al. "The OpenAirInterface 5G New Radio Implementation: Current Status and Roadmap". In: *WSA 2019; 23rd International ITG Workshop on Smart Antennas*. 2019, pp. 1–5.

[3] Igor Miladinovic et al. "Multi-Access Edge Computing: An Overview and Latency Evaluation". In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. Vol. 1. 2021, pp. 744–748. DOI: 10.1109/ICIT46573.2021.9453495.

[4] Tarik Taleb and Adlen Ksentini. "QoS/QoE predictions-based admission control for femto communications". In: *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*. IEEE, 2012, pp. 5146–5150.

[5] 3rd Generation Partnership Project (3GPP). *Technical Specification 23 501*. Tech. rep. 3rd Generation Partnership Project (3GPP), 2015. URL: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf.

[6] Pantelis A. Frangoudis et al. "An architecture for on-demand service deployment over a telco CDN". In: *2016 IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, May 22-27, 2016*. IEEE, 2016, pp. 1–6.

[7] ETSI. *ETSI MEC Framework Specification*. Tech. rep. MEC 003 V3.1.1. European Telecommunications Standards Institute (ETSI), 2022.

[8] P. Mockapetris. *Domain Names - Implementation and Specification*. Tech. rep. 1035. Network Working Group, Request for Comments, 1987. URL: https://www.ietf.org/rfc/rfc1035.txt.

[9] *Akamai CDN: Reference Architecture*. https://www.akamai.com/resources/reference-architecture/content-delivery.

[10] Masaki Suzuki and al. "Enhanced DNS Support towards Distributed MEC Environment". In: (2020).

[11] *Procedures for the 5G System (5GS)*. TS236502. Rev. 16.7.0. 3GPP. Jan. 2021.

[12] *Neasdf Dns Context*. https://forge.3gpp.org/rep/all/5G_APIs/-/raw/0505a68c3ad134a809c118136a0c2b5f693d8698/TS29556_Neasdf_DNSContext.yaml. 3GPP.

[13] ETSI. *Multi-access Edge Computing (MEC); Radio Network Information API*. Tech. rep. MEC 012 V2.1.1. European Telecommunications Standards Institute (ETSI), 2019.

[14] ETSI. *Multi-access Edge Computing (MEC); Location API*. Tech. rep. MEC 013 V2.2.1. European Telecommunications Standards Institute (ETSI), 2022.

[15] Navid Nikaein et al. "OpenAirInterface: A Flexible Platform for 5G Research". In: *SIGCOMM Comput. Commun. Rev.* 44.5 (Oct. 2014), pp. 33–38. ISSN: 0146-4833. DOI: 10.1145/2677046.2677053. URL: https://doi.org/10.1145/2677046.2677053.

[16] 3GPP. *Common API Framework for 3GPP Northbound APIs*. Tech. rep. 23.222 V16.9.0. 3rd Generation Partnership Project (3GPP), 2020.

[17] Simon Kelley. *Dnsmasq - network services for small networks*. http://www.thekelleys.org.uk/dnsmasq/doc.html. 2020.