Building an Intelligent Edge Environment to Provide Essential Services for Smart Cities

Gayathri Karthick
Middlesex University
The Burroughs Hendon
United Kingdom NW4 4BT
Email: gayuinfy@gmail.com

Glenford Mapp
Middlesex University
The Burroughs Hendon
United Kingdom NW4 4BT
Email: g.mapp@mdx.ac.uk

Jon Crowcroft
University of Cambridge
15 JJ Thomson Avenue Cambridge
United Kingdom CB3 0FD
Email: jon.crowcroft@cl.cam.ac.uk

ABSTRACT

Smart Cities will cause major societal change because they will provide a comprehensive set of key services including seamless communication, intelligent transport systems, advanced healthcare platforms, urban and infrastructure management, and digital services for local and regional government. Thus, a new service and networking environment which will provide low latency and sustainable high bandwidth is needed to build new applications and services for smart cities. In this system services will be managed from the edge of the Internet and not from the centre as they currently are. This represents a new computing paradigm which is called the Intelligent Edge Environment. This paper looks at how to build this new ecosystem. Firstly, a new framework which comprises seven layers is unveiled, showing the functions that must be supported to realise this brave new world. New mechanisms are then introduced and a small prototype is developed to support storage in highly mobile environments. The results show that this approach could be used to build smart city digital platforms. The paper ends by discussing the development of a Distributed Operating System for smart cities.

CCS CONCEPTS

• **Networks** → **Network services**; **Cloud computing**; **Location based services**; **Programmable networks**;

KEYWORDS

Intelligent Edge Environment, Mobile and Vehicular Communications, Capabilities, Service Management Framework, Docker, FUSE, Network Memory Servers, Distributed Operating Systems

ACM Reference Format:
Gayathri Karthick, Glenford Mapp, and Jon Crowcroft. 2023. Building an Intelligent Edge Environment to Provide Essential Services for Smart Cities. In *Workshop on Mobility in the Evolving Internet Architecture (MobiArch'23), October 6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3615587.3615987

MobiArch'23, October 6, 2023, Madrid, Spain

ACM ISBN 979-8-4007-0341-6/23/10.
https://doi.org/10.1145/3615587.3615987

1 INTRODUCTION

The Smart City agenda is now being pursued by numerous companies, institutions and governments because of the huge societal impact it will have. New key services will be developed. Firstly, ubiquitous communication based around the 4As paradigm: anywhere, anytime, anything, and anyhow will be provided. Intelligent Transport Systems (ITS) as well as Advanced Digital Medical Platforms (ADIMEP) will also play a big part in smart cities, along with smart grids, intelligent buildings and smart homes.

In order to fulfil these requirements, it is necessary to move services and servers closer to the user. Mobile Edge Computing (MEC) has evolved over the years to help deliver better services to users by offloading some of the work from central servers. However, to provide much lower latency and higher bandwidth, services must run from the edge of the network by default instead of from the centre of the network. This represents a new computing and networking paradigm which is called the Intelligent Edge Environment (IEE) and is shown in Figure 1.

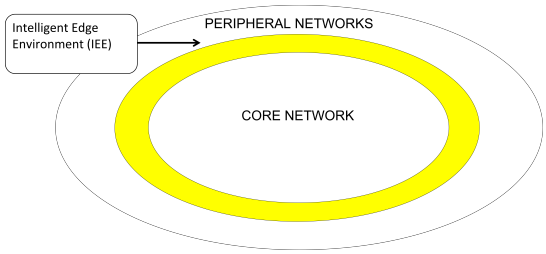

Figure 1: Intelligent Edge Environment

Several resources and mechanisms are required to build the IEE. In terms of resources, High Performance Edge Computing Systems (HPECS), fast storage systems using large Solid States Disks (SSDs), as well as Intelligent Caching Architectures and support for heterogeneous networking must form key components of the IEE. In addition, mechanisms including support for mobile services, where services can be dynamically migrated to support mobile users must also be available. Though migration mechanisms such as virtual machines [3], Docker [1] [21], LXD-CRIU [2], and Unikernels [12] can be employed to migrate services between different Cloud systems, Machine Learning (ML) and AI algorithms are needed to decide where is the best place to run a service at any given time. Furthermore, security protocols are also needed to ensure that servers are not hosted by rogue Cloud systems and that Cloud systems are not damaged by malicious servers. Finally, the IEE needs to support a microservice architecture allowing services to be quickly migrated, and more complex services to be built using other smaller services.

This paper explores the building of the IEE to support the development of applications and services for smart cities. It first looks at a new framework to define the functionality of the IEE. It then discusses the mechanisms needed for this new environment including a Resource Allocation Secure Protocol (RASP), Capabilities and a new Service Management Framework (SMF). Using these mechanisms, a prototype implementation employing a FUSE system with a Network Memory Server (NMS) to provide reliable storage is then detailed.

The contributions of this paper are detailed below:

- A new framework for the IEE is presented.
- A RASP system is developed which has been verified by Proverif.
- A new Capability System for Secure Storage is specified.
- A new SMF is examined.
- A prototype using FUSE and NMS is unveiled and tested.

The rest of the paper is as follows: Section 2 looks at Related Work. Section 3 details the new framework for the IEE while Section 4 examines its key system components. In Section 5, an initial prototype is developed and tested while Section 6 concludes the paper.

2 RELATED WORK

Mobile Edge Computing (MEC) was originally developed as an offloading mechanism to provide more computing resources at the edge of the network. In a survey of architecture and computation offloading in MEC, the authors in [11] explained that the current research being carried out regarding MEC is basically around how to guarantee service continuity in highly dynamic scenarios. In that regard, the authors in [22], proposed a vehicular offloading framework in a cloud-based MEC environment. They were able to investigate the computation offloading mechanism. The latency and resource limitations of MEC servers were taken into consideration which enabled the proposal of a computational, resource allocation and contract-based offloading scheme to be developed. Multi-access Edge Computing also called MEC, emerged as an enhanced paradigm of Mobile Edge Computing to also look at heterogeneous networking as a way of improving access to local services because new networks such as 5G and Fibre-to-the-Home have provided low-latency communications in the local environment. In [15], the authors examined the use of 5G and Wi-Fi networks for MEC; the results showed improved performance.

Along with advances in MEC, there have been considerable improvements in virtual machines and container technologies such as Docker, LXD and Unikernels. In [14], the authors compared the performance of these mechanisms and found that Unikernels had the best performance, especially for microservices. This was also validated in [19]. These technologies, therefore, led to the development of mobile services where services are migrated or replicated closer to the user. In [18], the author unveiled a new framework for mobile services and used it to experiment with determining whether or not a service can be successfully migrated, given the mobility of the user and the time it took to migrate the service [16]. However, this work did not consider security as well as issues of Quality of Service (QoS). Therefore, the authors in [5] looked at deep reinforcement techniques for service migration. In [10], a lightweight edge computing platform was explored using Raspberry Pis. Different types of orchestration techniques were explored. The project validates the approach taken in this paper. Though all these efforts were good and very useful, what is now needed is to combine these techniques as well to add new mechanisms to build a new computing and service environment at the edge of the network which is the focus of this paper.

3 A FRAMEWORK FOR THE IEE

By combining the issues discussed above, it is possible to specify a new framework for the Intelligent Edge Environment which is shown in Figure 2.

3.1 Layers of the IEE

The functions of each layer of the IEE are detailed below:

- Layer 1: Heterogeneous Networking Layer (HNL): A variety of networking technologies, including mobile networks such as 4G, 5G and CV2X as well as vehicular

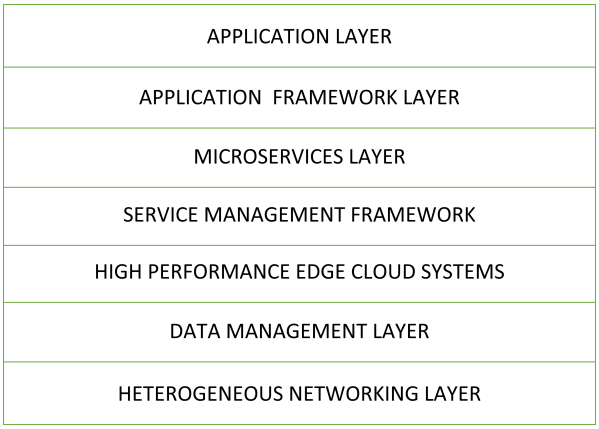

Figure 2: Intelligent Edge Environment Layers

networking technologies such as IEEE 802.11p and IEEE 802.11bd, are supported by the HNL. The HNL will support ubiquitous communication [8] using fast vertical handovers between the various networks and new transport protocols such as SLTP [7].

- Layer 2: Data Management Layer (DML): This layer uses many structures such as blocks, files, and databases to manage data. Individual data blocks are stored in a block storage system, where each block is encrypted for security and replicated for redundancy. For files and databases, meta-data storage is supported. Data is cached or replicated by an Intelligent Caching and Prefetching System to make sure it is always readily accessible. Machine learning (ML) techniques are used to analyse data access patterns.
- Layer 3: High Performance Edge Clouds (HPECS): This layer supports various Cloud-based services and offers processing and computational resources using VM techniques including support for Vmware and Citrix ecosystems. Clouds may also advertise their available resources to servers.
- Layer 4: Service Management Framework (SMF): This layer manages services and servers within the system. It offers mobile service support by migrating and replicating services using various migration techniques (Docker, KVM, Unikernels). Using AI techniques, it also determines the best location to run a service based on the QoS required, the location and mobility of the user as well as available Cloud resources at the edge of the network.
- Layer 5: Microservices Layer (MSL): This layer supports a large number of microservices. Microservices should be fast and small in order to be easily migrated.
- Layer 6: Application Framework Layer (APL): This layer uses the microservices layer to build systems and services for applications. It provides a new mobile virtual environment for processing, storage, GUI, HTTP, eCommerce platforms, etc.
- Layer 7: Application Layer (AL): This layer allows applications that have been built using the Application Framework Layer to be installed on the system and made available to users. Through this layer, users get applications that use all the resources of the IEE.

4 KEY SYSTEM COMPONENTS FOR THE IEE

In order to build the IEE, key system components must be developed.

4.1 New Resource Allocation Algorithm for the IEE

A primary goal is to ensure that services can be quickly migrated or replicated. In this system, Cloud Platforms (CPs), advertise their free resources in terms of CPU, memory, network, and storage. Servers therefore hear these advertisements and based on their own requirements in terms of CPU, memory, network and storage will decide whether there are enough free resources on the advertising CP. If the decision is to migrate the service to the advertised CP, the server will contact the Resource Allocation Server (RAS), also known as the Registry. With this approach, all CPs and services must register with the RAS. Hence, the RAS has data on all the maximum and allocated resources for all CPs as well as the requirements of services and thus the RAS can say whether the CP platform has the necessary resources. Once the server has checked with the RAS that the CP is registered and has the required resources, the server will send a transfer request to the advertising CP.

The CP will look at the server requirements in the transfer request and will contact the RAS to enquire if the server is a valid server. If the RAS indicates that the server is a valid server, the Cloud Platform will send a positive reply to the server's transfer request. The server will then begin to transfer the service to a new Cloud Platform. Each region will have a RAS server; large cities will have several RAS systems which will connected to form a distributed system to improve the scalability of the system.

The interactions between all parties and the transfer of the service are done using a Resource Allocation Secure Protocol (RASP). In RASP, a symmetric session key is used to encrypt and decrypt information to secure the transfer of services to the new Cloud infrastructure. ProVerif [4] is an automated reasoning tool to verify security properties using cryptographic tools. The RASP was developed and tested using Proverif. The results showed that the RASP system was safe. Details of the work are found in [9].

4.2 Capabilities

In any computing environment, issues of authentication, authorisation and accounting must be addressed. Because of the dynamic nature of the IEE where users, devices as well as services can be mobile, traditional AAA mechanisms such as the use of RADIUS Servers are no longer a good solution. It was therefore decided that AAA should be based around the subject or user rather than the object as this seems more scalable in the context of large systems. Thus capabilities are used to provide AAA for the IEE [20].

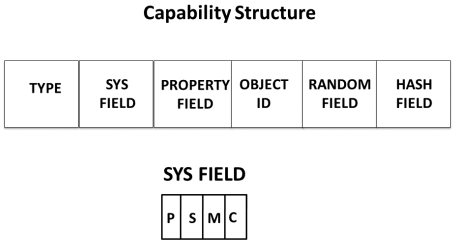

Figure 3: Capability Structure and SYS Field

4.3 Capability Structure

With the IEE, every object and its properties are identified using capabilities. Therefore, it is necessary that capabilities be carefully managed and be protected from being created or modified in an unauthorised manner and should be easily revoked. The format of the capability-based system is shown in Figure 3. The structure is explained below.

- Type Field (8 bits): This field is used to specify the type of object capability that is being used. Types include users, digital assets, facilities, etc.
- SYS Field (4 bits): This field is used to help manage capabilities. The capability related fields are given by four bits which are explained below.
- Property Field (12 bits): This field is used to define the properties of the object associated with the capability. This field is related to properties or functions of the object to which the capability refers.
- Object ID (72 bits): This field is used to uniquely identify the object in the system. A EUI-64 identification field is used to identify the object and a netadmin field (8 bits) is used to manage the object on a network.
- Random Bit Field (16 bits): The random bit field provides unforgeability. This field helps to uniquely identify the object. The random bit field is generated after the type field, SYS field, property field, and Object_ID field are created. When Proxy certificates are created, a new random field is generated. The random field also enables easy revocation of capabilities as this can be done by simply changing the random field and recomputing the capability, hence revoking previous versions of the capability.
- Hash Field (16 bits): The hash field is used to detect the tampering of capabilities. When a capability is created, the type field, sys field, property field, and Object_ID field are first generated, followed by the random bit field. Finally, these fields are used to generate a SHA-1 hash which is placed in the Hash Field of the capability.

4.3.1 *As shown in Figure 3, the SYS FIELD consists of:*

- The Private or P bit: This bit is used to restrict the list of people holding the capability. With a public capability, only the capability for the object must be presented. Because it is a public capability, anyone can hold this capability and hence, the sender's identification is not required. With a private capability, the capability of the object as well as the capability of the subject or user must be presented to ensure that the sender has the right to invoke that object.
- The System or S bit: This indicates whether the object involved has been created by the system, or by an application or user. A system capability cannot be modified or deleted by users or applications.
- The Master or M bit: This bit indicates that the capability was created by a Certificate Authority (CA). The Master capability is usually created when the object is created. If this bit is not set, it means that this is a Proxy capability. Proxy capabilities are derived from Master Capabilities and cannot be derived from other Proxy Capabilities.
- The Change or C bit: This bit is used to indicate whether this capability can be changed or not. This means that if this bit is set, the Proxy Capabilities can be derived from the Master Capability. If this bit is not set, it means that this capability must not be modified and hence Proxy capabilities cannot be generated.

4.4 New Service Management Framework

In order to allow services to migrate as users move around, it is necessary to have a robust service architecture [17], [6]. Hence, a new Service Management Framework (SMF) for mobile services has been developed and discussed in [13]. The layers of the SMF are detailed below:

- Application Layer (AL): This is the first layer of SMF and runs on the mobile node and invokes the service through the Service Management Layer (SManL), giving the Service name, Service_ID and the required QoS. When a service registers with the SMF for the first time,

it will be given a unique ID to identify the service. The service name indicates which type of service is required and the resources needed by the application such as CPU, Memory, or Storage must be specified.
- Service Management Layer (SManL): This layer is the management layer that administers the mobile service and is also responsible for the service subscription and service delivery. Service subscription refers to the Service-Level Agreement (SLA), which is also used for billing and accounting purposes. Service delivery describes how services should migrate from one location to another. Once this layer decides that a service should be migrated, it passes this information to the service migration layer (SML).
- Service Migration Layer (SML): This layer handles the migration requested by SManL and uses the RASP system for secure migration. In turn, the RASP protocol will use standard migration mechanisms such as Docker, KVM, LXD and Unikernels to do the actual migration. SML updates SManL when the migration is completed.
- Service Connection Layer (SCL): This layer monitors the connection status of the clients/application. It reports to the SManL when the mobile node is no longer available due to a handover to another network.

4.5 Microservices - FUSE and NMS

The Network Memory Server (NMS) is a network-based storage server that is used to provide blocks of storage to its clients. These blocks are held in non-volatile memory and thus can be used to provide low-latency, high-bandwidth storage for applications. The NMS is an example of a microservice because it is small and fast, and can therefore be easily migrated or replicated on different servers. FUSE is a library which is used to build user-space file systems. For this work, we have implemented a basic file system using FUSE with the NMS as the back-end providing permanent storage.

5 IMPLEMENTATION OF A PROTOTYPE IEE

5.1 Components of the Prototype

- SMF: The Service Management Framework: The service to be managed must first be registered with the SMF. The SMF is responsible for putting servers in touch with clients who require their service. The SMF will also migrate the service when required.
- The NMS: The NMS is the service that is being managed by the SMF. It is registered as a Block Storage Service. The NMS creates, deletes, reads from and writes to different data blocks on behalf of its clients. The NMS is migrated to different Roadside Units (RSUs).
- FUSE file system: This was built using the FUSE library and runs on the application machine. It contacts the NMS and performs operations on data blocks managed by the NMS.
- Docker: In our implementation, we use Docker as the migration mechanism to migrate the service. A Docker hub has public, private and container repositories that contain a collection of Docker images. We first created an account in the Docker hub. Once we logged in, we created a private repository to store our services and the computing resources to enable migration in Cloud environments.

5.2 Final results of SMF

The initial prototype was tested and showed that the SMF was able to start the service in another machine using Docker. The setup is shown in Figure 4. However, further development, testing and evaluation are currently taking place. Once fully tested, the prototype will be put in the public domain to allow further development.

6 CONCLUSIONS AND FUTURE WORK

This paper has examined a new computing and networking paradigm called the Intelligent Edge Environment (IEE) which is needed to support the development of essential services for smart cities. A new framework was unveiled and new mechanisms have been introduced. A prototype was built which showed that this framework could be used to build real systems. However, we believe that this work also points to the need to explore the development of a Distributed Operating System For Smart Cities so that essential services could be delivered in a secure, efficient, and reliable way for smart cities of the future.

REFERENCES

[1] 2019-08-02. Docker Technology. https://docs.docker.com/engine/reference/commandline/commit
[2] 2019-08-05. LXD Technology. https://ubuntu.com/blog/lxd-2-0-remote-hosts-and-container-migration-612(accessed20/08/19)
[3] 2019-08-05. Qemu or KVM Virtual Machines - Proxmox VE. https://pve.proxmox.com/wiki/Qemu/KVM_Virtual_Machines
[4] Bruno Blanchet et al. 2016. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends® in Privacy and Security* 1, 1-2 (2016), 1–135.
[5] Y. Cheng and X. Li. 2020. A compute-intensive service migration strategy based on deep learning algorithm. In *IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC*. 1385–1388.
[6] Qiang Duan, Yuhong Yan, and Athanasios V Vasilakos. 2012. A survey on service-oriented network virtualization toward convergence of networking and cloud computing. *IEEE Transactions on Network and Service Management* 9, 4 (2012), 373–392.

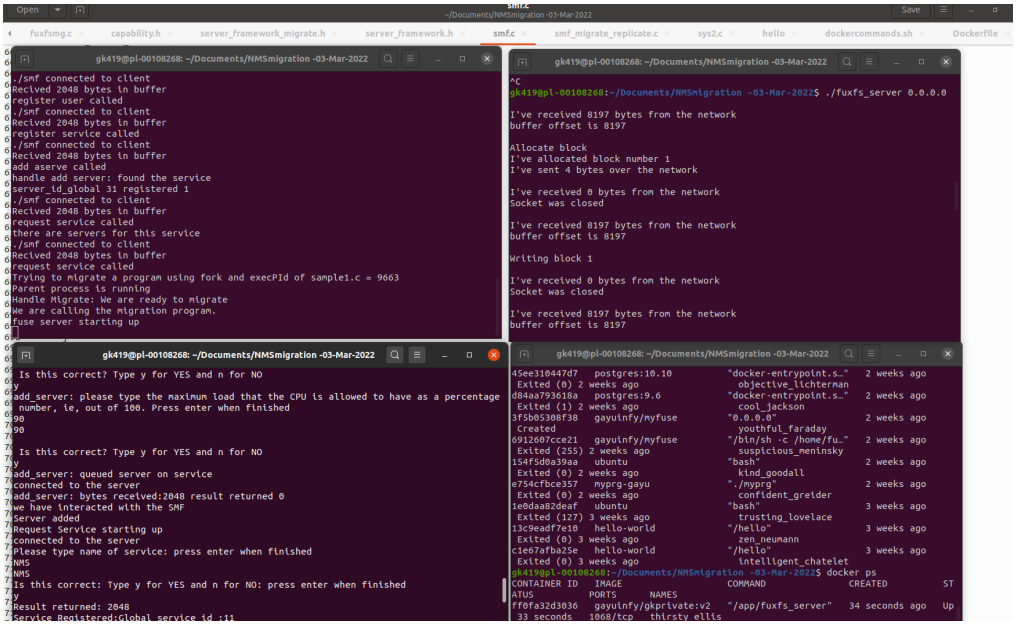

Figure 4: SMF (top left), FUSE Client (bottom left) and NMS (top right) and Docker Status (bottom right): NMS running at a different location using the SMF

[7] A. Ezenwigbo, V.V. Paranthaman, R. Trestian, G. Mapp, and F. Sardis. 2018. Exploring a new transport protocol for vehicular networks. In *Proceedings of the 5th International Conference on the Internet of Things*.

[8] Arindam Ghosh, Vishnu Paranthaman, Glenford Mapp, Orhan Gemikonakli, and Jonathan Loo. 2015. Enabling seamless V2I communications towards developing cooperative automotive applications in VANET systems. *IEEE Communications Magazine* 53, 12 (2015), 80–86. https://doi.org/doi:10.1109/MCOM.2015.7355570

[9] G. Karthick, G. Mapp, F. Kammueller, and M. Aiash. 2021. Modelling and Verifying a Resource Allocation Algorithm for Secure Service Migration in Commerical Cloud Environments. *Journal of Computation Intelligence* (Feb 2021).

[10] A. Lertsinsrubtavee, A.Ali, C. Molina-Jimenez, A. Sathiaseelan, and J. Crowcroft. 2017. Picasso: A lightweight edge computing platform. In *IEEE 6th International Conference on Cloud Networking (CloudNet)*. 1–7.

[11] P. Mach and Z. Becvar. [n.d.]. Mobile edge computing: A survey on architecture and computational offloading. *IEEE Communications Surveys Tutorials* 19, 3 ([n. d.]), 1628–1656.

[12] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. 2013. Unikernels: Library Operating Systems for the Cloud. *SIGPLAN Not.* 48, 4 (March 2013), 461–472. https://doi.org/10.1145/2499368.2451167

[13] A. E. Onyekachukwu, J. Ramirez, G.i Karthick, R. Trestian, and G. Mapp. 2020. *Exploring the Provision of Reliable Network Storage in Highly Mobile Environments*. Bucharest, Romania. https://doi.org/10.1109/COMM48946.2020.9142033

[14] J. Ramirez, O. A. Ezenwigbo, G. Karthick, R. Trestian, and G. Mapp. 2020. A new service managment framework for vehicular networks. In *The 23rd Conference on Innovation in Clouds, Internet and Networks Workshop (ICIN)*. 162–164.

[15] B. P. Rimal, D. P. Van, and M. Maier. 2017b. Mobile edge computing empowered fiber-wireless access networks in the 5g era. In *IEEE Communications Magazine*, Vol. 55. 192–200.

[16] Fragkiskos Sardis. 2015. *Exploring traffic and QoS management mechanisms to support mobile cloud computing using service localisation in heterogeneous environments*. Ph.D. Dissertation. Middlesex University. https://unihub.mdx.ac.uk/study/types/research-at-middlesex/research-repository

[17] F Sardis, G Mapp, and J Loo. 2011. On Demand Service Dilivery for Mobile Networks. In *Proceedings of the First International Conference on Mobile Services, Resources and Users (Mobility 2011)*. Barclona, Spain.

[18] F Sardis, G E Mapp, J Loo, M Aiash, and A Vinel. 2013. On the Investigation of Cloud-based Mobile Media Environments with Service-Populating and QoS-aware Mechanisms. *IEEE Transactions on Multimedia* (2013). https://doi.org/10.1109/T6MM.2013.224028

[19] Polychronis Valsamas, Lefteris Mamatas, and Luis Miguel Contreras. 2022. A Comparative Evaluation of Edge Cloud Virtualization Technologies. *IEEE Transactions on Network and Service Management* 19, 2 (2022), 1351–1365. https://doi.org/10.1109/TNSM.2021.3130792

[20] N. Vithanwattana, G. Karthick, G. Mapp, C. George, and A. Samuels. 2022. Securing Future Health in a post-COVID-19 world: Moving from Frameworks to Prototypes. *Journal of Reliable Intelligence Environments* (July 2022).

[21] B. Xu, S. Wu, J. Xiao, H. Jin, Y. Zhang, G. Shi, T. Lin, J. Rao, L. Yi, and J. Jiang. 2020. Sledge: Towards effective live migration of docker containers. In *IEEE 13th International Conference on Cloud Computing (CLOUD)*. 321–328.

[22] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang. 2016a. Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks. In *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*. 294–299.