Exploring the Provision of Reliable Network Storage in Highly Mobile Environments

Onyekachukwu Augustine Ezenwigbo, Jose Ramirez, Gayathri Karthick, Glenford Mapp, Ramona Trestian
Faculty of Science and Technology
Middlesex University
Email: oe130, jr901, gk419@live.mdx.ac.uk and G.Mapp, R.Trestian@mdx.ac.uk

***Abstract*—Computing is fundamentally about processing data which must be readily accessible to processing elements. Hence, the use of storage hierarchies plays an important role in the overall performance of computer systems. Recently, due to the deployment of fast networks, network storage has emerged as a viable alternative to large local storage systems. However, trying to provide reliable network storage in highly mobile environments, such as vehicular networks, results in the need to address several issues. This paper explores these challenges by first looking at the communication dynamics required for seamless connectivity in these networks. It then looks at how services can be migrated as users move around. The results of this analysis are applied to the migration of a simple Network Memory Server using different migration techniques such as Docker, KVM, LXD and Unikernels in an edge environment, represented by a real Vehicle Ad-Hoc Network. The results show that a proactive approach to service migration is needed to support such services in highly mobile environments.**

***Index Terms*—Network Storage, Service Migration, Network Memory Server, Docker, KVM, LXD.**

I. Introduction

Networks are getting faster, resulting in higher bandwidths as well as low latency. Hence new approaches to fundamental computing components such as storage have been sought. Network storage in which data is stored over a fast network is being used as the modus operandi in many systems, such as the Google Chrome Book. Emerging technologies such as 802.11p and 5G will give rise to the ubiquitous deployment of vehicular networks. A Vehicular Ad-Hoc Network (VANET/ ITS-G5) is an example of such systems. These networks work by using Road Side Units (RSUs), Access Points (APs) or Base Stations (BSs) on the road side infrastructure and Onboard Units (OBUs) in the vehicles or on cyclists and pedestrians.

However, providing reliable network storage in such environments raises a number of challenges. Firstly, there is a need to fully understand the communication dynamics with regard to ensuring seamless connectivity in such systems. This issue has been explored in the Y-Comm framework by using proactive handover techniques [1]. The second issue is about mechanisms that allow the intelligent migration of services to edge systems such as RSUs, APs and BSs. This will result in the ability to maintain a high Quality of Service (QoS) to users as they move around. This paper brings together these two issues to explore providing reliable storage for mobile nodes(MNs).

This paper, therefore, investigates the performance of different state-of-the-art service migration techniques, such as KVM, Docker [2], LXD and more recently Unikernels within the context of a real VANET experimental test-bed setup deployed at Middlesex University. Though mechanisms such as Kubernetes are available to manage workloads in a distributed environment, they do not take into account highly mobile environments such as vehicular networks. Hence, in order to better understand how reliable network storage as well as mobile services provisioning could be achieved within an dynamic edge-based VANET environment, two scenarios are considered, such as: proactive and reactive service migration.

This paper significantly contributes to our understanding of providing, not just storage, but mobile services in an edge environment for highly mobile systems. The rest of the paper is as follows: Section 2 looks at related work while Section 3 strives to understand the communication dynamics of vehicular networks. Section 4 explores the analysis of Edge-to-Edge Service Migration while Section 5 looks at the prototype environment for the new framework. Section 6 investigates different server migration mechanisms. The results are presented in Section 7 while Section 8 concludes the paper.

II. Related Work

A. Network Storage

Distributed Storage systems have been developed and built over many years. A major effort was the Serverless Network Filing System [3] or xFS, which looked at using a set of machines in a peer-to-peer fashion to provide storage for other machines over a wide area network. Like the Andrew File System (AFS) [4], the xFS design attempted to make extensive use of both memory caches and local on-disk caches at client nodes and used clustering as well as sophisticated cache coherency algorithms to eliminate the need for a central server at the core of the system. In [5], the authors showed that network storage could be implemented without the need for special hardware. The design supported mobile users by using a two-level system. The systems used a network memory cache or NMC which could be migrated to a machine on the same network as the mobile node and a persistent storage service or PSS which provided persistent storage as well as redundancy in the core network.

B. Migration of services

Authors in [6] proposed an optimal Web service migration framework, they discussed its service replication and service transferring strategies. They did experiments on different situations and acquired the appropriate service migration strategy to be adopted based on the current workload. The migration of the service for dynamic scheduling and dynamic deployment helped to ensure better QoS.

In [7], the authors proposed dynamic service migration in mobile edge computing based on a Markov decision process. This process defined mobile applications that utilize cloud resources. Such applications consisted of running a front end and their components running on Cloud servers, where the Cloud provided additional processing capabilities by using Media Edge Clouds (MEC) to address the challenges such as network overhead and latency by moving computation closer to the user. However, MEC now faces a new challenge of dynamic service placement and migration as mobile users move around. They formulated general cost models and a mathematical framework to design optimal service migration and approximated the underlying state space by the distance between user and service locations. Evaluations were based on real-world mobility traces of San Francisco taxis. The results showed that these mechanisms were faster than traditional methods.

Service migration has been proposed for many environments and is increasingly being used in Cloud environments that support virtualisation. This is possible because the virtual machine paradigm allows entire virtual machines to be migrated. Virtual machine migration can be expensive as the entire virtual machine has to be moved. The emergence of container technology, such as Docker [8], in which containers housing several services are migrated, is gaining in prominence. Unikernels [9] in which the operating system is bounded and customised to run a single main application is the next emerging specimen in this genre and should, from a management point-of-view, make server migration simpler. However, in vehicular environments, it is necessary to implement service migration in the context of the communication dynamics and hence these efforts are difficult to use without a wider service management framework.

III. Understanding the Communication Dynamics of Vehicular Networks

In this section, we introduce a set of network coverage parameters that will be used in the following sections to demonstrate the service migration in a highly mobile environment such as a vehicular network. The network coverage area is a region with an irregular shape where signals from a given Point of Attachment (PoA) i.e., Access Point or Base Station can be detected by a MN. The signals from the PoA are unreliable at the boundary and beyond the coverage area, the signals from the PoA cannot be detected. For seamless communication, handover should be finished before the coverage boundary is reached.

Therefore, two circles, denoted by the handover radius (R_H) and the exit radius (R_E), were defined in [10] to ensure smooth handover. The work states that the handover must begin at the exit radius and should be completed before reaching the handover radius boundary as shown in Fig. 1.

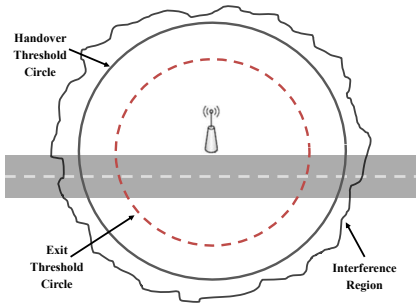

Fig. 1. Network Coverage

The exit radius will therefore be dependent on the velocity, ν, of the MN. If we represent the time taken to execute a handover by T_{EH}, then:

$$T_{EH} \leq \frac{(R_H - R_E)}{\nu} \tag{1}$$

Hence, exit radius can be given as shown in Equation (2)

$$R_E \leq R_H - (\nu * T_{EH}) \tag{2}$$

Our previous work on proactive handover in [11] showed that the above-mentioned coverage parameters can be segmented into communication ranges and presented an in-depth analysis of such segmentation and their importance in order to achieve a seamless handover as shown in Fig. 2. This segmentation can be put to effective use for achieving proactive handover, resource allocation, and service migration for a highly mobile environment.

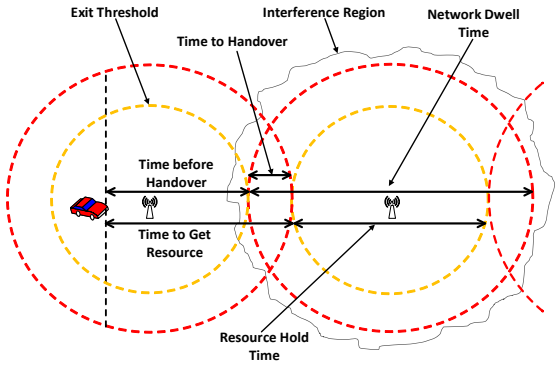

Fig. 2. Communication Range Segmentation

Time before handover (Υ) is the time after which the handover process should start and Time to handover ($\hbar$) is the time before which the handover to next coverage range has to be completed. Network Dwell Time ($\aleph$) is the time MN will spend in the coverage i.e., the Network Dwell Distance (NDD) of new network. Resource Hold Time ($\mathbb{N}$) is the resource usage

time or when actual exchange of data is taking place. $\hbar$ and $\mathbb{N}$ are the two key parameters that have to be considered for service migration in highly mobile networks. From an analysis of previous literature in [10], $\hbar$, is set to a maximum value of 4 seconds.

IV. Edge-to-Edge Service Migration

Along with the communication dynamics, it is also necessary to explore edge-to-edge service migration in order to support mobile services. Let us suppose the MN is travelling at a velocity, ν from one RSU's coverage region to the next RSU's coverage range; with an estimate of the Υ, $\hbar$ and $\mathbb{N}$, it is possible to decide whether a service should be migrated with the knowledge of the service migration time or SMT. Hence, SMT should be less than the sum of $\hbar$ and $\mathbb{N}$ in order to have effective service in the new network. If the SMT is greater than the sum of $\hbar$ and $\mathbb{N}$ then the MN will be out of coverage of the next RSU due to mobility by the time the service is migrated. This is expressed by Equation 3.

$$(\hbar + \mathbb{N}) > SMT \tag{3}$$

The above equation denotes a reactive approach. When the MN reaches the next coverage range, the service will be migrated to the next RSU. In summary, for this scenario, the communication handover and service migration begin at the same time, this is called a reactive service migration. This approach might disrupt the service due to mobility for services with high migration times.

Hence, for better QoS and Quality of Experience (QoE), we need to also consider proactive service migration. In this approach the service migration will begin before the communication handover as shown in Fig. 3. The point where the service is starting to migrate is called the proactive service migration time (X). When the service begins to migrate at point X before the communication handover, so the amount of time left is SMT-X. Thus proactive migration is shown in Equation 4. This also means that, X should be less than or equal to SMT, which will ensure that the service is not migrated far ahead before the MN reaches the coverage of the next RSU. However, in order to have a migration between adjacent RSUs, X also needs to less than $\mathbb{N}$ as shown in Equation 5. This is because service migration cannot begin before the communication resources of the current RSU are acquired.

$$(\hbar + \mathbb{N}) > (SMT - X) \tag{4}$$

$$X < \mathbb{N} \tag{5}$$

V. Developing a Prototype Environment for the new Framework

A. NMS and FUSE as a Service

Using this approach, we will now consider a Network Memory Server (NMS) that manages blocks of memory on behalf of its clients. The NMS creates, reads, writes and deletes blocks of memory using a simple socket interface. We have incorporated the NMS as a back-end to the FUSE file system which is a user-space file system commonly employed in Linux environment as shown in Figure 4. The diagram in Figure 5 depicts how the NMS and FUSE services are used in a Vehicular Network. FUSE runs on the MN and communicates with the NMS server running in the local network. As the MN moves within the vehicular network the NMS is migrated to a nearby RSU.

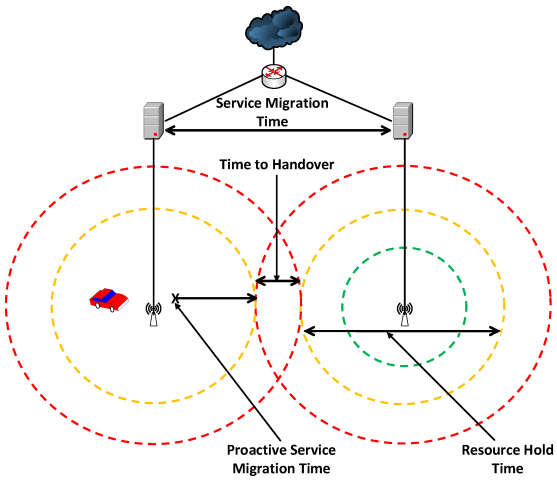

Fig. 3. Proactive Service Migration

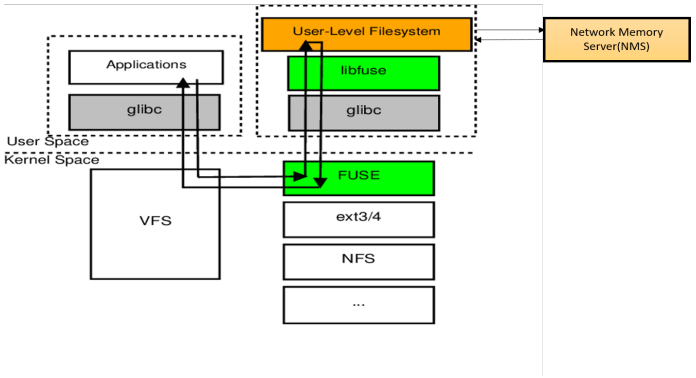

Fig. 4. FUSE and NMS integration

VI. Investigating different migration mechanisms for NMS

It is necessary to look at how the NMS will be migrated using different migration techniques. Four state-of-the-art migration techniques, such as KVM, LXD, Docker and Unikernels were deployed in order to test their performance. A logical diagram of the network implementation is shown in Figure 6. Two physical interfaces and one virtual bridge are attached to the host computer: wireless adapter (wlp0s1), Ethernet adapter (enp0s1), and one virtual bridge is created as part of the VM virtual network (br0). Two virtual Ethernet adapters are created per VM to represent the connections from

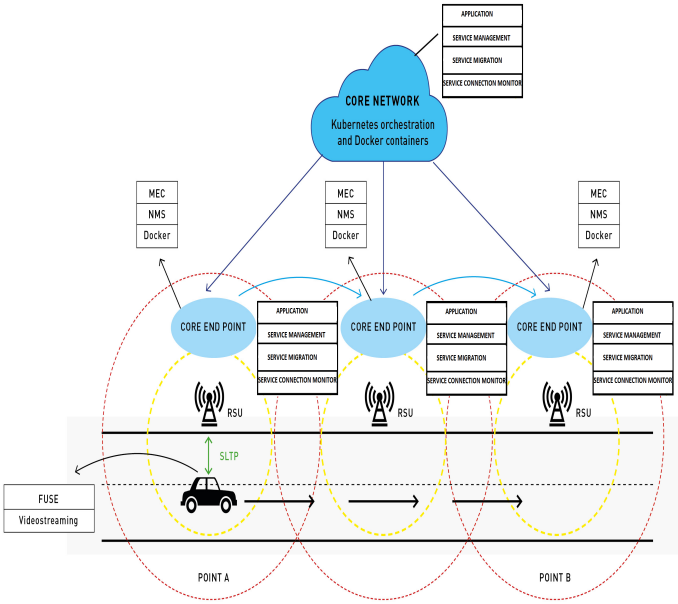

Fig. 5. NMS migration scenario within VANET

TABLE I
LINUX HOST USED FOR LIVE MIGRATION WITHOUT SHARED STORAGE

Linux-host (HWs)	Features
Device	Dell Inspiron 5559
CPU	Intel Core i5 6200U 2.4 GHz (4 cores)
RAM	16 GB RAM DDR3
Storage	256 SSD drive 1TB HDD drive
Host OS	Archlinux (Kernel version 5.2)
Network	Realtek 1Gbps Ethernet, Intel802.11ac wireless adapter

the VM to the host computer in the form of *veth* interfaces. Two interfaces are required, since one of them will be used for service traffic (enp0s8) and the second one (enp0s3) for management traffic. These *veth* interfaces are connected to the enp0s8 and enp0s3 interfaces inside the VM guest operating system.

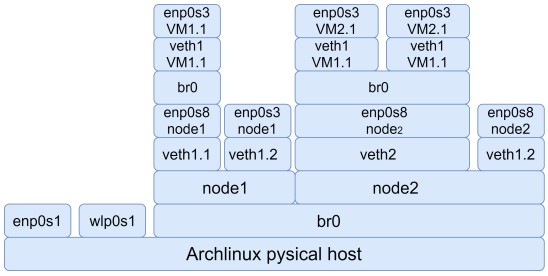

Fig. 6. Logical Implementation Diagram

LXD live migration is dependent on the Checkpoint/Restore in user-space (CRIU) library. Since CRIU must be aware of the particularities of the process to checkpoint and the way it utilizes the underlying resources. Thus, checkpoint creation has several restrictions. A manual compilation of the CRIU is required to guarantee a successful migration. In addition, even when using the latest compiled version available, live migration is not possible in all cases. In particular, any guest OS that uses **systemd** as the method to manage user processes fails to checkpoint due to **systemd's** use of shared filesystem mounts that interfere with the CRIU checkpoint process. Non-systemd distributions, such as Devuan or Alpine are available and can be migrated using this library.

A. KVM

Migration of the VM was done using the CLI with qemu+ssh. However, the virtual hard disk (vHDD) needs to be migrated in advance.

B. LXD

Migration over LXD is done using CRIU and an embedded functionality called LXC move. A preshared SSH key is required between two hosts to attempt migration. Then the container information and snapshots are migrated as a delta of the original image. If the migration is successful, the container is deleted from the source and started on the target server [12].

At the time of initiating the migration, the local LXD daemon checks for the existence of the declared container. A token is created by the local LXD daemon and sent to the remote target daemon, with the source URL and the local certificate identifying LXD local daemon. Then, the remote LXD daemon connects to the local daemon via a control websocket, using the provided token and the transfer mechanism is then negotiated depending on the backend storage being used.

C. Docker

Docker migration is not defined as a feature in Docker documentation. A checkpoint and restore capability using CRIU and runc is available under experimental conditions. This capability is not production-ready and it is evaluated for comparison purposes with other technologies. A checkpoint of the present state of the container is done in this scenario, which requires the container to be stopped in advance. Docker migration scripts were created using Bash scripting language to provide migration capabilities based on the work in [13].

D. Unikernels

A Unikernel was compiled as part of this research to evaluate the performance of live migration of Unikernel images over KVM. In order to compile a Unikernel, a target application needs to be defined, for this purpose, a simple NMS was used. OSv was successfully used to compile a Unikernel with the NMS. The migration algorithm for Unikernel images is identical to KVM images, since the result of the Unikernel compilation process is a QCOW2 KVM disk image. A VM with similar vCPU and vRAM was created using the QCOW2 image compiled.

TABLE II
VM SPECIFICATIONS

Name	ID	HDD	RAM	vCPU	Network	OS
VM	122	32 GB	2 GB	2	1Gbps	Ubuntu 18.04

VII. MIGRATION RESULTS

The specification for the VM being used to house the NMS is given in Table II. The four service migration methods were evaluated in terms of NMS migration time and the results are listed in Table III.

TABLE III
SERVICE MIGRATION RESULTS

Migration Mechanism	NMS Migration Time
Unikernel KVM	11.99 s
LXD CRIU	24.73 s
Docker Container	73.00 s
KVM	824.00 s

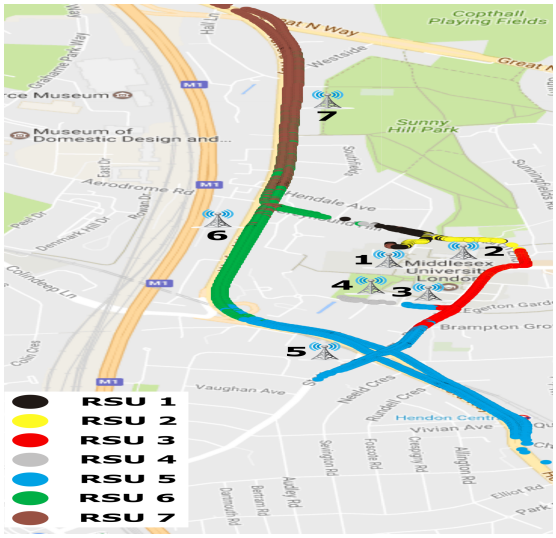

Fig. 7. Middlesex University VANET experimental test-bed

In order to understand the impact of the service migration time within a dynamic high mobility network environment, the VANET testbed deployed at Middlesex University was used. The real VANET experimental setup is illustrated in Figure 7. It consists of 7 RSUs with three RSUs: 5, 6 and 7, located along the A41 which runs behind the Hendon Campus. Using OBUs in vehicles, it was possible to measure the different coverage parameters at 30 mph and at 50 mph. These measurements are shown in Table IV.

Using the results from Tables III and IV, we can evaluate the performance of the four service migration methods under two scenarios: reactive and proactive service migration of NMS within a real dynamic VANET environment.

$$(\hbar + \mathbb{N}) > SMT \tag{6}$$

TABLE IV
COMMUNICATION COVERAGE SEGMENTATION DISTANCE AND TIME ($\hbar = 4s$).

RSU No.	NDD	30 Mph $\hbar$ + $\mathbb{N}$	50 Mph $\hbar$ + $\mathbb{N}$
RSU 1	300 m	22.37 s	13.42 s
RSU 2	456 m	34.00 s	20.40 s
RSU 3	517 m	38.55 s	23.13 s
RSU 4	248 m	18.49 s	11.09 s
RSU 5	974 m	72.63 s	43.57 s
RSU 6	1390 m	103.64 s	62.19 s
RSU 7	1140 m	85.00 s	51.00 s

TABLE V
REACTIVE SERVICE MIGRATION RESULTS

RSU	Unikernels	LXD	Docker	KVM
RSU1	Y	N	N	N
RSU2	Y	Y	N	N
RSU3	Y	Y	N	N
RSU4	Y	N	N	N
RSU5	Y	Y	N	N
RSU6	Y	Y	Y	N
RSU7	Y	Y	Y	N

A. Results for Reactive Migration of NMS

Using Equation 6, it is possible to calculate whether a reactive migration would succeed (given by Y) or would fail (given by N). The results are shown in Table V.

B. Results for Proactive Migration of NMS

In order to look at Proactive Migration, we calculate X as given by Equation 7.

$$(\hbar + \mathbb{N}) > (SMT - X) \tag{7}$$

The results are shown in Table VI. A value of zero indicates that no proactive service migration time is required.

In addition, we said that to have a service migrate from one RSU to an adjacent RSU, then $X < \mathbb{N}$. Thus using the results from Tables IV and VI, we can determine which migration technique should use proactive mechanisms for different RSUs. The results are shown in Table VII.

C. Evaluation of Overall Results

These results highlight key issues in providing services such as reliable storage in a highly mobile environment. Unikernels

TABLE VI
PROACTIVE MIGRATION RESULTS FOR X [SEC]

RSU	Unikernels	LXD	Docker	KVM
RSU1	0	2.36	50.63	801.63
RSU2	0	0	39	790.00
RSU3	0	0	34.45	785.45
RSU4	0	6.24	54.45	805.51
RSU5	0	0	0.37	751.37
RSU6	0	0	0	720.36
RSU7	0	0	0	739.00

TABLE VII
ADJACENT RSU SERVICE MIGRATION

RSU	Unikernels	LXD	Docker	KVM
RSU1	Y	Y	N	N
RSU2	Y	Y	N	N
RSU3	Y	Y	Y	N
RSU4	Y	Y	N	N
RSU5	Y	Y	Y	N
RSU6	Y	Y	Y	N
RSU7	Y	Y	Y	N

showed that it was the fastest migration mechanism and therefore could be used to provide reliable network storage using reactive or proactive migration techniques. LXD also showed good results as it is possible for it to be used by all the RSUs when employing proactive migration techniques. Docker showed mixed results and was unsuitable for RSUs with relatively small coverage ranges while KVM could not be used in both proactive or reactive migration mechanisms. The main reason for the long delay in KVM migration is because KVM attempts to migrate the whole Virtual Hard Disk (vHDD) rather than just the amount of disk and memory being used.

D. Towards a Service Management Framework

One possible solution is to use a shared memory architecture in which the RSUs are connected by a separate network which allows RSUs to share files. Preliminary results using shared memory show that migration times of LXD and Docker were measured at 5.6s and 11.6s respectively. The advantage of using shared memory is that the vHDD does not have to be migrated and so migration is much faster. However, this approach is challenging in the context of vehicular networks. The second approach is to look at using sophisticated caching techniques on the FUSE side of the architecture. This will increase the ability of the storage system to provide service even when the service is being migrated to another RSU. However, this work has revealed the need to look at building a Service Management Framework or SMF which will allow mobile services such as the Network Memory Server to be readily available to mobile users. The SMF will manage the availability, migration and replication of services to ensure that mobile users continuously obtain the required QoS for their applications and so guaranteeing a high QoE for mobile users. These efforts to build a prototype SMF are being pursued by the Networking Group at Middlesex University [14]. This work is looking to use commodity hardware to implement a system that could be built by many entities. In that regard the Raspberry Pi can potentially provide a way to build a distributed Service Management Framework that can be rolled out on a global scale.

VIII. CONCLUSIONS AND FUTURE WORK

Providing reliable network storage in highly mobile environment such as vehicular networks represents a significant challenge. This paper evaluated the performance of four state-of-the-art service migration methods, such as KVM, Docker, LXD and Unikernels within a real VANET experimental testbed setup. Two scenarios were considered, such as proactive and reactive service migration. The results showed that Unikernels outperforms the other schemes involved under both scenarios. While KVM is not a suitable solution in either of the scenarios as the delay introduced is too high. Work is now being done to build a Service Management Framework to support the provision of mobile services such as reliable network storage for mobile users.

REFERENCES

[1] O. A. Ezenwigbo, V. V. Paranthaman, G. Mapp, and R. Trestian, "Exploring intelligent service migration in vehicular networks," in *Testbeds and Research Infrastructures for the Development of Networks and Communities*, H. Gao, Y. Yin, X. Yang, and H. Miao, Eds. Cham: Springer International Publishing, 2019, pp. 41–61.

[2] S. Kumar Pentyala, "Emergency communication system with docker containers, osm and rsync," in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, Aug 2017, pp. 1064–1069.

[3] T. Anderson, M. Dahlin, J. Neefe, D. Pat-terson, D. Roselli, and R. Wang, "Serverless network file systems," in *In Proceedings of the 15th Symposium on Operating System Principles. ACM*, Copper Mountain Resort, Colorado, December 1995, pp. 109–126. [Online]. Available: citeseer.ist.psu.edu/anderson95serverless.html

[4] X. M. L. K., "Synchronization and caching issues in the Andrew File System." in *lTscnix Conference Proceedings lWinter). USENIX Assoc.*, Berkeley, Calif., 1988.

[5] G. Mapp, D. Thakker, and D. Silcott, "The Design of a Storage Architecture for Mobile Heterogeneous Devices," *ICNS2007*, vol. 0, p. 41, 2007.

[6] M. Yingchi, X. Ziyang, W. Longbao, and W. Jiulong, "An optimal web services migration framework in the cloud computing," in *2015 8th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, June 2015, pp. 153–156.

[7] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, June 2019.

[8] "Docker technology," 2016-11-29. [Online]. Available: https://www.docker.com/what-docker

[9] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, "Unikernels: Library operating systems for the cloud," *SIGPLAN Not.*, vol. 48, no. 4, pp. 461–472, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2499368.2451167

[10] G. Mapp, F. Katsriku, M. Aiash, N. Chinnam, R. Lopes, E. Moreira, R. P. Vanni, and M. Augusto, "Exploiting Location and Contextual Information to Develop a Comprehensive Framework for Proactive Handover in Hetegeneous Environments," *Journal of Computer Networks and Communications*, February 2012.

[11] A. Ghosh, V. V. Paranthaman, G. Mapp, and O. Gemikonakli, "Exploring efficient seamless handover in VANET systems using network dwell time," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, dec 2014. [Online]. Available: https://doi.org/10.1186%2F1687-1499-2014-227

[12] "Lxd technology," 2019-08-05. [Online]. Available: https://ubuntu.com/blog/lxd-2-0-remote-hosts-and-container-migration-612(accessed8.20.19)

[13] "Docker technology," 2019-08-02. [Online]. Available: https://docs.docker.com/engine/reference/commandline/commit

[14] J. Ramirez, O. A. Ezenwigbo, G. Karthick, R. Trestian, and G. Mapp, "A New Service Management Framework for Vehicular Networks," in *Proceedings of the 23rd Conference on Innovation in Clouds, Internet and Networks (ICIN 2020)*, February 2020.