Virtualisation of Swarms: Designing a digital twin prototype for MaritimeManet

NATHAN M. JONGEJAN, University of Twente, The Netherlands

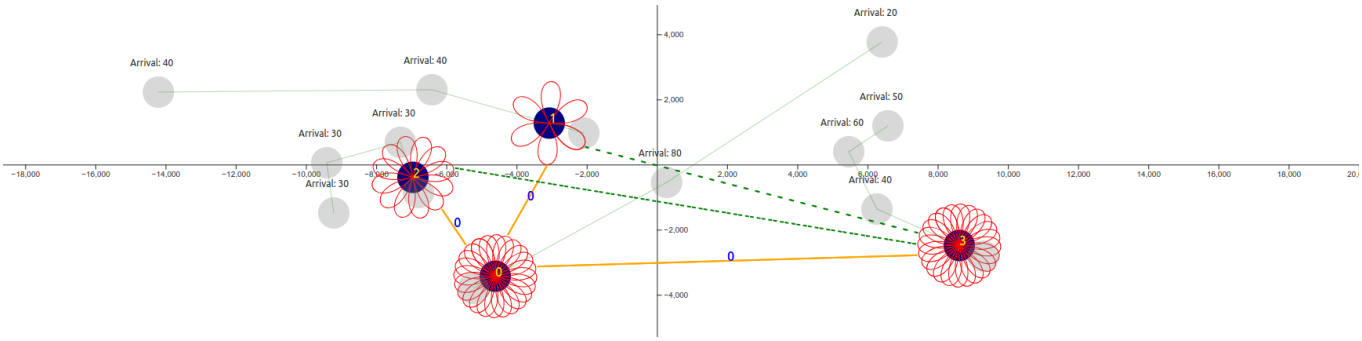

Fig. 1. MaritimeManet simulaiton

Thales is developing a new peer-to-peer distributed communications system for maritime purposes named MaritimeManet. MaritimeManet is a novel Mobile Ad-Hoc Network (MANET) designed with Multi-Beam Antennas in mind to circumvent the classical limitations of MANETs at these distances. For this new network, however, the need arises for a testing platform that allows iteration upon the distributed applications that will run on the network and the algorithms comprising MaritimeManet itself.

Additional Key Words and Phrases: MaritimeManet, Digital Twin Prototype, Network Simulation, Virtualization

1 INTRODUCTION

As we move to an ever more technological world, the value of connecting everything to the internet is becoming increasingly important. Conventionally, devices are connected to the internet through physical cabling or short-range wireless connections. This approach only works in some cases. There are other approaches to connecting systems. Wireless mesh networks are widely explored to connect platforms without centralized infrastructure. Their power lies in providing network access in places where using physical wires is challenging due to terrain. However, this flexibility often comes at the cost of link quality and bandwidth [7]. These problems are compounded when looking at the case of mobile ad hoc networks (MANETs), where the individual platforms are not fixed, and issues such as neighbourhood discovery and changing links come into play [1]. MANET routing algorithms such as BATMAN [4] have largely addressed these issues. However, unlike land-based mesh solutions, the naval space creates new challenges that are not met by current solutions. The distances in the maritime environment can be incredibly vast, limiting the availability of platforms and the bandwidth of the connections between them [3].

TScIT 41, July 5, 2024, Enschede, The Netherlands

To combat these issues, a new system called MaritimeManet has been proposed. This is a mobile ad-hoc network specifically designed for a maritime operating environment. MaritimeManet describes a specific protocol stack layer that combines to form a mesh network. In addition to this, the unique property that sets MaritimeManet apart from other MANETs is its use of multi-beam antennas (MBAs) [3]. MBAs are a set of directional antennas that combine to form a 360-degree area of coverage, similar to omnidirectional antennas. This way, the system can take advantage of this increased range and improved signal quality of directional antennas while still maintaining full coverage. However, this approach carries a significant limitation that MaritimeManet is designed to solve: what happens when a platform rotates away from the directional antenna currently holding the connection? For this, MaritimeManet can intelligently switch its connection from one antenna to whatever antenna would have the strongest signal strength to replace it seamlessly. This process is called a "handover." Along with this and the rest of the protocol stack, MaritimeManet can combine high capacity with high flexibility in the challenging maritime environment.

MaritimeManet is currently still in its early days of development. Therefore, it needs to undergo rigorous testing. This was initially done in collaboration with the Dutch Coast Guard by physically mounting the system on ships and testing them out at sea. Unfortunately, this approach is time-consuming and expensive. Deploying the system out to sea is a multi-day task, making it obstructive to testing smaller-scale applications. These ships also need crew to control the ship itself and the tests being run. Lastly, it is hard to have any real control and oversight over what is happening; you cannot control the environmental conditions, making repeatable experiments impossible. For this, a new testing environment was needed to solve these issues and to test both the system itself and how applications running on it will behave.

1.1 Research Questions

This led to the following research questions, which this paper will aim to answer:

- RQ1: What should the initial virtualisation architecture be, and how can this architecture be iterated upon to support the future direction of MaritimeManet?
- RQ2: What are the most appropriate hypervisor/container platform and virtual networking tools to implement the architecture accordingly?
  - SRQ2: By what criteria can we evaluate the functionality and performance of the tools for implementation?
- RQ3: How can the virtual environments dynamically be set up according to the desired configuration set by the Digital Simulator for MaritimeManet?

2 BACKGROUND

A series of systems were developed to address these issues to create a lab environment for performing tests. The first system is the Distributed Simulator for MaritimeManet (DSM). Its architecture is described in Fig 3. It is an implementation of MaritimeManet that can simulate how the system will perform and its decisions under changing topologies. However, the DSM had some inherent drawbacks; it had a simplified model of moving platforms, making it impossible to test complex changes in network topology. It was purely an implementation of the decision-making of MaritimeManet, excluding the possibility of testing applications and routing protocols that would be running on the network. The second system was the BATMAN Topology Configurator, designed to test the higher layer OSI layer functionality of MaritimeManet, such as routing and applications. It comprises embedded computers connected via cables that can virtually configure themselves to different topologies using a virtual ethernet switch [6], Fig 2. This approach still had drawbacks: it needed to be manually configured for different topologies, it could only differentiate between connected and not connected, as it could not simulate weak/low bandwidth connections. It was also physically large, comprising multiple pieces of hardware, with one new computer required for each additional platform to simulate the corresponding new configuration required.

The drawbacks of both systems and the need to test applications under more complex topologies led to the integration of both these systems. This integration used the output decisions of the DSM to dynamically configure the BTC by changing other properties, such as the bandwidth according to the signal strength. This system also introduced a more advanced movement simulator, allowing for larger, more complex topologies. This new system was still left with one of the main issues from the BTC, which was the physical hardware requirement. An embedded computer was still required to simulate each platform. It was integrated as described in Fig 4, by taking the core components of both systems and linking them together with the use of python scripts and commands sent through SSH.

2.1 Other existing approaches

Existing systems for simulating complex networks exist. Tools like NS3, QualNet, JSim, etc., all have some support for emulating full nodes and are very extendible [5]. the limiting factor of these network simulation tools is that they are designed to test new networks with existing protocols. They often lack support for complex arrays

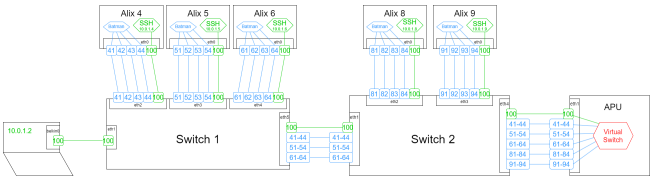

Fig. 2. Architecture of the BATMAN Topology Configurator

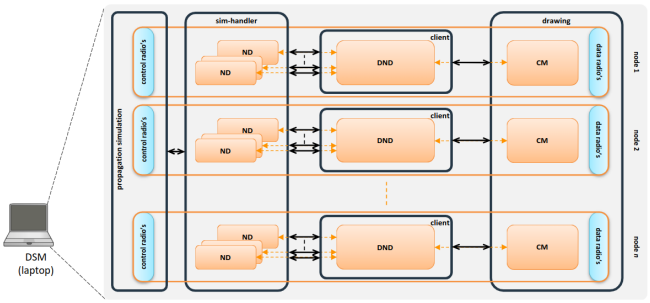

Fig. 3. Architecture of the Distributed Simulator for MaritimeManet

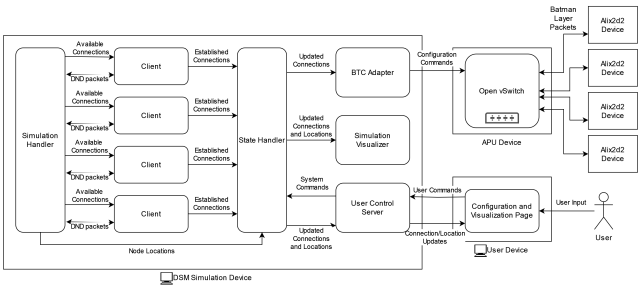

Fig. 4. Architecture of the integrated system of both the DSM and the BTC

of Multi-Beam antennas or have a high complexity for implementing mobile nodes. But lastly, MaritimeManet, being a new network with its own protocols for neighbourhood discovery, frequency selection, and routing, is not supported by the current set of tools out there. Leaving a space open where this Digital Twin can add value.

2.2 Configuration document

Many of the decisions made are guided by a document outlining the configuration of a complete MaritimeManet node. This document defines certain network configurations and IP allocations and describes what features a node can and cannot support. As the simulator aims to be a true Digital Twin Prototype, the design decisions will reflect these restrictions as much as possible for the simulator to maintain versatility and correctness.

3 METHODOLOGY

The first part of Research Question One is Identifying which principles a DTP for MaritimeManet should adhere to. This comes from two directions. Firstly, there are the initial requirements for this system: It should be usable by both technical and non-technical

persons and able to scale dynamically to adapt to different test cases. It should also be a digital twin closely resembling the final system so that MaritimeManet can be tested in an environment mimicking its final deployment environment. The last principle comes from good development practices: the system should withstand future requirements or be adaptable enough to adhere to those requirements with minimal change.

The system should be designed with three types of end users in mind: the developers of MaritimeManet, the developers of distributed applications intended to run on MaritimeManet, and lastly, people looking to give technical demonstrations of the system as a whole. For both types of developers, the system needs a way to quickly deploy the code to be tested and provide a testing interface that allows quick and long tests to be specified and reported upon. For demonstrations, the system needs to be visual enough to be understood and interpreted by people without a technical background or unfamiliar with the MaritimeManet project. The current system does this through its use of waypoints to define the movement of platforms within the simulation, which was designed based on feedback from a potential end user and how their domain of naval navigation interacts with systems. However, this should also be expanded upon to demonstrate the capabilities of the applications running on it.

Designing the system with the future in mind is crucial to ensure that the digital twin stays relevant and usable in the changing landscape and requirements of the final system. This is approached from two angles: identifying the direction MaritimeManet is currently headed in and how the digital twin can be designed with the near to mid-future requirements in mind. These are from how the digital twin can become a test bed for distributed applications and MaritimeManet itself to accommodating alternate MaritimeManet implementations. The second angle is that of using extensible patterns and interfaces and documenting how these interfaces work and how they were designed to be used. Giving space for future developers to expand upon the current system instead of redesigning large portions of it to fit future development directions.

Research Question Three aims to answer how the system can be set up dynamically at different scales. This is one of the initial requirements. It has to scale to extensive simulations involving many virtual platforms with complex distributed applications running simultaneously. However, what is important is that the digital twin should be able to scale to this size rapidly. This allows for quick iteration and changing of simulation size should the need arise. The design options will also be evaluated to determine how they will affect the Digital Twin's ability to achieve this goal.

To answer the second Research Question, we will use the decisions made in the architecture to inform what types of tools we will need to realise it.

4 OPTIONS

4.1 Methods of Isolation

There were three options for isolating the processes and routing from each other: using network namespaces, containers, and virtual machines. We will only consider the last two. Network namespaces do not allow for the dynamic scaling that is required for this virtualised simulator.

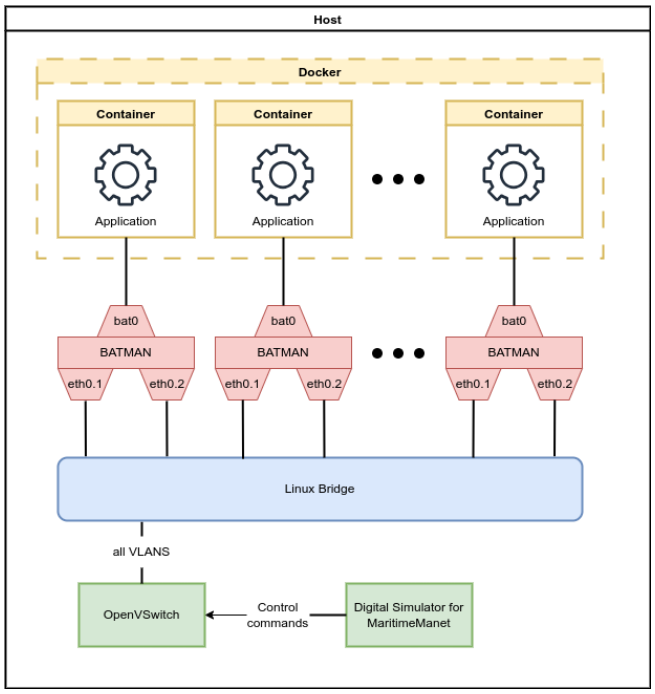

Fig. 5. Architecture of virtualisation using containers

4.1.1 Containerisation. One approach is to use containers to separate the processes from each other. Fig 5 shows a proposed architecture using containers. The docker host can dynamically create/destroy containers based on the current test's needs, meeting the scalability requirement. This approach has the advantage of using containers' fast and performant scaling. It's also relatively simple in terms of configuration. Containerised approaches tend to integrate well with existing test frameworks, allowing quick integration into workflows. This does, however, come at the cost of versatility. This can be seen in Fig 5 where the BATMAN processes fall outside the docker containers and are all kernel processes. This limits the versatility of testing other routing algorithms since they do not all necessarily integrate similarly. It also makes it difficult to scale the networking component without taking the whole network process of the host offline. Furthermore, moving to a more complete emulation of a MaritimeManet node will not be possible, making iterating on versions of MaritimeManet significantly more difficult and time-consuming. The system for control of using VLANs, bridges, an OpenVSwitch, and the DSM are all directly borrowed from the integration project.

4.1.2 Virtual Machines. Another approach is that of virtual machines coordinated by a hypervisor. Virtual machines allow for complete emulation of other computers, allowing multiple guests to run within a single host. Each computer is completely isolated from the others, having a separate virtual CPU, memory, storage, devices, and kernel. This level of isolation is quite useful for prototyping

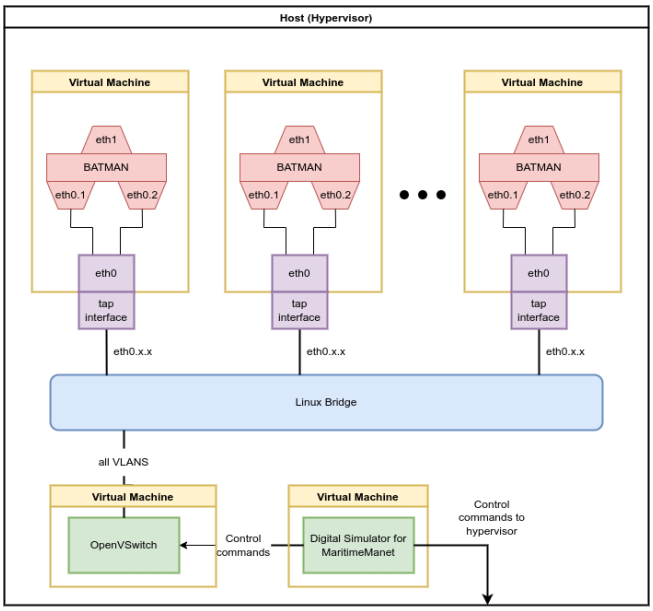

Fig. 6. Architecture of virtualisation using Virtual Machines

the MaritimeManet nodes. It allows complete images to be run and tested, requiring minimal to no changes before being deployed to production, minimizing the amount of real-world tests that need to be run. In addition to this, virtual machines can use optimization techniques like templating to replicate many nodes quickly for testing different-sized topologies. These improvements in versatility do, however, come at the cost of some complexity. Fig 6 showcases a potential architecture realized through the use of virtual machines. All computational tasks had to be done inside a virtual machine. While not strictly necessary, everything works similarly, making the setup easier to replicate onto new hosts. The creation of guest machines inside of the host also comes at the cost of a bit of performance, while this is is very low with modern virtualisation [8], it does mean that containers will be slightly more performant. The networking itself is also a bit more complex when using virtual machines. As can be seen in Fig 6, the network interfaces from the BATMAN processes to the Linux bridge include a middle step made up of two interfaces, one on the Guest, which presents itself as an ethernet interface and one on the Host which presents itself as a virtual TAP network interfaces. This, again, leads to more complexity when configuring the network.

4.2 Networking

A couple of conditions must be met for functional networking to be virtualised. Data sent from a sector has to be completely isolated from data from other sectors, even when coming from the same node until it reaches the OpenVSwitch. The method by which data is isolated must be dynamically addressable by a control process. There must also be room to scale the approach to work for different network topologies with a variable number of nodes present. This firstly requires a virtual link to carry the traffic from one port to the next, which will be done through Linux bridges, and also a plan for how the data carried will be isolated.

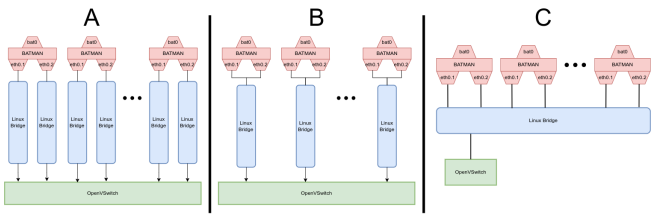

Fig. 7. Possible options for connecting interfaces using bridges

4.2.1 Different bridge configurations. We consider three different architectures for connecting the many interfaces of the BATMAN process to the OpenVSwitch responsible for establishing/maintaining the virtual connections. These options differ by how the traffic is isolated when travelling from the BATMAN process to the OpenVSwitch, characterised by how many bridges are used for their implementation. One bridge for each node sector, one bridge per node, or one shared bridge for all nodes.

Option A in Fig 7 shows each connection, giving a sector its own Linux bridge for traffic. This greatly simplifies any work that has to be done with setting up more complex ways of separating traffic, but it does have a few drawbacks. Firstly, one virtual network port is required per sector. This is unrealistic according to the final configuration of MaritimeManet, which defines the nodes as only requiring one physical ethernet port in total, not per sector.

Option B solves this by defining a Linux bridge per MaritimeManet node, with the traffic of the different sectors being isolated using VLANs. While this makes the architecture compatible with the configuration of a MaritimeManet node, it still has two tradeoffs. To create these bridges, the host's networking service has to be restarted, interrupting all connections to the simulation. Secondly, OpenVSwitch must be able to handle dynamically changing hardware interfaces. This can be supported through hot-plugging but can still have unexpected behaviours with interface naming, especially at larger scales.

Lastly, there is the option C. This approach only uses a single Linux bridge. It also uses many VLANs to separate network traffic. This approach involves embedding VLANs within an outer VLAN. The inner VLAN then describes the sector information as defined by a static configuration, while the outer VLAN indicates which node the traffic originates from. This practice has the name of QinQ, as VLANs are defined by the 8021q standard. It does require a more complex configuration within the OpenVSwitch to separate the VLANs and assign the correct VLANs to the correct MaritimeManet nodes. But in doing so, it does not require a full restart of the host's network service whenever the simulation scale changes.

4.3 Interfaces

There are two ways to implement QinQ networking for this use case. This is based on how the hypervisor handles virtualised network devices, of which a diagram can be seen in Fig. 8

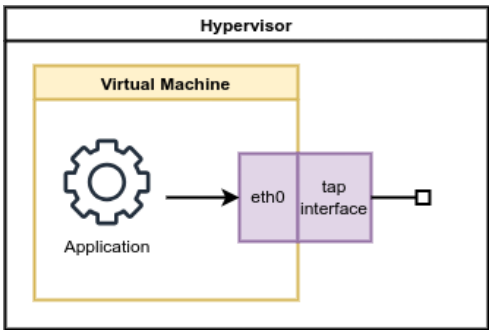

Fig. 8. Diagram of virtual machine network interface

4.3.1 QinQ on Virtual Machines. The first option is to define the QinQ network on the eth0 interface within the VM itself. This is the simplest option for quick configuration, as it is simpler than modifying the tap interface. However, it is less accurate to the final configuration of MaritimeManet, as it requires modifications to the virtual machine image either before the virtual machine is started or while it is running through a shell connection.

4.3.2 QinQ on TAP interfaces. The second option is defining the outer VLAN on the tap interface. Since this is transparent to the virtual machine itself, it allows the entire configuration of a MaritimeManet node to be used without modification. However, the unfortunate limitation is that configuring QinQ on a tap interface has proven more difficult than configuring it within the virtual machine.

4.4 Control plane

For the virtual simulator to function and allow the user the flexibility of manually inspecting their tests, data needs to flow between the control plane of the simulator, to the simulated entities themselves, and back. For a containerised approach, this is very simple, with the ability to inspect the files created by a test program, and spawning shells within each container. For the approach of virtual machines, this ability can be implemented in different ways, each with tradeoffs.

4.4.1 Serial. A serial port allows a shell to connect directly to a guest VM without requiring a network connection. This requires no configuration from the VM's side and very little effort from the hypervisor's side. However, a serial shell does have some big limitations. It does not allow easy file transfer, so copying logs is hard to impossible. It works by streaming only the raw text information from the serial port. So, there is no native way of checking the execution status or exit code from an executed command, only the textual output.

4.4.2 SSH. Another approach is to use a secure shell (SSH). This approach does not have the issues of using a serial port but at the cost of having requirements on what will need to be installed on the MaritimeManet node itself. It requires an active network connection configured in advance to give the control plane an IP address to connect to, as well as having to be on a bridge with every other device that has to be connected to via the control plane.

4.4.3 Guest Agents. Lastly, there are guest agents. These are small programs that need to be installed on the MaritimeManet nodes themselves, which can allow a hypervisor to interact with them. They support file transfer, command execution, and other methods to control the virtual machine itself. However, the API to communicate with them is quite a bit more cumbersome, and a guest agent will not allow a remote shell into the virtual machine, which prevents access to a powerful debugging method.

5 DEFINITIVE ARCHITECTURE

The definitive architecture can be seen in Fig 6. A hypervisor is used as the host, and virtual machines run the individual MaritimeManet nodes. This was chosen as it best allowed for the future direction of MaritimeManet. By being a closer representation of what the final product will look like, there is a greater versatility of tests which can be run—the greater level of isolation facilitates the ability to test different routing algorithms. This system also allows for an eventual shift of the control plane from the Distributed Simulator of MaritimeManet to the nodes themselves, allowing the neighbourhood discovery and channel selection production code to be run as if it were communicating out at sea.

Fig 6 also used the network type C as illustrated in Fig 7. Using a single bridge allows for the seamless scaling of the network component. The tradeoff of using multiple bridges can be justified by the complexity it adds in terms of the number of simulated Ethernet interfaces and the departure away from the final configuration as described in the configuration document. This, combined with configuring the VLANs on the tap interface (Fig. 8), allows for complete adherence to the configuration requirements of the final production system. It also prevents the need for having to modify the configuration files on the individual MaritimeManet nodes while setting up the tests, removing a step in the start-up process.

5.1 Choice of Hypervisor

The choice of hypervisor was based on a couple of criteria. It should have many network features to accommodate this use case. It should be well supported by a community with enough documentation to be integrated. And it should ideally be free and open source to make it accessible. For these reasons, Proxmox VE was chosen since it scores high on the desired categories [2].

6 RESULTS AND DISCUSSION

So far, a proof of concept has been developed, proving Proxmox to be a viable platform for building the Digital Twin. This proof of concept is based on the design in Fig 4, where the components are described thereby normal computers or embedded devices being virtualised one-to-one. This demonstrates the ability of the simulation to scale from one to four nodes with little effort from the user, as well as the control plane still functioning when virtualised. This, of course, is heavily limited in terms of the ability to scale beyond, but there is no longer a close theoretical/practical limit imposed by, for example, the cost of embedded computers or the amount of cables and physical space required to set up the simulation.

The ability to dynamically scale the number of virtual nodes has already been tested outside of integration with the DSM and BTC.

The system could scale to 40 nodes within two minutes and back from that in the same time period. This demonstrates that the chosen architecture answers the third research question by example.

6.1 Further works

The Digital Twin architecture can facilitate the following future improvements to the MaritimeManet testing environment: the integration of containers as application test environments and the movement of much of the core MaritimeManet logic and protocols from the simulator to the individual nodes themselves. Using separate containers for testing applications would be integrated by routing the traffic from each container to a separate MaritimeManet node, which acts as a gateway. As for moving the core protocol logic, with some adaptations, the DSM can simulate only the control radios, with the control logic (handovers, neighbourhood discovery, frequency decisions) being able to be moved to the final production location of the node itself.

A promising next step for the architecture itself is to replace the OpenVSwitch and the bridge network system with an event-based packet simulator. This way, the simulator can achieve even more granular control over exactly how the propagation of the packets is modelled, as well as artificially and in a controlled manner insert faults into the system to test resiliency.

7 CONCLUSION

The goal of this paper was to propose an architecture for a Digital Twin Prototype for MaritimeManet, what is required for this architecture to be scalable, and what technologies will be needed to implement it. We propose an architecture based on virtual machines that can handle the use case at scale while anticipating the future requirements that this system may face. This will reduce the costs and time needed for testing distributed applications in maritime environments and allow for more complex and longer-running tests on their functionality.

REFERENCES

[1] Ankur Bang and Prabhakar Ramteke. 2013. MANET: History,Challenges And Applications. (Sept. 2013).

[2] Taylor Chien. 2017. Comparing Pre-Built Hypervisors. https://dspace.sunyconnect.suny.edu/items/3bd8e377-131c-419a-b13a-39b212dddbe8

[3] J. H. Laarhuis. 2010. MaritimeManet: Mobile ad-hoc networking at sea. In *2010 International WaterSide Security Conference*. 1–6. https://doi.org/10.1109/WSSC.2010.5730256 Journal Abbreviation: 2010 International WaterSide Security Conference.

[4] K Kiran, N P Kaushik, S Sharath, P Deepa Shenoy, K R Venugopal, and Vignesh T Prabhu. 2018. Experimental Evaluation of BATMAN and BATMAN-Adv Routing Protocols in a Mobile Testbed. In *TENCON 2018 - 2018 IEEE Region 10 Conference*. IEEE, Jeju, Korea (South), 1538–1543. https://doi.org/10.1109/TENCON.2018.8650222

[5] Piotr Owczarek and Piotr Zwierzykowski. 2014. Review of Simulators for Wireless Mesh Networks. *Journal of Telecommunications and Information Technology* 3 (Sept. 2014), 82–89. https://doi.org/10.26636/jtit.2014.3.1037

[6] Michal Raczkiewicz. 2023. Better Approach To Mobile Ad-hoc Networking in MaritimeManet. http://essay.utwente.nl/96407/

[7] Syed Yasmeen Shahdad, Asfia Sabahath, and Reshma Parveez. 2016. Architecture, issues and challenges of wireless mesh network. In *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, Melmaruvathur, Tamilnadu, India, 0557–0560. https://doi.org/10.1109/ICCSP.2016.7754201

[8] Stefan Gabriel Soriga and Mihai Barbulescu. 2013. A comparison of the performance and scalability of Xen and KVM hypervisors. In *2013 RoEduNet International Conference 12th Edition: Networking in Education and Research*. 1–6. https://doi.org/10.1109/RoEduNet.2013.6714189