Evaluation of a Long-Distance IEEE 802.11ah Wireless Technology in Linux Using Docker Containers

Daniils Aleksandrovs-Moisejs*, Aleksandrs Ipatovs, Elans Grabs, Dmitrijs Rjazanovs
Institute of Telecommunications, Riga Technical University,
Azenes street 12, LV-1048, Riga, Latvia
daniils.aleksandrovsmoisejs@gmail.com

***Abstract*—Wireless technologies are essential for modern people to maintain uninterrupted connection to the Internet. The most popular standards for wireless technologies are standards of the IEEE 802.11 family. Currently, data transmission rate achievable by IEEE 802.11ac or 802.11ax standards can reach up to 10 Gbit/s. Different IEEE standards have specific data transmission rates. For example, the IEEE 802.11ah standard or Wi-Fi HaLow (code name) operates in the 900 MHz band, which is an unlicensed frequency band below 1 GHz, and is called the "Sub-1-GHz" range. In theory, this standard can provide coverage range of up to 543 meters indoors and data transfer rate of up to 347 Mbit/s (using a maximum of four spatial streams and 16 MHz channel bandwidth). The great benefit of the 802.11ah standard is low energy consumption, which enables communication between devices from the Internet of Things (IoT) over long distances without using a lot of energy. The Wi-Fi HaLow standard is being studied by the authors of the presented article in the ns-3 network simulation program with the 802.11ah module installed and implemented in Docker containers, VirtualBox Virtual Machines (VMs) with a running Linux operating system. During the simulations, results were obtained for the Docker containers simulation with a limited number of stations over different simulation times. These results have been studied in different scenarios. In the scenarios, the results of the Wi-Fi HaLow network simulation were converted into another simulation time, and thus were compared with each other.**

***Index Terms*—Linux; ns-3; Docker; Wi-Fi HaLow; IEEE; Wireless; 802.11.**

I. INTRODUCTION

Modern information technologies are evolving rapidly, wireless networks of the newest generation are becoming more common, powerful, and demanding. The mobility of the wireless communication networks is the most important benefit. Such networks are used to transmit data between two or more devices without using wired connections. Radio, optical, and laser waves can be used for data transmission. There are several types of wireless networks: WPAN, WLAN, WMAN, and WWAN networks. Users can use their network devices on WLAN and connect to existing Internet of Things (IoT) devices. It is called "internetworking of physical devices". IoT devices technology is equipped with sensors and other communication devices. This technology enables remote access to information from sensors equipped objects and remote control of these objects using existing network infrastructure [1]. IoT devices can be simulated using ns-3 network simulator in Linux operating system with Docker containers to study several types of simulation. The problem is implementing the IEEE 802.11ah protocol in simulation software to be researched in relation to the WLAN wireless network topology. For example, as a network simulator with required modules. In our case, multiple products must be used for such a simulation: Virtual Machines (VMs) with Linux, Docker containers software, Network Simulator ns-3 with experimental 802.11ah module, Wireshark packet analyzer software for testing connections and networking protocols for secured connection with M2M technology (e.g., Telnet or SSH).

Manuscript received 3 November, 2021; accepted 6 April, 2022.

II. LINUX OPERATION SYSTEM AND VMS

What is Linux? It is an open-source Unix-like operating system based on a Linux kernel, which is typically packaged as a Linux distribution. Distributions include the Linux kernel and additional software and libraries, some of which are provided by the GNU Project. The most popular Linux distributions are Ubuntu, Debian, and Fedora. These are Graphical User Interface (GUI)-based operating systems. There are also multiple Command Line Interface (CLI)-based operating systems, e.g., CentOS or Alpine. What is the difference between GUI and CLI operating systems? This depends on the goal of the user. GUI may be used to work with multimedia or graphical programs. CLI can be used for networking, e.g., to deploy a terminal operating system that works with a virtual machine or Docker container, to create an FTP or HTTP web server with a secured connection over SSH or Telnet protocols. For Linux administration in networks, it is better to use a CLI-based operating system and rely on some partitions of Linux networking - Low-Level Configuration, Local Network Servers, Internet Servers, and Network Security and Router Functions. In the author's opinion, Linux is more suitable for this task, compared to Windows or Mac OS, since Linux has some extended functionality: flexibility, stability, performance, it is networking friendly and secure [2].

Virtual Machine or Virtualization process is the provision of computing resources pool or logical relationship, which is separated from the hardware modifications and ensures isolation of logical relationships for computations over a single physical resource. There is a program for virtualization - a hypervisor or "virtual machine monitor" (Fig. 1). This is a program or hardware circuit that enables multiple operating systems to run in parallel on the same host machine. The hypervisor also provides the isolation of operating systems, their protection and security, resources sharing between different operating systems, and data management. In addition to that, the hypervisor must supply connectivity for operating systems from the host machine (e.g., FTP servers or secured network connections) in the same way as if these operating systems were to work on different physical computers.

There are plenty of criteria and reasons for using virtualization nowadays. For example, simultaneous use of multiple operating systems with virtual machine software installed on a single physical host machine, which must run multiple virtual operating systems. The performance of the operating system depends on computer resources, in particular, the volume of Random Access Memory (RAM), free hard disk space, etc. [3]. Each virtual machine can be used by another virtual machine; e.g., there can be a mail server, database server, or another application in a single virtual machine.

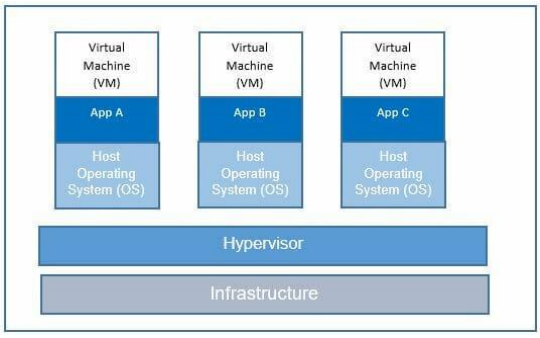

Fig. 1. VM Architecture.

In our case, VMs with Linux operating system were used with required packages installed: ns-3 discrete-event network simulator and additional pre-installed modules. Linux is a very suitable operating system for Python applications (ns-3 was written in Python programming language). VMs were used, as it was necessary to set up a connection between host-machine and VMs even when hypervisor VMs are in background mode (working without loaded GUI), and they must be connected by CLI program via SSH protocol with the host-machine.

Virtualization can significantly reduce hardware and energy consumption costs. In most cases, modern computers use only a fraction of their potential computational power and run with low average system load. Some hardware resources have been evaluated before the experimental part. Instead of running multiple such physical computers with partial load, they can be packaged (aggregated) in a form of various virtual machines run by a single powerful host computer with further balancing of VM loads [3]. The science of virtualization is making great advances, and there are several types of virtualization. In addition to independent virtualization, there is also another virtualization type today, containerization.

III. DOCKER CONTAINERIZATION

Containers have some differences compared to the structure of the virtual machine. Containers were designed, distributed, and operated for products. Software developers can build software locally and be sure that it will work in the same way regardless of the host environment. DevOps engineers can focus on the network deployment, host resources, and operation time allocation. Containers can be used and extended at phenomenal speeds throughout the industry. Docker is a "platform as a service, PaaS" assembly product, which uses operating-system-level virtualization to deliver software in packages. It is called a "container" [4].

The Docker containerization architecture is a "client-server" environment (Fig. 2). On the client side, containers communicate with Docker daemons on the host. The daemon and container can be installed on a single host and both can be controlled remotely. The daemon and containers communicate through the docker0 interface bridges (in Linux). Docker architecture consists of three components: Docker images, registers, and containers. The Docker image is a read-only template used for deployment of a Docker container. The image may contain, e.g., an operating system, a web server, or a mail server with additional pre-installed settings [4], [5].

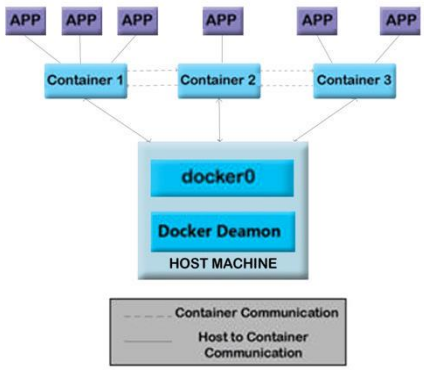

Fig. 2. Docker Architecture.

Containers are the form of application encapsulation with their specific dependencies. At first glance, it may seem to be a lightweight form of a virtual machine. Containers must have an isolated operating system that can be used to run applications. In various cases, containers have several benefits, which makes it difficult to use traditional VMs in a similar way, if not impossible. Containers share resources with a host operating system that makes their size more efficient. Containers can be started and stopped in a fast way. DevOps engineers can simultaneously run many containers on a single host machine, rather than using virtual machines alone [5].

Containers also have advantages for end-users and developers in cloud deployment. Container users can download and run complex applications without configuration and installation issues. Application developers can avoid containers problems caused by user environments

and accessibility dependence capacity [5]. More importantly, the main goals of virtual machines and containers are different. VM aims to completely reproduce another computer environment, whereas container environments aim to make applications portable and independent. For instance, containers can operate with the same OS that is loaded in physical memory, can communicate between sockets, bridges, etc. Also, containers have faster OS boot, compared to VM, file sharing is easy to implement, and lower memory usage to avoid additional storing.

IV. NETWORK SIMULATOR NS-3 AND 802.11AH MODULE

The kernel of simulation and C++ programming models are contained in a discrete-event network simulator ns-3. The network simulator ns-3 is designed as a library that can be dynamically or statically linked to the C++ library. The network defines the simulation. The simulator software can export all its APIs to Python programming language and allows importing Python libraries to ns-3 modules [6], [7].

The discrete-event network simulator ns-3 has been under development for several years. Each of its latest versions is supplemented by several modules that require various research protocols and IEEE standards. In the latest ns-3 version, developers and research staff can simulate an IEEE 802.11ax standard on a physical layer. This makes it easier to use different research protocols from IEEE and ITU standards.

A simulator consists of a kernel and its components, including common protocols, devices, and environmental models. The simulation kernel is specified in the application source code. Packets are fundamental objects of the simulator and are implemented over the network in the application source code. The main simulation modules (kernel and network) are intended to be independent and include a common simulation kernel that can be used by different types of networks. One more feature of the ideal simulation is its mobility. It holds static paths, signal handling distribution of points or packages, etc. Mobility and the internet can communicate with each other through packages that switch to all headers of the Internet protocol (e.g., IPv4 or MAC). The MAC header contains the source of the sender's IP address, the volume of transferred data [6], [7].

In addition to the above-mentioned network simulator kernels, the developers added two more modules. These modules supplement the main C++-based API. The discrete-event network simulator ns-3 can directly access all APIs. The fact is that the ns-3 simulator can write two APIs (or their combination) - and this is a fundamental aspect. Also, the focus can be made on two basic objects: Node and NetDevice. The basic objects of the network devices are the main sources of network emulation [6], [7].

The IEEE standards protocol 802.11ah mentioned before (code name Wi-Fi HaLow) is the category of IoT devices network and it combines the benefits of Wi-Fi and low-powered wireless sensors of network communications technology. The IEEE standard protocol 802.11ah is a WLAN protocol with PHY and MAC layers, which can operate in a 1 GHz frequency band (863 MHz–868 MHz in Europe and 902 MHz–928 MHz in North America) [7], [8]. Frequency bands below 1 GHz are intended for communications at a range of 1 to 3 kilometers and can transmit data with rate of up to 100 Kbps; at the same time, the maximum data transmission rate can reach 347 Mbps with four spatial flows using one 16 MHz bandwidth channel. Modulation schemes and coding rates for IEEE standards are specified by MCS indexes, e.g., at 100 Kbps of data transmission rate satisfactory throughput results can be reached [7], [9]–[12].

The 802.11ah MAC layer of the IEEE standard protocol introduces mechanisms such as hierarchical organization, the header of the short MAC layer, Restricted Access Window (RAW), Traffic Indication Map (TIM), Target Wake Time (TWT), and Modulation and Coding Scheme (MCS). The mechanisms described above support a limited number of stations. The RAW function is the distribution of station groups, by allowing one group to simultaneously access the channel, thus reducing the probability of collisions in networks [7], [8], [10]–[12].

Currently, the ns-3 simulator is supported by several IEEE 802.11 standards, including 802.11a, 802.11b, 802.11g, 802.11n, and modern 802.11ac protocols. It is composed of 4 main components:

– *WifiChannel* - physical part where data are transferred, analytical approximation of the physical medium over which data are transmitted, including propagation loss model and delay model. The propagation loss model describes the signal strength loss during transmission via air and the propagation delay model describes the transmission delay between two nodes.

– *Physical Wi-Fi* - physical part of the protocol, which controls processes of sending frames and receiving them. The PHY part of the IEEE 802.11ah module, where the format of PLCP Protocol Data Unit (PPDU) frame is defined, and frames are sent and received through the Wi-Fi Channel. It consists of the WifiPhy/YansWifiPhy, InterferenceHelper, and ErrorRateModel classes.

– *MacLow* - introduces RTS/CTS/DATA/ACK transactions, distributed coordination function (DCF) and enhanced distributed channel access (EDCA) functions, packet queues, fragmentation, retransmission, and transmission control.

– *MacHigh* - introduces management functions such as beacon generation, zoning, pooling, and authentication [6], [7], [9], [10], [12].

The main components of the discrete-event network simulator ns-3 are two primary PHY layers and MAC layers. The PHY components of the ns-3 simulator are parts of the 802.11ah PHY layers:

– *InterferenceHelper/ErrorRateModel* - defines a new batch type format, signal-to-noise ratio (SNR), and error rate calculation based on physical packets header and payload. ErrorRateModel packets for different models of error rate calculation.

– *WifiPhy* - defines modulation and coding schemes from MCS0 to MCS10 for channel bandwidth from 1 MHz to 16 MHz. The IEEE 802.11ah protocol standard uses

different packet format. Some formats were introduced to send and calculate the length of received packets in the preamble, header, and data loads.

– *Propagation Loss Model* - defines the signal strength in the wireless environment based on the distance between the transmitter and the receiver [7], [12].

MAC layer model components IEEE 802.11ah currently supporting the RAW model:

– *MacHigh protocol* - implements management functions such as beacon generation, probing, association, fast association, and part of RAW. It consists of the ApWifiMac (Access Point (AP)) and StaWifiMac (non-AP station) classes, which share a common parent class RegularWifiMac.

– *Double backup mechanism* - the two-stage back-off mechanism was implemented in the DcaTxop, EdcaTxopN, and DcfManager classes, supporting both Quality of Service (QoS) and non-QoS data transmissions. The start and termination of the two-stage back-off are managed by the DcaTxop and EdcaTxopN classes by sending instructions to the DcfManager class, which is changed to be able to store and restore back-off-related values [7], [12].

V. USED EXPERIMENTAL MEASUREMENTS SIMULATIONS FOR HARDWARE AND WIRELESS NETWORKS

To examine the state of the wireless network simulation, the auxiliary tools from previous sections must be used. All practical measurements of wireless communications network performance may be divided into two categories, system and operation measurements. System measurements are needed to evaluate or assess new network equipment (access points, switches, routers, servers, or host computers), and twisted pair cables with predefined technical type. The second type of measurement (operation) is performed during the cable and hardware mounting process. For example, during the setup of a router or switch in a wireless network. Operation measurements are needed to prevent collisions, accidents and to implement control. The case of accident measurements involves cable and hardware quality testing at local points. Preventive and control measurement tasks can be performed by hardware components, applications, and operational measurements [13]–[15].

The experimental part includes the development of a virtual wireless communications network testbed. During the experiment, a network topology was designed for performing the tasks. This network topology was implemented in two steps. The first step was implemented as a virtual machine with discrete-event network simulator ns-3 and an IEEE 802.11ah standard protocol module. The second step of an experimental part was implemented as a Docker container. It was developed based on defined tasks to evaluate data throughput for specified number of workstations, considering specified modulation and coding schemes according to IEEE 802.11ah standard protocol module for a network simulator ns-3.

Measurements were conducted for both experiments (virtual machine and containers), and the results must be compared according to specific parameters (simulation time and data transmission rate). The simulation time shows the operation speed from the access point to a limited number of stations. The processing time of one station was in the range of 10 to 60 seconds. All inaccuracies need to be assessed and results need to be obtained for different data transmission rates.

The figures below display methods of measurement, which were performed in two steps on the same host machine. The first step (Fig. 3) uses a computer with an installed hypervisor (VirtualBox) software. The VM operating system is Linux with additional software installed and configured in the hypervisor, i.e., the ns-3 discrete-event network simulator with the required IEEE 802.11ah protocol standard module and the connection setup between a virtual machine and host computer via the SSH protocol.

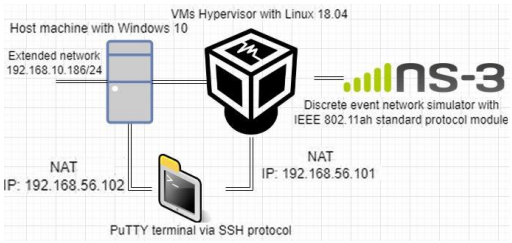

Fig. 3. First step of network scheme with virtual machine and network simulator ns-3 with built-in IEEE 802.11ah protocol standard module.

The second step (Fig. 4) of the network scheme is quite difficult but at the same time also interesting as an application.

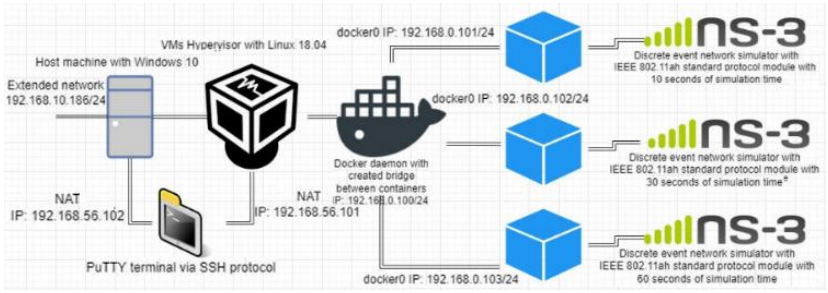

Fig. 4. Second step of network scheme with discrete events in the ns-3 simulator built-in IEEE 802.11ah protocol module for Docker containers.

The first scheme was complemented by Docker daemon and containers with Linux operating system and required programs pre-installed, which were used in the first step scenario. Three more containers with different IP addresses have been added to the Docker Daemon. In these Docker containers, a user or a developer can set up connections between containers themselves, as well as between containers and the Docker daemon. This technology is

called “Docker networking”. Docker includes support for network containers via network drivers. In our case, all containers were connected to virtual “bridges.” Using the same principle as in normal networks, more tools had been installed in containers to measure Wi-Fi HaLow parameters.

VI. Experimental Wireless Network Data Rate Measurements

In this section, the authors describe an experiment with data rate measurements from a wireless network. The goal was to determine the maximum number of stations capable of simultaneous interruptions of operation or overloading of the system, and the number of stations that can provide optimal throughput. This work studies different standard operating parameters of the IEEE 802.11ah protocol. Firstly, for modulation and coding scheme, the 1 MHz and 2 MHz bandwidth channels were used with 8 millisecond delay. Secondly, the distance between stations and access points was introduced by the parameter “rho” (its value was set at 50 meters). Thirdly, the next parameter used is a delay of response between stations (beacons) - 0.1024 seconds or 1024 nanoseconds. The main parameter used during the experiment was the number of stations. The experiment was constantly in progress and a total of 256 stations were used. It is the highest number of stations that can be processed by a host and hypervisor with a virtual machine. In theory, an even greater number of stations can be used, since the IEEE 802.11ah protocol standard can supply up to 8000 stations. To provide more stations for the system, the host computer with higher performance can be used with other operating systems (e.g., CLI or GUI with low consumption of resources), or cloud service servers (Amazon AWS) can be used as well. First, check the modulation and coding schemes selected in our experimental part and the corresponding data rates for such cases. The IEEE 802.11ah standard protocol is known to be based on orthogonal frequency-division multiplexing (OFDM) modulation with a maximum of four spatial flows using a single channel with a bandwidth of 16 MHz. In this experiment, however, we used 1 MHz and 2 MHz bandwidth channels for modulation and coding schemes, and also only a single spatial flow. For example, for the 2 MHz bandwidth channel, a Fourier transform with 56 OFDM subcarriers was used: 52 data subcarriers and 4 assisting subcarriers with a step of 31.25 kHz. Each of these subcarriers can be modulated by one of the following modulation types: binary phase-shift keying (BPSK), quadrature phase-shift keying (QPSK), quadrature amplitude modulation (16-QAM, 64-QAM, and 256-QAM) [7], [8].

For different channel bandwidth values, IEEE 802.11ah standard utilizes several types of modulation and coding schemes, coding rates, and frequency channels. Table I lists the supported data rates and their modulation and coding schemes for different channel bandwidths [7].

TABLE I. IEEE 802.11ah STANDARD PROTOCOL MODULATION AND CODING SCHEME RESULTS AT 1 MHZ AND 2 MHZ FREQUENCY CHANNELS.

MCS index	Modulation type	Coding rate	1 MHz channels	2 MHz channels
0	BPSK	1/2	0.3	0.65
1	QPSK	1/2	0.6	1.3
2	QPSK	3/4	0.9	1.95
3	16-QAM	1/2	1.2	2.6
4	16-QAM	3/4	1.8	3.9
5	64-QAM	2/3	2.4	5.2
6	64-QAM	3/4	2.7	5.85
7	64-QAM	5/6	3.0	6.5
8	256-QAM	3/4	3.6	7.8
9	256-QAM	5/6	4.0	-
10	BPSK	1/2 × 2	0.15	-

VII. Measurement of Experimental Wireless Network Data Transmission Rate in VMS and Docker

The experimental part has been performed in two scenarios: virtual machines and Docker containers. The simulation time and throughput of these technologies have been compared, and some conclusions were made on the technical part of these different technologies. It should be noted that it is impossible to rely completely on hardware and conclude that results are worse or better, since these results are affected by a host computer. The measurements were performed for different data transmission times (10 to 60 seconds of simulation time) for specified modulation and coding schemes with 1 MHz and 2 MHz bandwidth channels and single spatial flow. A different number of stations (up to 256 stations) was selected. The purpose of choosing a different number of stations was to explore how quickly the stations can simulate in one RAW group. Let us summarize experimental part results depending on modulation and coding scheme and number of stations. The analysis of obtained results was performed for all measurements of the entire step with further plots and tables obtained as a result (Tables II and III, Fig. 5).

The summary of obtained results shows that for small number of stations (i.e., 2 stations) the network throughput achieved value of up to 0.041 Mbps. If the hypothetical situation is considered, when a small number of stations is split into several groups, the network throughput of a small number of stations can be improved. For the case of a larger number of stations, the comparison was made for the results of Virtual Machine and Docker containers. The performance difference can be observed for a number of stations set to at least 64. For example, for MCS9 scheme, the throughput of the VMs was 0.592 Mbps and the throughput of the Docker was 0.613 Mbps (Table IV). Docker containers show better results not only for throughput tests, but also for simulation time. This is affected by a large number of factors, such as packet losses and defined modulation and coding scheme

parameters, simulation and operating system parameters. In addition, hardware performance is also a factor, in particular, writing/reading speed.

TABLE II. COMMON RESULTS OF THROUGHPUT (T.) AND SIMULATION TIME FOR VMS AT 1 MHZ.

MCS index	T. of 2 stations [Mbps]	Sim. Time [s]	T. of 64 stations [Mbps]	Sim. Time [s]	T. of 256 stations [Mbps]	Sim. Time [s]
MCS0	0.041	2.3	0.178	63.3	0.145	240
MCS1	0.041	6	0.258	73.3	0.198	260
MCS2	0.041	5.3	0.313	153.3	0.251	513.3
MCS3	0.041	8	0.377	190	0.304	593.3
MCS4	0.041	8	0.446	223.3	0.375	593.3
MCS5	0.041	8	0.53	256.7	0.427	560
MCS6	0.041	8	0.555	203.3	0.449	503.3
MCS7	0.041	8	0.582	330	0.48	520
MCS8	0.041	8	0.58	300	0.466	495
MCS9	0.041	14.3	0.592	266.7	0.481	600
MCS10	0.041	6	0.102	123.3	0.098	320

TABLE III. COMMON RESULTS OF THROUGHPUT (T.) AND SIMULATION TIME FOR DOCKER CONTAINERS AT 1 MHZ.

MCS index	T. of 2 stations [Mbps]	Sim. Time [s]	T. of 64 stations [Mbps]	Sim. Time [s]	T. of 256 stations [Mbps]	Sim. Time [s]
MCS0	0.041	2.7	0.184	43.3	0.158	220
MCS1	0.041	4	0.262	73.3	0.21	206.7
MCS2	0.041	4.7	0.319	100	0.257	280
MCS3	0.041	6.7	0.39	140	0.306	300
MCS4	0.041	6	0.476	140	0.375	320
MCS5	0.041	8	0.532	160	0.431	320
MCS6	0.041	6	0.56	160	0.453	340
MCS7	0.041	6.7	0.593	153.3	0.479	326.7
MCS8	0.041	8	0.586	220	0.482	376.7
MCS9	0.041	12.7	0.613	233.3	0.487	400
MCS10	0.041	6	0.105	123.3	0.07	320

TABLE IV. COMMON RESULTS OF THROUGHPUT (T.) AND SIMULATION TIME FOR VMS AT 2 MHZ.

MCS index	T. of 2 stations [Mbps]	Sim. Time [s]	T. of 64 stations [Mbps]	Sim. Time [s]	T. of 256 stations [Mbps]	Sim. Time [s]
MCS0	0.041	6	0.34	123.3	0.29	320
MCS1	0.041	4	0.458	73.3	0.369	196.7
MCS2	0.041	6	0.519	180	0.418	326.7
MCS3	0.041	8	0.55	220	0.459	420
MCS4	0.041	10.7	0.647	170	0.524	400
MCS5	0.041	10.7	0.681	170	0.55	400
MCS6	0.041	10.7	0.683	170	0.561	400
MCS7	0.041	10.7	0.703	223,3	0.579	433.3
MCS8	0.041	6	0.712	180	0.581	326.7

After performing the comparison, the results for the Docker container simulation look better (Table V) than those for the first step (VMs), where measurements were made for a single virtual machine. In both cases the same configuration was used; however, the plots clearly show that throughput was better by 0.1 Mbps to 0.2 Mbps for higher data transmission rate values. In a simulation, we may see that OFDM modulation changes after 3 Mbps of throughput and has an ideal level of packages sending without loss compared to the first simulation. It can be argued that energy consumption was also low for Docker container simulation, compared to the first scenario with VM, because it was less affected (Fig. 6). Assuming the power consumption of the single virtual station in the case of the virtual machine per station packet was 50 mJ to 100 mJ, the virtual station energy consumption of the Docker container for each station can be estimated as 30 mJ–80 mJ per station packet. Developing a modern device considers lower energy consumption, since it positively affects the performance of the device.

TABLE V. COMMON RESULTS OF THROUGHPUT (T.) AND SIMULATION TIME FOR DOCKER CONTAINERS AT 2 MHZ.

MCS index	T. of 2 stations [Mbps]	Sim. Time [s]	T. of 64 stations [Mbps]	Sim. Time [s]	T. of 256 stations [Mbps]	Sim. Time [s]
MCS0	0.041	6	0.34	123.3	0.29	320
MCS1	0.041	4	0.462	73.3	0.373	196.67
MCS2	0.041	6	0.534	180	0.421	326.67
MCS3	0.041	6	0.584	140	0.465	320
MCS4	0.041	10.7	0.651	170	0.528	353.33
MCS5	0.041	10.7	0.705	160	0.566	340
MCS6	0.041	10	0.704	140	0.578	276.67
MCS7	0.041	10.7	0.709	113.3	0.585	260
MCS8	0.041	6	0.718	180	0.588	326.7

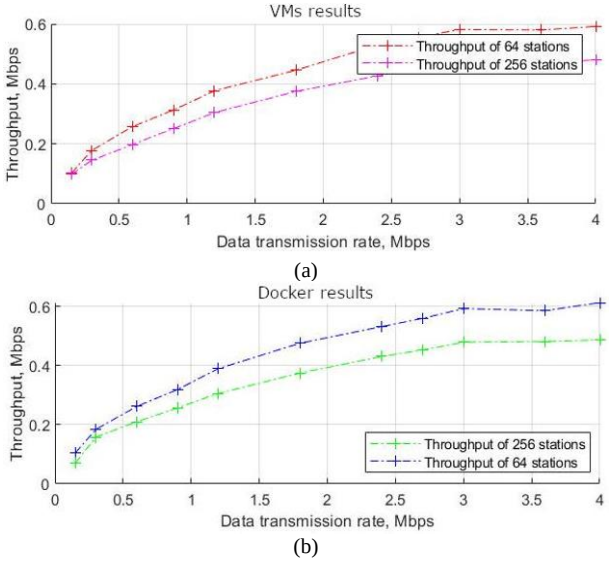

Fig. 5. Total throughput results of 64 and 256 stations with the MCS scheme at 1 MHz for (a) VMs and (b) Docker containers.

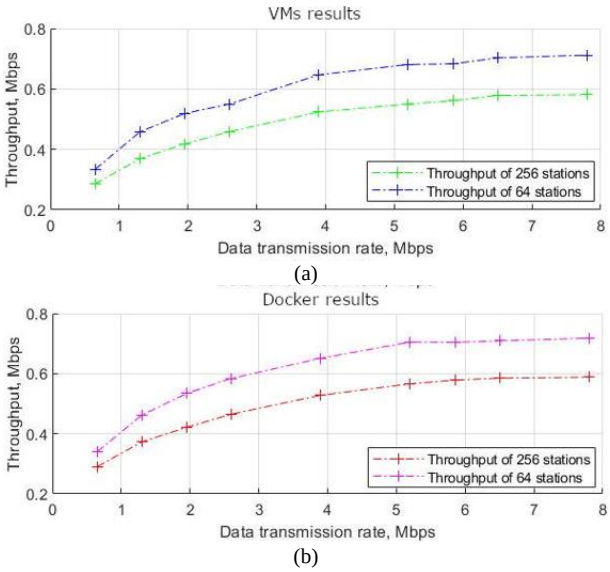

Fig. 6. Total throughput results of 64 and 256 stations with MCS scheme at 2 MHz for (a) VMs and (b) Docker containers.

VIII. RESULTS

The analysis of obtained results revealed that the Docker container configuration was found to show superior results. The experimental part includes measurements for each MCS diagram module for a specified number of stations. Both

steps used MCS schemes with 1 MHz and 2 MHz bandwidth channels, and the number of stations was up to 256. During the simulation, two scenarios have been set for the same configuration parameters (simulation time, station interval, station range). The paper also evaluates energy consumption of a single station during the simulation and the amount of power missing to implement the best operation of all stations. There are two options of IEEE 802.11ah module modulation and coding schemes: 1 MHz bandwidth with 8 ms operating interval and 2 MHz bandwidth with the same operation interval. Unfortunately, it was not possible to evaluate many available frequency bandwidths for the IEEE 802.11ah standard protocol with different MCS schemes (4 MHz, 8 MHz, 16 MHz, and 40 MHz). Still, it is expected that for increasing bandwidth, the data flows or throughput parameter indicators will also depend more on number of stations.

Summarizing the results of simulations for both simulation scenarios, it can be concluded that for 1 MHz bandwidth, both virtual machine and Docker containers show relatively equivalent results for number of stations in chosen range. The results are different only for the MCS10 scheme. When the number of stations is increased to 64, it can be concluded that the coding rate and modulation type for each scheme also adversely affect results for parallel internal processes. A similar situation was also observed for the 2 MHz bandwidth. This effect occurs because the substantial number of stations can still be operated with 5.2 Mbps data rate and the obtained results are not so satisfactory for throughput. The wider bandwidth of the channel provides a rapid data rate and elevated transmission quality.

IX. DISCUSSION

The problem of this paper is the comparison of virtual technologies and their throughout. Which technologies are better suited for simulation? Would a larger number of stations than in the paper's results affect throughput results for wider frequency bandwidth channels (20 MHz or 40 MHz)?

X. CONCLUSIONS

According to the work completed during the experimental part, the testbed was designed, and it supports the connection between hypervisor and host, as well as connection between virtual machine environments and Docker containers. All connections were proven by using the SSH protocol to connect from the host to a virtual machine using PuTTY.

The main conclusion on the simulation results is related to selection between Docker containers and VMs. If the priority is to use a better and faster architecture, it is better to use containers, since the simulation is managed faster compared to a single hypervisor. It can be seen that in the case of MCS8, where the encoding speed is lower (about 3/4 and 5/6), the results of the throughput and simulation time are better for 64 stations in containers, rather than VMs.

RELATED WORKS

The following work has been used for comparison: "Wi-Fi HaLow for the Internet of Things: An up-to-date survey on IEEE 802.11ah research" (2021). It is necessary for results comparing and to have a full-view abstraction for this paper.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] M. U. Farroq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A review on Internet of Things (IoT)", *International Journal of Computer Applications*, vol. 113, no. 1, 2015. DOI: 10.5120/19787-1571.

[2] M. K. Mishra and D. Goyal, "Security analysis in open-source Linux network", *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 8, pp. 1808–1812, 2013.

[3] *Oracle VM VirtualBox: User Manual*, Oracle Corporation, 2020.

[4] A. Mouat, *Using Docker: Developing and Deploying Software with Containers*. O'Reilly Media, 2016.

[5] J. Mulerikkal, "Evaluation of Docker containers based on hardware utilization", 2020.

[6] *ns-3: Network Simulators – ns-3 Manual*, ns-3 project, 2020.

[7] L. Tian, S. Santi, A. Seferagic, J. Lan, and J. Famaey, "Wi-Fi HaLow for the Internet of Things: An up-to-date survey on IEEE 802.11ah research", *Journal of Network and Computer Applications*, vol. 182, art. 103036, 2021. DOI: 10.1016/j.jnca.2021.103036.

[8] D. Perdana, S. Hafidzah, and B. Erfianto, "Analytical study on IEEE 802.11ah standard impact of hidden node", *International Journal of Intelligent Engineering & Systems*, vol. 14, no. 3, 2021. DOI: 10.22266/ijies2021.0630.44.

[9] M. P. Clark, *Data Networks, IP and the Internet: Protocols, Design and Operation*. John Wiley & Sons, Ltd, 2003. DOI: 10.1002/047086804X.

[10] *IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements*, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Computer Society, 2007.

[11] S. Banerji and R. S. Chowdhury, "On IEEE 802.11: Wireless LAN technology", *International Journal of Mobile Network Communications & Telematics (IJMNCT)*, vol. 3, no. 4, 2013. arXiv: 1307.2661. DOI: 10.5121/ijmnct.2013.3405.

[12] L. Tian, S. Latre, and J. Famaey, "An IEEE 802.11ah simulation module for ns-3", University of Antwerp, 2016. DOI: 10.13140/RG.2.1.1378.8244.

[13] Ģ. Ivanovs, G. Lauks, G. Liberts, and J. Poriņš, *Šķiedru Optikas Izmantošana Vietējās Sakaru Sistēmās*. Riga Technical University, Telecommunications Institute, Riga, 2004.

[14] И. Г. Бакланов, Технологии Измерений в Современных Телекоммуникациях. Эко-Трендз, 1998.

[15] D. Aleksandrovs-Moisejs, "Šķiedru optiskās pārraides līnijas vājinājuma novērtējums ar OTDR", B.S. thesis, Riga Technical University, Telecommunications Institute, 2019.