Modelling and Implementation Tools for SDWSN Smart Applications

Duaa Zuhair Al-Hamid, Pejman A. Karegar and Peter Han Joo Chong
Department of Electrical and Electronic Engineering Auckland University of Technology
Auckland, New Zealand
Email: {Duaa.alhamid, Pejman.Karegar, Peter.Chong}@aut.ac.nz

***Abstract*—With unprecedented advances in cutting-edge technologies such as Internet of Things (IoT), network virtualization, digital twin, etc., various applications such as vehicular networks (VN) need to be modelled and evaluated using suitable tools. Given the dynamic nature of VN, there is a need for a virtual flexible model that can be tested prior to implementation to avoid real-time network disruptions. It also offers network maintenance without requiring major modifications to the model in real-life execution. Therefore, software-defined networking (SDN) is one of the approaches that can provide flexibility to such applications as well as other applications such as clustering for trap monitoring using UAV. Applying this advanced technology to vehicular networks can involve significant costs and complex tools. Therefore, utilizing tools such as the Contiki-Cooja network simulator allows for trailing network scenarios and testing performance virtually to obtain the best network topology that can be applied to the physical network. The aim of the paper is to offer the modelling and implementation tools that can be employed towards concept development and testing. It provides insight into the ability of various accessible tools, including software and hardware, to investigate the concept of flexible vehicular network grouping and network re-orchestration.**

Keywords—Simulation tools, Wireless sensor network, Virtualization, SDN, Hardware

I. INTRODUCTION

The rapid advancement of technology has had a significant impact on the design, implementation and evaluation methods for any application, such as vehicular networks and environmental monitoring using unmanned aerial vehicles (UAV). Developed modelling, simulation tools, and hardware components are required for these applications to be intelligent and functional with the newly developed technologies such as the Internet of Things (IoT), and software-defined networking (SDN). For instance, dealing with high vehicle mobility requires a flexible mechanism to maintain network connectivity. Mainly, highway dynamics drive the demand for a real-time, self-organizing solution to handle the frequent occurrence of vehicles approaching and leaving the highway. As the topological network structure requires adaptation to road traffic and its dynamic challenges, traffic analysis modelling and full network connectivity can be provided using powerful communication techniques such as intelligent clustering (grouping) to provide flexible network structure and connectivity. Such requirements are in alignment with improvements in software-defined networking (SDN) and related dynamic network re-orchestration, which enable immediate reaction. This, in turn, enhances the concept of software-defined vehicular wireless sensor networks (SDVWSN), in which the key WSN functions (i.e., 'IoT gateway', 'Router', and 'Leaf') are integrated into the vehicular network via softwarization. This can be done by defining the functions of the nodes via software components in the virtual platform to enable a certain role for the node, such as the router and/or gateway roles. Herein, the cloud-based virtualization enables specific functional configuration parameters to be generated and provided to physical nodes in the physical network setup [1]. Such approaches may also benefit WSN ground setup for distributed traps in the forest, where the UAV can hover over the WSN for data collection. Herein, the SDWSN concept can offer flexible ground network orchestration to enhance and optimize the UAV path [2]. These intelligent applications require modelling and simulation to provide a reasonable approach for verifying a developed concept based on SDWSN and the associated virtualization. However, due to inadequate representation of networks like vehicular network (VN) specific constraints such as mobility, network size and adaptation to road dynamics, this falls short of providing realistic results. Real-world implementation and testbed, On the other hand, provide more accurate data to validate such concepts but are limited by size, cost, effort, and time factors.

This paper provides an overview of the current simulation tools along with a brief summary of related research on general VN and WSN simulation tools. Also, a comparison of network simulators based on their key features is highlighted. Based on the review analysis, the key components of the network simulator that can fit SDVWSN are highlighted to reflect most of the activities associated with the model. Suitable hardware with the selected simulator that can enhance network virtualization is then presented. Finally, a brief overview of additional hardware boards that can be utilized to improve any network modelling is offered.

The remainder of this paper is organized as follows: Section II provides an overview of the VN simulation tools. Section III describes the features of Contiki-Cooja network simulator. Section IV outlines the key components of the physical network, which include a variety of hardware boards. Finally, the conclusion of the paper is presented in section V.

II. OVERVIEW OF SIMULATION TOOLS: USE CASE FOR VEHICULAR NETWORKS

The vehicular network approaches, mainly clustering techniques, associated mobility and performance analysis can be

modelled and simulated using various simulators. Currently, the majority of the work deals with the design and analysis of vehicular network performance from the perspective of network and mobility simulators when the selected network simulator does not support mobility. However, networks like VN require tools that can provide full built-in support for dynamic features, scalability, and performance. Furthermore, as with the recent development of new technologies such as virtualization and SDN, the simulator used needs to reflect the ideology and conceptual implementation of these schemes. In a nutshell, while simulators are beneficial tools for VN, they need to be developed to better support its evolution. The most commonly used network simulators in VN and WSN include NS-3, NS-2, OMNeT++, Riverbed Modeler (formerly known as OPNET Modeler), MATLAB for any mathematical modelling and analysis, and Contiki-Cooja [3-7]. For mobility and traffic modelling, tools like VISSIM, VanetMobisim, and Simulation of Urban Mobility (SUMO) have been used as vehicular mobility generators [8-10].

Tools such as NS-2 [4] can provide a broad collection of models and allow users to simulate node mobility and traffic patterns; authors [11] use the network simulator to design and test their cluster-based data aggregation technique. This system organizes vehicles into autonomous clusters that are led by a cluster-head. In this case, data is routed to the cluster head, which aggregates and distributes it throughout the network. The data aggregation method was created as an NS-2 application/Agent that was connected to each node. The NS-2 (version 2.35) network simulator was used for performance validation in the work of Zhang et al. [12], which focused on the passive multi-hop clustering technique for picking the most stable node as the cluster head. Herein, the vehicle speed was limited to 36km/h to 126km/h, with a maximum of three hops. Since scalability is a major issue in VN, NS-2 is unsuitable for VN research as it consumes a higher percentage of CPU and memory when hundreds of nodes are simulated in NS-2. Furthermore, the tool is not user friendly, with poor GUI and debugging support. NS-3 [3] was introduced to provide additional features to handle the simulation complexity in VN. This, in turn, eases out the issues encountered in NS-2. NS-3 is a free and open-source discrete-event network simulator used for research and development as well as education. The simulation environment is built with C++ and includes a Python scripting interface. Drago et al. [13] provided a novel framework in NS-3 for modelling Artificial Intelligence (AI) algorithms by implementing a new geometry-based channel model and a V2X application, as well as a new intelligent unit (named RAN-AI) for optimizing wireless networks. However, most network protocols are not supported, visualization has limited support, no good network animation tool is supported for wireless scenarios, and limited SDN implementation as an OpenFlow module is only available for wired networks to enable SDN [14].

OMNET++ [5] is a discrete event simulator implemented in C++ and mostly used for queuing network simulations. It includes a GUI library for animation, tracing, and debugging. OMNeT++ was utilized in the work of [15] for modelling the vehicular communication network based on cluster-based routing. The tool can be integrated with SUMO [10] for road traffic simulation. However, no information is provided about the mobility modules used in this work.

Some research on VN and WSN applications has used OPNET Modeler [6], which employs the concept of modelling domains to describe the network, node, and process domains. Subnetworks, nodes (the specific capabilities of each node are described by a model), communication channels such as wireless (fixed or mobile), and geographical contexts are utilized when developing network topology. The internal architecture of the node is described in terms of functional elements and the data flow between them, whereas the process models are defined in a language called Proto-C, which is based on a combination of common features of the C or C++ programming language, Kernel Procedures, which is a library of high-level commands, and state transition diagrams (STDs): States, Events, Actions, Conditions and Transitions. Users can alter or develop numerous network models and customize the topology design based on these simulation models [16]. Figure 1 depicts a simplified scenario created in OPNET from a point to-point network of moving transmitter and reception vehicles.

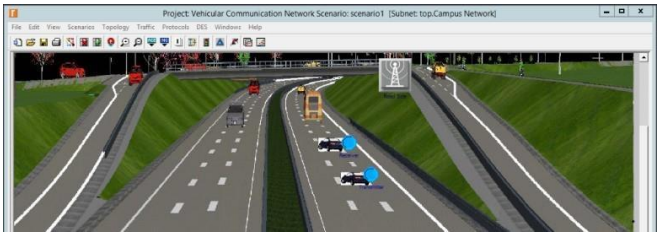

Fig. 1. The receiver and transmitter vehicles in OPNET

Chen et al. [16] evaluated the end-to-end delay, packet delivery rate, and routing overhead of their proposed Trunk Line Based Geographic Routing (TLBGR) protocol in VN using OPNET. Total number of vehicles is set to 100 vehicles and the trajectory of is determined by the trajectory function defined by OPNET. The vehicle speed range is from 30 to 60km/h with a packet generation rate of 1 to 10 packets/second. OPNET is a commercialized (non-open source) software that provides limited wireless mobility and protocol support, mainly for reconfiguring of a given protocol.

Concepts such as SDN and network virtualization require tools that can model and evaluate the network functionality on a virtual platform prior to actual implementation to prevent major adjustments that need to be performed on a physical network structure. Furthermore, dealing with the virtualization platform can facilitate dynamic planning for eventual network reorchestration as needed. The Contiki-Cooja virtualization tool (network simulator) [7] was utilized in the work of [17] to reflect some of the mentioned ideologies. Since Cooja serves as a virtualization platform, virtual nodes are built by compiling and configuring the same Contiki operating system firmware used to configure the actual target hardware platform of the Texas Instruments CC2538 sensor nodes. From utilizing the tool for a given application point of view, Karegar et al. [18] proposed a point-by-point air-to-ground communication system that considers the grouping structure for partitioning the ground network into small groups of sensor nodes distributed over large spaces with the support of an efficient communication between

sensor nodes and a UAV. UAV path flight is relaxed using the possible dynamics in WSN orchestrations as suggested in their approach. The Contiki-Cooja simulator has been utilized to set up a communication dialogue between the UAV and the ground WSN, with the communication scheduling method based on the RSSI measurements of the distance between the sensor nodes and the UAV.

Table 1 shows a comparison of network simulators, lists the tool's key features such as network scalability and mobility and highlights the tool's limitations.

TABLE I. COMPARISON OF NETWORK SIMULATORS

Simulators	Scalability	Mobility	Disadvantages es	Language e	License
NS-2	Small - Scale	Support	Very limited graphical support	C++ and Otcl/Tcl	Open-sourc e
NS-3	Large - scale	Support	Limited graphical support	C++/ Python	Open-sourc e
OMNeT ++	Small - scale	Support (dependent on external extensions such as Mobility Framework (MF)) [19]	Dependency on other frameworks such as INET framework for IP stack higher layers	C++/NED	Open-source (for education and research purposes)
OPNET	Enterprise se	Support	Limited wireless mobilit y support	Proto-C	Commercia l
Contiki - Cooja	Medium -Scale	Support	Maximum nodes to be displayed are 70 nodes	C	Open-sourc e

In keeping with the context of simulation tools, the vehicular mobility generator such as SUMO [10], a microscopic and multi-modal traffic simulation, can generate traces with detailed information for each individual vehicle, including position and speed. However, the traces generated are incompatible with various network simulation tools, including NS-2, Contiki-Cooja, and OPNET. Furthermore, while it includes examples of real-world scenarios, creating complex and large-scale scenarios is time-consuming. Moreover, SUMO does not support vehicle communication simulation. As a result, it should be used in conjunction with other network simulators to simulate vehicular communication. MATLAB is used by [20] to predict vehicle traffic in different scenarios based on mobility parameters such as vehicle arrivals.

In summary, the selected tools for the SDVWSN and road traffic analysis model presented in [21], [22] are Contiki-Cooja and MATLAB, respectively, based on the tool analysis provided above. The virtualization platform, which is one of the research elements, is provided by the Contiki-Cooja tool. It also contributes to the softwarization concept employed in this study by providing software definitions for the nodes in the vehicular group. It also provides a testing environment for the grouping strategy prior to physical network implementation with TI CC2538 hardware. The features of the tool and hardware are covered in the following sections. For traffic analysis, MATLAB is chosen to implement and test the proposed traffic analysis model in [23]. The mathematical and visual features of MATLAB fit well with the purpose of the traffic model as the required output to identify the road capacity, for example, is obtained

III. THE CONTIKI-BASED COOJA NETWORK SIMULATION TOOL

The development of tools such as Contiki, which allow the same program to be used in both hardware and simulation scenarios, has been made possible by advances in modelling and software technology. The Contiki OS [7] built-in "Cooja" tool, which is accessible via the "Instant-Contiki 2.7 IDE" that is installed in the VMware player, can be used to virtualize any IoT-based network, such as wireless sensor network and vehicular network. Based on the analysis of the most commonly used simulators, the Contiki-Cooja simulation tool is chosen for modelling the SDWSN, mainly the SDVWSN, as it is based on WSN core functions, and for serving as a virtualization platform where the network operational behaviour and its node's function(s) can be re-orchestrated prior to physical implementation.

The Cooja simulator's GUI offers a layout that allows user to set up a network with a given number of nodes. A simulation scenario can be created by adding 'Cooja Motes' each of which is programmed with a specific functionality, such as leaf node to serve the scenario's goal. Figure 2 depicts the simulation layout, with the 'Network' window providing options to view (by clicking 'View' in the 'Network' window) the related features such as Mote ID, Radio traffic, Positions, 10m background grid, Mote type, and Radio environment (UDGM). From the transmission range point of view, a green circle is displayed around each clicked node. This is the node's specific transmission range. The selected node (node ID 1 in figure 2) can communicate with all other nodes positioned within this range. The grey circle encircling the green circle represents the interference range of the selected node. If a node in this grey area broadcasts packets concurrently with another node, it cannot receive packets from that node. This is owing to interference. Nodes positioned in the white region are outside of the communication range of that specific node and are thus unaffected by any transmissions from that node. Figure 2 also shows the 'Simulation control' window, which offers the start button for running the simulation. This also includes the simulation time and the selected speed limit for running the simulation. The other window within the simulation layout is the 'Mote output', which reflects the time and data exchanged between nodes, such as node ID, sensing measures (e.g., vehicle speed, battery, RSSI, etc.) as shown in figure 2.

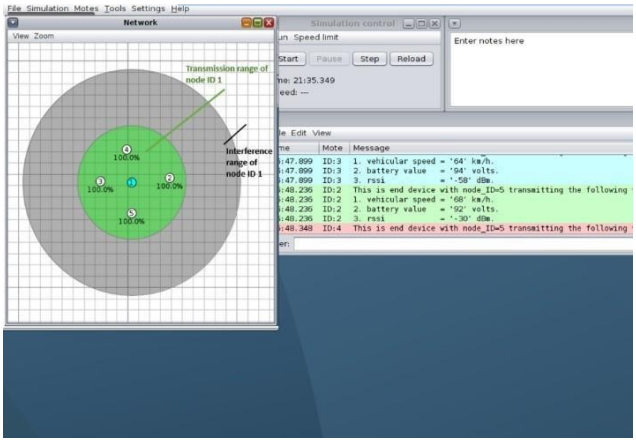

Fig. 2. Cooja simulation layout

From the configurability and softwarization point of view, the 'etimer' function in C code is used to configure a leaf node in Cooja to transmit a message at a specific rate to a recipient node. For example, a rate of 10 samples per second is added to the function 'etimer_set(&et, CLOCK_SECOND*0.1)' as a value of (0.1). Channel access methods such as TDMA (Time Division Multiple Access) and CSMA (Carrier Sense Multiple Access) can be invoked and switched as needed to control access to a shared communication medium from the MAC (Media Access Control) layer. These methods can be enabled in the Contiki software C code to enable a certain channel access method. This is applicable to both virtual motes and physical nodes (TI CC2538). Each of these has a different impact on network performance, depending on the requirements of network operations. For testing network scenarios built and simulated in Contiki-Cooja, the channel access methods at the MAC layer i.e., TDMA and CSMA were utilized. For the network packet rate, the sampling rate was set to 1 sample/second. For example, 30 leaf nodes (end devices) and one coordinator node were used to verify functionality and polling mechanism using TDMA. The initiation time of each node, including the coordinator, varies, resulting in latency in data dissemination in the first round of the simulation. However, the coordinator received all data during the overall simulation time, such as node ID, RSSI, of all nodes as shown in figure 3. In the CSMA method, the coordinator node with ID 31 received all data from all nodes in the first round as well as during the simulation runtime as shown in figure 4.

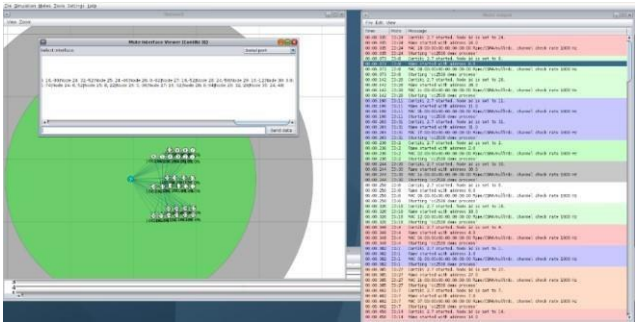
Fig. 3. Network with one coordinator node using TDMA.

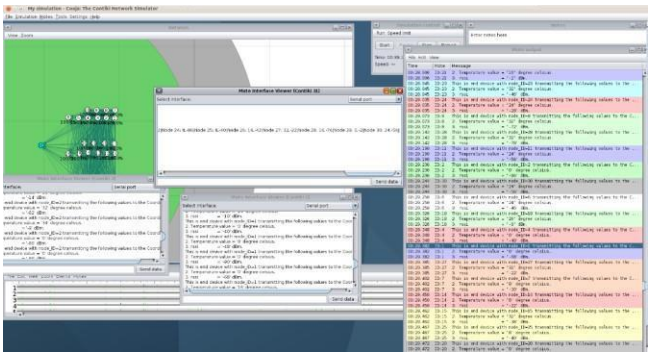
Fig. 4. Network with one coordinator node using CSMA.

It is worth mentioning that the timestamps of each node disseminating messages that appear in Mote's output can reflect the time intervals between nodes during their ongoing communications. To identify network packet loss, 'Counter_Packet_received++' can be added to the C code of the receiving node and 'Counter_Packet_sent++' to the C code of the sender node (i.e., leaf node).

IV. THE KEY COMPONENTS FOR PHYSICAL TESTBED

The hardware boards that can be used to validate a given model, such as SDWSN, represent the physical nodes within the physical network. The boards can be Texas Instruments CC2538 (TI CC2538), TI CC1352R, and Raspberry Pi.

A. Texas Instruments CC2538 Sensor Nodes

The on-chip RSSI (Received Signal Strength Indicator) sensor in the TI CC2538 wireless transceiver chips is used for sensing and transmitting real-world sensed data. It is an advanced chip with IP configurability. The CC2538 EM is used in conjunction with the SmartRF 06 Evaluation board for programming purposes. These features, as well as the hardware node configurability using the same program code as employed for the Cooja motes, encourage the use of these boards for physical network testbed implementation. Through the Contiki IDE, these sensor nodes can be configured with sensing, routing, or gateway functionality. The C code for hardware configuration 'include' the related libraries that enable these chips to be configured, such as '#include "cpu.h"', '#include "dev/uart.h"', '#include "dev/adc-sensor.h"', '#include "dev/sys-ctrl.h"'. Each physical node needs to be assigned a unique ID by accessing the 'contiki-conf.h' header file within Contiki, such as a coordinator node can have an ID of '0xDA'. Also, nodes can be assigned a channel to regulate traffic within the shared communication medium by accessing the same header file and assigning nodes to one of the available and desired channels. The TI CC2538 node can be configured to communicate over any of the 11 to 26 accessible channels. For example, channel 25 can be assigned as '#define CC2538_RF_CONF_CHANNEL 25'.

From the sensing measurements point of view, a scenario can be used where one of the nodes is turned into a mobile node by plugging a Micro USB charger to the chip to provide the mobility element for conducting outdoor experiments to test the impact of RSSI on the hardware network. Figure 5 depicts the TI CC2538 Evaluation Module (EM) with Micro USB. The other scenario is to manipulate the Radio Frequency (RF)

transmission power of the TI CC2538 module to reflect the impact on the network by providing different RSSI values [24].

Fig. 5. TI CC2538 board with Micro USB

B. Texas Instruments CC1352R Sensor Nodes

The Launchpad SensorTag kit CC1352R, also known as 'LPSTK-CC1352R', is a powerful Cortex-M4F MCU with integrated environmental and motion sensors, multi band wireless connectivity, and simple software for prototyping connected applications. Environmental sensors such as humidity and temperature sensors, ambient light and inertia sensors, a Hall-effect switch, and a 3-axis accelerometer are among the Launchpad modules that can be used in trap use cases. Mobility is provided by two AAA batteries or a CR2032 coin-cell installed on the module. These batteries can provide power for the transmission of their readings in a variety of settings. The Launchpad modules include a dual-band low power radio kit that allows Sub-1 GHz (868 MHz/915 MHz) and 2.4 GHz to run concurrently with Bluetooth Low Energy (BLE) in a single chip solution. The 15.4-Stack offers star topology networks for applications operating at sub-1GHz or 2.4GHz. The Sub-1 GHz version has several major advantages, including longer range and improved protection against in-band interference, as well as the ability to deliver Bluetooth Low Energy (BLE) beacon packets while running in dual-band mode on a Sub-1GHz 15.4Stack network. In this case, the dual-band functionality provides the ability to adjust the functionality of a specific SensorTag via over-the-air debugging and BLE configuration. Alternatively, an external programmer-debugger known as the 'CC1352R LaunchPad Development Board' can be used to reconfigure the SensorTag. The Launchpad SensorTag CC1352R and LaunchPad Development Board CC1352R are depicted in figure 6 and figure 7, respectively [24].

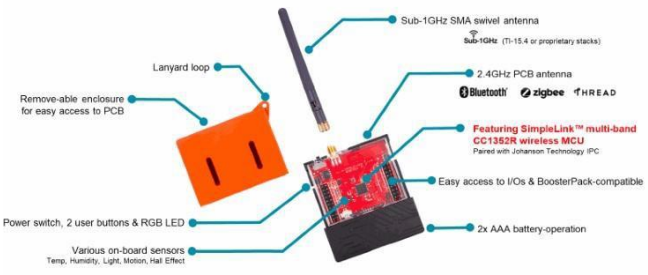

Fig. 6. The Launchpad SensorTag kit CC1352R

Fig. 7. The Launchpad Development Board CC1352R

C. Raspberry Pi

The widely used Raspberry Pi (RPi) is a compact single-board computer that is adaptable, affordable, fully customizable, and programmable with contemporary high-definition multimedia capabilities and Internet access, as shown in figure 8. It can be seamlessly integrated into the domain of WSN and VN applications for research and prototyping purposes. The RAM memory, the CPU (central processing unit), the power supply connector, the USB ports (to which the Wi-Fi USB dongle or adaptor can be attached), the Ethernet ports, the GPU (graphics processing unit), and the lower-level peripherals of General-Purpose Input Output (GPIO) pins that can be used for I2C, UART, SPI-based serial communication buses or interfaces, among other components, are all found on a RPi board. It has a slot for an SD card, allowing it to be utilized for datalogging or large-scale data storage (in cases of Internet outage). Operating systems like Raspbian, NOOBS, and others can be used to run RPi. The board can be used as IoT gateway-capable node to collect, store, and process the received data from sensor nodes in a local database within RPi called SQLite or to communicate the received data from the sensor nodes to a remote database such as MySQL depending on the application. Also, the board can be used as a mobile node by being connected to a portable battery HAT that can be placed on top of the RPI to provide 5v regulated power supply to RPi, allowing the board's mobility to meet a specific requirement, mainly for VN scenarios. It is worth mentioning that the stored real data in the remote database within the cloud can be retrieved and fed into the Contiki configuration unit using the appropriate database interface. Contiki uses the real-time data stored in the database from the most recent test of the physical network to compile and configure a given virtual Cooja mote. This can be placed in the simulation window within the simulation platform for virtual testing purposes, thus virtualizing an established physical network [24].

Fig. 8. Raspberry Pi board

V. CONCLUSION

This paper provides an overview of commonly used simulation tools, including NS-2, NS-3, OMNeT++, OPNET, and Contiki- Cooja. Scalability, mobility, and other features are practical and beneficial when designing a network model. The Contiki-Cooja tool provides a virtual platform to model and test a given network model such as VN-based WSN and SDVWSN prior to the physical implementation. Therefore, the Contiki-Cooja network simulation tool was utilized in this work, which enables virtualization, scalable networks, and dynamic networks as key features to be targeted in a tool. As the Contiki firmware is used by both the Cooja simulator and the Texas Instruments CC2538, the TI CC2538 is the target hardware for physical implementation and network performance validation.

ACKNOWLEDGMENT

The authors would like to thank the Department of Electrical and Electronic Engineering in the School of Engineering, Computer and Mathematical Sciences at Auckland University of Technology for providing hardware and advice for this research project.

REFERENCES

[1] Al-Hamid, D.Z.A., "Vehicular Dynamic Grouping: Virtualization and Network Re-orchestration". AUT. 2023.

[2] Karegar, P.A. and A. Al-Anbuky, "UAV-assisted data gathering from a sparse wireless sensor adaptive networks". *Wireless Networks*, 2023. 29(3): p. 1367-1384.

[3] (June 2023). NS-3 Network Simulator. Available: https://www.nsnam.org/.

[4] (June 2023). The Network Simulator - NS-2. 2022. Available: https://www.isi.edu/nsnam/ns/.

[5] (June 2023). OMNeT++. Available: https://omnetpp.org/.

[6] (June 2023). Opnet. Available: opnet_tutorial.pdf (carleton.ca).

[7] (June 2023). Cooja Simulator. Available: http://anrg.usc.edu/contiki/index.php/Cooja_Simulator.

[8] (June 2023). VISSIM. Available: https://www.tomfotherby.com/Websites/VISSIM/.

[9] (June 2023). VanetMobisim. Available: http://vanet.eurecom.fr/.

[10] (June 2023). Simulation of Urban MObility. Available: https://www.eclipse.org/sumo/.

[11] Shoaib, M., W.-C. Song, and K.H. Kim, "Cluster based data aggregation in vehicular adhoc network". in International Workshop on Communication Technologies for Vehicles. 2012. Springer.

[12] Zhang, D., et al., "New multi-hop clustering algorithm for vehicular ad hoc networks". 2018. 20(4): p. 1517-1530.

[13] Drago, M., et al., "Artificial Intelligence in Vehicular Wireless Networks: A Case Study Using ns-3". 2022.

[14] Mekki, T., et al., "Software-defined networking in vehicular networks: A survey". 2021: p. e4265.

[15] Alowish, M., et al., "Performance Evaluation of a Cluster Based Routing Protocol for VANETs". 2017. 12(2): p. 137-144.

[16] Chen, C., et al., "A geographic routing protocol based on trunk line in VANETs". 2021. 7(4): p. 479-491.

[17] Acharyya, I.S. and A. Al-Anbuky, "Software-defined Wireless Sensor Network: WSN Virtualization and Network Reorchestration". in SMARTGREENS. 2020.

[18] Karegar, P.A. and A. Al-Anbuky, "UAV as a Data Ferry for a Sparse Adaptive WSN". in 2022 27th Asia Pacific Conference on Communications (APCC). 2022. IEEE.

[19] Lessmann, J., et al., "Comparative study of wireless network simulators". in Seventh International Conference on Networking (icn 2008). 2008. IEEE.

[20] Jalel, B.O. and V. Véronique, "Queuing theory-based simulation model for vehicular mobility". in ICC 2021-IEEE International Conference on Communications. 2021. IEEE.

[21] Al-Hamid, D.Z. and A. Al-Anbuky, "Vehicular Grouping Protocol: Towards Cyber Physical Network Intelligence". in IEEE International Conferences on Internet of Things (iThings). 2021. IEEE.

[22] Al-Hamid, D.Z. and A. Al-Anbuky, "Vehicular Intelligence: Towards Vehicular Network Digital-Twin". in 2022 27th Asia Pacific Conference on Communications (APCC). 2022. IEEE.

[23] Al-Hamid, D.Z. and A. Al-Anbuky, "Vehicular Networks Dynamic Grouping and Re-Orchestration Scenarios". *Information*, 2023. 14(1): p. 32.

[24] Al-Hamid, D.Z., Karegar P.A. and PHJ Chong, "A novel SDWSN- Based Testbed for IoT Smart Applications". *Future Internet*, 2023. 15(9): p. 291.