Comparative Analysis of UPPAAL SMC, ns-3 and MATLAB/Simulink

Muhammad Naeem(✉), Michele Albano, Kim Guldstrand Larsen, and Brian Nielsen

Department of Computer Science, Aalborg University, Aalborg, Denmark
{mnaeem,mialb,kgl,bnielsen}@cs.aau.dk

Abstract. IoT networks connect everyday devices to the internet to communicate with one another and humans. It is more cost-effective to analyse and verify the performance of the designed prototype before deploying these complex networks. Network Simulator 3 (ns-3), MATLAB/Simulink, and UPPAAL SMC are three industry-leading tools that simulate communicating models, each with strengths and weaknesses. NS3 is suitable for large-scale network simulations, MATLAB/Simulink is suitable for complex models and data analysis, and UPPAAL SMC is efficient for real-time probabilistic systems with complex timing requirements, This paper presents a comparative analysis of NS3 and MATLAB/Simulink and UPPAAL SMC, based on a Sigfox-based case study, focusing on the behaviour of a single Sigfox node. The comparison is drawn on ease of use, flexibility, and scalability. The results can help researchers make informed decisions when designing and evaluating simulation experiments. They demonstrate that the choice of tool depends on the specific requirements of the simulation project and requires careful consideration of the strengths and weaknesses of each tool.

Keywords: WSN · Network Simulators · Sigfox · Energy Model · Network Modelling · IoT

1 Introduction

The Internet of Things (IoT) has seen significant growth in recent years, leading to the development of intelligent environments in areas like smart homes, energy, and industry [8]. As IoT devices are often used in sensitive areas to collect information and control the environment, designing an efficient model to reduce the error risk and ensure system security is crucial. Simulating the prototype's model during the design process is essential to analyse its performance, identify flaws, and overcome potential vulnerabilities. Several network simulation tools are available, but selecting the most suitable one can be difficult.

This paper presents a comparative analysis of three simulation tools: Network Simulator 3 (ns-3), UPPAAL Statistical Model Checker (SMC) [3], and MATLAB/Simulink [6], based on the simulation of an industrial case study aiming

J. Kofroň et al. (Eds.): ECBS 2023, LNCS 14390, pp. 153–169, 2024.
https://doi.org/10.1007/978-3-031-49252-5_12

to develop an energy-efficient wireless network for monitoring water levels in drainage lines.

The choice to compare these three tools is driven by their distinct network simulation and analysis capabilities. ns-3 excels in scalability and efficiency, making it ideal for large-scale wireless network simulations. Uppaal SMC's statistical model checking offers valuable formal verification capabilities, while MATLAB/Simulink's versatility in handling continuous and deterministic simulations adds another dimension to the comparison. This study aims to provide valuable insights into their performance and applicability for simulating energy-efficient wireless networks. The findings will help researchers and network administrators select the most suitable tool for their simulation needs.

The analysis involves the utilisation of these different tools to explore multiple aspects, including modelling complexity, simulation time, memory utilisation, and validation of the energy-efficient wireless network. The objective is to investigate the strengths and weaknesses of each tool and identify key considerations in selecting the most suitable tool for applications of this nature.

The rest of this paper is structured in the following manner: Sect. 2 provides an overview of the Related Work. Section 3 presents the tools overview used in this study. Section 4 presents the case study, focusing on the Sigfox sensor node. Subsequently, Sect. 5 describes the case study's modelling in ns-3, Uppaal SMC, and MATLAB/Simulink. Section 6 presents a comprehensive comparative analysis of the tools. Finally, in Sect. 7, we conclude the paper and propose avenues for future research.

2 Related Work

In recent years, the availability of various network simulation tools has provided researchers and network administrators with numerous options to choose from. However, the diversity of tools can complicate selecting the most suitable one for specific applications [12].

Nayyar and Singh [12] provided a comprehensive review of 31 simulators, aiming to clarify the features and limitations of each simulator to help new researchers in selecting the most appropriate simulation tool for their applications. The authors discussed the architecture of WSN simulators and proposed evaluation criteria, including the type of simulator, license, platform, ease of coding, tracing, debugging, popularity, and graphical support. The simulators were classified into three categories: generic simulators, code-level simulators, and firmware-level simulators. Generic simulators use high-level programming languages to simulate networking models but are considered less reliable compared to code-level and firmware-level simulators.

Xian et al. [17] compared OMNet++ simulators against other simulators such as OPNET and ns-2. The study demonstrated that OMNet++ outperformed both OPNET and ns-2 in terms of functionalities, including debugging, tracing, hierarchical modelling, and a powerful simulation library. The authors evaluated the performance of the simulators by implementing a well-known WSN protocol called directed diffusion and measuring performance metrics like total run time,

delivery rate, and memory requirement. The results showed that OMNet++ was the most powerful and efficient simulator.

In a study by Gnanaselvi [5], a survey was conducted to gain a better understanding of the current network simulation tools available and their features. Bakni et al. [1] presented a methodology for evaluating WSN simulators focusing on energy conservation. Kochhar and Kaur [7] proposed an approach to guide beginners in choosing an efficient simulator for designing a simulation environment based on their application area.

Our work presents the first comparative analysis of the network simulation tool ns-3 and MATLAB/Simulink with Uppaal SMC. None of the prior research considers the use of the model checker, which is a distinct feature in Uppaal SMC. The comparison is based on applying the three tools in an industrial case study.

3 Tools Overview

This section presents an overview of ns-3, Uppaal SMC and MATLAB/Simulink.

3.1 ns-3

ns-3 is an open-source Discrete Event Simulator (DES) released in 2008 [13]. It offers C++ simulation language with optional Python bindings, making it highly adaptable. It includes models for wired technologies, such as Ethernet networks with CSMA/CD protocols, and wireless technologies, like 802.11 MAC-level and 802.11a physical layer models. Its simulation library focuses on realism and reusability, allowing researchers to create complex network scenarios. ns-3 also supports NetAnim software, allowing for real-time experiments via emulation. ns-3 is a comprehensive and widely used network protocol design and evaluation platform because it integrates various simulation tools. The ns-3 simulator's basic architecture is depicted in Fig. 1 [15]. The figure shows that users create simulation programs that define network behaviour, utilising a simulation library with built-in models for nodes, links, channels, and protocols (ns-3 core). The engine executes these scripts to simulate the network. Data analysis modules offer statistics and performance metrics. Simulation outcomes are generated in a text file that can be analysed using the external graphing tool.

3.2 Uppaal SMC

Statistical model checking (SMC) advances the classic model checking technique [14]. SMC avoids the state-space exploration problem of the classic model checking, and it also comparatively consumes less time and memory in simulation. It simulates a model a number of times and uses statistical hypothesis testing for model checking. SMC technique can also estimate probabilistic systems' quantitative and qualitative properties.

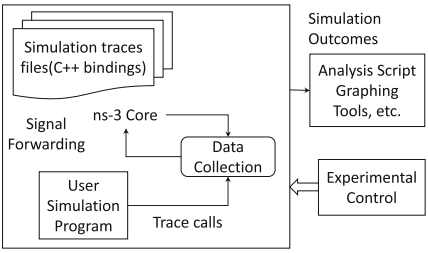

Fig. 1. Framework of NS-3 architecture [15]

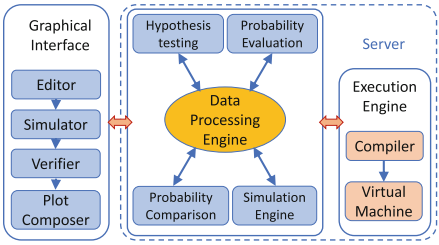

Fig. 2. Architecture of Uppaal SMC [4]

Uppaal SMC is an extension of Uppaal [3], and it models a system using priced timed automata. Uppaal SMC's model is based on stochastic and non-linear dynamic behavioural properties. Figure 2 depicts the Uppaal SMC's architecture [4]. The tool's interface allows users to create automata models in the editor and run simulations for the system's verification, validation and quantitative analysis. It supports visualising results as plots. The Uppaal SMC execution engine exploits the stochastic semantics of interacting stochastic hybrid automata to evaluate the performance queries.

3.3 MATLAB/Simulink

MATLAB/Simulink is a robust simulation methodology, combining MATLAB for matrix-based computation and Simulink for dynamic system design and simulation [6]. It offers a graphical programming language, visualisation tools, and extensibility through MATLAB integration, enabling efficient modelling, simulation, and analysis of diverse systems.

This integration offers researchers and developers in the embedded systems domain an efficient platform to model, simulate, and analyse complex embedded systems scenarios. With a graphical programming language and visualisation tools, MATLAB/Simulink enables the creation of intricate embedded systems models, including various network topologies and sensor node behaviours.

4 Case Study

The aim of the Distributed ONline monitoring of the Urban waTer cycle (DONUT) project is to develop a cost and energy-efficient IoT-based network to monitor the water cycle (See Fig. 3). The Montem Company (a project partner of the DONUT project) has developed a prototype of a digital wireless sensor network based on the Sigfox transceiver. The prototype includes a Sigfox transceiver, Atmega controller, ultrasonic sensor, Digital accelerometer, EEPROM, Regulator, and Battery (10,000 mAh). The controller uses the ultrasonic sensor to measure the water height and then analyses the data to determine the water height for a cycle. The processed value is stored in the EEPROM

before being transmitted to the base station through the Sigfox transmitter. The accelerometer is used to ensure the sensor node's position. Our project is focused on modelling the designed prototype's behaviour using a simulation tool to analyse the battery lifetime and investigate different transmission strategies to improve the overall node's lifetime.

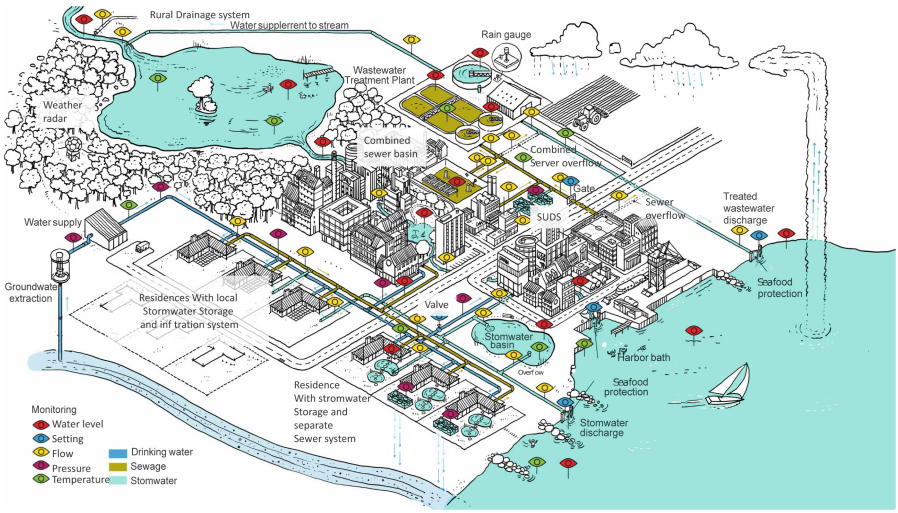

Fig. 3. DONUT-project's low cast sensor network provides holistic urban water system insights for better decisions. 200+ sensors monitor the water cycle, from groundwater to stormwater.

Sigfox is a low-power wide-area network (LPWAN) developed and operated by Sigfox, a company based in France. The basic structure of the Sigfox network is shown in Fig. 4. Sensor nodes use binary phase-shift keying modulation to communicate with the base station in a star topology. A Sigfox node broadcasts its message, which nearby base stations can receive, and these messages are then transferred to the Sigfox cloud. From there, they can be accessed by any IoT platform [16]. Sigfox specifications may vary depending on the region. The European part is the focus of this case study [16]. Sigfox restricts the messages a node can transmit to 6 per hour (144 per day) with a maximum payload of 12 bytes to reduce energy consumption. Additionally, nodes can receive up to 4 downlink messages per day.

5 Modelling the Case Study

In this section, we present the modelling of the DONUT case study utilising different tools, enabling us to conduct a comprehensive comparative analysis.

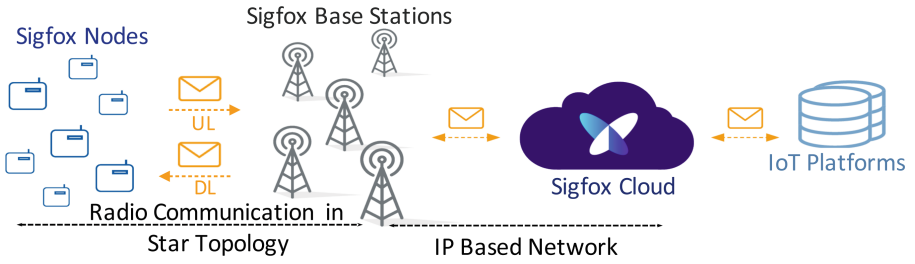

Fig. 4. Sigfox network Architecture

5.1 Implementation in ns-3

In [11], we have presented a Sigfox module for ns-3, and we also investigate the DONUT case study. The model is parametric concerning the hardware properties of the IoT device under research and includes all major energy-consuming states and actions of a Sigfox node. We built the energy model for the device based on data from the Sigfox radio specifications and power characteristics. We also used a novel battery model that considers the self-discharge current. Figure 5 depicts the class diagram of the C++-based designed Sigfox module, which includes the classes and functions that implement the core functionalities.

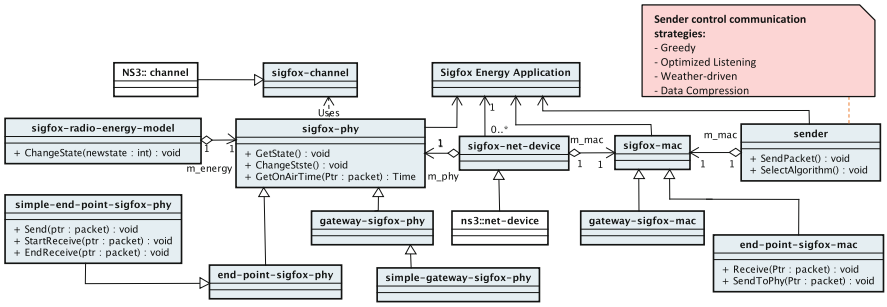

Fig. 5. Class Diagram of Sigfox Module in ns-3 [11]

Sigfox-phy: The `sigfox` module implementation features a PHY layer abstraction that models the interference between multiple colliding Sigfox transmissions to ensure appropriate behaviour when the simulation features large deployments. It also computes energy consumed by each state using subclass `sigfox-radio-energy-model` (See Fig. 5).

Sigfox-mac: The MAC protocol operates on top of the physical layer. As shown in Fig. 5, the implementation of this layer is divided into two classes, `EndPointSigfoxMac` and `GatewaySigfoxMac`, which model the MAC protocol for end node and Gateway separately. The behaviour of a node's MAC layer

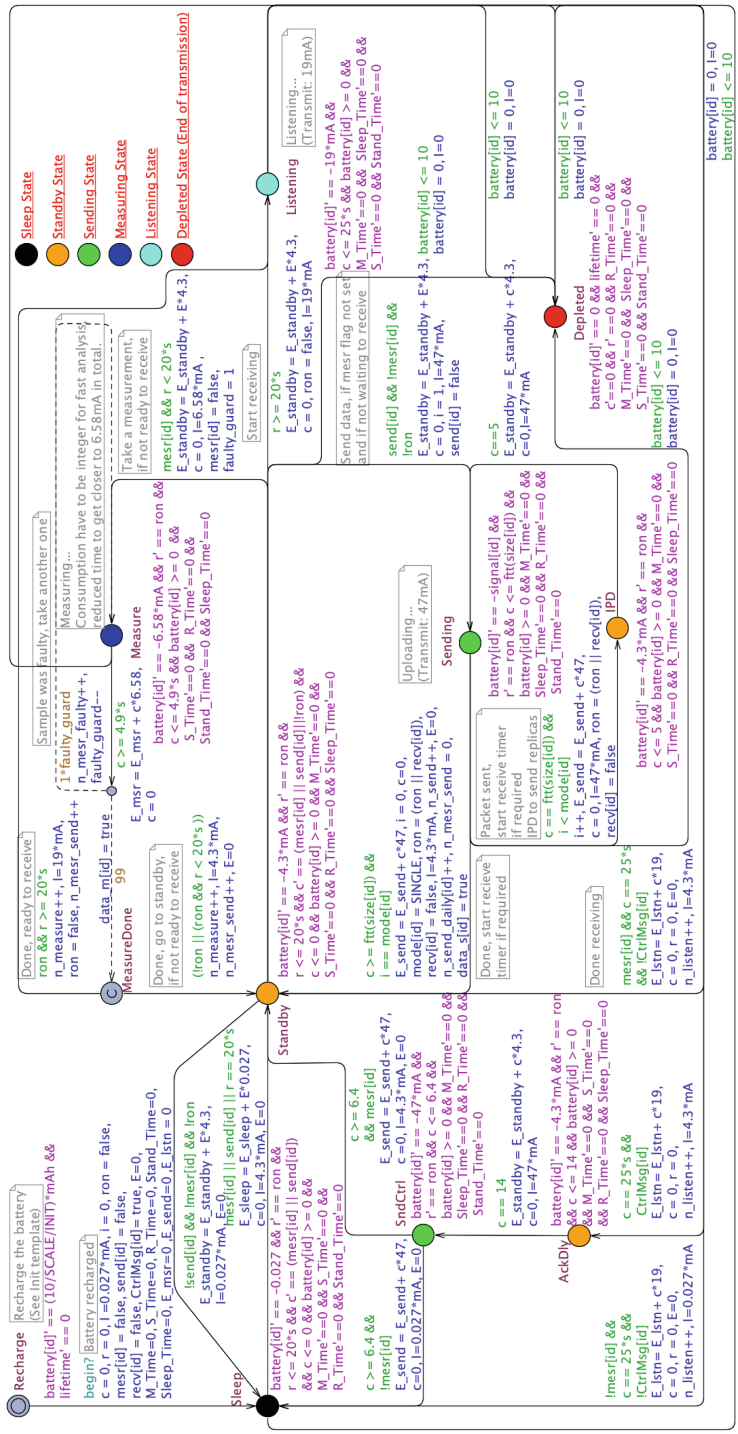

Fig. 6. Basic structure of the designed model in UPPAAL SMC [9]

implements the communication procedures (Uni-directional and Bi-directional), and it controls transmission strategies using subclass `sender` (See Fig. 5).

5.2 Implementation in UPPAAL SMC

In articles [9,10], we have presented the energy-aware analysis of this case study by designing and simulating its model in UPPAAL SMC. In the developed model, we only include the sensor node's behaviour, as the node's battery lifetime is unaffected by the remaining network elements following the Sigfox protocol. Unlike ns-3, we don't need to develop a complete network to simulate a node's behaviour but only a more abstract model capturing the system's behaviour.

The system's model includes four sub-process automaton models (`Initial`, `SensorNode`, `Self-Discharge`, and `Scheduler`), interconnected through shared variables and synchronisation channels to model the sensor node's energy behaviour effectively. `Initial` enables all other processes to an active state using a synchronisation channel, and the `Self-Discharge` model represents the battery's self-discharge behavior. The `SensorNode` automaton (shown in Fig. 6) models the behaviour of the Sigfox sensor node, and the `Scheduler` controls the actions of the `SensorNode`. The complete model is presented in paper [9].

The studies also investigate the different transmission strategies to optimise the battery lifetime. In UPPAAL SMC, as depicted in Fig. 6, users need to have proficiency in automaton modelling and a basic knowledge of the C language.

5.3 Implementation in MATLAB/Simulink

Figure 7 illustrates the behavioural model of the case study implemented in Simulink. We use the C Function block from the Simulink Library to build the Simulink model for the case study. It supports C programming to define the desired algorithm or functionality.

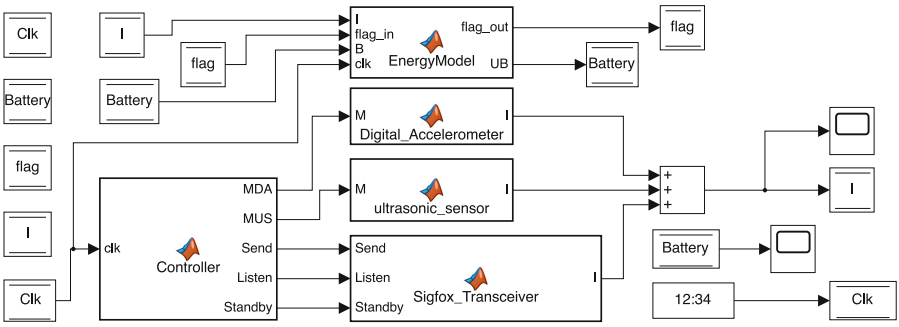

Fig. 7. Sigfox Sensor Node Energy Model in Simulink

The model comprises five main components: Controller, Sigfox_transceiver, Ultrasonic_sensor, Digital_Accelerometer, and EnergyModel. The Controller is

responsible for managing the operations of the active components. A `sum` block adds the current consumption by the Sigfox_transceiver, Ultrasonic_sensor, and Digital_Accelerometer. The EnergyModel utilises the combined system's current to update the battery level for every time unit and manage the self-discharge mechanism. By using a scope block, we can observe the behaviour of combined current and battery discharge.

6 Comparative Analysis of UPPAAL and ns-3

This section presents the comparative analysis of ns-3, Uppaal SMC, and MATLAB/Simulink, considering tool performance, simulation, validation, and usability. The research is based on the DONUT case study.

6.1 Classification of Network Simulation

Article [15] categorises simulations into different classes based on their application areas.

Continuous simulation: Continuous simulation is employed for models with dynamic state variables or parameters that change frequently over time. This type of simulation finds utility in diverse areas, such as military applications (e.g., simulating missile trajectories in WSN deployment).

DES: Discrete-event simulation is applied to systems with events occurring at discrete time intervals. Each change represents an event, with no expected changes between events.

Stochastic simulation: Stochastic simulation involves modelling probabilistic systems, such as evaluating telecommunication system latency, traffic flow in communication networks, and studying climatic changes. Monte Carlo simulation is a specific type of stochastic simulation.

Deterministic simulation: Deterministic simulation is employed in systems characterised by a lack of randomness. These systems possess pre-known inputs and yield unique sets of outputs.

Uppaal **SMC:** Uppaal SMC is a powerful tool that supports various simulations, including continuous, discrete, stochastic, and deterministic simulations [3]. It uses timed automata to model systems with precise timing and discrete events, allowing for continuous and discrete behaviour representation. With its support for continuous simulation, researchers can define clock variables to control the timing and duration of events. For stochastic simulation, Uppaal SMC introduces random variables and probability distributions, making it suitable for modelling systems with uncertainty and probabilistic outcomes. Additionally, Uppaal SMC can perform deterministic simulation, enabling researchers to precisely control the timing of events and verify the deterministic properties of real-time systems.

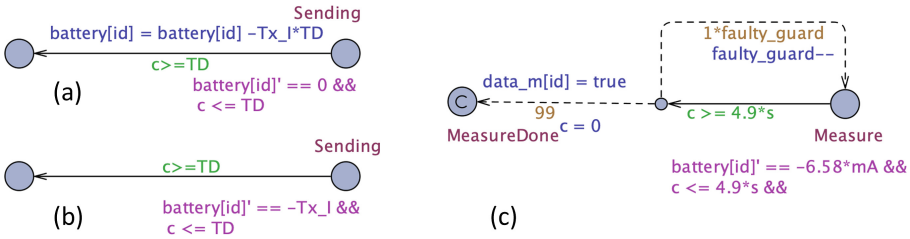

Fig. 8. Discrete and continuous behaviour (a,b) and Probabilistic choice (c) in Uppaal SMC

Figure 8 depicts a segment of our Uppaal SMC model. Figures 8(a,b) showcase discrete and continuous behaviours modelling. We update environment variables (Battery clock) after task completion to simulate discrete behaviour, considering the time spent at that location. Uppaal SMC supports ordinary differential equations to model continuous variable evolution while staying at a specific location. Figure 8(c) illustrates the implementation of a probabilistic choice for stochastic simulation in Uppaal SMC. The dotted line represents a probabilistic choice, where the model selects the next action based on probability. In our model, there is a 1% likelihood that the system measurement might be inaccurate. In this scenario, the model reverts to the measuring location. Otherwise, it will proceed to the following location.

Overall, Uppaal SMC is a versatile tool that provides comprehensive capabilities for analysing a wide range of real-time systems with different behaviours and uncertainties. One limitation of Uppaal SMC is that its continuous simulation is not as comprehensive as specialised tools like MATLAB/Simulink.

ns-3: ns-3 is a flexible and versatile network simulator that supports various types of simulation [13]. It can perform continuous simulation through event-based modelling, approximating continuous behaviour using small time steps. As a discrete-event simulator, ns-3 follows strict event scheduling for discrete simulation, making it suitable for modelling systems with specific time intervals for events.

In ns-3, we use Random Number Generator (RNG) (a built-in class) to model the probabilistic choice. It supports stochastic simulation by allowing researchers to introduce random variables and probability distributions. Additionally, ns-3 can perform deterministic simulation, where researchers can control the sequence of events and verify the deterministic properties of communication networks and protocols. It offers different algorithms to generate deterministic random variables.

While ns-3 offers a wide range of capabilities, it is important to note that continuous simulation in ns-3 is less comprehensive than in specialised continuous simulation tools, and stochastic simulation might require more manual intervention and configuration.

MATLAB/Simulink: Simulink is a robust simulation and modelling environment that extends the capabilities of MATLAB to support continuous, discrete, stochastic, and deterministic simulation [6]. It excels in continuous simulation by providing a graphical interface to model and simulate dynamic systems described by differential equations. Simulink's solvers can numerically solve these equations, allowing for the simulation of continuous behaviour over time. Additionally, researchers can use it to perform discrete simulations by specifying the sample time of blocks in the block diagram, enabling the simulation of systems with specific time intervals for events. It also supports stochastic simulation by allowing the introduction of random variables and probability distributions, and it can perform a deterministic simulation with precise control over the sequence of events. In this project, we use discrete modelling using an integer clock and schedule all events based on that, and we use a random variable to model stochastic behaviour. MATLAB/Simulink model is more abstract and simple in our case; however, the author [6] claims that building complex models in MATLAB/Simulink might require more time and effort than programming-based approaches.

6.2 Simulation Terms (Memory Consumption and Simulation Time)

The same model's memory consumption and simulation time can vary depending on the tool used. Figure 9 compares the tools regarding memory consumption and execution time while simulating the case study. ns-3 has less memory consumption, while Uppaal SMC has the shortest execution time. MATLAB has the lowest memory consumption but the longest execution time.

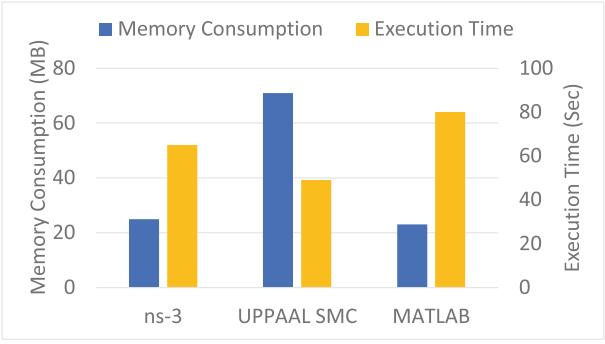

Fig. 9. Memory Consumption and Simulation time

System Configuration Details: The simulations for all models were conducted on a local machine with the following specifications: a MacBook Pro

(2019) workstation equipped with 16 GB 2133 MHz LPDDR3 memory and a 2.4 GHz Quad-Core Intel Core i5 processor. The machine ran macOS Monterey (Version 12.3.1) as the operating system.

6.3 General Comparison

This section provide a comprehensive comparison of the tools, with an abstract visual representation in Fig. 10 and a tabulated summary in Table 1. The Table presents how we categorise the different aspects of modelling a system in the tools focused on in this study; we've given them scores ranging from 0 to 10 (where 0 means challenging to use, and 10 means most accessible to use).

Expertise Required to Model: Uppaal SMC focuses on formal modelling and verification, making it suitable for researchers with a strong background in formal methods and automata theory. It also required a basic level of C++ to model actions behaviour. On the other hand, ns-3 demands a higher level of programming expertise in C++ and the core architecture of the network and protocols for developing network simulations. MATLAB/Simulink, in contrast, provides a higher level of abstraction and requires less programming expertise.

Other Expertise: Modelling in ns-3 only requires only good programming expertise. Uppaal SMC needs a good knowledge of automata models with the basic concept of programming to design a system model, and MATLAB/Simulink requires familiarity with Simulink's interface and its blocks.

Graphical User Interface (GUI) Support: Uppaal SMC provides a user-friendly GUI that simplifies formal modelling and verification tasks, allowing users to design and visualise timed automata models. In contrast, ns-3 does not have a built-in GUI, and users must write network simulations using C++ or Python, which requires advanced programming expertise. MATLAB/Simulink provides a complete GUI environment allowing users to visually represent complex system models using blocks and connections. This user-friendly interface benefits researchers with an engineering or numerical analysis background.

Availability of Good Online Documentation: Online network analysis and simulation documentation for ns-3 and MATLAB/Simulink is more detailed and readily available than for Uppaal SMC. The dedicated networking focus and their active community provide comprehensive tutorials and user guides. Many built-in libraries and baseline examples are also available to build the basic structure of the network and standard communication protocols. The specialised focus of Uppaal SMC on formal modelling and verification may result in limited specific documentation for network analysis and simulation tasks. But it provides a good user guide and online support group for efficient model design.

Table 1. Comparison of Uppaal SMC (U/S), ns-3(N) and MATLAB/Simulink(M/S)

		U/S	ns-3	M/S
Programming Expertise				
No	10	7	3	7
Basic (Conditions, loops, function)	7			
Expert (Classes and structures, Pointers, Memory Management)	3			
Other Expertise				
Only programming skill required	10	3	10	3
Multiple languages required	7			
Other modelling technique	3			
GUI Support				
Advanced	10	10	0	10
Moderate	5			
No	0			
Availability of good online documentation				
BaseLine Examples	10	5	10	10
Built-in libraries	8			
Strong Literature Review	6			
Week Literature Review	5			
Online Support Group	4			
User Guide	2			
No Help	0			
Scalability				
Allowed large scale network simulations	10	7	10	7
Allowed but simulation time increase more frequently	7			
Partiacialy allowed	3			
Not allowed	0			
Limitations in model Design				
Allowed most of the operation in networks	10	5	10	10
Limited tool set	5			
Not allowed	0			
Result Visualisation				
Optimisation of graphs	10	7	3	10
Graphical Representation & Text Data output	7			
Text Data	3			
Model Verification				
Rich proper language	10	10	4	4
Automatic analysis	8			
Model Validation for function requirements	6			
Ad Hoc Test Cases as Script	4			
Verbose output log enabling extend analysis	2			
Documentation of model / Representation of model				
Graphical representation	10	10	5	10
Document in the form of blocks or class diagram	5			
Code based representation	3			

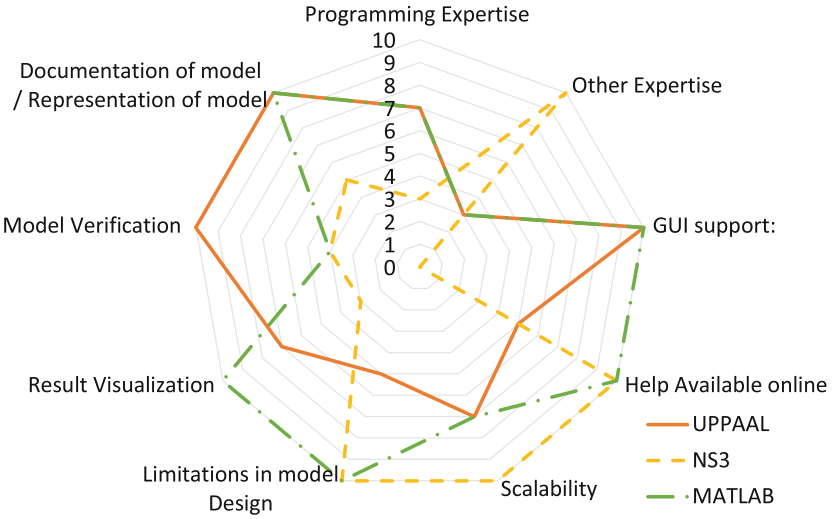

Fig. 10. General Comparison of Uppaal SMC, ns-3 and MATLAB/Simulink

The Sigfox module wasn't available in ns-3 for this case study, so we modified and customised a LoRaWan module for Sigfox. We built the models for MATLAB/Simulink and Uppaal SMC from scratch.

Scalability: Scalability was not within the scope of our case study, but in general, ns-3 is highly scalable and optimised for large-scale network simulations, making it an excellent option for simulations requiring much scalability. Uppaal SMC is well-suited for small to medium-sized systems, but its scalability may be limited for large and complex systems. Scalability to the very large system can be achieved by exploiting the possibility in the newly released UPPAAL 5.0 of linking to external compiled C-code (this was successfully done for the simulation of a model of Nothern Jutland of COVID-19 comprising more than 1 million components) [2]. The scalability of MATLAB/Simulink is generally good for small to moderate-sized models, but it may have performance limitations for large-scale simulations involving complex mathematical computations.

Limitations in Model Design: Both ns-3 and MATLAB/Simulink exhibit versatility in network analysis, allowing for robust modelling without significant design limitations. On the other hand, UPPAAL employs clock variables in timed automata, which evolve continuously through time derivative rates. It only supports integer values in clock rates and clock conditional statements.

In this case study, the node takes 4.9 s to gather measurements in the measuring state. However, Uppaal SMC doesn't support conditions with floating-point numbers, so we adjusted the base time clock from seconds to desi-seconds. This

conversion allowed us to represent the condition as 49 desi-seconds instead of 4.9 s. Simulating the node becomes more complicated when the base clock needs to be reduced to micro or nanoseconds to avoid floating point numbers, and the simulation time is in multiple years.

Result Visualisation: Uppaal SMC provides result analysis and visualisation through its built-in plot composer tool. However, it cannot zoom in on specific sections of the simulation graph for detailed analysis. In comparison, the ns-3 provides simulation results in a text data stream format that requires additional software like Gnuplot for graphical representations. We use MATLAB to visualise ns-3 simulation results and some simulation results from Uppaal SMC to highlight a specific plot section. However, MATLAB/Simulink stands out with its extensive visualisation functions and exceptional versatility in handling diverse simulation types, making it highly suitable for a wide range of result analysis and visualisation tasks.

Model Verification: Uppaal SMC is a specialised tool designed explicitly for formal model checking, making it a powerful choice for testing model correctness. It uses statistical model-checking techniques to verify if the system behaviour meets predefined requirements. This tool automatically checks for probabilistic systems' reachability, safety, and liveness properties, providing valuable insights into the model's correctness.

```
Pr[<=100*days](<> Sensors(0).Listening && r>20)          (1)
```

Equation 1 illustrates a query in Uppaal SMC, verifying the model's compliance with the requirement that it start listening (to receive a message) 20 s after sending an up-link. In this query, the clock variable `r` is a timer initiated upon transmitting the up-link frame. The query executes the model for hundred days to compute the possibility of reaching the state with the greater value of `r`. In this case, the calculated probability is zero, so the model is correct.

In contrast, ns3 and Simulink do not offer statistical model-checking capabilities like Uppaal SMC. However, researchers can still create test cases, analyse simulation results, and validate the system's behaviour against expected outcomes.

Documentation of Model/Representation of Model: We need good documentation of a designed model to present in front of others for many reasons (Publishing, collaborating or proving). The documentation must be graphical and more generic to make it more understandable for people from all domains.

Comparatively, the model in Uppaal SMC and MATLAB/Simulink has a graphical representation of the system's behaviour, and it is easy to convert into documentation in the form of states and actions.

7 Conclusions and Future Work

Uppaal SMC is a powerful tool for verifying early-phase design and identifying vulnerabilities in distributed communication systems. Its statistical model-checking capabilities ensure the correctness and reliability of the model, particularly benefiting users with expertise in Automaton models. ns-3 stands out for large-scale network simulations. Its event-driven architecture and visualisation modules are ideal for extensive network simulations and analysis. It is particularly well-suited for users proficient in C++ programming with a solid understanding of communication networks and protocols, enabling them to perform comprehensive network analyses. MATLAB/Simulink is versatile for simulating and testing communication networks. It has powerful simulation capabilities and visualisation functions, making analysis and visualisation of results easy. It's flexible for different types of simulations and result analysis. It is more suitable for users who know basic C code function blocks.

Future work can explore the co-simulation of Uppaal SMC, ns-3, and MATLAB/Simulink to leverage their strengths and enhance overall simulation capabilities. Integrating these tools can offer a more comprehensive approach to network simulation and analysis, allowing researchers to tackle complex scenarios more effectively.

References

1. Bakni, M., Chacón, L.M.M., Cardinale, Y., Terrasson, G., Curea, O.: WSN simulators evaluation: an approach focusing on energy awareness. arXiv preprint arXiv:2002.06246 (2020)
2. Bilgram, A., et al.: An investigation of safe and near-optimal strategies for prevention of covid-19 exposure using stochastic hybrid models and machine learning. Decis. Anal. J. **5**, 100141 (2022). https://doi.org/10.1016/j.dajour.2022.100141, https://www.sciencedirect.com/science/article/pii/S2772662222000728
3. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: UPPAAL SMC tutorial. Int. J. Softw. Tools Technol. Transfer **17**(4), 397–415 (2015)
4. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B., Sedwards, S.: Runtime verification of biological systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012. LNCS, vol. 7609, pp. 388–404. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34026-0_29
5. Gnanaselvi, S.: A study on various simulation tools for wireless sensor networks. Int. J. Eng. Res. Manag. (IJERM) **5**, 1–3 (2018)
6. Knight, A.: Basics of MATLAB and Beyond. CRC Press, Boca Raton (2019)
7. Kochhar, A., Kaur, P., Preeti.: Simulation platforms for wireless sensor networks: how to select?. In: Tuba, M., Akashe, S., Joshi, A. (eds.) Information and Communication Technology for Sustainable Development. Advances in Intelligent Systems and Computing, vol. 933, pp. 539–545. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-7166-0_54
8. Korala, H., Georgakopoulos, D., Jayaraman, P.P., Yavari, A.: A survey of techniques for fulfilling the time-bound requirements of time-sensitive IoT applications. ACM Comput. Surv. **54**(11s), 1–36 (2022)

9. Naeem, M., Albano, M., Larsen, K.G., Nielsen, B., Høedholt, A., Laursen, C.Ø.: Modelling and analysis of a sigfox based IoT network using uppaal SMC. IEEE Sens. J. **23**, 10577–10587 (2023)
10. Naeem, M., Albano, M., Larsen, K.G., Nielsen, B., Høedholt, A., Østergaard Laursen, C.: Battery aware analysis of sensor networks in uppaal SMC. In: 2021 10th Mediterranean Conference on Embedded Computing (MECO), pp. 1–6. IEEE Budva, Montenegro (2021)
11. Naeem, M., Albano, M., Magrin, D., Nielsen, B., Guldstrand, K.: A sigfox module for the network simulator 3. In: Proceedings of the WNS3 2022, pp. 81–88 (2022)
12. Nayyar, A., Singh, R.: A comprehensive review of simulation tools for wireless sensor networks (WSNS). J. Wirel. Netw. Commun. **5**(1), 19–47 (2015)
13. Riley, G.F., Henderson, T.R.: The ns-3 network simulator. In: Wehrle, K., Güneş, M., Gross, J. (eds.) Modeling and Tools for Network Simulation, pp. 15–34. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-12331-3_2
14. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 202–215. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_16
15. Sharma, R., Vashisht, V., Singh, U.: Modelling and simulation frameworks for wireless sensor networks: a comparative study. IET Wirel. Sens. Syst. **10**(5), 181–197 (2020)
16. Sigfox: Sigfox Radio specifications, February 2020. https://storage.googleapis.com/public-assets-xd-sigfox-production-338901379285/abaedf62-56de-402e-93c3-3a9c10a1cb49.pdf
17. Xian, X., Shi, W., Huang, H.: Comparison of Omnet++ and other simulator for WSN simulation. In: 2008 3rd IEEE Conference on Industrial Electronics and Applications, pp. 1439–1443. IEEE (2008)