Validation of ESDS Using Epidemic-Based Data Dissemination Algorithms

Loic Guegan*, Issam Raïs*, Otto Anshus*,
Department of Computer Science, UiT The Arctic University of Norway, Tromso, Norway*
Corresponding authors: loic.guegan@uit.no, issam.rais@uit.no

***Abstract*—The study of Distributed Systems (DS) is important as novel solutions in this area impact many sub-fields of Computer Science. Although, studying DS is not an easy task. A common approach is to deploy a test-bed to perform a precise evaluation of the system. This can be costly and time consuming for large scale platforms. Another solution is to perform network simulations, allowing for more flexibility and simplicity. Simulators implement various models such as wired/wireless network models and power consumption models.**

Extensible Simulator for Distributed Systems (ESDS) is a simulator designed for simulation of systems that include edge platforms, namely Internet of Things (IoT), Wireless Sensor Networks (WSN) and Cyber-Physical Systems (CPS). ESDS uses coarse-grained (flow-level) models for wired and wireless networks, and provides nodes power consumption models. However, to ensure accurate predictions, these models must be validated.

In this paper, we propose to validate the flow-level wireless model and the power consumption model of ESDS using epidemic-based data dissemination simulations. We show that ESDS has similar predictions than another validated flow-level network simulator, in terms of network performance and energy consumption.

***Index Terms*—network simulation, distributed systems, data dissemination, Internet of Things, Wireless Sensors Networks, Cyber-Physical Systems, Energy Consumption**

I. Introduction

Distributed Systems (DS) consist in the combination of distributed computers that communicates through a distributed network. The nature of the distributed network varies across applications. It can be wired (Cloud Computing, High Performance Computing etc.), wireless (WSN, IoT, CPS etc.) or both. With the popularization of WSN and IoT technologies, research on DS based on wireless and heterogeneous (wired and wireless) networks is crucial and targets many applications such as energy efficiency [1], node clustering [2], data dissemination [3], monitoring [4, 5, 6] etc.

Depending on the use-case, setting up such platforms can be difficult. As an example, monitoring the Arctic Tundra has been a critical need for the COAT initiative [7]. In [8], authors investigate an update system for mostly sleeping CPS nodes in the context of the monitoring of the AT. This type of work could greatly benefit from using simulation. However in this context, having access to the real environment and conducting real node deployments is time consuming and costly. Depending on the studies, performing real node deployment is not always required and many solutions can be studied through emulation [9] or simulations [10, 11]. It allows researchers to save time and potentially money.

Emulation replicates the entire deployment environment which includes software (SW) and hardware (HW). However, achieving emulation is difficult in scenarios with high variability in SW and HW. Performing experiments with exotic HW is not always possible with emulation.

To work around these issues, researchers can rely on network simulations. Compared to real deployments, simulation provides more flexibility. Researchers are then not limited by the physical platform. In addition, simulation allows for more exhaustive studies compared to real deployments and achieve better reproducibility compared to emulation [9]. However, to ensure that results from simulations are reliable and usable, their models must be validated.

In this paper, we propose a first set of validation experiment for the Extensible Simulator for Distributed Systems (ESDS). The aim is to provide validated models that are usable for research purposes. This validation targets the ESDS's event scheduler, its wireless communications performance model and its node power consumption model. To conduct this validation, an implementation of two epidemic-based data dissemination algorithms is studied through ESDS. A comparison between the results obtained with ESDS and a validated simulator from the literature is performed. We also show that simulation permits a fine grain analysis and comparison of dissemination algorithms.

This paper is organized as follows. Section II presents the state of the art of existing validation methods for network simulators and motivate the work. Section III presents the overall architecture of ESDS. Section IV details the validation process that is used to validation our simulator. Section V and VI presents the validation results for the *PF* and *SIR* algorithms. Section VII discuss about the overall results. Finally, Section VIII concludes the work.

II. Related Work

A. Validation methods

In the literature, the first validation method consists in reproducing the simulated platform and scenarios by deploying real nodes [12] where relevant data concerning the behavior of the system is collected. Obtained results are compared with results from simulation of the system. This approach compares the ground-truth to the simulated-truth. It assesses how accurate the simulator is. However, it requires access to a test-bed of nodes. Such a test-bed may not be available or be costly and time-consuming to set-up. In [13, 14], the

authors validate the ns-3 implementation of TCP congestion algorithms by using real experiment results. In [15], real experiments are used to validate the performance and energy consumption prediction models of Message Passing Interface (MPI) applications available in SimGrid.

The second validation method uses an analytical model of the system. This model can be used to produce data about the system. The results obtained from the analytical model are compared with results from the simulation. This approach compares the analytical model-truth to the simulated-truth. When the system is very simple, accurately reflecting its behavior with an analytical model is simple. However, for more complex systems, an analytical model can be difficult to construct. In [16], authors use an analytical model to validate the 802.11ax OFDMA implementation of ns-3. In [17], authors validate the LTE device-to-device model of ns-3 using an analytical model.

The third validation method consists in reproducing the simulations in another network simulator [18]. Results from the non-validated simulator are compared with the results from a validated simulator. This approach assumes that the reference simulator has been validated accurately. This is a low-cost and efficient way of documenting the accuracy of a simulator. In [19], authors validate the wired network energy models of the SimGrid simulator using the ECOFEN module of ns-3. In [20], the Wi-Fi model of ns-3 is used to validate a flow-level Wi-Fi model implemented in SimGrid. In [21] the authors validate the 802.11 model of Komondor using ns-3.

B. Motivations

Despite being a critical need for scientific research, validation of network simulator models are not always performed even though they are widely used in the literature [22]. To ensure accurate predictions of ESDS models, this work propose to validate ESDS using the third approach: comparing ESDS to another validated simulator.

The reference simulator used for the validation is SimGrid [23]. Such as ESDS, SimGrid provides similar network models granularity (flow-level), and a node power consumption model. SimGrid has been widely used and validated [24, 25]. Consequently, it is a suitable simulator to use as reference.

III. Simulator architecture and design

The ESDS simulator comprises two major components: 1) The Simulation Orchestrator (SO) 2) The Simulated Nodes (SN).

The SO is the main component responsible for implementing the main simulation loop. It initialises the network (e.g bandwidths and latencies), collects and processes the events (e.g communications, turn on/off nodes). First, the SO instantiates the network platform (i.e description of the network characteristics) defined by the user. Second, it binds the SN implementations to each simulated nodes. It then instantiates every SN and starts their concurrent executions. Finally, the SO orchestrates SNs by means of synchronization mechanisms. It collects and processes SN events, until no more events are generated by the SNs.

TABLE I
Simulated Nodes API calls

Call	Description
`send()`	Send data
`sendt()`	Send data with a timeout
`receive()`	Wait for and fetch incoming data
`receivet()`	Wait for and fetch incoming data with a timeout
`wait()`	Wait for a specific amount of simulated time
`wait_end()`	Wait until the end of the simulation
`log()`	Print a message in the SO standard output
`read()`	Read in the SO current state
`turn_on()`	Turn the node on
`turn_off()`	Turn the node off

The network platform comprises network interfaces defined by two matrices: a latency matrix $\boldsymbol{L}$ and a bandwidth matrix $\boldsymbol{B}$. As in [26], the duration T_c for communication c from SN i to SN j, containing n bytes of data can be defined as:

$$T_c = \frac{n}{\boldsymbol{B}_{(i,j)}} + \boldsymbol{L}_{(i,j)} \tag{1}$$

Note that if $\boldsymbol{B}_{(i,j)} = 0$, SN i and SN j are considered out of range. ESDS does not implement any network protocol such as TCP. If specific communication protocols are required, they can be implemented in the SN, as it is done in real networks. It is up to the user of ESDS to provide such implementation, for example through plugins (detailed later). This simple network platform representation allows the user to have fine control over data transmitted during communications.

A. Simulated Node (SN) API

As in most network simulators, ESDS provides a common API for the implementation of each SN. In our case, this API performs requests to the SO in order to generate and/or consume events. The API follows existing communication mechanisms from other network simulators. However, we strive to keep it small, while being able to implement most DS scenarios. This API is detailed on Table I. It comprises mechanisms to perform communications, changing nodes states (on and off), simulation wait calls, accessing to the simulator state and logging.

These API calls are meant to provide mechanisms to implement the vast majority of scenarios that are part of CPS, WSN and IoT. Nevertheless, the user is in charge of implementing the use-case specific features. If these features are meant to be reused, they can be implemented in ESDS, as a *SN plugins*.

B. SN Plugins

Aside from the presented simulation API, additional features may be required to study phenomena that are part of several research experiments. The SN energy consumption is an example of such feature. Since we want to keep the standard API small, these additional features are implemented by means of *SN plugins*. A *SN plugin* is a specific node feature that is implemented only once and can be reused across several

nodes and experiments. The user can extend the features of ESDS by creating its own plugins and use them in the SN implementations.

Currently, ESDS provides a plugin that model the node energy consumption. It is based on the power state model [27]. This approach consists in approximating the energy consumption of a node based on the hypothesis that a node passes throughout several discrete power states during its up-time. Hence, a fixed power value is assigned to each state. The power state plugin of ESDS allows the user to define an arbitrary number of power states. Each SN switches between its available power states according to the implementation defined by the user. To account for the energy consumed by the network interfaces, power state values can be attached to communication interfaces. The plugin currently defines three power states per interface namely P_{tx} (transmission), P_{rx} (reception) and P_{idle}.

In this section we presented the simulator architecture and design, including its main features. To be accurate, the models described above must be validated carefully. The next section presents the approach followed for their validation.

IV. Validation Protocol

A. Scenarios

The core of a discrete-event network simulator comprises an event scheduler [28] and a network model. During the simulation of dense network scenarios, thousands of events (e.g communications) can be generated. The event scheduler is in charge of ordering, prioritizing and processing these events. The network model is used to predict the communication duration. To ensure a correct event scheduling and accurate communications duration predictions, it is important to validate both the event scheduler and the network model.

Catching event scheduling and network performance errors in predictions can be done by simulating scenarios with cascading events that include communications. Data dissemination [29, 30] is of interest in distributed systems and can generate cascading events. Hence, we compare results from SimGrid and ESDS by simulation of data dissemination algorithms.

In [30], authors study epidemic-based data dissemination using a WSN test-bed. They study two data dissemination algorithms: Probabilistic Flooding (PF) and the Susceptible-Infected-Recovered (SIR) model.

In a wireless context, PF data dissemination algorithm works as follow: to disseminate data, an initiator node sends a message to all its neighbors. To mitigate re-transmissions, each neighbor forwards the message received from the initiator node based on a probability q. The smaller q is, the less likely it is for a message to reach all nodes of the system. On the other hand, a greater value of q allows to reach more nodes in the network, but leads to more re-transmissions.

In a wireless context, a SIR data dissemination algorithm defines 3 node states. *Susceptible*: a node can be infected with a probability β upon receiving a message. *Infected*: at each unit of time ϵ, the node forwards the message. *Recovered*: after being infected, a node can be recovered at each ϵ unit of time with a probability α. Upon recovery, the node stops forwarding the message. The Algorithm 1 details the implementation of a SIR algorithm on a node starting in the *Susceptible* state. The spread of the message is driven by the reproduction number $\tau = \frac{\beta}{\alpha}$, which determines the expected number of infections generated by the first infected node. Hence, the message is expected to spread when $\tau \geq 1$.

Algorithm 1 Susceptible-Infected-Recovered

Require: $\alpha, \beta \in [0, 1]$
Require: $state \in \{susceptible, infected, recovered\}$
1: $state \leftarrow susceptible$ ▷ Initiator node starts in the $infected$ state
2: **while** $state$ **is not** $recovered$ **do**
3: $message \leftarrow$ Receive() ▷ Synchronous call
4: **if** BernoulliTrial(β) **then**
5: $state \leftarrow infected$
6: **while** $state$ **is** $infected$ **do**
7: Send($message$) ▷ Synchronous call
8: **if** BernoulliTrial(α) **then**
9: $state \leftarrow recovered$
10: **end if**
11: **end while**
12: **end if**
13: **end while**

To perform the validation of the event scheduler and the network performance model, PF and SIR algorithms are implemented in ESDS and SimGrid. The PF implementation is combined with the power states plugin to perform its validation. Note that the energy consumption study is presented for PF only, due to a lack of space. Since the simulated communications are wireless in our scenarios, only the wireless model of ESDS is validated in this work.

B. Metrics

In our study of data dissemination algorithm, two important network related metrics are considered (extracted and derived from [30]): (1) The network coverage represents the proportion of nodes in the network that received the message disseminated by the initiator node. Lets define n, the number of nodes in a platform and m, the number of nodes that actually received the disseminated message ($n \geq m$). The network coverage is then defined as $C = \frac{m}{n} \in [0, 1]$. (2) The number of transmissions that occur during the data dissemination, noted N_{tx}, measures the dissemination efficiency. Indeed, for an equal network coverage and in a similar context (i.e each transmission has the same characteristic and occurs in the same platform), a dissemination algorithm with less transmissions should be considered, to have a more efficient usage of the network.

The last metric used for the validation experiments is the SN energy consumption, noted E_{sn}. It quantifies the energy consumed by the nodes, and allows for comparison between the power state models in SimGrid and ESDS.

TABLE II
SIMULATION PARAMETERS

Parameters	Value	Description
n	25	Number of nodes
P_{idle}	$0.4W$ [31]	Idle power consumption
P_{tx}	$0.16W$ [31]	Transmitting power consumption
P_{rx}	$0.16W$ [31]	Receiving power consumption
BW	$50kbps$ [31]	Bandwidth
L	$0s$ [31]	Latency
r	1 000	Number of runs per scenario
	Probabilistic Flooding	
q	$\{0.2, 0.3, 0.4, 0.5, 0.6\}$	Forward probability
	Susceptible-Infected-Recovered	
A_1	$\{0.6\}$	α set 1
A_2	$\{0.5\}$	α set 2
B_1	$\{0.2, 0.4, 0.5, 0.55, 0.6, 0.65, 0.7\}$	β set 1
B_2	$\{0.7, 0.8\}$	β set 2
(α, β)	$A_1 \times B_1 \cup A_2 \times B_2$	Values of α (recovery probability) and β (infection probability)

C. Simulation Parameters

The simulation parameters are reported in Table II. The simulated platform is similar to the one used in [30]. Its complete adjacency matrix, termed T, is available in [32]. The platform consists of 25 nodes that communicate through wireless technologies. To stay within plausible technologies at the edge, we assume that all nodes communicate with a bandwidth of 50kbps, similar to the performance of LoRa, as detailed in [31]. We also instantiate the power states model of SimGrid and ESDS with values extracted from [31].

Both network simulators, ESDS and SimGrid, are deterministic. That is, for the exact same inputs they produce the same outputs. Each scenario is run with 1 000 different seeds used by a random number generator. It impacts the forward of the message for both *PF* and *SIR*, and the modification of the state for *SIR* (susceptible, infected or recovered). All in all, 700 000 simulations are carried out.

V. VALIDATION: PROBABILISTIC FLOODING

A. Network Coverage

Figure 1 presents the network coverage metric for *PF* as a function of the initiator node and the forwarding probability q. Note that ESDS and SimGrid provide the exact same results. Hence, the results are overlapping in the figure.

Results for each initiator show a similar trend: the network coverage increases linearly with the forward probability q. However, depending on the initiator node, the impact of q varies. A key factor for this observed variation is the node degree of connectivity (or node degree), that corresponds to the number of nodes directly reachable (i.e through a single hop) from a given node. The network coverage is higher for scenarios having initiator nodes with higher degrees, compared to scenarios with lower degrees. One direct reason is that initiator nodes with a high degree are able to cover a large part of the network at the beginning of the simulation.

Scenarios with high initiator node degrees have lower standard deviations. This is clearly visible when comparing the results for initiator nodes 8 and 22. For initiator nodes with a low degree of connectivity (e.g node 8,9 and 10), the chance of infecting all nodes is lower. However, if a single transmission can reach a node having a high degree of connectivity (e.g node 0 and 18), it significantly increases the network coverage, leading to a higher standard deviation. For scenarios with initiators having a high degree of connectivity, the opportunity of improving the network coverage is lower since they already perform well, leading to a lower standard deviation. Consequently, starting the data dissemination with initiator nodes having a high degree of connectivity consistently leads to high network coverage.

Figure 2 presents the results for the number of transmissions (N_{tx}) when simulating *PF*. It shows a correlation between the forward probability q and the average number of transmission N_{tx}, which characterises *PF*. The greater q is, the higher N_{tx} is. The lower the network coverage is, the more important it is to have high value of q for a sufficient network coverage. It confirms the importance of the initiator node degree when using *PF*. The standard deviations are not presented for readability, however, they are exactly the same for ESDS and Simgrid.

Simulation for SimGrid and ESDS provide the same results for the network coverage and N_{tx}, over all simulations. For readability, Figure 1 and 2 present results for both simulators without distinction, as there is a perfect overlap. This implies that both simulators scheduled the same communication events with similar network performance, providing the same outcomes. Consequently, ESDS has the same prediction abilities than SimGrid in terms of event scheduling and network performance using *PF*.

B. Energy Consumption

Figure 3 presents the node energy consumption reported by the power state plugin of ESDS using *PF*. The standard deviations are not presented for readability, and they are exactly the same for both simulators. Results from SimGrid are not provided since they correspond to results from ESDS with a constant offset of $\Delta E = 9.92 \times 10^6$ J. This offset is due to SimGrid's implementation of *PF* that follows a specific communication schedule to avoid deadlocks as explained in Section II. This leads to a larger simulated time (due to longer idle times), corresponding to an energy overhead of ΔE.

These graphs show a correlation between energy consumption E_{sn} and the initiator node degree. The higher the initiator node degree is, the larger E_{sn} is. As depicted on Figure 2, nodes with high initiator degree transmit more and thus consume more energy. In addition, larger values of q lead to higher network coverage and higher energy consumption.

Both simulators provide the same results for the E_{sn} metric in every simulated scenarios. Consequently, the power state model of ESDS is validated against SimGrid.

VI. VALIDATION: SUSCEPTIBLE-INFECTED-RECOVERED

This section presents results from simulating the *SIR* algorithm by ESDS and SimGrid.

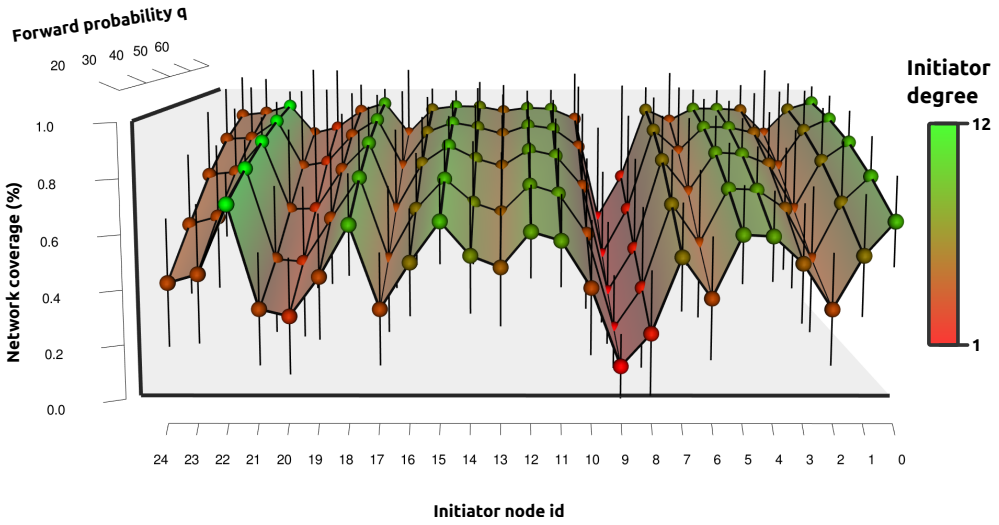

Fig. 1. ESDS results of the network coverage evaluation using *PF* as an average over 1 000 runs. The results for each initiator node is shown. The vertical bars provide insights on the standard deviation for each scenario over 1 000 runs. The degree of connectivity with the neighbors of every initiator node is highlighted with a color scale (red for low degree and green for high degree). Note that results for SimGrid and ESDS are overlapping since they are exactly the same.

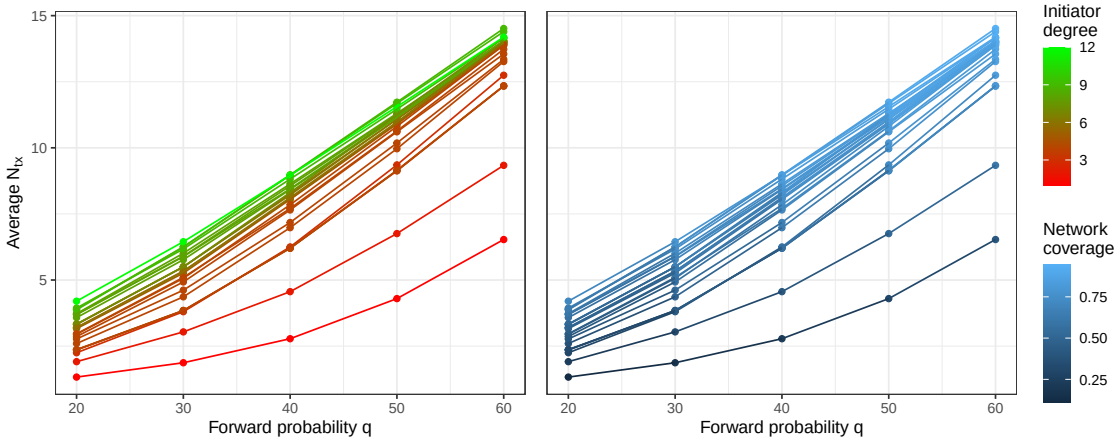

Fig. 2. Results of the N_{tx} metric evaluation using *PF*. These results are the average over 1 000 runs. Each curve is specific to one initiator node. The initiator degree and network coverage are shown using color scales. Note that results for SimGrid and ESDS are overlapping since they are exactly the same.

Figure 4 depicts the results for the network coverage simulation of *SIR*. The initiator node and the reproduction number, τ are the factors in the simulations. The degree of connectivity of the initiator node is shown with a color scale (red for nodes with a high degree and green for nodes with a low degree).

The network coverage results show that *SIR* data dissemination model is less impacted by the initiator node degree, compared to *PF*, for a given reproduction number. The network coverage is strongly affected only on extreme node degrees. For example, node 9 has a degree of 1, leading to a lower network coverage, even for large values of τ. Similar trends are visible for other nodes with low degree of connectivity (e.g nodes 8, 20 and 21).

Overall, the *SIR* data dissemination model offers more network coverage than *PF* for this specific set of reproduction number τ and network platform. Most of the scenarios result in a network coverage of more than 60%.

Figure 5 depicts results for the N_{tx} metric using the *SIR* model. The standard deviations are not presented for readability and are exactly the same for both simulators. As expected, the initiator node degree has a limited impact on the number of transmission that occurs during the simulation. Most curves are close to be overlapping. Similarly, the reproduction number τ has a limited impact on the network coverage. Compared to *PF*, the *SIR* model leads to a greater value of N_{tx}, in average (with a minimum of $N_{tx} = 20$). With the chosen values of τ, *SIR* adds between 3 to 20 times the amount of transmissions

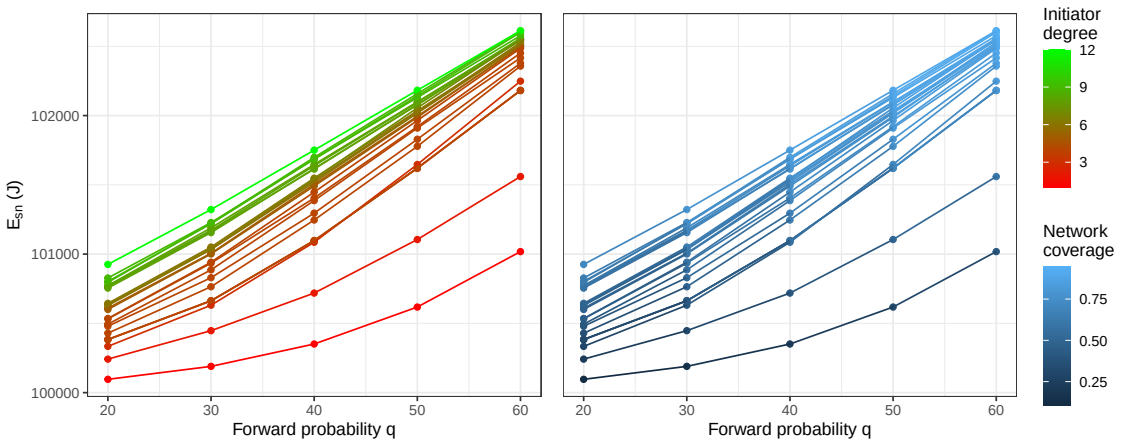

Fig. 3. Evaluation of the network platform energy consumption in ESDS and its power states plugin using *PF*. These results are the average over 1 000 runs. Results for SimGrid are not provided since they correspond to the ESDS results with a constant offset of $\Delta E = 9.92 \times 10^6$ J that is due to the limitations encounter in SimGrid for the scenario implementation.

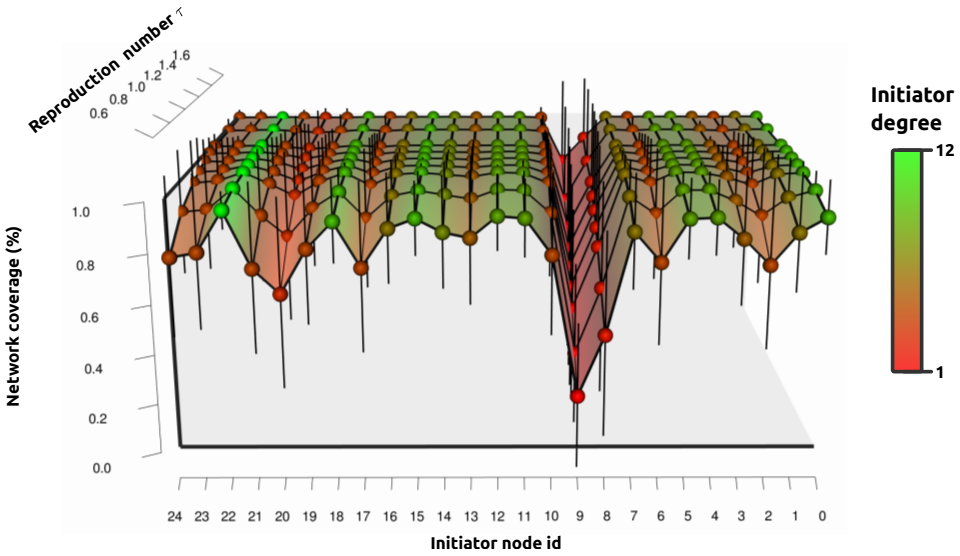

Fig. 4. Results of the network coverage evaluation using *SIR* as an average over 1 000 runs. Results for each initiator node is shown. The vertical bars provide insights on the standard deviation for each scenario over 1 000 runs. The degree of connectivity with the neighbors of every initiator node is highlighted with a color scale (red for low degree and green for high degree). Note that results for SimGrid and ESDS are overlapping since they are exactly the same.

of *PF*. Consequently, most of the nodes in the network are receiving the message spread by the sender, but at the cost of more re-transmissions.

Simulations for SimGrid and ESDS provide the same results, over all simulations using the *SIR* model. For readability, Figures 4 and 5 present results for both simulators without distinction, as there is a perfect overlap. Consequently, the event scheduler and the network model of ESDS provide similar predictions to SimGrid.

VII. Discussion

A. Data dissemination

Simulation results reveal interesting properties of *PF*. First, it is highly sensitive to the initiator nodes degree. Having a fixed value of q can lead to an inefficient data dissemination, as quantified by the network coverage and energy consumption. This phenomenon might be more significant on dynamic network platforms where the nodes degrees of connectivity are not fixed due to their mobility and reachability. Still, *PF* is able to cover a major part of the network platform (more than 75%), with relatively few data transmission in average (less than 15). In addition, scenarios with an initiator node with a high degree tend to increase the energy consumption from $q = 50$ to $q = 60$, but not necessarily improving the network coverage by a significant amount. It suggests that q could be adjusted according to the initiator node degree. A future work can include a more energy efficient data dissemination.

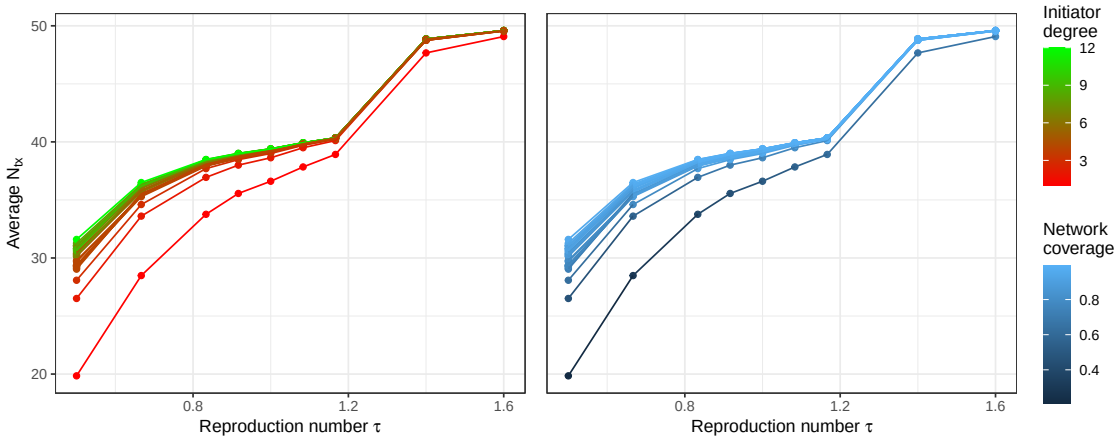

Fig. 5. Results of the N_{tx} metric evaluation using *SIR*. These results are the average over 1 000 runs. Each curve is specific to one initiator node. The initiator degree and network coverage are shown using color scales. Note that results for SimGrid and ESDS are overlapping since they are exactly the same.

The simulation study of *SIR* data dissemination model shows that, for a given set of reproduction number, *SIR* is less sensible to the initiator node degree, compare to *PF*. However, *SIR* has significantly higher N_{tx} values in average, as it generates more transmissions.

B. Validation

As explained, two epidemic-based data dissemination algorithms (*PF* and *SIR*) were implemented using ESDS and SimGrid. For both of these algorithms, network coverage C and number of transmissions N_{tx} is studied. A comparison of the power state implementation of ESDS against SimGrid is performed on the *PF* algorithm. Setting up the power state implementation of SimGrid to work with the *SIR* algorithm was time consuming. Each scenario is run with 1 000 seeds resulting in 700 000 simulations. Both simulators perform the same, as quantified by the three presented metrics. With the same inputs, they predict the same coverage, number of transmissions and energy consumption in every cases. Consequently, the wireless network model and the power state model of ESDS is validated against SimGrid.

VIII. Conclusion

In this paper, we propose a first set of validation experiment for the ESDS simulator. The event scheduler of ESDS, its wireless network performance model and its power state model are validated. We compare the simulation outcomes of ESDS to another validated network simulator called SimGrid. The simulated scenarios implement data dissemination algorithms, that generate cascading events. The results show that ESDS is able to provide the exact same predictions as SimGrid for the various metrics presented in the paper. ESDS can be reliably used to conduct energy consumption studies, as we also provide a validation of the implementation of the power state model. The validation experiments are entirely available online [33], along with the ESDS source code [34].

We show that ESDS is the proper tool to study topics like data dissemination. Compared to a real deployment [30], ESDS allows to vary additional parameters. Its results for *PF* and *SIR* reveal that the initiator node degree has a major impact on the data dissemination efficiency. They suggest that further improvements in energy efficiency and network communications performance are possible with *PF* and *SIR*.

Future works include an improvement of the validation of ESDS by providing new dimensions of validation experiments. In fact, this work do not cover the validation of its flow-level wired network model. Since studying large scale DS is important, evaluating ESDS in terms of scalability is also required. Finally, we are planning to use ESDS to perform various research studies on CPS, IoT and WSN.

References

[1] Eljona Zanaj et al. “Energy Efficiency in Short and Wide-Area IoT Technologies—A Survey”. In: *Technologies* 9.1 (Mar. 2021).

[2] Amin Shahraki et al. “Clustering Objectives in Wireless Sensor Networks: A Survey and Research Direction Analysis”. In: *Computer Networks* 180 (Oct. 2020).

[3] Issam Rais, Loic Guegan, and Otto Anshus. “Impact of Loosely Coupled Data Dissemination Policies for Resource Challenged Environments”. In: *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, May 2022.

[4] Brooke Potter et al. “Environmental Monitoring Using a Drone-Enabled Wireless Sensor Network”. In: *2019 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, Apr. 2019.

[5] Rizky Pratama Hudhajanto et al. “Real-Time Monitoring for Environmental Through Wireless Sensor Network Technology”. In: *2018 International Conference on Applied Engineering (ICAE)*. IEEE, Oct. 2018.

[6] Issam Rais et al. “UAVs as a Leverage to Provide Energy and Network for Cyber-Physical Observation Units on the Arctic Tundra”. In: *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, May 2019.

[7] Rolf A. Ims et al. *Science Plan for COAT: Climate-Ecological Observatory for Arctic Tundra*. 2013.

[8] Roberth Tollefsen and Otto Anshus. “Detecting the Arrival of an Update at Mostly Sleeping Cyber-Physical IoT Nodes”. In: *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Marina del Rey, Los Angeles, CA, USA: IEEE, May 2022. (Visited on 04/17/2023).

[9] Shrey Baheti, Shreyas Badiger, and Yogesh Simmhan. “VIoLET: An Emulation Environment for Validating IoT Deployments at Large Scales”. In: *ACM Transactions on Cyber-Physical Systems* 5.3 (July 2021).

[10] Lelio Campanile et al. “Computer Network Simulation with Ns-3: A Systematic Literature Review”. In: *Electronics* 9.2 (Feb. 2020).

[11] Adrien Gougeon et al. “Impact of Wired Telecommunication Network Latency on Demand-Side Management in Smart Grids”. In: (2021).

[12] Thanh-Hai To and Andrzej Duda. “Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA”. In: *2018 IEEE International Conference on Communications (ICC)*. IEEE, May 2018.

[13] Siddharth Gangadhar et al. “TCP Westwood(+) Protocol Implementation in Ns-3”. In: ().

[14] Mark Claypool, Jae Won Chung, and Feng Li. “BBR’: An Implementation of Bottleneck Bandwidth and Round-Trip Time Congestion Control for Ns-3”. In: *Proceedings of the 10th Workshop on Ns-3 - WNS3 ’18*. ACM Press, 2018.

[15] Franz Heinrich et al. “Predicting the Performance and the Power Consumption of MPI Applications With SimGrid”. In: ().

[16] Davide Magrin et al. “Validation of the Ns-3 802.11ax OFDMA Implementation”. In: *Proceedings of the Workshop on Ns-3*. Virtual Event USA: ACM, June 2021. (Visited on 04/13/2023).

[17] Richard Rouil et al. “Implementation and Validation of an LTE D2D Model for Ns-3”. In: *Proceedings of the Workshop on Ns-3 - 2017 WNS3*. Porto, Portugal: ACM Press, 2017. (Visited on 04/13/2023).

[18] Loic Guegan et al. “A Large-Scale Wired Network Energy Model for Flow-Level Simulations”. In: *Advanced Information Networking and Applications*. Ed. by Leonard Barolli et al. Vol. 926. Springer International Publishing, 2020.

[19] Loic Guegan et al. “A Large-Scale Wired Network Energy Model for Flow-Level Simulations”. In: (2019).

[20] Clément Courageux-Sudan et al. “A Flow-Level Wi-Fi Model for Large Scale Network Simulation”. In: (2022).

[21] Sergio Barrachina-Munoz et al. “Komondor: A Wireless Network Simulator for Next-Generation High-Density WLANs”. In: *2019 Wireless Days (WD)*. IEEE, Apr. 2019.

[22] Pedro Velho et al. “On the Validity of Flow-Level Tcp Network Models for Grid and Cloud Simulations”. In: *ACM Transactions on Modeling and Computer Simulation* 23.4 (Oct. 2013). (Visited on 01/18/2019).

[23] Martin Quinson, Cristian Rosa, and Christophe Thiéry. “Parallel Simulation of Peer-to-Peer Systems”. In: IEEE, May 2012.

[24] Kayo Fujiwara and Henri Casanova. “Speed and Accuracy of Network Simulation in the SimGrid Framework”. In: *Proceedings of the 2nd International ICST Conference on Performance Evaluation Methodologies and Tools*. ICST, 2007.

[25] Franz Christian Heinrich et al. “Predicting the Energy-Consumption of MPI Applications at Scale Using Only a Single Node”. In: IEEE, Sept. 2017.

[26] Pedro Velho and Arnaud Legrand. “Accuracy Study and Improvement of Network Simulation in the SimGrid Framework”. In: *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques*. ICST, 2009.

[27] He Wu, Sidharth Nabar, and Radha Poovendran. “An Energy Framework for the Network Simulator 3 (Ns-3)”. In: *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. ACM, 2011.

[28] Thomas J. Schriber and Daniel T. Brunner. “How Discrete-Event Simulation Software Works”. In: *Handbook of Simulation*. Ed. by Jerry Banks. John Wiley & Sons, Inc., Aug. 1998.

[29] Umesh Bodkhe and Sudeep Tanwar. “Secure Data Dissemination Techniques for IoT Applications: Research Challenges and Opportunities”. In: *Software: Practice and Experience* 51.12 (Dec. 2021).

[30] Andreana Stylidou et al. “Evaluation of Epidemic-Based Information Dissemination in a Wireless Network Testbed”. In: *Technologies* 8.3 (June 2020).

[31] Issam Raıs, Loic Guegan, and Otto Anshus. “Impact of Loosely Coupled Data Dissemination Policies for Resource Challenged Environments”. In: ().

[32] Aikaterini Georgia Alvanou et al. “CaBIUs: Description of the Enhanced Wireless Campus Testbed of the Ionian University”. In: *Electronics* 9.3 (Mar. 2020).

[33] Guegan Loic. *ESDS - Wireless Validation Experiments*. https : / / gitlab . com / manzerbredes / esds - validation - wireless.

[34] Guegan Loic. *ESDS - Extensible Simulator for Distributed Systems*. https://gitlab.com/manzerbredes/esds.