Frequency-Based Partitioning for Modular Validation of Stochastic Petri Net Digital Twin Models

Ashkan Zare
University of Southern Denmark
Odense, Denmark
zare@mmmi.sdu.dk

Sanja Lazarova-Molnar
Karlsruhe Institute of Technology
Karlsruhe, Germany
University of Southern Denmark
Odense, Denmark
lazarova-molnar@kit.edu

Abstract

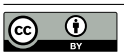
As technological advancements enable the manufacturing industry's adoption of cyber-physical production systems, the critical need for Digital Twins has also become more apparent. They allow for optimization, flexibility, and continuous improvement by replicating physical systems in a digital environment. However, they can only be utilized if their underlying models remain a valid representation of the corresponding real-world systems. As manufacturing systems and their Digital Twins grow in complexity, model validation and decomposition become increasingly challenging, particularly due to heterogeneous system dynamics. In complex manufacturing systems, different events occur at different rates, and a uniform validation approach of the corresponding Digital Twin models may fail to account for these dynamic behaviors. Depending on the rate of occurrence of different events, validation mechanisms must decide whether to preserve, recalibrate, or re-extract different parts of a model. To address this challenge, a modular validation framework was introduced, partitioning Digital Twin models into sub-models and dynamically adjusting validation policies. The effectiveness of such a framework, however, critically depends on how the model is partitioned and whether the partitions capture heterogeneous system behavior while preserving dependencies. In this paper, we present a data-driven, behavior-based approach for partitioning stochastic Petri net Digital Twin models that automatically identifies meaningful partition boundaries by analyzing reachable marking patterns and transition firing frequencies. The approach preserves inter-partition dependencies while enabling independent validation of sub-models. We demonstrate the method through a reliability-focused manufacturing case study, showing that frequency-based partitioning can capture heterogeneous dynamics while preserving system behavior.

CCS Concepts

• **Computing methodologies** → **Model verification and validation**; **Discrete-event simulation**.

Keywords

Digital Twin, Continuous Validation, Stochastic Petri Nets, Manufacturing Systems



SIGSIM-PADS '26, Vienna, Austria

ACM ISBN 979-8-4007-2648-4/26/06
https://doi.org/10.1145/3806789.3810257

ACM Reference Format:
Ashkan Zare and Sanja Lazarova-Molnar. 2026. Frequency-Based Partitioning for Modular Validation of Stochastic Petri Net Digital Twin Models. In *40th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '26), June 24–26, 2026, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3806789.3810257

1 Introduction

Today's manufacturing industry needs to respond to rapid market changes, and the digitalization of the industry takes a step towards addressing this challenge. One example of digitalization is the adoption of cyber-physical production systems and Digital Twins (DTs). DTs have emerged as a powerful tool in the manufacturing industry as near real-time virtual replicas of the physical systems to help identify inefficiencies and optimize production [6]. The effectiveness of DTs relies on the accuracy of their underlying models compared to the actual system. Therefore, as manufacturing systems evolve and go through changes, corresponding DT models need to be continuously updated to maintain their fidelity with the physical systems [8]. In essence, continuous validation of DT models is an enabler of DTs and a significant differentiating factor from traditional simulation models.

The dynamic and near-real-time nature of DTs makes traditional simulation-validation techniques unsuitable for keeping DT models' validity over time. Traditional validation methods are one-time and all-at-once approaches that lack the capability of continuously reflecting the real systems' changes in the DT models. Furthermore, complex manufacturing systems consist of different components that evolve and change with varying rates [26], making all-at-once validation approaches insufficient. Therefore, robust validation mechanisms are necessary to address the challenges of validating different components according to their needs and characteristics. From a simulation perspective, this also raises the question of how complex discrete-event models can be partitioned based on behavioral dynamics rather than purely structural criteria, while preserving correctness and inter-component dependencies.

To this end, we previously proposed a modular validation framework where DT models are partitioned into sub-models, and each sub-model is validated according to its characteristics [23]. The approach presented how each sub-model is determined to be recalibrated, re-extracted, or kept intact based on its needs. It further showed how the dependencies and the interactions among the partitioned sub-models are preserved when partitioning [25]. However,

the framework does not provide a systematic method for automatically identifying the partitions, and addressing this limitation is the focus of our work.

An automated partitioning method is essential because, as noted earlier, processes and variations in manufacturing systems happen at vastly different rates. These differences directly influence the requirements for validation. For example, consider a robotic assembly arm, depicted in Figure 1, where the robotic arm rarely breaks down. This means assembly processes performed by the robotic arm are highly frequent while faults are rare events. Therefore, when validating the corresponding DT model, the validation process of the model corresponding to the assembly process should differ from the fault model. Not only is the frequency of the two events different, but also the time it takes to gather the sufficient data is incomparable. This is a critical consideration as immature validation with insufficient data leads to false validation results and raises the risk of losing certain parts of the DT model. As a result, validation process must decide whether to preserve, recalibrate, or re-extract the different parts.

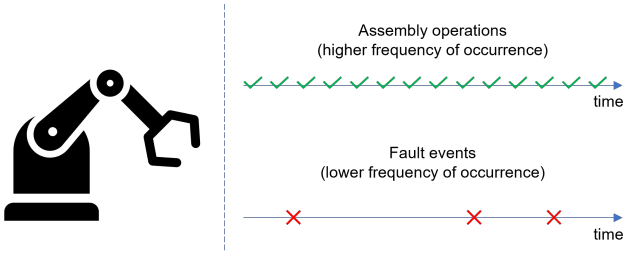


Figure 1: An example of events with varying rate of occurrence.

In this paper, we extend the previously introduced modular validation framework and propose a data-driven approach that automatically partitions DT models into sub-models based on frequency. The chosen modeling formalism of our proposed approach is Stochastic Petri Nets (SPNs) as it is well-suited for capturing and describing processes in discrete-event systems such as manufacturing systems. In SPNs, transitions represent activities and markings represent the state of a system. We utilize these components to automatically partition the models. In addition, we employ the Petri Net Markup Language (PNML) to formalize our SPN-based approach, resulting in structured representation and tool-independence [1].

The remainder of this paper is as follows. Section 2 reviews the background and related work on DT model validation, SPNs, and model partitioning. In Section 3, we present our methodology on automatic frequency-based partitioning DT models for enabling modular validation. We further evaluate our partitioning approach through a case study in Section 4. In Section 5, we discuss the challenges and limitations. Finally, Section 6 concludes the paper and discusses future research directions.

2 Background

In the following, we review the current literature on DT model validation, formally describe stochastic SPNs and the Petri net markup language, and finally give an overview of model partitioning and SPNs' decomposition.

2.1 Digital Twin Model Validation

DTs need to constantly reflect the current state of the real-world systems they are representing. Therefore, DT models need continuous and ongoing validation mechanisms to ensure the models' accuracy remains acceptable within their domains of applications [21]. Simulation model validation techniques are well-established [20], however, they cannot be directly applied to DTs as static model validation is insufficient for dynamic and evolving DT models [24].

Recent studies have explored different approaches for dynamic and ongoing validation techniques suitable for Digital Twins. Combining data collected through Internet of Things (IoT) devices and human expert knowledge, Hua et al. proposed a hybrid validation approach for DT models [8]. Furthermore, Lugaresi et al. utilized timed series analysis for an online validation approach [12]. Mertens and Denil introduced a workflow for continuous model validation and reusing existing techniques to detect anomalies [13, 14]. A modular validation framework, formalized through SPNs, was proposed by Zare and Lazarova-Molnar in which the validation is done in two phases [23]. The first phase focuses on initial validation of the model, while the second phase's focus is on continuous validation of the model. As shown in Figure 2, in the second phase, the model is partitioned into sub-models, for continuous validation. Once partitioned, the dependencies among the sub-models are preserved through introduction of interface places and token generator transitions, ensuring the dynamics of the system remain intact [25]. Finally, each sub-model is assigned a validation policy based on its needs and validated individually.

In this paper, we build on this framework and propose our data-driven approach on automatically identifying the partitions to help with continuous validation of the DT models.

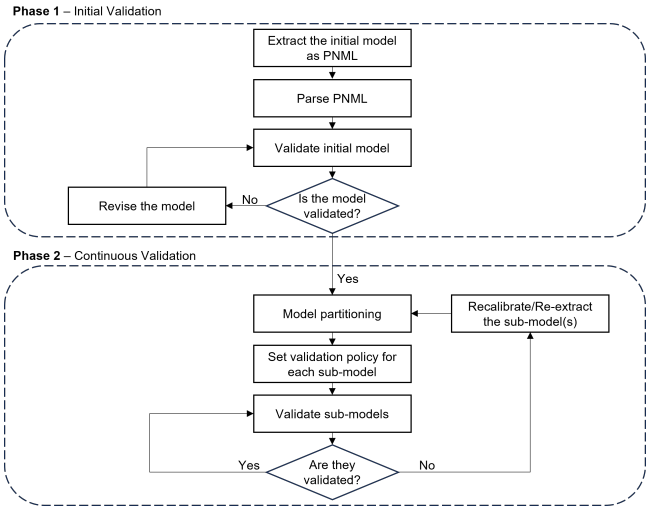


Figure 2: Modular validation framework [23].

2.2 Stochastic Petri Nets

As noted previously, the chosen modeling formalism of our approach is SPNs. Petri nets are used for description of processes in discrete-event systems, making them a great fit for our application in manufacturing systems [19]. SPNs, specifically, are an extension of Petri nets that allow transitions to be assigned probability distribution functions determining their firing delays. This characteristic enables modeling of processes in manufacturing systems as timed transitions. SPNs are defined by places, transitions and their associated guard functions, arcs, and an initial state referred to as the initial marking of the SPN [11].

$$\text{SPN} = (P, T, A, G, m_0) \tag{1}$$

where P is the set of places, T is the set of transitions (either immediate or timed with associated probability distribution functions), A is the set of input, output, and inhibitor arcs, G is the set of guard functions, and is m_0 the initial marking. In our framework, we support any class of distribution (e.g., exponential, Weibull, lognormal, or gamma) for assigning firing delays. In addition, the firing delay distributions are associated with transitions and do not depend on the number of tokens in places.

Furthermore, to describe SPNs, we use the Petri net markup language (PNML), which is a tool and platform independent universal XML-based interchange format [1]. XML-elements, also called PNML elements, are used to describe the objects of a Petri net (places, transitions, etc.). Labels are also used to provide further meaning to the objects. Figure 3 shows a graphical representation of a stochastic Petri net model and a snippet of its corresponding PNML file. The PNML syntax is tailored to stochastic Petri nets. For every transition, we have an "attribute" type that specifies whether a transition is immediate (type = "I") or timed (type = "T"). Furthermore, time transitions have a distribution element which specifies the type of probability distribution as well as its parameters. For example, in Figure 3, transition T_1 is a time transition with an exponential distribution with parameter a = 0.5.

2.3 Model Partitioning

The advancement in technologies such as cyber-physical systems and IoT has significantly increased the complexity of manufacturing systems [22]. This increased complexity is subsequently reflected in the corresponding simulation models. Decomposing large models into multiple smaller, more manageable sub-models can aid in both resource management and validation. To the best of our knowledge, partitioning has mostly been employed as a means for resource management, such as load-balancing, distribution of workload, and run-time reduction [2]. We, however, aim to utilize partitioning for modular validation of DT models [15].

Within the context of optimizing computation efficiency and reducing simulation time, GloMoSim library was developed for parallel simulation, enabling computational load distribution by partitioning and decomposing simulation models [27]. Furthermore, a dynamic partitioning algorithm for computational workload optimization was proposed by Peschlow et al. [18]. Another proposed method applies dynamic decoupling principles and relevant time scales to partition the models [16]. Clustering algorithms have also been utilized for partitioning to improve computational efficiency

```
<?xml version="1.0" encoding="UTF-8"?>
<pnml xmlns="http://www.pnml.org/version-
2009/grammar/pnml">
  <net id="SimpleWarehouse“
  type="http://www.pnml.org/version-2009/grammar/ptnet">
    <name>
      <text>Simple Warehouse Process</text>
    </name>
    <page id="page1">
      <!-- Places -->
      <place id="P1">
        <name><text>warehouse</text></name>
        <initialMarking><text>0</text></initialMarking>
      </place>
      <place id="P2">
        <name><text>finished product</text></name>
        <initialMarking><text>0</text></initialMarking>
      </place>
      <!-- Transitions -->
      <transition id="T1" type="T">
        <name><text>arrival of task</text></name>
        <distribution>
          <type>exponential</type>
          <parameters>
            <a>0.5</a>
          </parameters>
        </distribution>
      </transition>
```

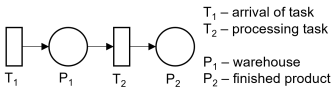


Figure 3: Graphical representation of a stochastic Petri net model and a snippet of its corresponding PNML file.

[4, 7]. Focusing on Petri nets, a partitioning algorithm utilizing net splitting operation based on valid cut sets for concurrent execution was proposed [3].

In this paper, we focus on partitioning stochastic Petri net models for validation of DT models as proposed by the modular validation framework [23]. Therefore, the focus is on behavioral partitioning based on frequency for validation rather than traditional graph partitioning methods that focus on balancing computational load [17].

Unlike traditional partitioning approaches in parallel and distributed simulation that primarily target computational load balancing, our work focuses on behavior-driven partitioning to support continuous validation, while still addressing dependency preservation that is central to distributed simulation.

3 Methodology

In this section, we present our approach to automatic partitioning of DT models based on behavioral frequency analysis to enable modular validation. The partitioning step is a critical step within our validation framework for DTs. In a manufacturing environment, data is continuously generated, and the corresponding operational DT must be validated against this new data. However, different components of the system operate and generate data at different rates and targeted validation processes are necessary to ensure DTs validity throughout their life cycle. Therefore, the main goal behind our partitioning approach is to preserve, recalibrate, or re-extract different parts of the model depending on their occurrence frequency. Therefore, our focus is on improving validation as opposed to enhancing computational load.

As noted earlier, SPNs are our adopted modeling formalism. In our approach, we utilize the structural and behavioral characteristics of SPNs, namely marking and transition firing. By analyzing markings and transition firings, we get insight into the behavior of the system. This enables us to identify meaningful boundaries for transitions, along with their associating places and arcs, yielding sub-models.

We continue the section with a brief overview of our previously proposed modular validation framework [23], and go in the details of our approach on analyzing markings and transitions, detection of interface places to preserve dependencies after partitioning, and finally generating the sub-models as PNML files for individual validation.

The proposed modular validation framework is a two-phased validation approach where in the first phase the developed initial model is validated through standard validation techniques. In the second phase, the model is partitioned into sub-models for continuous modular validation. Finally, each sub-model is validated individually depending on its needs. In this paper, we focus on the second phase of the framework, specifically the model partitioning after validating the initial model.

Figure 4 provides an overview of the partitioning with each color-coded box indicating the subsection in which the corresponding step is described (blue: Section 3.1, green: Section 3.2, orange: Section 3.3).

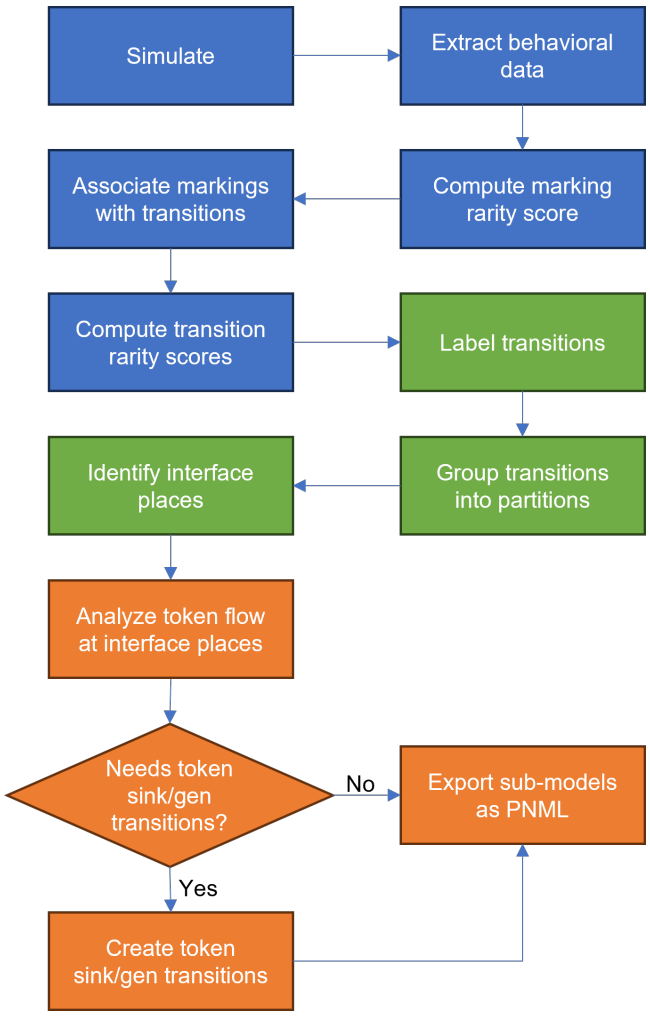


Figure 4: Overview of the automatic partitioning process.

3.1 Marking and Transition Analysis

To automatically partition the model, we utilize markings and transition firing of SPN. As described previously, a stochastic Petri net is defined as:

$$\text{SPN} = (P, T, A, G, m_0)$$

where $T = \{T_1, T_2, \ldots, T_m\}$ is the set of transitions, A is the set of arcs $A = A^I \cup A^O \cup A^H$, and m_0 is the initial marking. Furthermore, the set of all reachable markings of an SPN is denoted by $M = \{m_1, m_2, \ldots\}$.

As noted earlier in the section, we look at the behavior of the system to find the boundaries for partitioning. Different operational states have different occurrence frequency, and in SPNs, we need to look at the set of reachable markings to identify and analyze these frequently different states.

To discover the set of reachable markings of a model, we execute N independent simulation replications of the initially validated model, each for duration $t_{\max}$, to collect the necessary behavioral data. From the resulting logs of the simulation replications, we derive the following datasets, *event log* and *marking change*. The *event log* $E = \{e_1, e_2, \ldots\}$ captures the discrete events, transition firings, in all simulation replications. Each event, $e_i = (T_j, t_i, r_i)$, represents the transition T_j firing at time t_i in simulation replication r_i.

The *marking change* records the state of the system over time. Every time a transition fires, tokens are created/destroyed at places, changing the state (marking) of the system. Here, we capture how with each firing of a transition, the state of the system and its corresponding marking are affected. The *marking change* $H = \{h_1, h_2, \ldots\}$ where each observation $h_i = (m_i, t_i, d_i, r_i)$ shows the marking m_i at time t_i for duration of d_i in simulation replication r_i.

The combination of the two datasets provides us with an insight into the system's behavior. Based on the two sets, we can see how each firing of a transition changes the state of the system. Furthermore, through associating transition firings (discrete events in the system) to the markings (system's states), we can identify meaningful partitions by frequency, aiding with achieving targeted validation.

In manufacturing systems, events occur at different rates. While some events are common, others may be less occurring or rare. In Petri net terms, these different frequencies of firing of transitions result in certain markings (states) being occupied for longer periods of time than others. Therefore, the validation requirements for parts involved in rare states may differ from those involved in common states. To capture these behavioral differences, we look at the portion of time spent in different markings, as well as the number of times markings are reached to define marking rarity score. We formally define the marking rarity score as:

$$\rho(m) = \frac{1}{\sqrt{p_{\text{visit}}(m) \times p_{\text{time\_spent}}(m)}} \tag{2}$$

where:

$$p_{\text{visit}}(m) = \frac{\#\{h_j \in H : m_i = m\}}{|H|} \tag{3}$$

is the ratio of total number of visits to the marking m_i to total visits to all markings, and

$$p_{\text{time\_spent}}(m_i) = \frac{\sum d_{m_i}}{\sum d} \tag{4}$$

is the ratio of total time spent in the marking m_i to total time spent in all markings (total time of simulation replications).

The marking rarity score is inversely proportional to marking frequency, meaning common markings receive lower scores while rare markings receive higher scores. Algorithm 1 presents our approach for calculating the marking rarity scores. We aggregate the time spent and number of visits for each marking (lines 3-6) as well as total time spent and total number of visits (lines 7 and 8). Finally, we calculate the rarity score for each marking (lines 11-14). It is also important to note that we do not consider vanishing markings in our calculations (line 4).

Algorithm 1 Calculating marking rarity scores

Require: Marking observation H
Ensure: Rarity score $\rho(m)$ for each unique marking
1: Initialize time_spent$(m) \leftarrow 0$ and visit$(m) \leftarrow 0$ for all markings m
2: Initialize total counters: total_time $\leftarrow 0$, total_visits $\leftarrow 0$
3: **for** each observation $h = (m, t, d, r)$ in H **do**
4: **if** $d > 0$ **then**
5: time_spent$(m) \leftarrow$ time_spent$(m) + d$
6: visit$(m) \leftarrow$ visit$(m) + 1$
7: total_time $\leftarrow$ total_time $+ d$
8: total_visits $\leftarrow$ total_visits $+ 1$
9: **end if**
10: **end for**
11: **for** each marking m **do**
12: $p_{\text{time\_spent}}(m) \leftarrow$ time_spent(m)/total_time
13: $p_{\text{visit}}(m) \leftarrow$ visit(m)/total_visits
14: $\rho(m) \leftarrow \frac{1}{\sqrt{p_{\text{visit}}(m) \times p_{\text{time\_spent}}(m)}}$
15: **end for**
16: **return** all rarity scores $\rho(m)$

While the marking rarity score gives us an overview about the frequency of different states of the system, we still need to find the transitions associated with each marking to help identify and label the transitions. With each firing of a transition, the state of the system changes. Therefore, for each firing, we have a source marking, before firing of the transition, and a target marking, after firing of the transition.

To find the associated markings of each transition, we go through *marking change* of each transition to record the marking before and after firing of the transition. For each transition firing at time (t) in simulation replication (r) we have:

$$M_{\text{source}}(T) = m \text{ at time } (t - \delta) \text{ in replication } r \tag{5}$$

$$M_{\text{target}}(T) = m \text{ at time } (t + \delta) \text{ in replication } r \tag{6}$$

where $\delta > 0$ is a small, fixed time offset to find the markings before and after the firing. Therefore, the set of all markings associated with a transition is:

$$M(T) = \left\{M_{\text{source}}(T) \cup M_{\text{target}}(T)\right\} \tag{7}$$

The pseudocode for finding all associated markings for each transition is presented in Algorithm 2. We process each event in the log and identify the markings immediately before and after each transition firing, storing all associated markings (lines 1-7). To identify the marking, we scan the marking observation for each replication and keep track of the most recent marking (lines 9-12). Once the time exceeds the target time, the function stops and returns the last marking tracked (lines 13-17).

Algorithm 2 Finding associated markings for each transition

Require: Event log E, marking observation H
Ensure: Mapping links(T) of source and target markings for each transition T
1: Initialize links$(T) \leftarrow \emptyset$ for all transitions T
2: **for** each event $e = (T, t, r)$ in E **do**
3: $M_{\text{source}} \leftarrow$ FindMarking$(H, r, t - \delta)$
4: $M_{\text{target}} \leftarrow$ FindMarking$(H, r, t + \delta)$
5: links$(T) \leftarrow$ links$(T) \cup \{M_{\text{source}}, M_{\text{target}}\}$
6: **end for**
7: **return** links
8: **function** FindMarking(marking observation, replication, target time)
9: $m_{\text{current}} \leftarrow$ `undefined`
10: **for** each $h = (m, t, d, r_h)$ in H with $r_h = r$ **do**
11: **if** $t \leq$ target time **then**
12: $m_{\text{current}} \leftarrow m$
13: **else**
14: **break**
15: **end if**
16: **end for**
17: **return** m_{current}
18: **end function**

Finally, with the marking rarity score and the markings associated with each transition, we define a final metric, transition rarity score, that integrates both the marking rarity score and firing frequency of the transitions to label each transition for partitioning. We formally define the transition rarity score for labeling transitions as:

$$S(T_j) = \bar{\rho}_{M(T_j)} \times \rho_{\max(M(T_j))} \times (1 - \alpha \times freq_{T_j}) \tag{8}$$

where $\bar{\rho}_{M(T_j)}$ is the weighted average of rarity score of all markings associated with T_j, normalized by maximum rarity score $\rho_{\max(M(T_j))}$, $freq_{T_j}$ is the firing frequency of T_j, normalized by maximum frequency of transitions, and a weight parameter $\alpha \in [0, 1]$ controlling the impact of the firing frequency. The appropriate value for of the weight parameter is system dependent. For example, in reliability-focused systems, a high α value is suitable because the frequency of failures is low. To automatically select the suitable α, methods such as cross-validation and sensitivity analysis can be utilized. Furthermore, human-expert knowledge can also be used for selection.

Finally, the calculated metric of each transition is compared against defined thresholds to decide what the transition's label

should be. The thresholds can be predefined by experts or automatically derived using established statistical methods such as clustering [10]. Furthermore, the number of the resulting partitions is not fixed. For example, in reliability models, the distinction between frequent and rare is clear and the threshold can be used to create two partitions, rare and frequent, while in more complex models more than one threshold and therefore more than two partitions may be needed to ensure validity. However, increasing the number of partitions increases complexity. Therefore, a trade-off exists between granularity and efficiency.

3.2 Partitioning and Interface Place Detection

Once all transitions are labeled, the process of creating the partitions and identifying the interface places starts. Interface places and token generator transitions, as a means of preserving dynamics of the system, were introduced in our previous work [25]. Here, we introduce a data-driven method for identifying interface places and token generator transitions, further explained in the next subsection.

First, based on the transitions' labels, we group the transitions and record all connected places and arcs associated with each transition, creating the initial partitions. After creating the initial partitions, we need to ensure the dependencies among the partitions are preserved. To achieve this, we begin the process of identifying the interface places. A place is an interface place if it is connected to transitions in different partitions. Therefore, we go through all the places and if a place is found in more than one partition, we identify it as an interface place.

Furthermore, we need to identify the token flow direction to accurately reflect the behavior of one partition in another. In another word, we need to analyze which partition produces tokens at the interface places and which consumes them. This is crucial for the validation as the flow direction dictates how token generator transitions should be created and updated. Algorithm 3 presents a pseudocode for identifying the interface places. The algorithm begins with constructing partitions of transitions according to the labels (lines 1-4), then adds associated places and arcs to each partition (lines 5-10). Finally, it identifies interface places as those appearing in multiple partitions and returns both the interface places and the partitions (lines 11-17).

3.3 Token Generator Transitions and Final Sub-Models Generation

The last step of the approach is the creation of token generator transitions and the partitioned sub-models as PNML files. Token generator transitions are used as a mechanism to replicate the dynamics of the system during independent validation of sub-models. By recreating the token flow at the interface places, token generator transitions help maintain the creation and destruction patterns of tokens to preserve the dynamics of SPNs.

To create the token generator transitions, we look at the captured data of token creation and destruction at each interface place and estimate the underlying probability distribution that best fits the data. Furthermore, the token flow direction at the interface places determines which partitions the token generator transitions need to be added to. It is also important to note that there may be a need

Algorithm 3 Identifying interface places

Require: SPN, defined or derived transition labels
Ensure: Partitions π, Interface places P_{int}
1: Initialize partitions $\pi \leftarrow \emptyset$
2: **for** each transition T in the SPN **do**
3: Assign T to partition $\pi(\text{label}(T))$
4: **end for**
5: **for** each partition $\pi_k \in \pi$ **do**
6: **for** each transition $T \in \pi_k$ **do**
7: Collect all places connected to T via arcs in $A^I \cup A^O \cup A^H$
8: Add these places and their arcs to partition π_k
9: **end for**
10: **end for**
11: Initialize $P_{\text{int}} \leftarrow \emptyset$
12: **for** each place P in the SPN **do**
13: **if** P appears in more than one partition **then**
14: $P_{\text{int}} \leftarrow P_{\text{int}} \cup \{P\}$
15: **end if**
16: **end for**
17: **return** (P_{int}, π)

for a token sink transition to avoid token accumulation and halting the model. This is especially highlighted in reliability cases where failures/repairs halts and resumes the system, and this behavior can be replicated in partitions through both token generator and sink transitions. For example, in reliability terms, token generator transition replicates failures while token sink transition replicates repairs. Therefore, we expand on our previous work [25], and identify and add token sink transitions to ensure preservation of dynamics and dependencies among partitions.

To determine the flow, we consider the arcs connected to interface places in the partitions. In each partition, output tokens show token production at the interface place, input arcs show token consumption at the interface place, and inhibitor arcs show inhibition of a transition by the interface place. Based on the flow, a partition needs token generator when it either inhibits or consumes tokens but does not produce any. Alternatively, a partition needs token sink transition when it either inhibits or produces tokens but does not consume any. Formally:

$$\text{needs}_{\text{generator}} = (I \neq \emptyset \vee C \neq \emptyset) \wedge (P = \emptyset) \tag{9}$$

$$\text{needs}_{\text{sink}} = (I \neq \emptyset \vee P \neq \emptyset) \wedge (C = \emptyset) \tag{10}$$

where I, C, and P denote the sets of inhibitor, consumer (input), and producer (output) arcs connected to the interface place within each partition.

Once all the token generator/sink transitions are created and added to their corresponding partition, the resulting partitions are saved as PNML files for further validation steps according to the modular validation framework. For each partition, a validation policy is set determining the frequency with which the partition goes through the validation process, and if needed goes through recalibration to ensure the model remains an accurate reflection of the system.

It is important to note that manufacturing systems evolve and change over time [26]. When a system evolves, its corresponding DT model and the subsequent partitions and their associated token generators may need to be updated. This is handled through

the continuous validation loop within our proposed modular validation framework (illustrated in Figure 2). When a sub-model is no longer valid, it goes through a recalibration process using data collected from the system, updating the firing distribution of the affected transitions as well as the associated token generator/sink transitions. However, if recalibration fails, for example, in case of structural changes in a system, re-extraction of a new model from newly gathered data and repartitioning may be necessary. This re-extraction and repartitioning process is done without the need for manual intervention within our validation loop, keeping the partitioned DT model inline with the systems.

4 Case Study

To demonstrate our automatic partitioning approach based on frequency, we conducted an experiment using a reliability case study as a proof-of-concept. Reliability models are great examples for assessing our approach as they have significant varying frequencies of occurrence of events. Here, we focus on a simple manufacturing line consisting of an assembly operation. The setup is grounded in Industry 4.0 Lab at the University of Southern Denmark [9]. The lab provides different capabilities such as warehousing and production supported by collaborative robots and IoT enabled data collection infrastructure. Our case study uses a robotic arm for its assembly operation. For simulation and validation, we utilized the PySPN library, a lightweight tool for simulation of Stochastic Petri nets [5]. We further extended the library to add the capability of translating PNML files into executable SPN models for PySPN.

First, we parsed the PNML file of our valid SPN model, graphically depicted in Figure 5, to PySPN and conducted 100 terminating simulation replications to gather the necessary data.

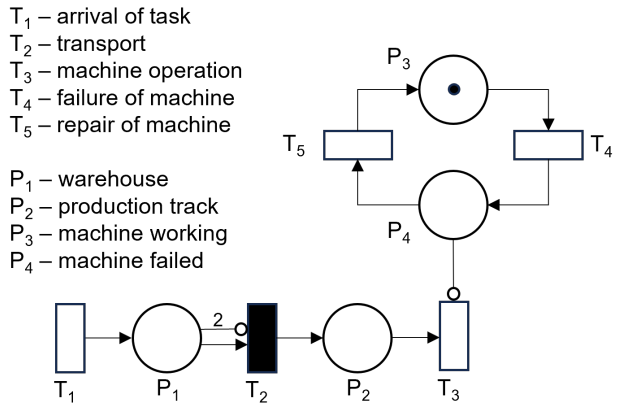


Figure 5: Graphical representation of the reliability case study.

Based on the data, demonstrated in Table 1 and Table 2, we then analyzed the marking change to find the time spent in each marking as well as number of visits for calculating marking rarity score, illustrated in Table 3.

The next step was to find all markings associated with each transition to calculate the transition rarity score for our partitioning. To achieve this, we looked at all firings of each transition from the *event log* and recorded the markings before and after each firing. Once all associated markings for each transition were found, we calculated the transition rarity score by using the weighted average marking rarity score and number of transition firings. In this study, we chose $\alpha = 0.9$ based on expert knowledge to mitigate the influence of frequent transitions on the transition rarity score. A high α value is suitable for our reliability case study as failures are rare events and the frequency of transitions associated with them is low. Table 4 shows the transition rarity score and number of associated markings of each transition.

Table 1: Snippet of event log

Transition	Time (t)	Replication (r)
arrival_of_task	76.66	0
transport	76.66	0
machine_operation	86.40	0
arrival_of_task	124.00	0

Table 2: Snippet of marking change

Marking (m)	Time (t)	Duration (d)	Replication (r)
$(0, 0, 1, 0)$	0.00	76.66	0
$(0, 1, 1, 0)$	76.66	9.74	0
$(0, 0, 1, 0)$	86.40	37.60	0
$(0, 1, 1, 0)$	124.00	12.57	0

Table 3: Marking rarity score for each marking

Marking	Total time spent in marking	Total visits	Marking rarity score
$(0, 0, 1, 0)$	103873.55	2889	1.74
$(0, 1, 1, 0)$	30732.09	2854	3.21
$(0, 1, 0, 1)$	4633.94	210	30.49
$(0, 0, 0, 1)$	3734.41	215	33.57

Table 4: Transition rarity score for each transition

Transition	Number of associated markings	Transition rarity score
T_1 – arrival of task	8	1.106
T_2 – transport	6	0.773
T_3 – machine operation	4	0.673
T_4 – failure of machine	4	4.116
T_5 – repair of machine	8	10.245

We set the partitioning threshold using the mean transition rarity score, separating above-average rarity from below-average rarity. This is particularly effective in our reliability case study as frequent production processes have lower transition rarity scores while rare failures have high scores, creating a natural boundary. Therefore, using the mean, as shown in Figure 6, transitions T_1, T_2, and T_3 were

labeled *frequent*, green rectangles in the figure, while transitions T_5 and T_5 were labeled *rare*, red rectangles in the figure. Finally, place P_4, blue circle in the figure, was identified as interface place as it connects to transitions in two different partitions.

To preserve the dynamics of the system within the partitions, we analyzed the token flow to determine which partition needs token generator/sink transitions. As the rare partition consumes and produces tokens at the interface place, it does not need token generator/sink transitions. However, the frequent partition needs both token generator and token sink transitions as it satisfies the conditions for token generator and token sink transitions. The final sub-models are illustrated in Figure 6, frequent transitions as green rectangles, token generator/sink transitions as purple rectangles, and interface place as a blue circle.

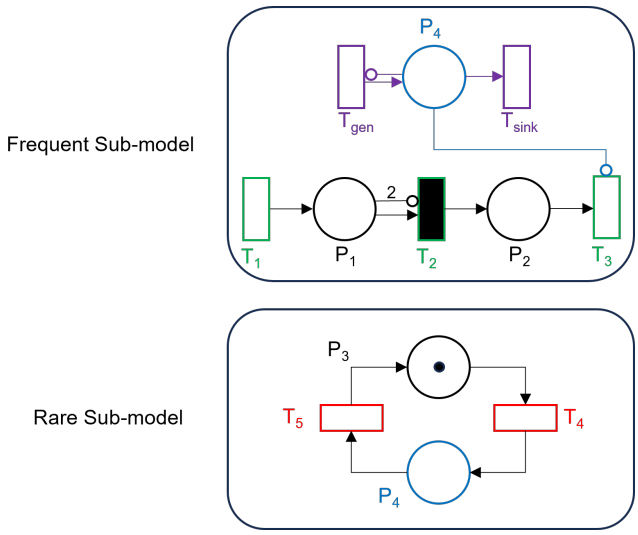


Figure 6: Graphical representation of the final sub-models including interface places and token generator/sink transitions.

To ensure that the dynamics and dependencies of the original model are preserved, we validated the resulting partitions against the original model as well as the ground truth. Models are deemed valid if the simulation results fall within the confidence intervals of the ground truth. For the frequent partition, we chose number of completed operations as our key performance indicator, and for the rare partition, the selected KPI was mean time between failures. To perform the comparison, we conducted 100 independent simulation replications on the original non-partitioned model, and the partitioned sub-models. To conduct the simulations, we parsed the PNML files, snipped shown in Figure 7, of each partition in our chosen simulation tool, PySPN. The comparison of the simulation results of the frequent partition to the original non-partitioned model as well as the ground truth, illustrated in Figure 8, shows validity of the partition as the results fall within the confidence intervals of both the original non-partitioned model and the ground truth.

For the rare partition, the simulation results, seen in Figure 9, further illustrate preservation of dynamics and behavior with the

```
<?xml version="1.0" ?>
<pnml xmlns="http://www.pnml.org/version-
2009/grammar/pnml">
  <net id="rare_subnet"
type="http://www.pnml.org/version-
2009/grammar/ptnet">
    <name>
      <text>rare Subnet</text>
    </name>
    <page id="page1">
      <place id="P3">
        <name>
          <text>machine_working</text>
        </name>
        <initialMarking>
          <text>1</text>
        </initialMarking>
      </place>
      <!--Interface place connecting to other
        partitions-->
      <place id="P4" interface="true">
        <name>
          <text>machine_failed</text>
        </name>
        <initialMarking>
          <text>0</text>
        </initialMarking>
      </place>
```

Figure 7: Snippet of the rare partition model as PNML with the interface place identified.

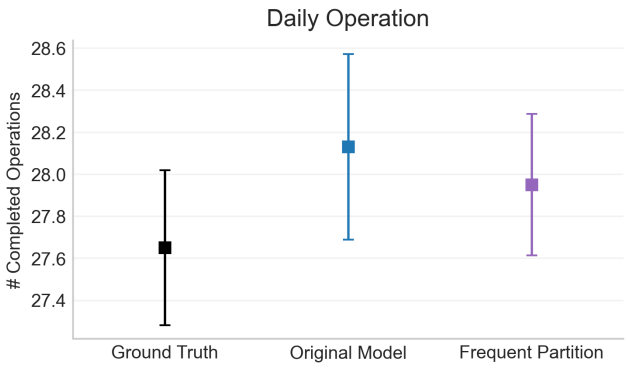


Figure 8: Comparison of simulation results with the ground truth for daily operation (error bars representing 95% confidence interval for 100 independent replications).

original model, as the results of the rare partition fall within the confidence intervals of the original model as well as the ground truth.

It is worth mentioning that the simulation results of the partitions were not expected to be exactly as the original non-partitioned model due to stochastic nature of the simulation. The partitioning result and KPI comparisons demonstrated that our approach was able to partition the model based on frequency while preserving and replicating the dynamics and behavior of the system within the partitioned models, further aiding in modular validation of Digital Twin models.

5 Challenges and Limitations

While our study shows promising results for enabling modular validation through automatic partitioning, it is not without limitations.

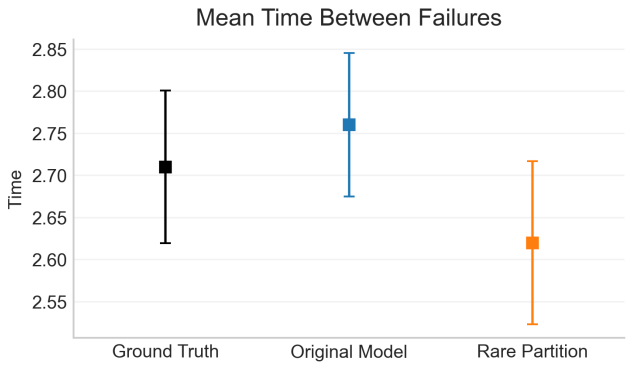


Figure 9: Comparison of simulation results with the ground truth for mean time between failures (error bars representing 95% confidence interval for 100 independent replications).

Therefore, when evaluating the usefulness of the approach to a specific system, the following limitations should be considered:

- **Data Availability:** Our approach relies on availability of sufficient data which may be hard to obtain, and addressing this limitation is a challenge for certain systems.
- **Assumptions:** In our proposed approach, we assume that an initially valid model exists and the behavioral analysis is based on the valid model. Furthermore, while parameter selection can be automated, we relied on expert knowledge for determining the partitioning thresholds for the case study.
- **Limited Scope:** To showcase our approach, we limited the scope of the case study to reliability models. Scalability and integration need further investigations.
- **Validating Reachability Information**: A critical consideration is validating that all practical reachable markings are discovered for partitioning. Therefore, sufficient simulation replications should be conducted to gather the reachability information. In our confined case study, the total number of reachable markings was known beforehand, and we were able to validate that with 100 simulation replications we discovered all possible reachable markings. However, in larger nets, this approach is not feasible. In these cases, we can monitor the rate of discovery of new markings as we conduct replications. For example, when no new markings are discovered over a number of replications, we can consider the replications sufficient and the reachability information valid.
- **Data Quality:** Our approach is dependent on having high quality data to be able to correctly identify partitioning points. Poor data quality can result in invalid or inaccurate sub-models.
- **General Applicability:** Our proposed approach is limited and best suited to stochastic Petri nets. However, the general approach to automatic partitioning based on frequency can be tailored to other modeling formalisms.

6 Conclusion

As technological advancements such as cyber-physical production systems transform manufacturing, Digital Twins have emerged as essential tools and enablers through the continuous reflection of real-world systems in the virtual world. However, the value of Digital Twins depends heavily on their fidelity to the physical systems, and ensuring continuous validity of the Digital Twin models remains a challenging task. As today's complex manufacturing systems consist of many different activities with vastly different rates of occurrence, the validation process of their corresponding Digital Twins must be capable of distinguishing the significance of these different and dynamic behaviors. To this end, we previously proposed a modular validation framework that aims at validating Digital Twin models by partitioning them into multiple sub-models, validating each sub-model individually depending on its characteristics.

In this paper, we build on the modular validation framework by presenting a data-driven approach to finding the sub-models based on the rate of occurrence of events. In our proposed approach, based on the stochastic Petri nets modeling formalism, we utilize transition firings and marking patterns to determine the partitions. We introduce a metric, transition rarity score, to determine to which partition each each transition belongs. To calculate the transition rarity score, we analyzed frequency of firing of transitions, frequency of reaching a marking, and time spent in a marking. Based on the score, transitions, along with their associated arcs and places, are partitioned into sub-models.

After partitioning, to preserve dependencies and replicate the dynamics of the system within the sub-models, we identified interface places that connect to transitions in different sub-models. Furthermore, we created token generator/sink transitions to recreate the behavior of the system at interface places, ensuring synchronization among the sub-models. Finally, through a case study in reliability, we demonstrated that our approach can automatically identify meaningful partitions based on frequency, thereby supporting the modular validation of Digital Twin models.

While our study contributes to making Digital Twin validation more robust, certain limitations remain. The approach is tailored to stochastic Petri nets, relies on the availability of sufficient historical data, and does not consider computational costs. Therefore, as future work, we suggest several directions. First, extending the approach to other modeling formalisms beyond stochastic Petri nets. Second, exploring hybrid approaches that combine our proposed behavioral analysis with reachability analysis. Third, integrating data with human expert knowledge to mitigate the effects of insufficient data. Finally, evaluating the approach on larger and more complex case studies.

Acknowledgments

This research is part of the ONE4ALL project, funded by the European Commission, Horizon Europe Programme under the Grant Agreement No 101091877.

References

[1] Jonathan Billington, Søren Christensen, Kees Van Hee, Ekkart Kindler, Olaf Kummer, Laure Petrucci, Reinier Post, Christian Stehno, and Michael Weber. 2003.

The petri net markup language: Concepts, technology, and tools. In *International Conference on Application and Theory of Petri Nets*. Springer, 483–505.

[2] Azzedine Boukerche and Sajal K Das. 1997. Dynamic load balancing strategies for conservative parallel simulations. In *Proceedings of the eleventh workshop on Parallel and distributed simulation*. 20–28.

[3] Anikó Costa and Luis Gomes. 2009. Petri net partitioning using net splitting operation. In *2009 7th IEEE International Conference on Industrial Informatics*. IEEE, 204–209.

[4] Gabriele D'Angelo. 2017. The simulation model partitioning problem: an adaptive solution based on self-clustering. *Simulation Modelling Practice and Theory* 70 (2017), 1–20.

[5] Jonas Friederich and Sanja Lazarova-Molnar. 2023. Pyspn: an extendable python library for modeling & simulation of stochastic Petri nets. In *International Conference on Simulation Tools and Techniques*. Springer, 70–78.

[6] Michael Grieves et al. 2014. Digital twin: manufacturing excellence through virtual factory replication. *White paper* 1, 2014 (2014), 1–7.

[7] Jiaoyang He, Yanxi Zhao, Ping He, Minglei Yu, Yan Zhu, Weixing Cao, Xiaohu Zhang, and Yongchao Tian. 2025. Rice Yield Prediction Based on Simulation Zone Partitioning and Dual-Variable Hierarchical Assimilation. *Remote Sensing* 17, 3 (2025), 386.

[8] Edward Y Hua, Sanja Lazarova-Molnar, and Deena P Francis. 2022. Validation of digital twins: challenges and opportunities. In *2022 Winter Simulation Conference (WSC)*. IEEE, 2900–2911.

[9] Sune Chung Jepsen, Torben Worm, Aslak Johansen, Sanja Lazarova-Molnar, Mikkel Baun Kjærgaard, Eun-Young Kang, Jonas Friederich, Juan Esteban Heredia Mena, Reza Soltani, Sune Lundø Sørensen, et al. 2021. A research setup demonstrating flexible industry 4.0 production. In *2021 International Symposium ELMAR*. IEEE, 143–150.

[10] Adán José-García and Wilfrido Gómez-Flores. 2021. A survey of cluster validity indices for automatic data clustering using differential evolution. In *Proceedings of the genetic and evolutionary computation conference*. 314–322.

[11] Sanja Lazarova-Molnar. 2005. *The proxel-based method: Formalisation, analysis and applications*. Ph. D. Dissertation. Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek.

[12] Giovanni Lugaresi, Sofia Gangemi, Giulia Gazzoni, and Andrea Matta. 2023. Online validation of digital twins for manufacturing systems. *Computers in Industry* 150 (2023), 103942.

[13] Joost Mertens and Joachim Denil. 2023. Digital-twin co-evolution using continuous validation. In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*. 266–267.

[14] Joost Mertens and Joachim Denil. 2025. Reusing model validation methods for the continuous validation of digital twins of cyber-physical systems: J. Mertens, J. Denil. *Software and Systems Modeling* 24, 5 (2025), 1427–1449.

[15] Florian Netter, Frank Gauterin, and Björn Butterer. 2013. Real-data validation of simulation models in a function-based modular framework. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*. IEEE, 41–47.

[16] Alessandro Vittorio Papadopoulos and Alberto Leva. 2015. A model partitioning method based on dynamic decoupling for the efficient simulation of multibody systems. *Multibody System Dynamics* 34, 2 (2015), 163–190.

[17] Siddheshwar V Patil and Dinesh B Kulkarni. 2020. Graph partitioning using heuristic kernighan-lin algorithm for parallel computing. In *Next Generation Information Processing System: Proceedings of ICCET 2020, Volume 2*. Springer, 281–288.

[18] Patrick Peschlow, Tobias Honecker, and Peter Martini. 2007. A flexible dynamic partitioning algorithm for optimistic distributed simulation. In *21st International Workshop on Principles of Advanced and Distributed Simulation (PADS'07)*. IEEE, 219–228.

[19] Carl Adam Petri. 1962. Kommunikation mit automaten. (1962).

[20] Robert G Sargent. 2020. Verification and validation of simulation models: an advanced tutorial. In *2020 winter simulation conference (WSC)*. IEEE, 16–29.

[21] Stewart Schlesinger. 1979. Terminology for model credibility. *Simulation* 32, 3 (1979), 103–104.

[22] Wei Xiang, Kan Yu, Fengling Han, Le Fang, Dehua He, and Qing-Long Han. 2023. Advanced manufacturing in industry 5.0: A survey of key enabling technologies and future trends. *IEEE Transactions on Industrial Informatics* 20, 2 (2023), 1055–1068.

[23] Ashkan Zare and Sanja Lazarova-Molnar. 2024. Modular Validation within Digital Twins: A Case Study in Reliability Analysis of Manufacturing Systems. In *2024 Winter Simulation Conference (WSC)*. IEEE, 2903–2914.

[24] Ashkan Zare and Sanja Lazarova-Molnar. 2024. Validation of digital twins in labor-intensive manufacturing: Significance and challenges. *Procedia Computer Science* 238 (2024), 623–630.

[25] Ashkan Zare and Sanja Lazarova-Molnar. 2025. Preserving Dependencies in Partitioned Digital Twin Models for Enabling Modular Validation. In *2025 Winter Simulation Conference (WSC)*. IEEE, 2836–2847.

[26] Ashkan Zare and Sanja Lazarova-Molnar. 2025. System Variations and Their Impact on the Validation of Digital Twin Models in Dynamic Manufacturing Environments. In *2025 30th International Conference on Automation and Computing (ICAC)*. IEEE, 1–6.

[27] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. 1998. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proceedings of the twelfth workshop on Parallel and distributed simulation*. 154–161.