Abstract—Just like any other algorithm Dynamic Thermal Management (DTM) schemes for multi-core architectures are susceptible to errors. Moreover, due to the wide spread usage and safety-critical nature of these schemes, there is a key demand for robust verification of these schemes before deployment. Traditional analysis techniques, like simulation and emulation, are inherently incomplete and therefore they cannot guarantee a complete absence of bugs. In this paper, we present a generic formal verification methodology, based of probabilistic model checking, for verifying decentralized DTM schemes. The paper provides a general modeling approach for developing a Markovian model of any decentralized DTM scheme. Moreover, we identify a set of generic probabilistic properties that can be of an interest to DTM scheme designers. For illustration purposes, the proposed methodology is used to verify Thermal Aware Agent Based Power Economy (TAPE), which is a state-of-the-art decentralized DTM scheme.

I. INTRODUCTION

Elevated temperatures, resulting from increased on-chip power densities, pose new challenges for multi-core system designers. At the same time, due to the increased complexity and scalability issues of modern many-core architectures, traditional centralized Dynamic Thermal Management (DTM) schemes [8] [15] can no longer be used to control these elevated temperatures. As a result, decentralized DTM schemes [6] have emerged as a new paradigm. These schemes tackle the problem of excessively high temperature spots, known as thermal hotspots, by focusing on local decision making. This way the usage of decentralized DTM schemes significantly reduces the overhead of monitoring the entire chip. However, this kind of distributed decision making brings up many new verification challenges.

Traditional analysis methods, like simulation and emulation, are inherently in-exhaustive and incomplete, i.e., there is always a possibility that an error may escapes the verification. Specially when considering many-core architectures, the possible number of system configurations, e.g., task-to-core mappings, increase exponentially with the increasing number of cores. Moreover, the decentralized nature of DTM techniques further complicates the test vector generation problem. Simulation based methods can thus never guarantee the coverage of all possible scenarios due to lack of scalability. Therefore, no matter how intelligent the test-bench and generator are, validating the design intent through simulation is inherently incomplete for large complex systems. On the other hand formal methods [16], like model checking [10], are well-known for their exhaustive nature and 100% completeness of analysis. A model checker exhaustively traverses through all possible system states to ascertain the correctness of model against the given specifications.

Traditional model checking has been employed successfully in the analysis of few Dynamic Power Management (DPM) schemes [11]. Probabilistic model checking of DPM schemes for many-core architecture has also been presented in [9]. Recently, a traditional model checking technique has also been introduced in analysis of a decentralized DTM scheme [4]. However, conventional model checking techniques are not useful for analyzing quantitative properties. In this paper, we propose a probabilistic model checking technique for analysis and verification of decentralized DTM schemes. Our choice of probabilistic model checking over traditional model checking is motivated by the characteristics of the problem we want to verify. A decentralized many-core system designer can be more interested in quantitative knowledge rather than simple yes/no answers. For instance, in case of thermal hot spots are formed, the probability of occurrence of this event is more useful than simply knowing the absence or presence of this event. Similarly, the probability that the system reaches a stable configuration in say $x$ time units can be an interesting property in the context of verifying decentralized DTM schemes.

Another key factor for employing probabilistic model checking in analysis of decentralized DTM scheme is the randomness in behavior of decentralized schemes. Due to decentralized nature of these schemes and absence of global system state knowledge, the workloads distribution and in some cases the initial task mapping across the many-core system, is random [6]. Ignoring this randomness in the behavior of decentralized DTM schemes during verification can compromise the analysis results. Since an error may remain undetected during verification if the input space is not sampled completely. These uncaught bugs during verification may lead to runtime failures, wastage of resources (i.e., time, money, and effort) or delays in the deployment of DTM schemes later on, as happened in the case of Foxton DTM that was designed for the Montecito chip [2]. Probabilistic model checking not only caters the randomness but also provide detailed statistics and hence, it can play a vital role in the design of the next generation DTM schemes for many-core systems. Moreover, due to absence of global communication in a fully decentralized system, there is an essential need for the exchange of state information across regions to achieve an optimal distribution of temperatures across the chip. A key challenge for multi-core system designers is the choice of tuning parameters for this negotiation [15]. Our proposed methodology helps the designers of decentralized DTM schemes to identify the optimal values of these tuning parameters.

A. Our Novel Contributions and Concept Overview

In this work, we present a generic trailblazing approach to formally analyze any decentralized DTM scheme and thus overcome the above-mentioned challenges. This work is an extension of our previous work [1], in which we have presented a case study on formal probabilistic analysis of a state-of-the-art decentralized DTM scheme, namely, Thermal aware agent based power economy (TAPE). After adding some new steps to our verification process, we present a generic methodology and a Markov Chain model, which can be used in the analysis and verification of any decentralized DTM scheme. The overall verification process is divided into three main phases. In the first phase, we judge the verification scalability of the given decentralized DTM scheme. Once the verification-feasibility is ensured, the DTM scheme is taken to the next phase of our methodology in which we perform probabilistic model checking by constructing the Markovian model of the given distributed DTM scheme. Besides probabilistic model checking, the proposed methodology also makes use of approximate model checking [12] in order to analyze the scheme on large-sized grids (i.e., increased number of cores).

Our proposed methodology utilizes a well-known probabilistic model checking tool, i.e., the PRISM model checker [3], which allows...
both the probabilistic and approximate verification. In order to illustrate the effectiveness of our methodology, we perform a case study on the formal verification of state-of-the-art agent-based decentralized DTM scheme, namely Thermal-aware Agent-based Power Economy (TAPE) [6]. As an extension of our previous work [1], we also provide some new experimental results for TAPE. Our proposed verification methodology allowed us to identify a couple of issues with TAPE during verification, which could not have been caught using traditional model checking techniques in [4].

II. BACKGROUND AND PRELIMINARIES

A. Probabilistic Model Checking

Probabilistic model checking [7] is an extension of traditional model-checking techniques [4] for the integrated analysis of both qualitative and quantitative properties of systems that exhibit stochastic behavior. A model checker exhaustively searches all possible input and state conditions for failures and ensures 100% completeness of analysis results. One of the drawbacks of model checking is the extensive utilization of memory due to large state-space of some systems, which sometimes leads to state-space explosion problem. This problem is usually resolved by developing abstract, less complex, models of the system or by using approximate model checking.

B. PRISM Model Checker

PRISM [3] is a well-known probabilistic model checker. The system to be verified by PRISM is first described as a probabilistic variant of Reactive Modules [3]. It provides support for analyzing different kinds of probabilistic models, such as discrete-time Markov chains (DTMCs) [5], in which time is modeled as discrete steps. A vantage of using PRISM is that it allows accurate computation for a wide range of numerical properties and it forms a complete analysis, which is a very useful feature for analyzing the best and worst case scenarios. Moreover, PRISM provides support for approximate model checking as well and thus the same model of the given decentralized DTM scheme can be used for both approximate and probabilistic model checking.

C. Approximate (Statistical) Model Checking

Approximate model checking [12] is a simulation-based alternative to probabilistic model checking; it can be used for analyzing the decentralized DTM schemes on large-sized grids, as the latter becomes infeasible due to state-space explosion problem. Unlike conventional probabilistic model checking, which uses numerical approaches to iteratively compute the exact values of probabilities, statistical model checking is achieved by sampling. It generates large number of sample paths through the system, evaluates the results of the given properties at each run and then uses the information to infer whether these sample paths provide a statistical evidence to the satisfaction or violation of the specification. Since all execution paths of the model are not analyzed, there can be a likelihood of error occurrence which can be bounded. These bounds determine the verification accuracy and with sufficiently large number of sample paths we can get very close to accurate results.

III. PROPOSED METHODOLOGY

The most critical aspect in the design of a new DTM scheme is verifying that the DTM scheme is both functionally correct and performance efficient. The decentralized DTM schemes involve runtime tasks migration and the most decisive functional aspect of a decentralized DTM is its ability to attain a stable and nearly optimal system configuration (uniform distribution of temperature across multiple cores and elimination of thermal hotspots) from all possible scenarios. Moreover, the stability should be attained within a reasonable amount of time. Therefore, both functional and timing properties should be checked while assessing the feasibility of a DTM technique. The main three phases of our proposed methodology, depicted in Fig.1, are described in the following.

A. Model Construction and Verification-Feasibility Analysis

This is the first phase in which an abstract model of the given decentralized DTM scheme is constructed, using the PRISM language, and then analyzed for verification feasibility. The behavior is expressed by a set of guarded commands in PRISM. For example,

$$[[\text{guard} \rightarrow \langle \text{prob} \rangle : \langle \text{action} \rangle + \ldots + \langle \text{prob} \rangle : \langle \text{action} \rangle];$$

The guard is a predicate over all variables of the system and if it is true then a transition will take place depending upon the probabilities. An example command is given as under:

$$[\text{y=1} \rightarrow 0.2: (y'=4) + 0.8: (y'=5);$$

The above-mentioned command states that if the expression ‘y = 1’ holds true, then (→ >) with probability 0.2, the next value of y (i.e., y’) would be equal to 4 or with probability 0.8, the next value of y (i.e., y’) would be equal to 5.

Due to randomized nature and high number of task-to-core mappings, the reachable state-space of the given DTM model is usually very large. Moreover, DTM schemes mostly involve several unknown parameters in their cost function for making decisions [6] [14], which significantly affects the thermal behavior of the chip and the performance of DTM scheme at runtime. In order to find the optimal values for these unknown parameters, we analyze the behavior of the given DTM scheme over a range of possible values for such parameters. If the resulting reachable state-space of the given DTM scheme is very high then it becomes computationally infeasible to find the optimal values for unknown parameters. In this case, the algorithm may be optimized further to facilitate its verification. Therefore, it is very important to check the verification feasibility of the DTM scheme during the early design stages as it facilitates the development of a well-defined DTM algorithm that results into a manageable consumption of verification resources.

Following basic steps are involved during this phase:

1) Model Construction: A decentralized DTM algorithm can be modeled as a Discrete Time Markov Chain Model (DTMC) since, in most decentralized DTM schemes (e.g., [6]), the thermal

---

Fig. 1. Proposed Methodology
measurements are received after discrete time intervals. The transition
probabilities of this DTM can be derived on the basis of the total
number of cores and a set of allowable task-execution times. For
instance, if a total number of tiles \( x \) are equally probable for mapping
a task, then the transition probability of mapping the task on one
particular tile can be computed as \( 1/x \). For example, \( \lbrack \) guard ->
\( 1/x : mapping_{\text{core}} = 11 + \ldots + 1/x : mapping_{\text{core}} = 21 \). In a similar manner, we can define a set of allowable execution
times for the tasks. Then the algorithm can randomly select a task with a random execution time. For example, \( \lbrack \) guard -> 0.25 :
\( \text{task} \text{time} = 1 + \ldots + 0.25 : \text{task} \text{time} = 4 \). An
excerpt of Markov chain model is presented in Fig.2, where the set
of allowable execution time is \( \{1,2\} \) and the grid size is \( 2 \times 2 \). The transition probability of mapping the task on a particular core can change depending upon the decentralized DTM scheme. We propose to use the PRISM Preprocessor tool [3] to develop decentralized
DTM models as this choice allows us to use generic variables for various parameters of the given algorithm, which facilitates the
code reuse during the probabilistic and approximate model checking
techniques. Depending upon the size of the model, various factors
can be kept random in the analysis of the given decentralized DTM
diagram. These factors include task mapping, amount of workloads,
task execution times and initial temperature of different chips.

2) Calculate States and Transitions: After constructing a high
level model, we perform a quick analysis on the size of state-space
of the model by building the model with the PRISM engine for
different values of the total number of tasks. If the reachable states
and transitions increase exponentially with the number of tasks then
we may need to generate a newer reduced model of the given design,
as depicted in Fig.1. Various techniques can be used for reducing
the size of the model, such as abstraction and symmetry reduction.
For instance, a smaller set of integer or rational data values can be
used to represent the real data set. Similarly, we suggest to reduce
the number of probabilistic choices in case of scalability issues.

B. Probabilistic Model Checking

This phase allows the designer to verify probabilistic properties of
the given decentralized DTM algorithm by executing its individual
components in parallel using the PRISM model checker. A vantage
of using PRISM is that it supports the verification of timing
properties without the overhead of modeling time separately (as
like “expected time” and “expected temperatures” can be computed
using costs and rewards in PRISM. The basic steps involved during
this phase are as under:

1) Verification of functional properties: First of all, we evaluate
the probabilities associated with the desired functional properties
by expressing them in probabilistic LTL. For example, the probability
that the algorithm eventually terminates can be expressed as:

\[
P =? \ [F \ \text{“terminate”}] \]

Similarly the probability that eventually the tasks become zero can
be expressed as:

\[
P =? \ [F \ \text{tasks} = 0] \]

Moreover, in the context of verifying decentralized DTM schemes,
it is important to verify that the system attains a stable configuration
in long run. This can verified by expressing following probabilistic
property:

\[
S =? \ [\text{isStable} = \text{True}] \]

Where \( S \) is known as the steady-state probability. It is the probability
of being in a state where the condition “\( \text{isStable} = \text{True} \)” holds
true at time \( T \), as \( T \) goes to infinity.

2) Verification of timing properties: Time-based properties can
be verified in PRISM by augmenting the models with costs and
rewards, i.e., real values associated with certain states or transitions.
For this purpose we extend the model with following reward structure:

\[
\text{rewards} \quad \text{true} : 1; \quad \text{endrewards} \]

This reward assigns a real value of 1 to every state of the model.
The reward gets accumulated over time and in order to
calculate the number of discrete steps required to reach a state where
z equals to 2 from any state \( s \) by expressing the following property:

\[
R =? \ [F \ z = 2] \]

In case of decentralized DTM schemes, we can replace \( z = 2 \)
with the stability condition and hence calculate the number of steps
before the system attains a stable configuration.

3) Running experiments to determine the optimal values of
unknown parameters: We run many iterative experiments to evaluate
the performance of decentralized DTM scheme at different values of
unknown parameters. In this way, we can find the optimal values
for these parameters, which are best-suited for the underlying DTM
scheme.

If the desired properties are not verified or the desired performance
characteristics are not met then first we should look for any inconsis-
tencies in our DTMC model as depicted in Fig.1. If the validity of
model is ensured then we have to go back to the design phase to
fix the bugs and unwanted behaviors, which should be removed from
the system.

C. Approximate Model Checking

The previous two steps may not cater for large models due to
the inherent state-space explosion problem of model checking [13].
So we propose to use them for verifying the decentralized DTM
schemes for small grids. Once the schemes are rigorously verified
on the smaller grids, we propose to test them for large-sized grids
(i.e. with large number of cores) using approximate model checking.

1) Rebuilding the DTM model using preprocessor for higher
number of cores: Using the preprocessor generated model for the
DTM scheme, the steps in this phase become very straightforward
as we can generate a model for PRISM with an updated large value
for the number of cores.
2) Analysis of functional and time-based properties: Once the model is ready, we perform different experiments to verify the desired functional and timing properties of the given DTM scheme on large grid sizes. We can also test if the optimal values of parameters remain the same for larger grids or not.

All the three phases in our proposed methodology are deemed important. The first phase facilitates the designer to develop an efficient DTM algorithm which is formally verifiable. The second phase allows the designer to verify the correctness of the given decentralized DTM scheme along with a detailed quantitative analysis and the third phase allows the designer to analyze the already verified algorithm for larger grids without having the state-space explosion problem.

IV. CASE STUDY ON PROBABILISTIC FORMAL ANALYSIS OF TAPE DTM

For illustration purposes, we used the proposed methodology to formally analyze the Thermal-aware Agent-based Power Economy (TAPE) [8], which is a state-of-the-art agent-based DTM scheme.

A. Introduction to TAPE

TAPE is an advanced technique which utilizes the concepts of decentralized multi-agent systems in order to keep the multi-core system working below a safe thermal threshold. According to the TAPE algorithm, an agent is associated with each tile of the multi-core system. At startup, all these agents acquire an equal number of power units. If a power unit is utilized for task execution by the Processing Element (PE), it is termed as a used power unit, otherwise all power units are known as free power units. At startup, each tile is equally probable to map a task, therefore, one is chosen randomly among all. Otherwise a tile with the maximum number of free power units is chosen for task mapping. According to the TAPE algorithm, it is the tile with maximum \( sell_{n} = buy_{n} \) value.

Based on the number of free and used power units, the agent \( n \) first calculates the following buy and sell values at each time interval:

\[
\begin{align*}
Sellbase &= w_{u,s} \times used_{n} + w_{f,s} \times free_{n} \quad (1) \\
Buybase &= w_{u,b} \times used_{n} - w_{f,b} \times free_{n} \quad (2) \\
Buy modifier : buy_{n} &= Buybase - a_{b} \times T_{n} \quad (3) \\
Sell modifier : sell_{n} &= Sellbase - a_{s} \times T_{n} \quad (4)
\end{align*}
\]

where \( a_{s}, a_{b}, w_{u,s}, w_{f,s}, w_{u,b}, w_{f,b} \) (see sub-section C for details) are known as weight parameters and \( T_{n} \) is the current temperature of the corresponding core. Then the agents compare their own resulting modified buy/sell values with the buy/sell values of their neighbors using the following equation which provides a stability criteria:

\[
(sell_{n} - buy_{n}) - (sell_{i} - buy_{i}) < T_{n} \quad (5)
\]

If the above-mentioned stability equation is not satisfied by an agent \( n \) and its neighboring the agent \( i \), then the agent \( n \) gives away a free power unit to agent \( i \). If the agent \( n \) has no free power unit available, then the agent gives up one of its used power units and task migration is performed. The task is remapped to the tile with maximum \( sell_{n} = buy_{n} \) value.

B. Verification properties for the TAPE DTM scheme

The most important property in analyzing TAPE DTM scheme is to ensure that no thermal hotspots are formed. Moreover, it is important ensure that no circular loops are formed during the power trade. Another major property of interest is to identify the values of unknown weight parameters (i.e., \( a_{s}, a_{b}, w_{u,s}, w_{f,s}, w_{u,b}, w_{f,b} \)) at which TAPE algorithm behaves optimally. Apart from these functional properties, it is very important to determine the efficiency of the TAPE algorithm (i.e., time required to attain a stable system configuration). We have successfully analyzed all these properties using our proposed methodology.

C. Modeling TAPE in PRISM

Using our methodology, we described the behavior of the TAPE algorithm running on a many-core chip as a DTM. The grid is modeled as a two dimensional array of distributed nodes. For every node, there is an agent for (re-) mapping and a power trading agent for the exchange of power units. Every agent keeps a count of its free and used power units. The tasks are initialized in a separate module and the cores are chosen using the following criteria:

1) If all the nodes are equally probable for task mapping then one of them is chosen randomly with probability equal to one divided by the total number of cores (See Algorithm 1, Lines 1-5).

2) Otherwise, the task is (re-) mapped to the tile with the maximum \( sell_{n} = buy_{n} \) value. (See Algorithm 1, Lines 6,7).

Once a core is selected, its corresponding agent increments the used units and decrements the free units, depending on the task execution time. It is assumed that only 1 power unit is consumed for 1 time unit. Moreover, the temperature variable (Tm) of the corresponding tile is incremented by \( 4^\circ K \) (as done in [4]). Algorithm 2 shows the implementation of one agent module. At every interval, the buy and sell values of neighboring agents are compared using the stability equation. If the stability equation is not satisfied, then power trading agents will trade the corresponding power units (Algorithm 3). The details on formal modeling of TAPE algorithm can be found in [1].

**Algorithm 1** Mapping Software

**Module (re-) mapping**

1: \( [\text{Condition}_1 = \text{true}] \rightarrow > \)
2: \( 1/tiles : (\text{remap_core}_{n} = 11) \)
3: \( +1/tiles : (\text{remap_core}_{n} = 12) \)
4: \( +1/tiles : (\text{remap_core}_{n} = 21) \)
5: \( +1/tiles : (\text{remap_core}_{n} = 22) \)
6: \( [\text{Condition}_2 = \text{true} \& \text{max_diff} = \text{diff}_{11}] \rightarrow > \)
7: \( (\text{remap_core}_{n} = 11) \)
8: ... endmodule

**Algorithm 2 (Re-) Mapping Agent**

**Module (re-) mapping_agent_n**

1: \( [\text{remap_core}_{n} = n] \rightarrow > \)
2: \( (\text{free}_{n} = \text{free}_{n} - \text{remap_time}) \& \)
3: \( (\text{used}_{n} = \text{used}_{n} + \text{remap_time}) \& \)
4: \( (\text{temp}_{n} = \text{temp}_{n} + \text{remap_time} \times 4) \& \)
5: \( \text{(sync} = 0); \)
endmodule

**Algorithm 3** Power Trading Agent

**Module power_trading_agent_n**

1: \( [\text{dec}_{n} = \text{true}] \& (\text{free}_{n} > 0) \rightarrow > \)
2: \( (\text{free}_n = \text{free}_n - 1) \& (\text{dec}_{n} = \text{false}); \)
3: \( [\text{dec}_{n} = \text{true}] \& (\text{free}_n < 0) \rightarrow > \)
4: \( (\text{used}_n = \text{used}_n + 1) \& (\text{dec}_n = \text{false}) \& \)
5: \( (\text{temp}_n = \text{temp}_n - 4) \& \)
6: \( (\text{remap_tasks}_n = \text{remap_tasks}_n + 1) \& (\text{remap_state}_n = 0); \)
7: \( [\text{inc}_n = \text{true}] \rightarrow > \)
8: \( (\text{free}_n = \text{free}_n + 1) \& (\text{sync} = 0) \& (\text{inc}_n = \text{false}); \)
endmodule
1) Characteristics of weights \((a_a, a_b, w_{u,a}, w_{f,a}, w_{u,b}, w_{f,b})\): The most important challenge in the verification of the TAPE DTM scheme was to identify the values of weight parameters at which the algorithm behaves optimally. The values of these parameters remain constant throughout the execution of the algorithm. Our proposed methodology allowed us to identify certain combinations of these parametric values at which (i) circular loops are formed during power trade \([1]\), (ii) thermal hotspots are created and (iii) the algorithm requires infinitely long to reach stability. These statistics indicates the usefulness of our methodology. We reported the optimal values of these parameters in \([1]\) (see Fig. 3).

D. RESULTS

We used version 4.1 of the PRISM model checker along with Windows 7 professional OS running on a core i5-3210 CPU at 2.50 GHz with 8.00 GB of RAM. The verification is done for a 3x3 and a 4x4 grid using probabilistic model checking. Approximate model checking is performed on a 9x9 grid. It has been formally verified, using PRISM, that with the discovered combinations of parametric values \([1]\), the system not only maintains lower temperatures (Fig. 4), but also requires less number of steps to attain a stable configuration (Fig. 5).

Verification of functional properties: In our previous work \([1]\), we verified the following functional properties for TAPE:

1) Stability: The probability that in the long run the stability equation will be satisfied can be expressed for node \(ij\) and \(ji\) as follows:

\[
S=\{ \text{diff}_{ij} - \text{diff}_{ji} < Tn \}
\]

where \(\text{diff}_{ij}\) is the \(\text{sell}_{r_n} - \text{buy}_{r_n}\) value of core number \(ij\).

2) Thermal hotspots: The probability that in steady state the maximum temperature will not exceed the threshold temperature can be expressed as follows.

\[
S=\{ \text{MaxTemp} < 62 \}
\]

As an extension of the previous work, we have verified the following new properties and found some interesting results that are shown in Figs. 4-7.

1) Steps to stability: An important property in context of decentralized DTM schemes is to calculate the number of steps before a stable system configuration is reached. In order to do so, we extend the model with reward structure as explained earlier. Then, the steps to stability are calculated by expressing the following property:

\[
R"\text{steps}"=\{ \text{stepsToStability} \}
\]

2) Effects of tasks on maximum temperature: A system designer can be interested in analyzing the overall maximum temperature on the chip as more and more tasks are initialized. Again we extend the model with the following reward structure:

\[
\text{rewards } \text{"maxtemp"}
\]

\[
\text{true} : \text{max}_{\text{temp}};
\]

\[
\text{endrewards}
\]

This reward assigns a real value, equal to the maximum temperature, to every state of the model. Thus, in order to calculate the maximum temperature of the chip after stable system configuration is attained, we express the following property:

\[
R"\text{maxTemp}"=\{ I= \text{stepsToStability} \}
\]

This type of reward known as “Instantaneous reward”, refer to the reward of a model at a particular instant in time. Therefore, it gives the value of maximum temperature. Steps to stability can be calculated using the above-mentioned property.

Fig. 4 presents the effect of the total number of tasks on maximum temperatures. These results are observed on a 3x3 grid for value of \(a_a = 0.1, a_b = 0.1, w_{u,a} = 2/7, w_{f,a} = 2/7, w_{f,a} = 2/7\) and \(w_{u,b} = 8/7\). It shows that the maximum temperature increases with the increasing number of tasks but even with 54 tasks, the maximum temperature does not exceed beyond 60°C. This is the minimum possible maximum-temperature, which is attained due to the even distribution of the tasks around the grid.

Fig. 5 presents effects of the number of tasks on the steps to attain stable configuration. It can be observed that at the optimal values of the parameters, the algorithm attains a stable configuration in lesser number of steps. Whereas, at some values, the number of steps goes to infinity. In reality, the algorithm enters into a never ending loop due to back and forth trading between two agents and hence never attains stability. These statistics clearly show the importance of probabilistic mode checking in the context of verifying decentralized DTM schemes, since these insights were not gained in the traditional model checking based analysis of the same algorithm \([5]\).

Thermal hotspots: In our previous work \([1]\), we reported that an
The proposed methodology mainly utilizes PRISM model checker for the verification of functional and timing properties. For illustration purposes, the paper presents a successful probabilistic analysis of TAPE, which a state-of-the-art decentralized DTM scheme. Our proposed methodology allowed us to identify certain glitches in TAPE DTM scheme, which were not caught by a traditional model checking based analysis of the same algorithm. This fact clearly indicates the applicability and usefulness of our methodology in the context of verifying decentralized DTM schemes.

ACKNOWLEDGMENT

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program “Dependable Embedded Systems” (SPP 1500 - spp1500.itec.kit.edu) and in parts by the DAAD “Deutsch-Pakistansche Forschungscooperationen” project.

REFERENCES