Synthesizing power management strategies for wireless sensor networks with UPPAAL-STRATEGO

Shengxin Dai, Mei Hong and Bing Guo

Abstract
Effective power management has become a key concern in the design of wireless sensor networks. Dynamic power management refers to strategies which selectively switch between several power states of a device during the runtime in order to achieve a tradeoff between power consumption and performance. In this article, we present a novel methodology that exploits current model-checking technology for automatic synthesis for dynamic power management. The generic system model for dynamic power management is modeled as a network of timed games. And the synthesis objectives are expressed as synthesis queries. Subsequently, automatic synthesis of power management strategies is performed by UPPAAL-STRATEGO with respect to the synthesis queries. Once a strategy has been constructed, its performance can be analyzed through statistical model-checking using the same tool. The modeling and synthesizing procedures are illustrated with a running example. Finally, the applicability of the methodology is assessed by synthesizing and evaluating a range of power management strategies for a concrete sensor node. Our methodology can be employed to help designers in constructing dynamic power management strategies for wireless sensor networks in practical applications.

Keywords
Strategy synthesis, dynamic power management, model-checking, wireless sensor networks, UPPAAL-STRATEGO

Introduction
Wireless sensor networks (WSNs) are one of the technologies that are gaining considerable attention. They are composed of distributed autonomous nodes which are capable of organizing themselves to establish a network and carry out reliable sensing. However, when considered individually, each node is a simple device. A wireless sensor node is typically battery-powered and therefore energy constrained. Thus, the reduction of power consumption is crucial to increase the lifetime of a sensor node. The problem of reducing power consumption of WSNs has been addressed in two different ways in the literature. First, a mass of energy-efficient communication protocols that take the peculiarities of WSNs into account have been proposed. Second, local dynamic power management (DPM) strategies are developed to reduce the power consumption within an individual node. In this article, we focus on the second approach and consider the individual sensor node as a power-managed device.

Numerous DPM techniques have been proposed, in order to achieve an optimal tradeoff between power consumption and performance in general battery-powered embedded systems. DPM refers to strategies...
which selectively switch between several power states of a device during the runtime in order to minimize power consumption, in the meantime guarantees the performance constraints or vice versa. A device may have several operational states that are characterized by a service rate and a power consumption value. The power manager is a controller that controls the transitions between these operational states. The criterion used for making state transition decisions is called power management strategy. The design problem is to formulate a strategy for transitioning between states so as to maximize energy efficiency. The best known standard for low-power hardware and software design is ACPI (Advanced Configuration and Power Interface), which introduces an abstract and flexible interface between the power-manageable components and the power manager. However, ACPI does not provide insight into constructing the power management strategy. The power-aware sensor model is similar to the system power model in the ACPI, which has been studied in Benini et al.1

The previous works on constructing DPM strategy are based on analytical approaches, such as linear programming1,7 and reinforcement learning–based methods.3,8 The analytical approaches require a mathematical modeling of the system, which is an intricate process. Thus, their complexity results in poor usability that impedes them from practical usage. In addition, the analytical approaches are based on Markov models (Markov chain or Markov decision process), one key assumption made by the Markov models is that the transitions between the states follow the exponential distribution. However, the reality is not always the case, and it has been proven that potential errors would be made if the transitions between states were approximated using the exponential distribution.9 The learning-based approaches demand a large number of training examples to construct the strategy which is time-consuming, and they require offline data collection.

Automatic synthesis is a recent research direction in the field of model-checking.10,11 In the setting of automatic synthesis, the controller is regarded as a controllable player, and the environment is regarded as another uncontrollable player (or a set of players). The specification is cast as a game objective that should be enforced by the controlled player at all the time. Thus, strategy synthesis is the process of constructing a winning strategy for the controller that will satisfy the desired objective in an adversarial environment. Benefit from the development of several tools (such as UPPAAL-TIGA12 UPPAAL-STRATEGO,13 and PRISM-games14,15) is that strategy synthesis has been applied successfully to the real-life case studies.16–18

The scientific contribution of this article can be summarized as follows:

1. We present a novel methodology for DPM, which is based on current model-checking technology for automatic synthesis. It illustrates the applicability of automatic synthesis to the area of DPM. We illustrate the system modeling procedure and query formulating process, and show how UPPAAL-STRATEGO can be used to generate the optimal strategy. To the best of our knowledge, this is the first research effort in synthesizing power management strategies with model-checking technology.

2. The generic system model for DPM1 is modeled in the form of timed games, and the modeling procedure is illustrated with a running example, so that other systems can be handled in a similar manner.

3. The strategy synthesis and evaluation queries are presented. The strategy synthesis queries are utilized for generating strategies, and they provide a flexible way to express the requirement of power-performance tradeoff. And the strategy evaluation queries are utilized for verifying and evaluating the generated strategies.

4. The applicability of the methodology is assessed by synthesizing and evaluating several power management strategies for a concrete sensor node. And the efficiency of the synthesized strategies is demonstrated by comparing them with realistic strategies.

5. Our methodology can be used to help designers in constructing effective power management strategies.

There are several advantages of our methodology. It assists in designing power management strategies with the recent branch of the UPPAAL tool suite, UPPAAL-STRATEGO, which supports for automatic synthesis of safe and near-optimal strategies using a combination of symbolic synthesis and reinforcement learning. Symbolic synthesis is exhaustive in state-space searching that guarantees the generated strategies are safe. And the learning algorithm uses a simulation-based method for learning near-optimal strategies for a given price metric. Afterwards, the synthesized strategies can be analyzed through statistical model-checking with the same tool, which avoids extensive simulation. Compared with the analytical approaches, our modeling procedure is more intuitive, that is, modeling the system with timed games is relatively easier than formulating that with mathematical formulas. The default uniform and exponential distribution are available in UPPAAL-STRATEGO. What’s more, one can encode any distribution into the model with the help of user-defined functions. UPPAAL-STRATEGO automatically utilizes the reinforcement learning approach to generate the optimal strategies and do not require
additional tool support. In addition, the synthesized strategies can be analyzed with the same tool, which avoids extra prototyping and measurement using a specific simulator.

This article is organized as follows. In section “DPM,” we give a short introduction of the DPM problem. In section “Timed games, synthesis, and UPPAAL-STRATEGO,” we present the definition of timed games and synthesis, and then introduce the UPPAAL-STRATEGO tool. In section “Strategy synthesis for DPM with UPPAAL-STRATEGO,” we illustrate our methodology with a running example. In section “Sensor node case study,” we apply our methodology to a concrete sensor node. In section “Related work,” we review related work in the field of DPM and strategy synthesis. Finally, section “Conclusion” concludes this article.

DPM

DPM exploits the fact that certain devices may have several operational states that have different service rate and power consumption value, so they could be switched into low-power states during inactive periods to save energy. However, when a service request arrives, the device needs to be brought back into the active state, and this transition incurs additional latency. The low-power state saves more energy, but leads to higher transition latency at the same time. Therefore, it is an elaborate work to make decisions on when and which transition should be enforced.

The generic system model for DPM was originally proposed in Benini et al.1 This model, as depicted in Figure 1, consists of four components: Service Requester (SR), Service Queue (SQ), Service Provider (SP), and Power Manager (PM). The SR represents the application that generates requests for the SP, for example, data transfer. The SQ stores requests that cannot be serviced immediately. The SP is the request-processing device (i.e. the sensor node in WSN) that is under power manager control. The PM issues state transition commands to the SP, based on observations of the system and the DPM strategy.

It has been proven that DPM is an effective way for reducing the power consumption in computing systems.2,3 According to the level (e.g. component and system) that the DPM is applied, DPM can have different embodiments. And DPM can be classified into timeout, predicative, stochastic, and machine-learning approaches. Timeout strategy2,8 switches a device to a low-power state after it has been idle for a certain time period. Timeout strategy is easy to implement, but the drawback of this strategy is that it wastes power while waiting for the timeout to expire. Predictive strategies2,19,20 forecast the length of idle periods based on the history of past idle periods, and switch the device to a low-power state if the predicted idle period is longer than a certain threshold. Predictive strategies do address workload uncertainty, but an incorrect estimate can cause both performance and energy penalties. Both timeout and predictive strategies are heuristic in nature and, thus, do not provide mechanisms for controlling the power-performance tradeoff.9 Stochastic strategies1,7,9 are based on stochastic models that can guarantee optimal results, and they do provide a flexible way to control the power-performance tradeoff. However, solving the stochastic optimization problem to find the optimal DPM strategy with linear programming is a complex process. Machine-learning strategies3,8 apply machine learning to learn the request arrival patterns for DPM; they perform well under various workload conditions. But these strategies require offline data collection and training of a classifier that is a time-consuming and complex work.

We present an alternative methodology to the above-mentioned problems, which is based on current model-checking for automatic synthesis. Benefit from the user-friendly interface of UPPAAL-STRATEGO13 is that the power management problem can be formulated in a convenient manner, and the optimal strategies will be automatically obtained in the sequel. Our methodology provides a flexible way to control the power-performance tradeoff, and with the system model and queries as inputs, the optimal strategies are automatically generated without any complex calculation or data collection. In general, our methodology belongs to stochastic strategies since it is based on stochastic modeling of the system. However, model-checking technology is utilized for generating the optimal strategies instead of linear programming in our methodology, which reduces the complexity of the problem formulation and increases the scalability of our methodology.

Timed games, synthesis, and UPPAAL-STRATEGO

Let $X$ be a finite set of real-valued variables called clocks. We define $B(X)$ as the set of constraints by the
Strategy synthesis for DPM with UPPAAL-STRATEGO

In this section, we use a four-state device power management problem as a running example to illustrate our methodology. We model the problem, explain the procedure, and show how UPPAAL-STRATEGO can be used to solve the problem.

Illustration of the methodology

In this subsection, we introduce our methodology briefly. The block diagram of our methodology is shown in Figure 2. What should be emphasized is that the shaded blocks represent manual steps, while white boxes are fully automated steps. That is, the translations of the description and specifications are done manually.
The idea of DPM strategy synthesis with UPPAAL-STRATEGO is to model the service requests arrival scenario and the WSN systems with timed games, and to leave the switching decisions between states open (modeled as controllable transitions in timed games). Then formulates the optimal objective as a synthesis query. Subsequently, UPPAAL-STRATEGO can be utilized to construct the strategy with respect to the synthesis query. Specifically, for a given service requests arrival scenario and sensor node configuration, UPPAAL-STRATEGO manipulates the controllable transitions to control the switching between the power states. After random simulations and reinforcement learning process, UPPAAL-STRATEGO provides a guidance to manipulate the controllable transitions so that the power consumption of the sensor node can be minimized, which is referred to as the optimal DPM strategy.

Our methodology requires three inputs: a system description, a strategy specification (specifies the strategic requirements, for example, minimizing the power consumption under performance constraints), and a performance specification. The system description provides the information needed to formulate the system model, such as the service rate and the power consumption value. The system is specified as a set of timed games that associated with the system parameters. Besides, the synthesis specifications are expressed as the synthesis queries. With the system model and queries as inputs, the tool automatically generates the strategy (if exists) and satisfies the requirement; otherwise, it reports the requirement cannot be satisfied.

Once a strategy has been constructed, it can be analyzed through statistical model-checking with respect to the strategy evaluation queries that translated from the performance specifications. With the results provided by the tool, further analysis and comparison can be accomplished.

### Modeling DPM in UPPAAL-STRATEGO

We adopt the generic system model for DPM proposed in Benini et al., which has been introduced in section “DPM”. In this subsection, we describe the modeling procedure of this model in detail. The modeling procedure is described through a four-state device, which has also been used as a case study in Qiu et al.7 and Simunic et al.9

The device has four states: Active, Idle, Standby, Sleep. The Active state is the only state that the device can provide service (with a service rate of 125), and Idle is the state that the device is active but has no request in the system. When the SP is in the Standby or Sleep state, it is inactive and working in a low-power model. The SP makes transitions automatically between the Idle and Active states. The transitions between the Sleep, Standby, and Idle states are controlled by the PM. The PM can issue a command $a \in A = \{ \text{goSleep}, \text{goIdle}, \text{goStdby} \}$ to the SP that enforces it to make transition accordingly. The average power consumption of each state is shown in Table 2. The length of the service queue is 20.

#### Table 2. Average power consumption for each state of the SP

| State  | Sleep | Standby | Idle | Active |
|-------|-------|---------|------|--------|
| Average power (W) | 0.13 | 0.35 | 0.95 | 2.15 |
The average power consumption and the average times to translate between each pair of states are shown in Tables 3 and 4, respectively. In addition, it has been observed in Simunic et al. that the transitions between the states of this device are best described using uniform distribution. And the transition times from the Sleep and Standby states into the Idle state have a deviation of 0.5s, and the remaining transition times have a deviation of 0.1s.

The service requests arrival scenario has also been observed in Simunic et al. the requests inter-arrival time in the active state (the state where one or more requests in the service queue) follows the exponential distribution, and that follows the Pareto distribution during the idle periods. Specifically, the inter-arrival time in active state follows an exponential distribution with the mean value \( \lambda = 100/72 \), where the probability of the SP receiving a request within time \( t \) follows the cumulative probability distribution \( F_{\text{SR}}(t) = 1 - e^{-\lambda t} \), and that during the idle periods follow a Pareto distribution with the cumulative probability \( F_{\text{SR}}(t) = 1 - a^{-t} \). What’s more, after collecting of 11 hr request trace for the device with standard software, they obtained the parameters for the distributions \( \lambda = 100/72 \), \( a = 0.9 \), and \( b = 0.65 \).

### Modeling the service requester

The SR generates requests according to the specific scenario, as shown in Figure 3. Initially, the SR stays in the Idle state, and the actual delay in this state is computed and stored into a clock delay. The function Pareto() computes this delay with respect to the Pareto distribution. After releasing the first request, the SR moves to the Active state, and generates requests follows an exponential distribution with a mean value 0.72, that is implemented by assigning the rate of exponential of the state with 100/72. Whenever the service queue is empty, the SR moves back to the Idle state. Note that it is straightforward to refine this model further to capture the actual requests arrival scenario in practical applications, by explicitly introducing user-defined functions and transitions.

### Modeling the service queue

The SQ is modeled as a timed game with a parameter QUEUESIZE that indicates the maximum length of the queue, as shown in Figure 4. Initially, the SQ stays in the Empty state until a request arrives, after that the SQ moves to the Queue state. Thereafter, with every arrival of a request, the variable count increases by 1 up to the capability of the queue. Whenever the number of enqueued requests exceeds the capability, the requests are lost, and the number of lost requests is recorded by lostCount. Whenever a request is fetched, count decreases by 1, until the queue is empty, and the SQ moves back to the Empty state.

Note that the variable totalDelay records the total waiting time of all the requests, thus it has a growth rate equals to count that is the current number of requests in the queue.

### Modeling the service provider

The SP is modeled as a timed game, as shown in Figure 5, in which the solid
lines indicate the controllable transitions that may be triggered by the PM, and the dashed lines indicate the uncontrollable transitions.

As shown in Figure 5, the four operational states of the SP are modeled as locations in the timed game, and the transient states ($Idl_2Sl$, $Sl_2Idl$, ..., $Idl_2St$) represent the transition latencies that cannot be controlled. The $Active$ state is the only state that the request can be serviced; it releases $Ser$ signal following an exponential distribution with mean value $1/\eta$, where $\eta$ is the service rate, which means the service speed of the SP for requests in the $Active$ state. Hence, $1/\eta$ gives the average time that is needed by the SP to complete a service. In addition, since the transitions between the states follow the uniform distribution, a function $Uniform()$ is implemented to compute the actual delay for the transient states.

What’s more, each state is associated with a power consumption value. It represents the power consumption of the SP during the time it occupies the state. The
automation has one continuous variable \(\text{Energy}\) which records the total amount of energy used by the SP.

**Modeling the power manager.** The PM controls the SP to translate between the low-power states by issuing a command \(a \in A\) to it. The PM, as depicted in Figure 6, consists of three locations \((\text{Idle}, \text{InSleep}, \text{InStandby})\) corresponding to the status when the SP is in the \(\text{Idle}\), \(\text{Sleep}\), and \(\text{Standby}\) states, respectively. The outgoing transitions of the locations represent the available commands when the PM is in that state. The guard \(!\text{inTr}\) guarantees that no command can be issued when the SP is in transient states. And the guard \(!\text{exSer}\) ensures that when the SP is in the \(\text{Idle}\) state and there exists at least one request in the system, it cannot move to low-power states except moving to the \(\text{Active}\) state.

**Strategy synthesis with UPPAAL-STRATEGO**

In this subsection, we introduce the synthesis queries in detail. Intuitively, the strategy synthesis should carry over an infinite time horizon. However, the reality is not exactly the case. The system lifetime is limited by the battery capability. This is the case of our assumption, where the time span of synthesis is a finite time window. Hence, we assume the system is not operating over an infinite horizon, but only over a finite time scope \(\text{BOUND}\).

There are two kinds of constraints, namely, performance constraint and energy constraint, which may be involved in specifying the synthesis specification. We believe that, in order to devise an efficient power management strategy, the performance constraint should be considered, especially in sensor networks that strongly depend on an application. Performance constraint consists of three constraints including the service lost ratio, the average queue size, and the average waiting time in the queue. And the Energy constraint is represented by the energy consumption of the system by time \(\text{BOUND}\). The constraints can be formulated with the following formulas with the variables in the model:

- Energy consumption constraint:
  \[\text{Energy} \leq E;\]
- Service lost ratio constraint:
  \[\text{loSeR} = \text{lostCount}/\text{totalCount} \leq C;\]
- Average queue size constraint:
  \[\text{avQuS} = \text{totalDelay}/\text{BOUND} \leq Q1;\]
- Average waiting time in the queue constraint:
  \[\text{avWaT} = \text{totalDelay}/(\text{totalCount} - \text{lostCount}) \leq Q2.\]

In which, \(E, C, Q1, Q2\) are constants that should be specified by the user. And \(\text{Energy, lostCount, totalCount, totalDelay}\) are variables in the system model.

**Figure 6.** Model of the power manager.

With the combination of the performance constraint and power constraint, we can formulate two kinds of the constrained optimal strategy synthesis problem: performance optimization under energy constraint and energy optimization under performance constraint. We only list out two queries in the following, other queries can be specified in a similar manner.

- Minimizing the service lost ratio under the energy constraint

  \[
  \text{strategy} \; \text{str}_0 = \text{minE(loSeR)}[ < = \text{BOUND}] : < > \text{done} \tag{1}
  \]

  where \(\text{done} = (\text{Energy} < = E)\) is a boolean variable that turn to true if the constraint is satisfied at time \(\text{BOUND}\).

- Minimizing the energy consumption under the average queue size constraint

  \[
  \text{strategy} \; \text{str}_1 = \text{minE(Energy)}[ < = \text{BOUND}] : < > \text{done} \tag{2}
  \]

  where \(\text{done} = (\text{avQuS} < = Q1)\).

**Strategy evaluation with UPPAAL-STRATEGO**

Once a strategy has been constructed, it can be analyzed through statistical model-checking approach. We use the average queue size as the primary performance metric and the average power consumption as the power metric, and set the time horizon \(\text{BOUND} = 1000\), which have also been used in Benini et al.,1 Dhiman and Rosing,3 and Qiu et al.7 We vary the performance constraint in the synthesis queries and generate the
strategies accordingly. We want to stress the difference between the performance metric and the performance constraint; the former represents the average case, while the latter restricts the worst case. Thus, the performance metric of the synthesized strategy is smaller than the corresponding performance constraint. The synthesized strategies are evaluated using UPPAAL-STRATEGO.

The first question we could ask is what the probability that the synthesized strategy satisfies the desired constraint. We use the strategy generated with “minimizing the energy consumption, meanwhile, guaranteeing the average queue size is smaller than 1.0” as an example, that is $Q1 = 1.0$ in equation (2), the synthesized strategy can be verified using the following query

\[
Pr[Q > BOUND] < avQuS < 1.0 \quad \text{under } str_1
\]

The probability is in the interval $[0.0001, 0.0002]$ with a confidence of 99.9%, which means the synthesized strategy satisfies the desired constraint.

In order to quantify the performance of the synthesized strategies, we compare them with the existing strategies. We modeled the following strategies for comparison, which have also been used in Benini et al.\textsuperscript{1} and Qiu et al.\textsuperscript{7}

- Timeout strategy: the PM switches the SP from \textit{Idle} to \textit{Sleep} after the SP has been in the \textit{Idle} state for a time period $T_{out}$. And the PM switches the SP from \textit{Sleep} to \textit{Idle} immediately when a request arrived. $T_{out}$ is varied during the experiments to examine the performance and power characteristics of the timeout strategy.
- $N$ strategy: the PM switches the SP from \textit{Idle} to \textit{Sleep} immediately after the SQ is empty. And the PM switches the SP from \textit{Sleep} to \textit{Idle} only after there are more than $N$ requests waiting in the SQ. $N$ is varied in the range of [1,10] during the experiments.
- Greedy strategies: the PM switches the SP from \textit{Idle} to the low-power states (\textit{Sleep} or \textit{Standby} state) immediately when the SP is in the \textit{Idle} state and the SQ is empty, and switches the SP back to \textit{Idle} after a request arrived.

In order to compare the strategies, we evaluate the performance metric and power metric under different strategies. We find the mean value of the metrics from a number of runs $N(N = 10,000$ in our experiments) under the strategy (here, we use strategy $str_1$ as an example) by time $BOUND$ using the following queries:

- Average power consumption:
  \[
  E[< = BOUND; N] \quad (\text{max : Energy/BOUND}) \quad \text{under } str_1;
  \]
- Service lost ratio:
  \[
  E[< = BOUND; N] \quad (\text{max : loSeR}) \quad \text{under } str_1;
  \]
- Average queue size:
  \[
  E[< = BOUND; N] \quad (\text{max : avQuS}) \quad \text{under } str_1;
  \]
- Average waiting time in the queue:
  \[
  E[< = BOUND; N] \quad (\text{max : avWaT}) \quad \text{under } str_1;
  \]

The power-performance tradeoff curves for the DPM strategies are shown in Figure 7, and the average waiting time in the queue under the corresponding strategies is shown in Figure 8, note that the X-axis is log scaled. In addition, the service lost ratio is 0 under all the DPM strategies, thus we omit the corresponding figure here. This is because the service queue is long enough to buffer the requests until the SP can switch back to the \textit{Active} state.
From Figures 7 and 8, we have the following conclusions. In most situations, the synthesized strategies outperform the existing strategies. The synthesized strategies provide a good power-performance tradeoff. And the existing strategies such as the timeout strategy cannot always provide a valid tradeoff. That is, in the experiments, decreasing the timeout threshold $T_{out}$ do not necessarily decrease the performance and power metrics. It can be observed that there exist an optimal timeout threshold; however, there is no formal way to find out this optimal threshold. The average waiting time in the queue has a simple relationship with the average queue size, that is the former increases as the latter increases since they are proportional to totalDelay. And the average waiting time in the queue under the synthesized strategies is larger than that under the remaining strategies, this is because the SP spends more time in the low-power states under these strategies, thus the requests stay in the service queue for a longer time.

### Sensor node case study

We have illustrated our methodology with a running example. In this section, we demonstrate the applicability of the methodology to a concrete sensor node, which has also been used in Draeger et al.\textsuperscript{23} Compared with the running example, the model of the sensor node has more transitions that can be controlled by the PM, and the service requests arrival scenario is different. In addition, the transitions between the states are assumed to follow the exponential distribution. We present this case system as a pilot study, so that other WSNs can be handled in a similar manner.

The sensor can operate in five different operational states, labeled, Active, Idle, LpIdle, Standby, Sleep, and only in the Active state that the sensor can provide service and its service rate is 10. In the Idle state, the sensor is listening, whereas some of the electronic components are turned off. The LpIdle state is similar to the Idle state except that it has decreased power dissipation. In the Standby or Sleep state, the sensor is inactive. The average power consumption for each state is shown in Table 5. The average transition times between each pair of states are shown in Table 6. When the sensor is in transient states, the behavior of the sensor is not controlled by the PM, and the sensor has zero service rate but its power consumption is high (1.0 W). The length of the service queue is 2.

What’s more, the service requests arrival scenario is obtained based on time-stamped traces measured on real machines and profiling. The SR has two states: Idle where no request is generated and Req where one request is generated per time step (1 s). The probabilities of the transitions between the two states are described with a matrix as follows:

\[
M = \begin{bmatrix}
0.902 & 0.098 \\
0.52 & 0.48 
\end{bmatrix}
\]

Note that the transitions between the states are assumed to follow the exponential distribution in Draeger et al.\textsuperscript{23} we follow this assumption when modeling the SR, SP, and PM. According to the proposed methodology, the model of the SR, SP, and PM are shown in Figures 9–11, respectively, and the model of the SQ is the same as the one introduced in the third section.

Apart from the simple strategies introduced in section “Timed games, synthesis, and UPPAAL-STRATEGO,” we add several realistic strategies for comparison. The involved strategies are the adaptive timeout strategy of Shih and Wang,\textsuperscript{20} the predictive strategy of Hwang et al.,\textsuperscript{19} the stochastic strategy of Benini et al.,\textsuperscript{1} and the reinforcement learning strategy of Triki et al.\textsuperscript{24} (“RL” in the figures). Note that we employ a heuristic timeout strategy instead of the original timeout strategy, so that multiple low-power states can be utilized. What’s more, since the capacity of the

| State  | Sleep | Standby | LpIdle | Idle | Active |
|--------|-------|---------|--------|------|--------|
| Average power (W) | 0.1 | 0.3 | 0.75 | 0.95 | 1.5 |

| Average time (s) | Active | Idle | LpIdle | Standby | Sleep |
|-----------------|--------|------|--------|---------|-------|
| Active          | /      | 0.001| 0.004  | 0.19    | 0.4   |
| Idle            | 0.004  | /    | 0.004  | 0.19    | 0.4   |
| LpIdle          | 0.004  | /    | /      | 0.19    | 0.4   |
| Standby         | 0.19   | /    | /      | /       | 0.4   |
| Sleep           | 0.4    | /    | /      | /       | /     |
service queue is small, the service lost ratio could be a serious constraint. We add the “UPPAAL-Con” strategy in the experiments, which is generated with “minimizing the power consumption, meanwhile, guaranteeing the average queue size is smaller than $Q$ and the service lost ratio is smaller than 0.005,” that is, $\text{done} = (\text{avQuS} < = Q)$ & $(\text{loSeR} < = 0.005)$ in the synthesis query. The choice of 0.005 is arbitrary in the experiment, and this value could be configured in practical applications.

**Figure 10.** Model of the service provider for the sensor node.
The power-performance tradeoff curves for the DPM strategies are shown in Figure 12; the average waiting time in the queue under the corresponding strategies is shown in Figure 13; and the service lost ratio under the corresponding strategies is shown in Figure 14 (the Y-axis is also log scaled in this figure).

From the figures, we have the following conclusions. It can be observed that when the average queue size is small (smaller than 0.02), the performance difference among the DPM strategies is not obvious. This is because, in order to achieve the desired performance metric, the SP seldom moves to the low-power states (Standby and Sleep states). Thus, all the DPM strategies have a similar behavior. The synthesized strategies outperform the timeout strategies and the stochastic strategies, especially when the average queue size is bigger than 0.05. And the synthesized strategies have comparable performance to the realistic strategies. We remark, however, the realistic strategies do not take the performance constraints into account. That is, the corresponding strategy construction approaches do not allow the designer to automatically contain the performance constraints when constructing the strategies.

Thus, the constructed strategies may violate the constraints. Note that UPPAAL-Con strategy has a higher power consumption than the remaining strategies when the average queue size is approximately 0.03. This is the consequence of the requirement of the service lost
ratio should be smaller than 0.005. This is also the reason for the absence of the UPPAAL-Con strategy when the average queue size is larger than 0.04. Therefore, it can be observed that the service lost ratio under the UPPAAL-Con strategy is smaller than that under the remaining strategies when the average queue size is approximately 0.03. And as the average queue size increases, the service queue has a higher probability to be full that leads to a higher service lost ratio, thus there exists an obvious tendency in Figure 14.

**Related work**

Simunick et al.\(^9\) provided a comprehensive assessment of state-of-the-art of DPM in WSNs. The aspects of power dissipation in a node are investigated, which consists of the power dissipated by the processor subsystem, communication subsystem, communication interfaces, and memory. Then, the strength and limitations of selective switching and dynamic scaling are investigated. Finally, some concerns in the implementation of the DPM techniques are proposed. This article presents a complete investigation of DPM in WSNs; our work is inspired by this article.

Benini et al.\(^1\) proposed a stochastic model for the formulation of DPM policy optimization based on Markov decision process. The policy optimization is cast into a linear programming problem. This problem is solved in polynomial time by efficient interior point algorithms. They demonstrate the flexibility of the approach with three case studies belonging to different classes of systems.

Qiu et al.\(^7\) formulated the system-level power management problem as an optimal policy selection problem based on the theories of continuous-time Markov decision process and stochastic networks. And the problem is solved by using linear programming and policy iteration. They solve the problem in continuous time domain instead of discrete time domain, which increases the accuracy. In addition, they consider the priority queue model that extends the scalability.

The above two articles deal with the DPM problem as an optimization problem, and solve this problem using linear programming. Linear programming is a reasonable way to solve this problem; however, it requires a mathematical model as an input, which is a complex and non-trivial work that limits its feasibility. In addition, they assume the transitions between the states follow the exponential distribution, that does not give an accurate representation of the real system.

Norman et al.\(^25\) illustrated the applicability of probabilistic model-checking to the area of DPM analysis. They model the system with the PRISM language in the form of DTMC and CTMC. After that, the optimization problem is passed to the MAPLE symbolic solver. The policies are automatically constructed with the solver. Once a policy has been constructed, its performance can be analyzed with the PRISM model checker. Irani et al.\(^26\) studied the adversarial modeling approach for power management, and stochastic model-checking for the evaluation of the effectiveness of DPM algorithms. In addition, Sesic et al.\(^27\) extended the work of Norman et al.\(^25\) with two different request priorities. Their works can be generalized to multiple request priorities and different queue length.

These works are similar to our approach; however, we apply the automatic synthesis approach to constructing the strategy, and they use an external symbolic solver to construct the strategy. They translate the results provided by the solver into the PRISM model in order to quantify the performance of the constructed strategies. By contrast, we employ only one tool to synthesize and evaluate the strategies, and no translation is required during the process.
Larsen et al.\textsuperscript{17} illustrated the usability of UPPAAL-STRATEGO for automatic synthesis of the optimal yet safe controller. They demonstrated the applicability on an Adaptive Cruise Control problem. They model the system components in the form of stochastic-priced timed games. Afterwards, UPPAAL-STRATEGO is used for strategy synthesis and analysis.

Wognsen et al.\textsuperscript{28} proposed a wear score function to compare and evaluate the relative impact of usage profiles on battery deterioration. The system is modeled as a network of stochastic-weighted timed games. With the combination of wear score function with the system model, and then subjected to UPPAAL-STRATEGO for optimization using reinforcement learning. So that they examine the tradeoff between short-term dynamic payload selection and the operational life of the battery.

Larsen et al.\textsuperscript{18} suggested a general and scalable methodology for controller synthesis for hybrid switched systems. They proposed an online synthesis methodology instead of offline synthesis, which periodically computes the controller only for the near future. What’s more, they propose and apply a compositional synthesis approach that improves the scalability. Finally, they demonstrate the applicability of the methodology with a case study to a floor-heating system of a real family house.

Applying model-checking for strategy synthesis has become a hot topic in recent years. These works are similar to our work, except that they deal with the different problems.

**Conclusion**

Optimizing power management is a critical issue in WSNs. In this article, we proposed a novel methodology for synthesizing near-optimal power management strategies for individual sensor node. The individual sensor node is regarded as a power-managed device. The generic system model for DPM was modeled in the form of timed games. And the synthesis objectives were expressed as the synthesis queries. Subsequently, a range of near-optimal strategies were constructed using UPPAAL-STRATEGO with respect to the synthesis queries. Once a strategy has been constructed, it was analyzed by the same tool. The methodology was illustrated with a running example, and its applicability was demonstrated by synthesizing a range of power management strategies for a concrete sensor node.

Our further work includes utilizing the online synthesis to extend the time horizon that can be handled. And it is possible to apply compositional synthesis to improve scalability. We are also considering applying our methodology to the systems that consist of multiple service queues and sensors. What’s more, investigating the impact of the local decisions made by a node upon the global WSN with statistical model-checking could be a potential research direction.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the State Key Program of National Natural Science Foundation of China under grant no. 61332001, the National Natural Science Foundation of China under grant no. 61272104, and the Basic Research For Application Foundation of Sichuan Province under grant no. 2014JY0112.

**References**

1. Benini L, Bogliolo A, Paleologo GA, et al. Policy optimization for dynamic power management. *IEEE T Comput Aid D* 2000; 18(6): 813–833.
2. Benini L, Bogliolo A and Micheli GD. A survey of design techniques for system-level dynamic power management. *IEEE T VLSI Syst* 2000; 8(3): 299–316.
3. Dhiman G and Rosing T. System-level power management using online learning. *IEEE T Comput Aid D* 2009; 28(5): 676–689.
4. Zhuravlev S, Saez JC, Blagodurov S, et al. Survey of energy-cognizant scheduling techniques. *IEEE T Parall Dist* 2013; 24(7): 1447–1464.
5. Dargie W. Dynamic power management in wireless sensor networks: state-of-the-art. *IEEE Sens J* 2012; 12(5): 1518–1528.
6. Sinha A and Chandrakasan A. Dynamic power management in wireless sensor networks. *IEEE Des Test Comput* 2001; 18(2): 62–74.
7. Qiu Q, Qu Q and Pedram M. Stochastic modeling of a power-managed system-construction and optimization. *IEEE T Comput Aid D* 2001; 20(10): 1200–1217.
8. Khan UA and Rinner B. Online learning of timeout policies for dynamic power management. *ACM T Embed Comput S* 2014; 13(4): 96:1–96:25.
9. Simunic T, Benini L, Glynn P, et al. Dynamic power management for portable systems. In: *Proceedings of the 6th annual international conference on mobile computing and networking (MobiCom '00)*, Boston, MA, 6–11 August 2000, pp.11–19. New York: ACM.
10. Svoreov M and Kwiatkowska M. Quantitative verification and strategy synthesis for stochastic games. *Eur J Control* 2016; 30: 15–30.
11. Jacobs S, Bloem R, Brenguier R, et al. The first reactive synthesis competition (SYNTCOMP 2014). *Int J Software Tool Tech Tran*. Epub ahead of print 9 April 2016. DOI: 10.1007/s10009-016-0416-3.
12. Behrmann G, Cougnard A, David A, et al. UPPAAL-Tiga: time for playing games! Berlin, Heidelberg: Springer, 2007, pp.121–125.
13. David A, Jensen PG, Larsen KG, et al. UPPAAL STRATEGO. Berlin, Heidelberg: Springer, 2015, pp.206–211.
14. Kwiatkowska M, Parker D and Wiltsche C. PRISM-Games 2.0: a tool for multi-objective strategy synthesis for stochastic games. Berlin, Heidelberg: Springer, 2016, pp.560–566.
15. Chen T, Forejt V, Kwiatkowska M, et al. PRISM-games: a model checker for stochastic multi-player games. Berlin, Heidelberg: Springer, 2013, pp.185–191.
16. Chen T, Kwiatkowska M, Simaitis A, et al. Synthesis for multi-objective stochastic games: an application to autonomous urban driving. Berlin, Heidelberg: Springer, 2013, pp.322–337.
17. Larsen KG, Mikučionis M and Taankvist JH. Safe and optimal adaptive cruise control. Cham: Springer International Publishing, 2015, pp.260–277.
18. Larsen KG, Mikučionis M, Muñiz M, et al. Online and compositional learning of controllers with application to floor heating. Berlin, Heidelberg: Springer, 2016, pp.244–259.
19. Hwang YS, Ku SK and Chung KS. A predictive dynamic power management technique for embedded mobile devices. IEEE T Consum Electr 2010; 56(2): 713–719.
20. Shih HC and Wang K. An adaptive hybrid dynamic power management algorithm for mobile devices. Comput Netw 2012; 56(2): 548–565.
21. David A, Larsen KG, Legay A, et al. Uppaal SMC tutorial. Int J Software Tool Tech Tran 2015; 17(4): 397–415.
22. David A, Jensen PG, Larsen KG, et al. On time with minimal expected cost! Cham: Springer International Publishing, 2014, pp.129–145.
23. Draeger K, Forejt V, Kwiatkowska M, et al. Permissive controller synthesis for probabilistic systems. Log Meth Comput Sci 2015; 11(2): 1–34.
24. Triki M, Ammari AC, Wang Y, et al. Reinforcement learning algorithms for dynamic power management. In: Proceeding sof the 2014 world symposium on computer applications & research (WSCAR), Sousse, Tunisia, 18–20 January 2014, pp.1–6. New York: IEEE.
25. Norman G, Parker D, Kwiatkowska M, et al. Using probabilistic model checking for dynamic power management. Form Asp Comput 2005; 17(2): 160–176.
26. Irani S, Singh G, Shukla SK, et al. An overview of the competitive and adversarial approaches to designing dynamic power management strategies. IEEE T VLSI Syst 2005; 13(12): 1349–1361.
27. Sesic A, Dautovic S and Malbasa V. Dynamic power management of a system with a two-priority request queue using probabilistic-model checking. IEEE T Comput Aid D 2008; 27(2): 403–407.
28. Wognsen ER, Haverkort BR, Jongerden M, et al. A score function for optimizing the cycle-life of battery-powered embedded systems. Cham: Springer International Publishing, 2015, pp.305–320.