Dependability Analysis of Automated Lighting Control System by Probabilistic Model Checking using the PRISM Model Checker

Archana Narayanan* , Hemanthkumar†
Department of Computer Science, University of Virginia
Charlottesville, VA, United States
Email: {*an2adv, †hg6va}@virginia.edu

***Abstract*—Automated Lighting has become a part of our daily routine, not being constrained to home use but also in healthcare, education, and every other corporate infrastructure. In such scenarios, it becomes necessary and crucial to evaluate the dependability of the system in order to make it more safe and reliable. We aim to utilize probabilistic model checking, one of the formal verification methods particularly using Continuous Time Markov Chains (CTMC'S) to study the dependability properties of this system and evaluate its reliability.**

***Index Terms*—Probabilistic Model Checking, PRISM, CTMC, Dependability analysis, Reliability check**

I. Introduction

At a time when everything around us is connected and driven by automated systems and the world is moving towards Internet-of Things, Automated Lighting Systems play a crucial role in every domain including smart homes, healthcare infrastructure, education etc. Automated Lighting is convenient, cost-efficient, and enhances security. The aged and disabled also benefit greatly from an automated lighting system. It also plays a crucial role in emergency situations.

Being able to predict the timeline and probabilities of failure of the components in automated lighting control, i.e., the sensors, actuators, I/O processors and main processors through the dependability analysis provides the framework to select the best possible iterations of the components and also to be well informed about maintenance intervals. This helps deploy a safe and secure system in place with increased efficiency.

So far, formal methods have been used to evaluate systems such as autonomous driving vehicles, pacemakers, power management systems, etc. We perform an experimental investigation on using probabilistic model checking using Markov modelling on an Automated Lighting System and evaluate the results obtained. The analysis could also be extended to other automated systems whose behavior is highly reliant on controllers and sensors.

II. Literature Survey

Probabilistic Model checking plays a significant role in predicting the reliability of a system and it does not come as a surprise that different types of Probabilistic Model checking has been explored and understanding its merits has been a topic of discussion for decades. In [1], Kwiatkowska et al. have explored the dependability properties of software based control systems using the probabilistic model checking. [2] discusses the latest formal methods for validating VLSI systems and software systems. [3] explores the conditions under which digital clocks are sufficient for performance analysis of probabilistic timed automata. [4] utilizes probabilistic model checking using PRISM to identify the best and worst case performance analysis of device discovery using Bluetooth.

All of these literature readings helped us understand the efficacy and merits of utilizing probabilistic model checking using PRISM for controller and software based systems.

III. Novelty

The systems analysed so far are very simple with few sensors and actuators(less than or equal to 3). The systems we studied in the literature survey contain bare minimum components essential for a working system such as sensors, controllers buses and actuators only and remain non-application specific. There has also been no formal verification method applied to studying Automated Lighting systems to the best of our knowledge.

The novelty of our verification model lies in the fact that it includes a cloud based storage model which is of prime importance as we migrate towards the Internet-of-Things. We also integrate a voltage check model that monitors fluctuations in output lights making the system more safe and reliable.

The proposed model we study consists of 10 sensors, 5 actuators, an input processor, main controller, output processor, bus module, a cloud storage module and voltage check module making the structure more complex mirroring real life scenarios.

IV. Aim of the project

The aim of our project was to utilize probabilistic model checking, for studying the dependability properties of automated lighting system and evaluate its reliability. As a result, we aimed to predict the timeline of failures for various components of an automated lighting system. This would lead to the deployment of a safe and secure system in place with increased efficiency. Furthermore, we also analysed the results to extract information that would help be well informed about maintenance intervals for this system.

V. Proposed Model for Analysis

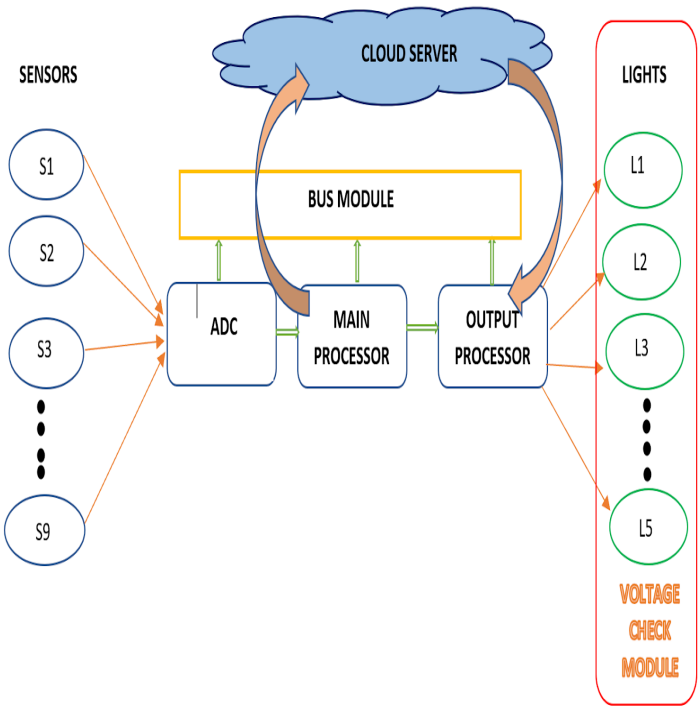

Figure 1. Proposed Model for analysis

Our proposed model encompasses a complex structure that consists of 9 sensors, an input processor, a main processor, an output processor and 5 actuators along with a bus module, cloud storage module and voltage check module as shown in the above figure. The functionality of each component is as explained,

- Sensors: Sensor components detect change in environment and send data to the input processors. The sensors include motion detection sensors, contact sensors et al. We include 9 sensors in our system to enhance the complexity of our system.
- Input processor : The input processor is an analog to digital converter(ADC) that gathers analog information from the sensors and converts it to digital values for further processing. The output of the analog to digital converter is fed to the main processor module.
- Main processor module : The main processor module is the brain of the system that produces output results based on the information obtained from the analog to digital converter. Furthermore, the main processor also sends information to the cloud storage module to facilitate data access for other devices connected via the Internet-of Things.
- Output Processor : The output processor receives results from the main processor and also the cloud module to trigger the actuators.
- Actuators: The actuators in our system consists of lights connected to the output processor. There are 5 lights connected to the output processor in our system.
- Voltage Check Module : The voltage check module checks for voltage fluctuations in actuators and their working conditions for correctness before activating the lights.
- Bus Module : Finally the bus module comprises of data and instruction buses that connect the analog to digital converter, main processor and the output processor.

VI. Methodology

The working of the system is reliant equally on all the components of the system as each component has its own significant role to play in the complete functioning of the system. Data is collected from sensors such as motion sensors, contact sensors etc. Input processor collects this data and transmits it to main processor. The main processor then analyses this information and sends to cloud module and output processor. Data from cloud module is retrieved by the output processors in real-time along with results from main processor. Lastly, the lights are activated based on the results after voltage check pass. This process occurs in a cycle repeatedly, with the length of the cycle controlled by a timer in the main processor

A. Working Conditions

We define a set of conditions for the system to continue its functionality taking into account all the components present in the system. Any of the 9 sensors can fail, however a minimum of three sensors are required to be working for the analog to digital converter module to gather adequate information to proceed. If more than 3 sensors become non functional, then the ADC module reports this to the main processor module and the system is shut down. In the same manner, one out of 10 lights are required to be working for the system to remain functional. However, if this condition is not satisfied them the output processor directs the main processor to shut down the system.

The analog to digital converter module and the output processor can also fail. They are classified into two types of failures, transient failure and permanent failure. In the former case, the system reboots itself to return to functional state and the situation is rectified. In latter case, this would result in the main processor not being able to either receive information from the analog to digital converter module or send data to the output processor and a cycle of operation is skipped.

We identify this permanent failure by defining a maximum number of times that the modules can skip cycles before they shut down. If the number of times the main processor skips its cycle of operation is greater than the maximum skip cycle count defined, the system is shut down. The same process holds good to detect cloud module failure as well.

The main processor and voltage check module can themselves fail, in which case the system is completely shut down.

VII. Probabilistic Model Checking

Model checking is a well established method for the verification of finite state concurrent systems. Model checking involves the construction of a state transition diagram which represents the system in real time. This state transition diagram consists of all possible states that the system can be in and all the possible transition occurrences between the states. The system specifications are expressed using temporal logic formulas which are sent to a model checker and efficient symbolic algorithms are used to navigate through the model while checking if the defined conditions are satisfied. This system is used in various industries to verify largely complex systems in real-time.

Probabilistic Model Checking is a variant of model checking wherein it involves the modelling and analysing of systems that exhibit probabilistic behaviour. It facilitates the automatic formal verification of systems that exhibit stochastic behaviour. We built a formal model for the system where the model is provided with probabilistic information such as the likelihood of failures. We utilize Continuous Time Markov Chains(CTMC's) which are a class of probabilistic models that are used for dependability analysis of probabilistic systems.
A CTMC comprises a set of states S and a transition rate matrix $R : S \times S \rightarrow IR \geq 0$.

The rate R defines the delay before which a transition between states s and s' is enabled.

A CTMC makes transitions from state to state, independent of the past, according to a discrete-time Markov chain (DTMC), but once entering a state remains in that state, independent of the past, for an exponentially distributed amount of time before changing state again.

This is suitable for our model since it contains exponentially distributed delays which are apt for modelling component lifetimes and inter-arrival times. This can also be used to approximate more complex probability distributions.

Other models include the Discrete Time Markov Chains(DTMCs), wherein the probability of moving between states is calculated in discrete time-steps, and Markov decision processes (MDPs), which are used to model systems which exhibit both probabilistic and non-deterministic behaviour.

VIII. PRISM

PRISM is a tool used for probabilistic model which supports the construction and analysis of probabilistic model such as CTMCs, DTMCs and MDPs and many more. Models are described using the PRISM language, a simple, state-based language. Each module comprises of a component in a real life system. The property specification language incorporates the temporal logics PCTL, CSL, LTL and PCTL, as well as extensions for quantitative specifications and costs/rewards as mentioned in [5]. Each module also has a set of variables that represents the state that the model is in and used for transition between states. The full complete module is the parallel construction of many singular modules integrated together. PRISM also computes all states which are reachable from the initial state and helps in the identification of states that are in a deadlock unable to proceed. It calculates the probability of the defined property in place and also helps us to visualize the trends in the form of graphs. Below attached figure shows the GUI of PRISM tool with an experiment in progress.

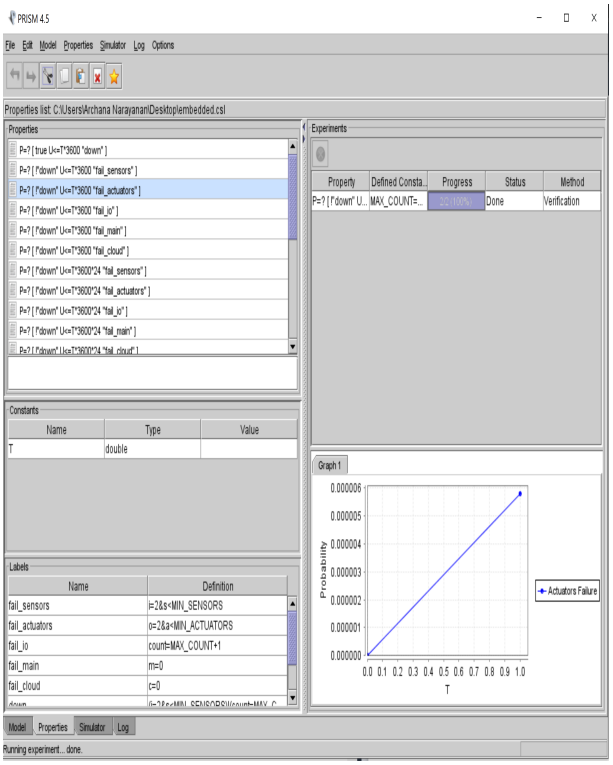

Figure 2. PRISM Tool GUI

IX. Modules

The Automated Lighting system is modelled in PRISM as a set of modules i.e Sensors, I/O Processors, Actuators, Main Processor, Bus, Cloud and Voltage Regulator. The mean times of failure for a sensor, actuator, processor, cloud and the voltage (fluctuation at the actuator where the voltage is not in the operational range) are 1 month, 2 months, 1 year, 1 hour, 1 month and 1 week respectively. On an average, a transient fault occurs at the I/O processors once a day and the mean times for a timer cycle expiry, a cloud re-transmission and a processor reboot are 30 seconds, 30 seconds and 1 minute respectively.

The PRISM code for the sensor module is shown in figure [3]. It contains a variable 's' which represents the number of active sensors. The action s' = s-1 decreases the count of the active sensors at a rate $s*\lambda_s$ and the command "$s > 0$" states that the action can occur only when the number of sensors is greater than zero.

Figure [4] shows the PRISM code for the input processor module. The variable "i" holds the value *0*, *1* or *2* depicting the state of the processor as *failed*, *transient fault* and *ok* respectively. The input processor can fail (new state, i' = 0) at a rate of λ_p, when the current state is not failed ($i > 0$) and the

```
// sensors
// a sensor fails once a month on average
const double lambda_s = 1/(30*24*60*60);
module sensors
        // number of sensors working
        s : [0..9] init 9;
        // failure of a single sensor
        [] s>0 -> s*lambda_s : (s'=s-1);
endmodule
```

Figure 3. PRISM code for sensor module.

```
// input processor
// (takes data from sensors and passes onto main processor)
const double lambda_p = 1/(365*24*60*60); // 1 year
const double delta_f = 1/(24*60*60); // 1 day
const double delta_r = 1/30; // 30 secs
module proci
        // 2=ok, 1=transient fault, 0=failed
        i : [0..2] init 2;
        // failure of processor
        [] i>0 & s>=MIN_SENSORS -> lambda_p : (i'=0);
        // transient fault
        [] i=2 & s>=MIN_SENSORS -> delta_f : (i'=1);
        // reboot after transient fault
        [input_reboot] i=1 & s>=MIN_SENSORS -> delta_r : (i'=2);
endmodule
```

Figure 4. PRISM code for I/O Processor module.

total number of working sensors are greater than the minimum threshold. Transient fault can occur at a rate of δ_f, when the input processor is in *ok* state (i=2) the number of working sensors is above the threshold. Once a transient fault occurs, the input processor takes δ_r time to reboot and to be back into working condition (i'=2). Figure [5] shows the PRISM code for an actuator module. It is similar to the sensor module, with a variable a holding the number of active actuators and the actuators fail with a probability λ_a. The output processor is modelled similar to the input processor.

As shown in figure [6], the main processor module has a variable *m* to hold the state, 0 for *failed* and 1 for *ok*.

```
// actuators
// an actuator fails once every 2 months on average
const double lambda_a = 1/(2*30*24*60*60);
module actuators
        // number of actuators working
        a : [0..10] init 10;
        // failure of a single actuator
        [] a>0 -> a*lambda_a : (a'=a-1);
endmodule
```

Figure 5. PRISM code for actuator module.

```
// main processor
module procm
        m : [0..1] init 1; // 1=ok, 0=failed
        count : [0..MAX_COUNT+1] init 0;
        [] m=1 -> lambda_p : (m'=0);
        [timeout]  comp -> tau : (count'=0);
        [timeout] !comp -> tau : (count'=min(count+1, MAX_COUNT+1));
endmodule
```

Figure 6. PRISM code for main processor module.

```
//The cloud module can fail once every month
const double lambda_c = 1/(30*24*60*60);
module cloud
        c : [0..1] init 1;
        count_c : [0..MAX_COUNT+1] init 0;
        //failure to transmit to cloud
        [] c=1 -> lambda_c : (c' = 0);
        // transmission completed before timer expires
        //- reset skipped cycle counter
        []  comp -> time_c : (count_c'=0);
        // transmission not completed before timer expires
        //- increment skipped cycle counter
        [] !comp -> time_c : (count_c'=min(count_c+1, MAX_COUNT_CLOUD+1));
endmodule
```

Figure 7. PRISM code for cloud module.

The variable *count* holds the number of consecutive cycles skipped by the processor. The main processor can fail with a probability of λ_p. When the I/O processors enter a transient state, the main processor cannot read from the input or write to the output processors and is forced to skip the current cycle. The main processor waits for a time period τ before deciding to skip the cycle. If the processing is completed before the time-out, the variable *count* is set to zero. Else, the *count* variable is increased by one. If the count of consecutive cycles skipped crosses a threshold, the processor is shut down.

The PRISM code for the cloud module is shown in figure [7]. The cloud module contains a variable *c* which holds the state (1 for *working*, 0 for *failure*) and the variable *count* to keep track of the number of re-transmissions. The cloud module can fail with a probability of λ_c. The cloud module waits for a time period τ_c for a successful transmission of data to the cloud. When the data transmission is unsuccessful within the time-out period, the cloud module increases the *count* variable and attempts re-transmission. The module is deemed to be failed if the count of the consecutive re-transmissions cross a threshold value. The voltage regulation module represents the fluctuation of available voltage at the actuator and the module fails with a probability of λ_v which signifies that the voltage available is not in the operational range of the actuator.

The determination of the state of the system, whether *up*,

```
formula down = (i=2&s<MIN_SENSORS)|(count= MAX_COUNT+1)
|(o=2&a<MIN_ACTUATORS)|(m=0)| (count_c= MAX_COUNT_CLOUD+1)|(c=0);
formula danger = !down & (i=1 | o=1 | v=0 );
formula up = !down & !danger;
```

Figure 8. PRISM code for defining the states.

down or *danger* can be seen in figure [8]. The failure of any single module will result in the system going *down*. The scenarios where the number of active sensors and actuators go down the threshold or the number of cycles skipped in the processor or the number of re-transmissions in cloud module go above the threshold also pushes the system into *down* state. The system goes into *danger* state when the I/O processors go into the transient state or when the voltage regulation check at the actuator fails. The system is said to be in the *up* state when it is not in both *down* and *danger* states.

X. Results

PRISM model checker is used to construct the CTMC model which represents the automated lighting control system and to analyse the dependability properties using probabilistic model checking. We use different CSL properties to determine failures and expected time in a state. The following is a CSL property for failure of one of the subsystems - sensors (1), I/O processors (2), actuators (3), main processor (4), cloud (5)

$$P_{=?}[\neg\ down\ U^{\leq T} fail_{\mathrm{j}}]$$

where j =1..5 refers to one of the five failures mentioned above, T is the number of hours simulated and *down* denotes that one of the failures has occurred. The probability of shut down due to each single cause and the probability of a total failure is simulated over a period of 24 hours and the results are shown in figure [9].

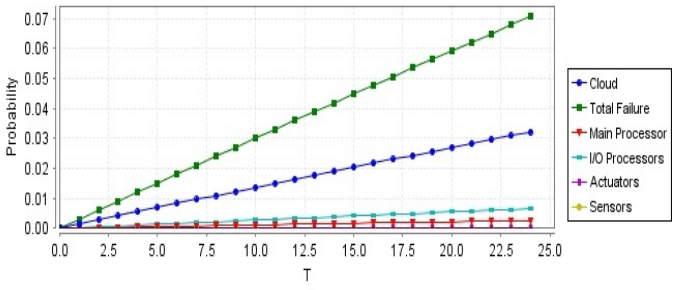

Figure 9. Probabilities of failure due to each single cause.

The system is classified to be present in any of the following three states, *up*, *down* or *danger*. Time spent in each state is calculated by assigning a cost of 1 to that particular state and assigning a cost of 0 to the other states, and calculating the total reward. The CSL property to estimate the time spent in a state until time T is,

$$R =?[C \leq T]$$

Using this property, the expected time spent by the autonomous lighting system in each of the three states is simulated for 24 hours and the results are shown in figure [10].

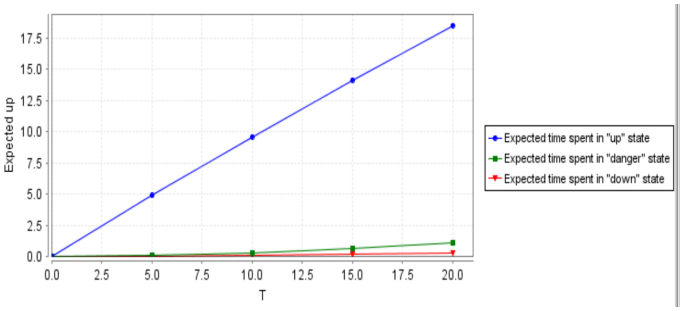

Figure 10. Expected time of system spent in each state by time T.

Another CSL property which is used to determine the long run probability of each failure type occurring is defined as,

$$P_{=?}[\neg\ down\ U fail_{\mathrm{j}}]$$

where j=1..5 refers to one of the aforementioned five failures. These long run probabilities of each failure are shown in Table 1.

Table I
Long Run Probabilities of Failure

Failure Type	Probability
Cloud	0.371
Sensors	0.162
Actuators	0.120
I/O Processors	0.083
Main Processor	0.030

A. Analysis

From the *figure 9* above it can be observed that the probability of failure of cloud, main processors, I/O processors, actuators and sensors is linearly increasing with time. This suggests that the difference in probabilities between any two time intervals is constant. However, the rate of increase of probability of failure for each component is different. This comes as a result of defining different mean times of failures for these components.

The slopes of this graph clearly delineates the fact that the probability of failure for cloud module increases at a faster rate than rest of the components. The total failure denoted is the sum of all component failures satisfying *shut down* condition described in the previous section.

We can infer from these results that the total failure of the system can be decreased effectively to a large extent by preventing cloud module failure with regular maintenance at frequent intervals. In our simulation, the cloud module is expected go down once in a month. This probability is inferred from the frequent disruptions for the Computer Science department servers at the University of Virginia.

From *figure 10* it can be observed that at any point of time the duration of the system in *up* state is comparatively much

higher than the duration for which it will be in *down* or *danger* state. This plot also suggests that the probability of the system being in either *down* or *danger* state is negligibly low during the first 10 hours. After this time period, the probability of the system to be in *danger* state increases linearly at a faster rate than the probability of the system being in *down* state. This will give us enough time for monitoring the system and serve as a warning period during which action is required to be taken. This statistics generated for a longer period of time will give adequate results to predict the failure of the system. The system is likely to shut down very soon after it enters the *danger* state.

XI. Conclusion and Future Work

The automated lighting control system is devised as Continuous Time Markov Chain models and the dependability, reliability properties of the system are analysed by probabilistic model checking using the PRISM model checker. Temporal Logic is used to define properties for verification. Novel models like cloud module and voltage check constraint are included in the system. With this framework of probabilistic determination of timelines of failure for every component and the system as a whole, potential system outages can be identified in advance and maintenance schedules can be drawn out efficiently. In our model, the failures of individual components are determined using probabilities. Future work might include developing sub-modules to determine the individual component failure in a more non-probabilistic, realistic manner. Our work can be extended to additional modules that can be included to represent IoT systems and their dependability can be studied.

References

[1] Kwiatkowska, M., G. Norman and D. Parker, "Controller dependability analysis by probabilistic model checking", Control Engineering Practice Volume 15, Issue 11, November 2007, Pages 1427-1434

[2] Alur, R. and Henzinger, A., "Formal methods in system design".

[3] Kwiatkowska, Marta and Norman, Gethin and Parker, David and Sproston, Jeremy, "Performance Analysis of Probabilistic Timed Automata Using Digital Clocks".

[4] Duflot, Marie and Kwiatkowska, Marta and Norman, Gethin and Parker, David, "A Formal Analysis of Bluetooth Device Discovery".

[5] M. Kwiatkowska and G. Norman and D. Parker., "4.0: Verification of Probabilistic Real-time Systems".