Master's Thesis

Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction

Jiwon Choi (최 지 원)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2018


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

컨텍스트 추출을 기반한 스마트 홈 IoT 장치의 결함 감지 및 식별

Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction

Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction

by

Jiwon Choi
Department of Computer Science and Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Computer Science and Engineering

Pohang, Korea
12. 26. 2017
Approved by
Jong Kim
Academic advisor

Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction

Jiwon Choi

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master's degree from POSTECH

12. 26. 2017

Committee Chair Jong Kim

Member Chanik Park

Member Hanjun Kim

MCSE 최 지 원. Jiwon Choi
20162498
Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction,
컨텍스트 추출을 기반한 스마트 홈 IoT 장치의 결함 감지 및 식별
Department of Computer Science and Engineering , 2018, 43p, Advisor : Jong Kim. Text in English.

ABSTRACT

A fast and reliable method to detect faulty IoT devices is indispensable in IoT environments. In this thesis, we present DICE, an automatic method to detect and identify faulty IoT devices with context extraction. Our system works in two phases. In a precomputation phase, the system precomputes sensor correlation and the transition probability between sensor states known as context. During a real-time phase, the system finds a violation of sensor correlation and transition to detect and identify the faults. In detection, we analyze the sensor data to find any missing or newly reacting IoT devices that are deviating from already grouped correlated sensors, and state transition to find the presence of an abnormal sequence. Then, the system identifies the faulty device by comparing the problematic context with the probable ones. We demonstrate that DICE identifies faulty devices accurately and promptly through the evaluation on various fault types and datasets.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

Contents

List of Tables

List of Figures

I. Introduction

Internet of Things (IoT) has proliferated significantly and has pervaded various areas of human lives. IoT environments such as Smart Homes and Smart Cities are becoming increasingly popular and have received growing attention over the past few years. Using sensors and actuators, these IoT applications offer convenient services to users. An example of such services would be a smart bulb that turns on when a motion sensor detects a nearby motion.

However, IoT devices are lightweight devices exposed to harsh environments and have limited computational capabilities, which cause frequent device failure [1–3]. Sensors in general suffer from many types of faults due to hardware failure, limited battery life, or human mistakes [4, 5]. These sensor faults in IoT environments are particularly more critical and may lead to dire consequences because their negative effects do not remain in the cyber system, but extends to the physical domain. For example, if a Smart Blind in a Smart Home is drawn up at night due to a faulty light sensor, the privacy of the user may be exposed to the neighbors. Thus, preserving the integrity of the data is an essential requirement in IoT environments.

Recent studies have proposed solutions to detect faulty sensors in an IoT environment [5–9] by using the correlation either between activities and sensors [5,6] or between actuators and sensors [7]. However, such studies are not suitable for actual IoT environments for the following reasons. First, existing solutions require user intervention. They require users to annotate each activity during the training period or provide additional information such as the sensor location. User intervention not only makes the system less usable, but also increases chances of error due to user mistakes [1,5,8]. Second, the experiment settings of the existing

solutions do not represent actual IoT environments, and thus have a generalization fallacy. Real-world Smart Homes are generally deployed with various sensors and actuators [10]. Device faults in real-world often occur in an unpredictable manner. However, existing solutions only consider a few specific types of sensors or faults. Third, existing solutions require too much computation cost or make unrealistic assumptions. Some solutions require a considerable number of sensors to operate, and yet the complexity of the system grows exponentially with the types of sensors [5]. Last, the time to detect a faulty sensor is overly long for some of the existing work. When the detection time takes too much time, the failure may cause serious problems [11]. For such reasons, all prior art is not adequate for the IoT environments. A novel IoT device fault detection method that resolves such problems is necessary.

In this thesis, we propose DICE, an automatic faulty device **D**etection and **I**dentification solution based on the **C**ontext **E**xtraction. Our devised method does not require recognition of user's activity, and works for various device and fault types. Furthermore, our system is faster in detecting and identifying a faulty device than state of the art. Our system consists of two phases: a precomputation phase and a real-time phase. The real-time phase is divided into two steps: detection and identification. During the precomputation phase, DICE precomputes the correlations among sensors and makes sensor states with correlated sensors. Next, we calculate the transition probability of the sensor states. During the real-time phase, DICE detects the presence of device failure by performing a correlation check that detects correlation violations among sensors and a transition check that detects transition violations of the sensor state sequence. Once a sensor fault is detected, DICE identifies which device has generated the faulty data by comparing the problematic context with the most probable context.

To evaluate our system, we implemented a Smart Home testbed that is

deployed with various types of sensors and actuators to represent real-world environments. Furthermore, we not only tested our system on our own datasets, but verified our system with the five publicly available Smart Home datasets. The evaluation results shows that our system successfully detects faults with an average of 94.9% precision and 92.5% recall from both the internal and public third party datasets. Thus, adopting our system highly enhances the reliability of IoT environments.

Our main contributions are summarized as follows:

- **Novel design.** We propose a novel faulty device detection and identification method that enhances the reliability of Smart Homes.

- **Usability.** We devise a context extraction method that do not require any form of user intervention. Using our method, a user does not have to annotate the activities or provide additional information about the deployment. Our method automatically extracts the context (*i.e.,* correlation and transition probability) from Smart Home data.

- **Generality.** Our method works for both binary and numeric values. Note that the more heterogeneous the sensors are, the more challenging it is to extract the correlation of sensors because different types of sensors react differently to the same event. We have used datasets with varying types of sensors and actuators. Thus, this demonstrates that our system is easily adoptable to real-world Smart Homes. We have tested our system on various fault types that frequently occur in real-world Smart Homes. Our experimental result confirms that our method can effectively detect and identify device faults of different types with high accuracy.

- **Feasibility.** Our method requires a small computation time. We eliminated any use of complex algorithms or techniques to simplify the process.

- **Promptness.** Our method can detect device faults promptly. Our method takes an average of 3 minutes to detect sensor faults. This is much faster than the fastest reported average detection time of prior art which is 12 hours. For identification, our method takes an average of 28 minutes. In optimal conditions, the time can be reduced down to 7 minutes.

The rest of the thesis is organized as follows. In Chapter II, we review the prior art that performs sensor failure detection and identification. Chapter III presents the detailed architecture and design of DICE. Chapter IV shows the datasets we used and the deployment settings of our Smart Home testbed. Chapter V explains the evaluation results of DICE, which is followed by discussion and conclusion in Chapter VI and VII, respectively.

II. Related work

Sensor failure detection and identification have received considerable attention in the research community. Particularly, three streams of work have been conducted for automatic sensor failure detection and identification: network-level approach, homogeneous approach, and heterogeneous approach.

2.1 Network-level Approach

Network-level solutions have been proposed for sensor failure detection. [12–14] develop a network management system to detect anomaly in the network level by monitoring packets. [12] devises a distributed sensor node failure detection method, in which the sensor nodes in the network cooperatively monitor each other to detect problematic sensor nodes. [15] aggregates distributed data at the sink and detected failure by finding insufficient flow of incoming data. [16] uses Markov models to characterize the normal behavior of sensor networks. It builds a Markov model at each sensor node to estimate anomaly-free probabilities from its past observation traces and derive optimal anomaly detection rules for sensor failure detection. These solutions are effective for finding sensor failure in the network level, but the failure correspond to fail-stop sensor faults that stop generating values after the failure. When there are non-fail-stop faults that continues to report incorrect but technically reasonable values, such network-level solutions cannot be enforced.

2.2 Homogeneous Approach

Homogeneous approach deploys multiple sensors of the same type to detect and identify a sensor that shows an anomalous behavior. Most work exploits the fact that the same type of sensors that are spatially close to each other generate similar values. In contrast, a faulty sensor will show an uncorrelated behavior with other closely located sensors. From this intuition, [17] detects an abnormal sensor if the value of the sensor shows significant deviation from the values of other nearby sensors. [18,19] apply the majority rule to detect and localize faulty sensors in wireless sensor networks. It locally performs majority voting, where each sensor node makes a decision based on comparison between its own sensor reading and sensor readings of one-hop neighbors.

Another body of work analyzes time-series data collected from multiple sensors to identify the faulty sensor. [20,21] use Auto Regressive Integrated Moving Average (ARIMA), a time-series model that compares the predicted measurement with the reported measurement. [20] uses a previously reported value of one sensor to estimate the future reading of another sensor. [21] compares the sensor measurement against its predicted value using the time series forecasting method to detect a fault.

There are several other solutions besides the majority-rule-based or time-series-analysis based solutions. [22] proposes a reputation-based framework, which uses the correlation between neighboring sensors of the same type to detect non-fail-stop sensor faults. [23] detects a non-fail-stop failure based on the assumption that sensor readings reflect the distance from the nodes to the event. [24] applies Dynamic Bayesian Network (DBN) to analyze and diagnose anomalous wind velocity data. It builds individual sensor models, on top of which a coupled DBN model is learned to represent the joint distribution of two sensors.

Substantial studies take the homogeneous approach to resolve sensor failure

problems. However, the redundancy of deploying multiple sensors of the same type spatially close to each other increases the cost of deployment. Furthermore, most of the majority voting solutions take threshold-based decisions, but finding an optimal threshold is difficult. Improper threshold values decrease the accuracy of the system. Such limitations of the homogeneous approach restrain them from being deployed to actual IoT environments.

2.3 Heterogeneous Approach

Heterogeneous approach combines different types of sensor data to detect sensor failure. The approach has only become popular recently with the emergence of IoT environments that are deployed with various types of sensors. SMART [5] develops a system that detects sensor failure based on the classifier outputs. It trains the classifier to recognize the same set of activities based on different subsets of sensors. At run-time, its system detects failure by analyzing the relative behavior of multiple classifier instances trained to recognize the same set of activities based on different subsets of sensors. FailureSense [7] explores the relationship between the electrical appliances and sensor data to detect a sensor failure at home. First, its system learns the regular behavior of a sensor with respect to electrical appliance activation at home. Then it continuously monitors the behavior of sensors and alarms users when sensor reaction shows significant deviation from the regular behavior. IDEA [6] performs activity recognition based on the sensor data and computes the sensor's impact for each activity. It computes the conditional probability of a sensor not to be triggered with the given activities, and regards the sensor as failure when the probability is lower than a certain threshold. CLEAN [8] suggests a clustering-based outlier detection technique to detect anomalies in event-driven binary sensors. It uses Least Common Subsumer (LCS) to quantify the similarity of any two sensor events.

Based on the LCS distance, it clusters sensors and detects outliers in clusters as failure. 6thSense [9] devises a context-aware intrusion detection system for smart devices by monitoring changes in sensor data. It creates a contextual model to distinguish benign and malicious behaviors of sensors and utilizes three different machine learning based detection mechanisms (i.e., Markov Chain, Naive Bayes, and LMT) to detect a malicious behavior associated with sensors.

While these heterogeneous solutions effectively detect sensor failure without using redundant homogeneous sensors, they left much to be desired. The followings are requirements a solution should satisfy:

- **Usability.** Usability is the condition of which a system does not require any user intervention in the form of either requiring labeled training or supplying additional information. For instance, requiring users to perform different activities and annotate the start and end time of each activity during the training period greatly harms the usability.

- **Generality.** Generality is the condition of which a system considers various sensor types and fault types that represent real-world settings. In real-world settings, Smart Homes are equipped with not only binary sensors but also numeric sensors. In fact, the most commonly used sensors in Smart Homes are infrared sensors, RFID sensors, motion sensors, accelerometers, cameras and microphones [10]. A solution should be able to cover both binary and numeric values. Furthermore, sensor faults in real-world often occur in an unpredictable manner. A system that considers only specific sensor types or faults is either inapplicable to real-world Smart Homes or susceptible to real-world faults.

- **Feasibility.** Feasibility is the condition of which a system does not require too expensive computation cost or not make chimerical assumptions that

Table 2.1: Analysis of Heterogeneous Approach

	Usability	Generality	Feasibility	Promptness
SMART [5]	✗	✗	✗	✗
FailureSense [7]	✓	✗	✗	–
IDEA [6]	✗	✗	✓	✗
CLEAN [8]	✗	✗	✓	–
6thSense [9]	△	✗	✗	–
DICE	✓	✓	✓	✓

are far from reality.

- **Promptness.** Promptness is the condition of which a system detects or identifies a faulty sensor promptly. When a system takes too much time to identify the faulty sensor, the fault may cause serious problems due to propagation of the fault effect.

Table 2.1 shows whether the aforementioned heterogeneous sensor failure detection methods meet these requirements, along with our proposed scheme. Most of the methods fail to meet the usability requirement. SMART [5] and IDEA [6] which detect sensor failure based on the activity and sensor correlation require users to annotate each and every activity during the training period. 6thSense [9] also uses labeled training because it examines the correlation between a user's activity and sensors. It also proposes a Markov Chain-based method that does not require annotation. Thus, we mark them as '△' in the table. CLEAN [8] eliminates the burden of annotating activities, but instead requires users to provide information about the location and the hierarchy of sensors.

All five methods do not satisfy the generality requirement. All heterogeneous solutions except 6thSense only consider binary sensors. Moreover, no prior art

satisfies the generality requirement of sensor failure detection by lack of consideration of either sensor types or fault types. IDEA only detects fail-stop faults, while SMART and CLEAN only detect non fail-stop faults. 6thSense has no consideration of different fault types.

SMART, FailureSense [7], and 6thSense do not satisfy the feasibility requirement. SMART require a considerable number of sensors to operate. However, the complexity of constructing the classifier profile grows exponentially with the number of sensors. FailureSense makes an assumption that a sensor and an event in an appliance must always have a connection. Thus, the detection accuracy of the sensors unconnected to an appliance is 0% in its experiment result. 6thSense defines a set of activities composed of nine daily activities humans perform while using a smartphone. Thus, the false positive rate of 6thSense could be high for real-life situations because it regards any activities that do not belong to the existing nine activities as an error.

Only two among the five studies investigate the time required for sensor fault detection and identification. We mark the studies that do not perform any evaluations on the time as '–' in the table. Two studies, SMART and IDEA, perform evaluations on time but do not meet the promptness requirement. They require at least 12 hours for detecting and identifying a faulty sensor. 6thSense detects the presence of a faulty sensor but does not identify the sensor that has caused a failure, which implies a manual inspection of the identification of a faulty sensor. Thus, these schemes that require too much time or do not identify the faulty sensor are hard to fix or replace the faulty sensor promptly enough before serious problems occur.

III. DICE: Detection & Identification with Context Extraction

In this chapter, we present a detailed design and architecture of DICE, an automatic detection and identification method for faulty IoT devices.

3.1 Overview

The typical Smart Home we consider (Figure 3.1) consists of diverse IoT devices, a home gateway, and a network connecting all devices and outside world [25]. The home gateway works as a central control point for collecting sensor data and sending control commands to IoT devices. All IoT devices in Smart Home are connected to this home gateway directly or indirectly. The home gateway is connected to a cloud server as an option to provide services linked to other IoT service providers. DICE will be installed on this home gateway.

Figure 3.2 shows the overview of our DICE system. The system consists of two phases: the precomputation phase and the real-time phase. During the precomputation phase, our system extracts the correlation and sensor state transition information from the collected IoT sensor and actuator data. During the real-time phase, our system performs a correlation check and a transition check to detect faults in real-time based on the precomputed correlation and transition. After detection, the system identifies the faulty devices by analyzing the context.

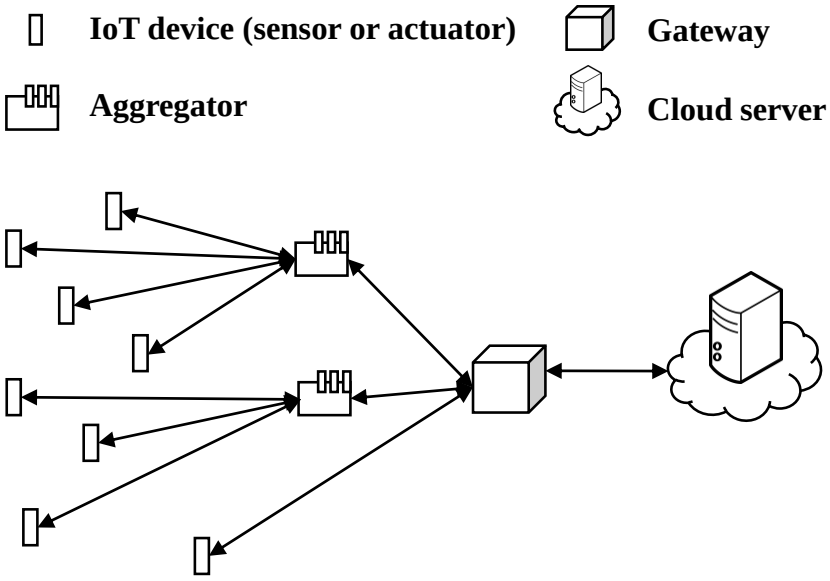


Figure 3.1: Typical Smart Home

3.2 Precomputation Phase

During the precomputation phase, DICE collects the normal behaviors of the sensors. DICE records the context of the IoT environment by extracting the correlation among sensors and by computing the transition probability of the sensor state sets and actuators. Since actuators affect sensor readings, the extraction of only sensor correlation is sufficient. It reduces the complexity of precomputation. We assume that the data collected during the precomputation phase preserve its integrity and the IoT devices are without faults.

3.2.1 Correlation Extraction

The sensor correlation is extracted from raw sensor data. We denote the sensors as $S : S_{t,1}, S_{t,2}, ..., S_{t,k}, ..., S_{t,n}$, where t represents the time at which the data was collected, k represents the ID of a sensor, and n represents the number of sensors. Similarly, we denote the actuators as $A : A_{t,1}, A_{t,2}, ..., A_{t,j}, ..., A_{t,m}$, where j represents the ID of an actuator and m represents the number of ac-

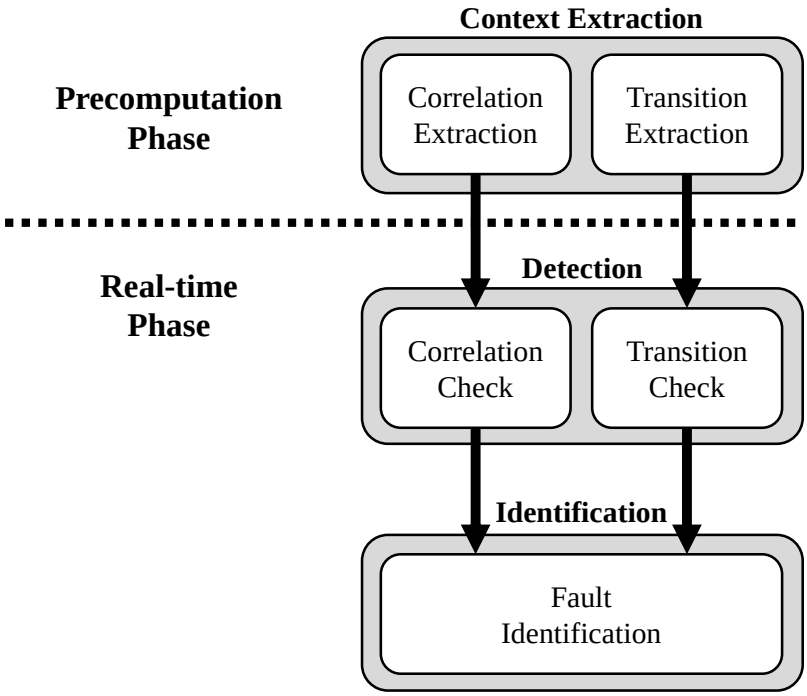


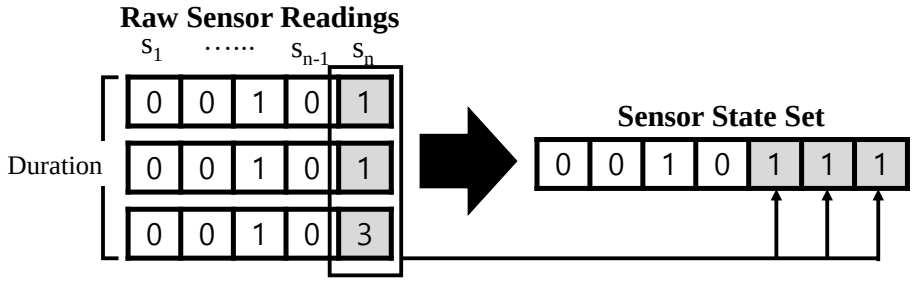
Figure 3.2: DICE Overview

tuators. DICE constructs a sensor state set from the incoming raw sensor data (Figure 3.3a). A sensor state set is composed of bits that represent the activation status of each sensor within a predefined duration of time. We set the duration of the sensor state empirically in our experiment, e.g. one minute. There are two classes of sensors we consider: a binary sensor and a numeric sensor. DICE applies different techniques to set activation status on the two sensor classes. A single binary sensor is represented by a single bit:

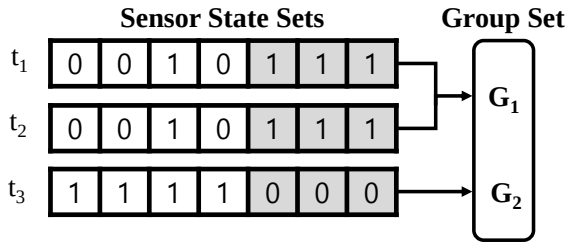
$$|S_{t,k}|S_{t+1,k}|....|S_{t+d,k}| \tag{3.1}$$

where d is the duration of the sensor state.

When a binary sensor is activated at least once during the predefined duration of time, the bit of the corresponding sensor is set to '1' and otherwise 0' (Eq. (3.1)). We use a bit-wise 'or' operation to implement it.



(a) Construction of Sensor State Set



(b) Group Generation

Figure 3.3: Correlation Extraction

When a numeric sensor is activated, the system applies three formulas (Eqs. (3.2)-(3.4)) to deduce the activation status, in which the results of each formula represent a bit in the state set.

$$E\left[\left(\frac{S_k - \mu^3}{\sigma}\right)\right] > 0 \tag{3.2}$$

$$S_{t+d,k} - S_{t,k} > 0 \tag{3.3}$$

$$\frac{\sum_{i=t}^{t+d} S_{i,k}}{d} > valueThre \tag{3.4}$$

where $valueThre$ represents a threshold value.

Eq. (3.2) determines if the skewness of the data during the duration exceeds zero. Eq. (3.3) examines the increasing or decreasing trend of the sensor values. Eq. (3.4) determines if the mean sensor value exceeds the threshold, $valueThre$. We set $valueThre$ as the corresponding sensor's mean value of the data collected

during the precomputation phase. The conversion of a numeric sensor value into binary values enables DICE to be applied to any sensors regardless of their classes, while minimizing the computation cost. Thereby, we achieve the generality and feasibility requirements.

Each unique sensor state set during the precomputation phase is given a unique ID and is referred to as a **group** (Figure 3.3b). We denote a set of groups as $G : G_1, G_2, ..., G_x, ..., G_l$, where x represents the ID of a group and l is the number of groups. We denote the bits in the group as $B : B_1, B_2, ..., B_y, ..., B_p$, where y represents the ID of a bit and p is the number of bits. The number represents the ID of the group.

$$G_x = \{B_1, B_2, ..., B_y, ..., B_p\} \tag{3.5}$$

where $B_y = 0$ *or* 1.

The final output of the correlation extraction stage is a set of groups (*i.e.,* unique sensor state sets). Note that a single sensor can belong to multiple sensor state sets. Also, a sensor state set may be a subset or superset of another sensor state set.

Our correlation extraction method does not require any labeled training because we do not analyze the correlation of the sensor and the activities. All previous methods performed activity recognition have requested users to perform a list of activities several times and annotate the start and end time of each activities. Furthermore, some previous work had users provide the location and hierarchy information of the sensors, which were used for clustering the sensors. However, the correlation extraction method we devised runs automatically without any need for labeled training or supplementary information. Therefore, DICE achieves the usability requirement.

3.2.2 Transition Extraction

To extract the sensor state transition information, DICE computes three transition probabilities: group-to-group (G2G), group-to-actuator (G2A), and actuator-to-group (A2G) transition probabilities (Figure 3.4). A G2G is the transition probability of the sequence of groups obtained in the correlation extraction stage. For example, if group 2 always appears after group 1 during the precomputation phase, the transition probability of group 1 to group 2 is 100%. A G2A is the transition probability from a group to an actuator. For example, if actuator 1 is always activated after group 1, the transition probability of group 1 to actuator 1 is 100%. Similarly, an A2G is the transition probability from an actuator to a group. If the activation of actuator 1 always causes changes in the sensor readings to shift to group 2, the transition probability of actuator 1 to group 2 is 100%. We represent G2G, G2A, and A2G with three two-dimensional matrices used in the Markov Chain [26]. Each cell in a matrix holds the transition probability of the row's state to the column's state. Thus, the final outputs of the transition extraction stage are three matrices, a G2G, a G2A, and a A2G, that hold the transition probabilities. Since actuators affects sensor readings, G2G, G2A, and A2G can be substituted for actuator-to-actuator (A2A) transition probability. Therefore, we do not examine A2A to reduce calculation time.

3.3 Real-time Phase: Detection

In real-time, DICE looks for signs of deviation from the normal behaviors collected in the precomputation period to detect the presence of faults. DICE conducts a correlation check to detect correlation violations and a transition check to detect transition violations.

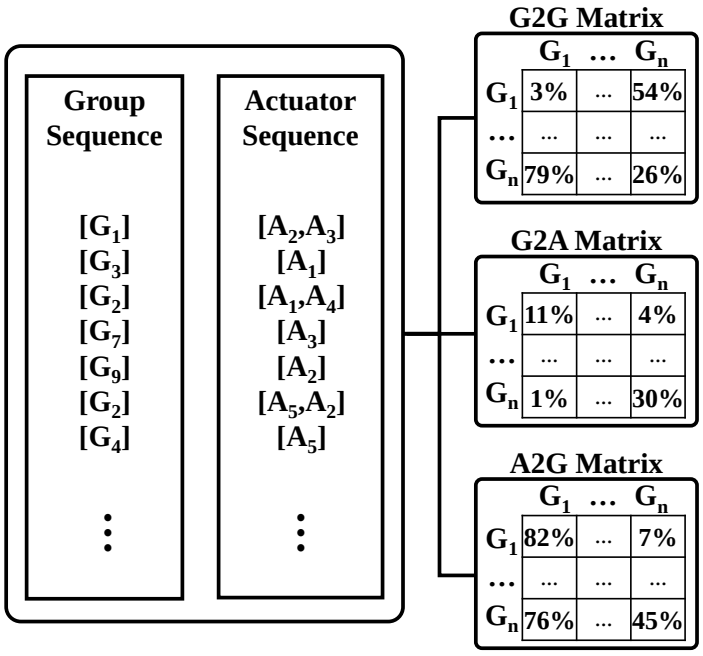


Figure 3.4: Transition Extraction

3.3.1 Correlation Check

DICE converts the raw sensor data into a sensor state set as described in Subsection 3.2.1. Then, DICE compares the obtained sensor state set with each of the groups obtained during the precomputation phase and makes candidate groups upto a certain distance (Figure 3.5). The distance between two different groups is measured by the difference in the bits. For example, if $G_1 = \{0, 0, 0, 0, 0\}$ and $G_2 = \{0, 0, 0, 1, 1\}$, G_1 and G_2 are apart by a distance of two. The distance threshold of the candidate groups is determined based on fault models considered. For example, if the system considers only a single fault for the collection period, we select the groups with less than two distance as the candidate groups. Among the candidate groups, a group that perfectly matches with an incoming real-time sensor state set (*i.e.,* has a distance of zero) is referred to as a **main group**. The remaining candidate groups are referred to as **probable groups**. When there exists no main group, this implies that a new sensor correlation combination that

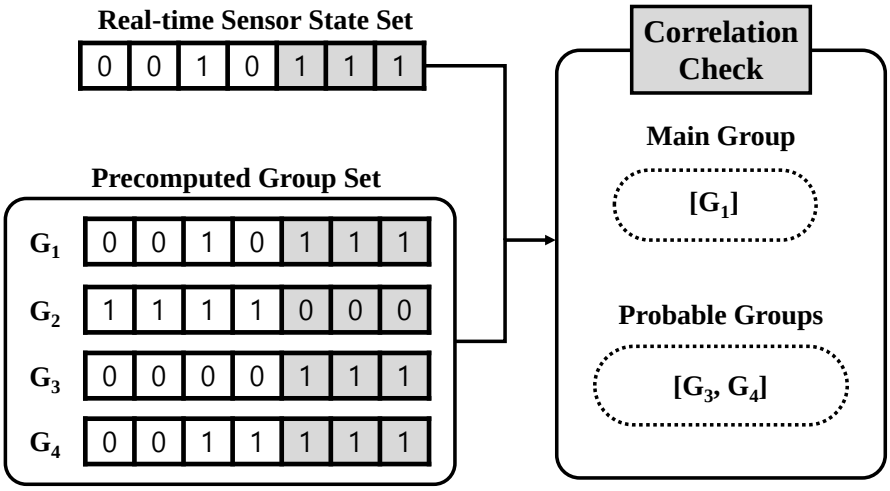


Figure 3.5: Correlation Check

has not been observed during the precomputation phase has occurred. Thus, DICE regards it as a correlation violation and proceeds to the identification step. If there exists a main group, DICE proceeds to the transition check.

3.3.2 Transition Check

In the transition check, DICE detects a transition violation based on G2G, G2A, and A2G obtained from transition extraction (Figure 3.6). The transition check is necessary because non-fail-stop faults often maintains the correlation information even after the fault. Therefore, the transition check captures the faults that were not caught during the correlation check. DICE examines the following three cases:

- **Case 1.** The transition probability of the previous sensor state set at time t_{i-1} and the current sensor state set at time t_i is zero in the G2G matrix.
- **Case 2.** The transition probability of the sensor state set at time t_{i-1} and the activated actuator at time t_i has a probability of zero in the G2A matrix
- **Case 3.** The transition probability of the activated actuator at time t_{i-1}

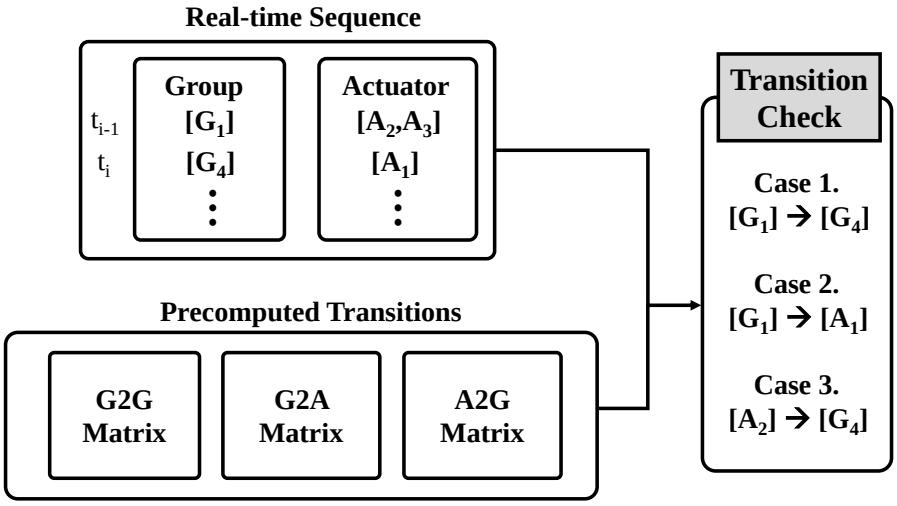


Figure 3.6: Transition Check

and the sensor state set at time t_i has a probability of zero in the A2G matrix

T_{i-1} denotes the $T_i - duration$. If the transition belongs to one of the three cases, DICE regards it as a transition violation. We skip the A2A transition probability by the same reason in the transition extraction.

3.4 Real-time Phase: Identification

When a violation is detected, DICE identifies the faulty devices by diagnosing the problematic context in the identification stage. When a correlation violation is detected due to the missing of main group, DICE compares the problematic real-time sensor state set with the list of probable groups. DICE examines the bits that differ and finds the probable faulty sensors (Figure 3.7). If the different bit, B_i, is a binary sensor, we derive the corresponding binary sensor, S_i as the probable faulty sensor. However, for a numeric sensor S_j, three bits constitute for a single numeric sensor. Thus, if one among the three bits is the different bit, we derive the corresponding S_j, as the probable faulty sensor.

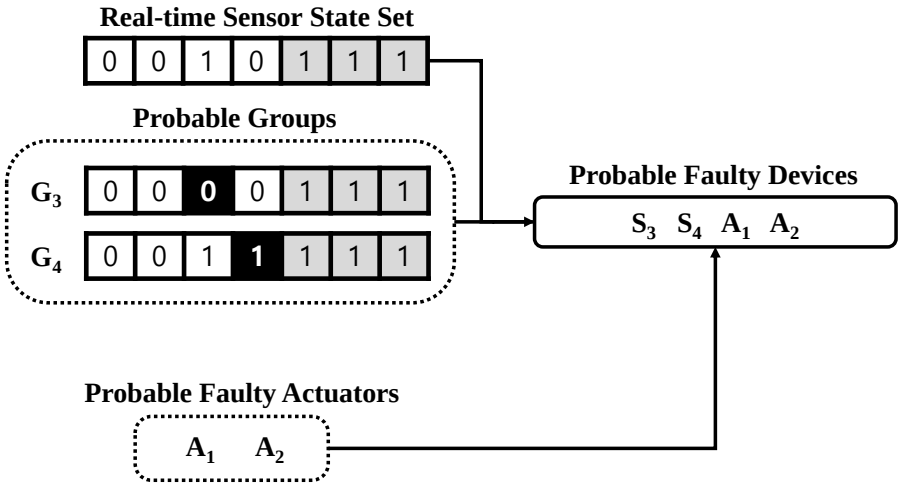


Figure 3.7: Identification

When there is only one probable group, DICE ends the identification step and notifies the user of the probable faulty devices. (We may add an additional attestation step for a verification purpose.) If there are two or more probable groups, DICE checks the transition probability from the previous group at time t_{i-1} to the current probable groups at time t_i. The groups that have no transition probability are removed from the probable group.

The similar process is applied when a transition violation is detected. When detecting a G2G violation during the transition check (case 1), DICE applies the same identification process as the correlation check case. When detecting a G2A or A2G violation during the transition check (case 2 and 3), DICE also selects probable faulty actuator DICE regards the present activated actuators (G2A) or the previously activated actuators (A2G) as faulty actuators and adds them to the probable faulty devices set.

We also applied an additional sophisticated technique when looking for probable faulty sensors. When there are multiple probable faulty sensors, DICE cannot instantly determine the faulty sensor. However, it is likely to appear in the probable faulty sensor set repeatedly, because a problematic sensor is likely to

generate faults continuously. Therefore, DICE repeats the identification process until the intersecting sensors of all the probable faulty sensors are below a threshold, *numThre* in this case. The *numThre* value is determined by the number of faults the system considers. When the system considers a single-fault case, *numThre* is set to 1. During the repetition, DICE skips the correlation and transition check but proceeds directly to the identification step because a fault has already been detected. Suppose probable faulty sensors at time t_i, t_{i+1}, and t_{i+2} are $\{S_1, S_2, S_3\}$, $\{S_1, S_2, S_4\}$, and $\{S_1, S_5, S_6\}$, respectively. At time t_i, the number of probable faulty sensors is three, so DICE repeats the identification step. At time t_{i+1}, the number of intersecting sensors of the two sets are two (S_1 and S_2), so DICE again repeats the step. At time t_{i+2}, S_1 is the only intersecting sensor in the three sets. Thus, DICE outputs S_1 as the faulty sensor and starts detecting faults from the top. Similar to sensor fault identification, DICE selects the faulty actuators that repeatedly appears in the probable faulty actuator set.

IV. Experimental Setup

This chapter describes the datasets used to validate DICE and the details of the Smart Home testbed we have implemented including the deployment setting and experiment design.

4.1 Data Acquisition

To show that our scheme is applicable to real-world Smart Home environments deployed with various sensor types, we used diverse datasets that have different sensor/actuator deployments and activity lists. We used a total of ten datasets; details of the datasets are shown in Table 4.1. The top five datasets are the publicly available third-party datasets; the bottom five datasets are the data collected from our Smart Home testbed.

4.1.1 Third-party Smart Home Data

We used five third-party datasets collected and distributed by Intelligent Systems Lab Amsterdam (ISLA) [28, 29] and Washington State University (WSU) [27, 30]. These datasets have also been used in state of the art sensor failure detection studies [5, 6, 8]. All five datasets have Smart Home sensor data of users performing daily human activities such as dish washing, opening the refrigerator, and sleeping. The datasets have different experiment setups and testbeds. The number and type of the activity list differs in each dataset. The sensor deployment setups are also diverse – the location and the type of deployed sensors are different in each dataset. We are convinced that we have objectively tested the performance of our system on diverse Smart Home settings.

Table 4.1: Datasets

	Hours	Binary sensors	Numeric sensors	Actuators	Activities
houseA [27]	576	14	0	0	16
houseB [27]	648	27	0	0	25
houseC [27]	480	23	0	0	27
twor [28]	1104	68	3	0	9
hh102 [28]	1488	33	79	0	30
D_houseA	600	6	31	8	16
D_houseB	650	6	31	8	14
D_houseC	500	6	31	8	18
D_twor	1200	6	31	8	9
D_hh102	1500	6	31	8	26

4.1.2 Our Smart Home Data

Although the five third-party datasets recreated the real context of use in the testbeds, the existing third-party datasets lacked realism. They either did not consider numeric sensors (**houseA, houseB, and houseC**) or deployed unreasonably large number of sensors (**twor** and **hh102**). Furthermore, existing studies claimed that their methods work in heterogeneous environments, but their dataset used only a few types of sensors. Among the two datasets that used numeric sensor data, **twor** only used three numeric sensors. Even though **hh102** deployed 79 numeric sensors in their testbed, they were all one of the three types – a battery sensor, light sensor, and temperature sensor. Therefore, to collect heterogeneous and realistic Smart Home data, we implemented our own Smart Home testbed. We deployed 31 numeric sensors, 6 binary sensors and 8 actuators in our Smart Home (Table 4.1). Figure 4.1 shows the floor plan and details of

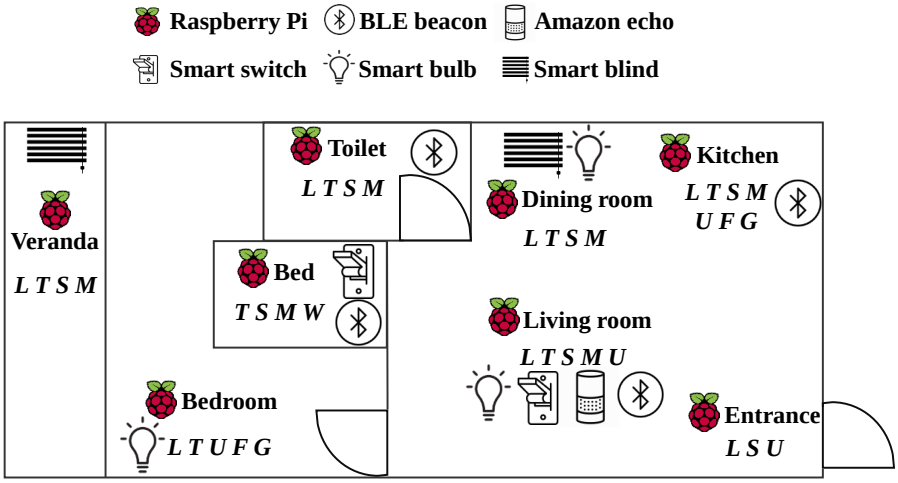


Figure 4.1: Floor Plan of the Smart Home Deployment (Each character stands for the following sensors. L: light, T: temperature, S: sound, M: motion, U: ultrasonic, F: flame, G: gas, W: weight)

the sensors and actuators deployed in our testbed.

Deployment Setting Our deployment setting demonstrates that our Smart Home data reflects real-world Smart Home environments. An IoT environment in general is composed of sensors, actuators, aggregators, a server and a communication channel [25]. We deployed 37 sensors, 8 actuators, and 8 aggregators with an Ubuntu server that runs DICE. Raspberri Pi boards were used as the aggregators. The server used MATLAB software to compute the correlation and transition extraction, as well as to run the DICE's detection and identification modules. We built our system on IoTivity [31], which communicates with the CoAP protocol. We explain the type of sensors and actuators deployed in our Smart Home in more detail.

Sensors. We used a total of nine sensor types in our testbed. Note that the more heterogeneous the sensors are, the more challenging it is to extract the correlation because different sensor types react differently to the same event. Thus, to collect data from a truly heterogeneous environment, we deployed diverse

types of sensors that include light, temperature, humidity, motion, ultrasonic, flame, gas, weight, and location sensors. The humidity sensors are not shown in the floor plan in Figure 4.1 because temperature and humidity sensors are contained in a single chip. However, the chip generates separate data for the temperature and humidity, so we regarded them as two different sensor types. We used beacons' Received Signal Strength Indicator (RSSI) values received in a smartphone to retrieve user's location information. We deployed four beacons, each in the kitchen, the bathroom, the bedroom, and the living room.

Actuators. We deployed the following actuators: three smart bulbs (Philips Hue), one smart speaker (Amazon Echo), two smart switches (WeMo Switch), and two smart blinds. The actuators were programmed to react to the connected sensor's values. Hue's light came on when a connected motion sensor detected a nearby motion. WeMo switches activated a fan or a humidifier based on the sensor readings of the connected temperature and humidity sensors. We installed motors to the blinds and programmed it to pull up when the light sensor value is low, and pull down otherwise.

Experiment Design We collected time-series sensor and actuator values from the deployed devices periodically while performing different human activities. We recruited five volunteers to join the experiment. All experiments were approved by the Institutional Review Board (IRB) of our university. To add objectivity to the data, we imitated the list and sequence of activities performed in ISLA and WSU datasets in our testbed. Each of the five subjects performed a series of activities that has the same sequence of activities in the third party dataset. We denoted each dataset collected the five subject from the testbed as **D_'dataset'**. The 'dataset' indicates the name of each simulated dataset.

We also covered a few exceptions. We removed the activities that could

not be reproduced in our testbed such as watering the baobab tree or playing a piano. Furthermore, for many datasets, there were less number of activities in the actual data than the number of activities stated in the official document. For this reason, the simulated dataset have less number of activities than the corresponding third-party dataset. In case when the third-party datasets had two toilets in their Smart Homes, we regarded the two toilets as one in our data.

Prior to the experiment, we developed a smartphone application that sequentially displays the name of the activity with its sequence equal to that of the third-party dataset. The application also received data from the beacons and recorded the beacons' RSSI values. The following is the detailed process of the experiment that the subjects performed. We requested the subjects to perform the activity indicated on the application. The subjects carried their smartphones and freely performed the activities without any designated place or time limit.

4.2 Faults Generation

Sensor faults are categorized into two classes: fail-stop and non-fail-stop faults. A fail-stop fault occurs when a device completely shuts down or its function ceases to generate any data. A non-fail-stop fault occurs when a device exhibits an abnormal behavior and generates incorrect data. Although the fail-stop faults occur in an unpredictable manner, [4] have categorized them by their characteristics and listed the most frequently observed non-fail-stop fault classes, which are outlier, stuck-at, high noise or variance, and spike. An outlier is a sensor fault where the sensor value at some point spatially or temporarily generates an anomalous value. A stuck-at-fault is when a series of output values is unaffected by the input and remains the same. A high-noise or variance fault is when sensor values have a noise or variance beyond the expected degree. A spike is when multiple data points are greater than the expected value, making

a shape like a spike. We referenced these four commonly occuring fault types and inserted faults in our collected data. The sensor type, fault type, and the insertion time were chosen randomly.

V. Evaluation

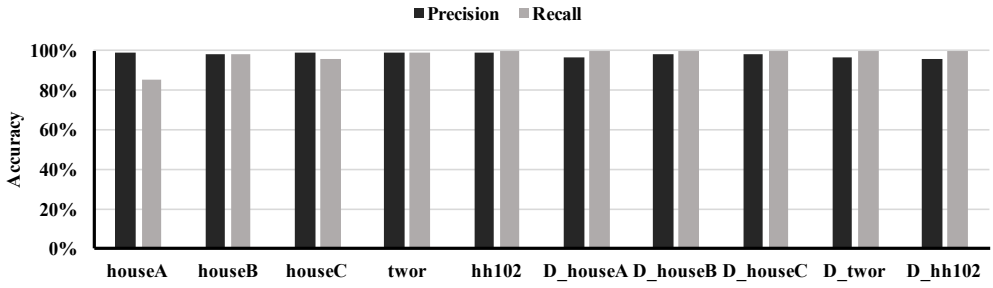
We evaluated DICE with the five third-party datasets and five datasets collected from our testbed to demonstrate the accuracy and promptness of our system. We used the first 300 hours in the dataset as the precomputation period, and used the rest of the data as the real-time data. We divided the real-time data into segments that have six hours of length. We then duplicated the segment and inserted one of the four faults in the segment mentioned in Chapter IV. The original segment was used as a faultless data to examine the precision (*i.e.,* false positive rate) of DICE. The duplicated segment was used as a faulty data to examine the recall (*i.e.,* false negative rate) of DICE. In total, we tested 100 faultless data, and 100 faulty data for each of the five datasets.

5.1 Accuracy

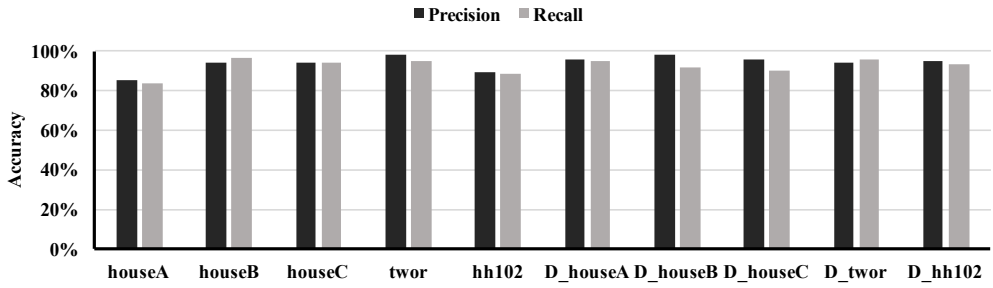
We measured the accuracy of our system when detecting and identifying device faults in ISLA, WSU, and the reproduced datasets (Figure 5.1).

5.1.1 Detection Accuracy

Fig. 5.1a shows the detection accuracy of each dataset, which measures how much DICE successfully detects the presence of faults. We used false negative and false positive rate to quantify the detection accuracy. False negative rate is a measure of how much faulty segment DICE failed to detect. False positive rate is a measure of how much faultless segment DICE incorrectly detected as faulty. DICE achieved an average precision of 98.20% and an recall rate of 97.91% for the ten datasets. The precision of the ten datasets were exceed 96%, and the recall of the five datasets collected in our Smart Home testbed were exceed 99%.



(a) Detection



(b) Identification

Figure 5.1: Detection and Identification Accuracy of the Ten Datasets

Therefore, we are convinced that DICE is highly accurate in detecting sensor faults in Smart Homes. We explain why some datasets show higher or lower detection accuracy than others in Subsection 5.4.

5.1.2 Identification Accuracy

Fig. 5.1b shows the identification accuracy of each dataset, which measures how much DICE successfully identifies the sensor that has generated faults. We used precision and recall to quantify the identification accuracy. Precision is the percentage of the actual faulty sensors among the identified sensors. Recall is the percentage of the identified faulty sensors among the actual faulty sensors. The evaluation results show that DICE achieved an average of 94.9% precision and 92.5% recall. Therefore, we conclude that DICE identifies the problematic sensors with high accuracy. **houseA** relatively showed lower precision and recall

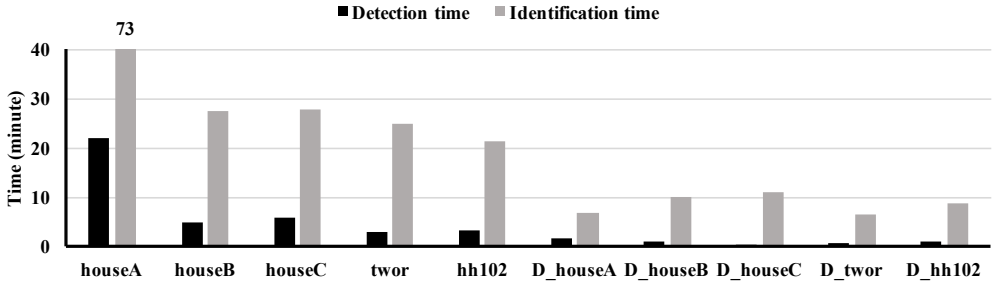


Figure 5.2: Detection and Identification Time (Detection time: the time to detect a fault, Identification time: the faulty sensor)

than other datasets; the reason is analysed in Subsection 5.4. The identification accuracy was lower than the detection accuracy in general. The reason is simply because detection only examines whether DICE detects the presence of faults, while identification examines whether the identified sensor matches the actual faulty sensor.

5.1.3 Actuator Faults

We also evaluated the accuracy of DICE on actuator faults. Since datasets collected from our testbed was the only available dataset that contained the actuator data, we evaluated the actuator faults from the **D_'dataset'**. DICE identified the problematic actuators with 92.5% precision and 94.9% recall on average. We detect and identify actuator faults as well with high accuracy.

5.2 Detection and Identification Time

We measured the average detection time and identification time of DICE in each dataset (Fig. 5.2). Detection time is the time DICE takes to detect the presence of faults since the occurrence of the fault. Identification time is the time DICE takes to identify the faulty sensor since the occurrence of the fault. In general, the correlation degree and the detection/identification time

Table 5.1: Detection Time of the Correlation Check and Transition Check

	Correlation check (mins)	Transition check (mins)
houseA	10.5	29.0
houseB	2.8	5.3
houseC	3.4	9.9

were proportional. Except for **houseA**, our system detected faults of the nine other datasets in 10 minutes and identified them within 30 minutes at most. For **houseA**, DICE took 21.88 minutes and 72.82 minutes on average to detect and identify the faulty sensor, respectively. Nevertheless, the slowest detection time of DICE was still much faster than the fastest reported average detection time of prior art which was 12 hours. Therefore, we are convinced that DICE identifies the faulty sensors promptly enough in real-time. We explain why the different datasets show varying detection and identification time in Subsection 5.4.

We also compared the detection time of the correlation check and transition check in **houseA**, **houseB**, and **houseC** (Table 5.1). The fault detected by the transition check was detected approximately three times slower than the fault detected by the correlation check. This is because the correlation violation is detected almost instantly, while the transition check detects a fault after a contrasting transition occurs. In general, a sensor state set retained its value for several rounds (*i.e.,* minutes) of time, which did not cause any drastic change in the transition. As a result, DICE detects a fault after the transition violation which took much more time than the correlation check.

Table 5.2: Correlation Degree and the Number of Sensors of the Datasets

	houseA	houseB	houseC	twor	hh102	DICE
Correlation degree	1.4	2.9	4.6	7.2	3.8	10.6
Number of Sensors	14	27	23	71	112	37

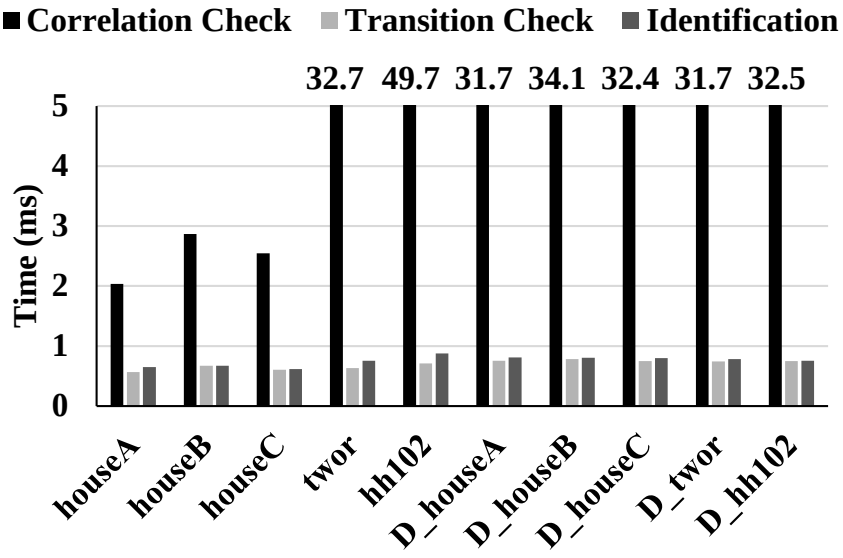


Figure 5.3: Computation Time

5.3 Computation Time

We measured the computation time of the correlation check, the transition check, and identification from the ten datasets to verify the feasibility (Figure 5.3). DICE spent shorter correlation check time in **houseA**, **houseB**, and **houseC** than the time in the other datasets. In detail, the correlation check time was most influenced by obtaining probable groups and the others took similar time in all datasets. The number of sensors affected the time to obtain probable

groups because the distance in Figure 3.5 was calculated for all probable groups. From the 37 sensors in our Smart Home testbed, the number of bits converted by the numeric sensors are over 100. This bit makes the correlation check time much longer. Nevertheless, the maximum computation time per one sensor state set in real-time was below 50 ms for all datasets, it is reasonable. The time of the transition check and identification was negligible and similar among the datasets due to simple probability comparisons with the transition matrix and bit value comparisons, respectively.

5.4 Correlation Degree

We calculated the correlation degree, which is an indicator of how much correlation exists among the sensors. The more the sensors there are that react together, the higher the correlation degree. Thus, to quantify the degree of correlation, we calculated the average number of activated sensors per group (*i.e.,* unique sensor sets). Table 5.2 is the summary of the correlation degree of each dataset. **houseA** had the lowest correlation degree, which was 1.4. In other words, for every unique sensor state set, one or two sensors have been activated simultaneously on average. The **DICE** datasets for a real-world Smart Home had the highest correlation degree of 10.6, and **twor** and **hh102** had more number of deployed sensors. This shows that the number of sensors and the correlation degree are not directly proportional. Instead, the accuracy and the detection/identification time was dependent on the correlation degree of the datasets. Datasets with higher correlation degree achieved better detection and identification accuracy. In the same way, our system detected and identified the faulty sensors of the **DICE** datasets much faster than those of the other datasets with lower correlation degree. In **houseA**, which had the lowest correlation degree, we had lower accuracy and slower detection/identification time than other

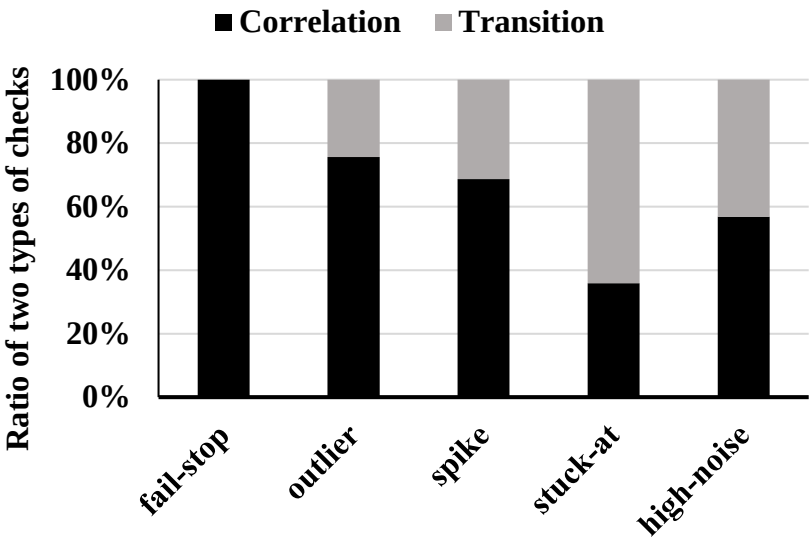


Figure 5.4: Ratio of Detection by Correlation Check and by Transition Check based on Fault Types

datasets. Nevertheless, the accuracy of detection and identification in **houseA** is acceptable.

5.5 Ratio of Detected Faults

We analyzed the ratio of faults detected in the correlation check and transition check based on fault types (Fig. 5.4). All fail-stop faults were detected during the correlation check, while most stuck-at faults were detected in the transition check. The reason is because fail-stop faults easily altered the correlation among sensors; thus, they were easily discoverable by the correlation check alone. However, the correlation relationship was maintained even with the presence of stuck-at faults that continuously reported the same values after the fault. DICE detects them with the transition check that captured the abnormal behaviors of such sensors. We are convinced that simple fail-stop faults may be detected by techniques proposed in prior art, but transition check is required to detect non-fail-stop faults of various types.

VI. Discussion

Multi-user cases. We considered a Smart Home with one or two occupants in our experiments. Note that all five heterogeneous sensor failure detection solutions also targeted a single or a two-occupant Smart Home. However, DICE works for Smart Homes with multiple residents when the residents are present during the precomputation phase. In such a case, the number of unique sensor state sets may grow exponentially with the increasing number of residents due to the increase in the possible combination sets.

For example, the sensors in the kitchen show a high correlation. In a single-resident case, other sensors are not likely to react to the user; thus, only the kitchen sensors are likely to be marked as activated in the sensor state set. However, in a multiple-resident case, other sensors in the bathroom or bedroom may react to a different user simultaneously, which increases the possibility of different sensor combinations. This may decrease the accuracy or increase the computation cost and time of DICE.

To resolve the problem, a user may group the sensors that are spatially closely located and connect each group to DICE individually to restrain the growing number of combinations. For example, the sensors in the kitchen can be grouped together and run DICE separately from the sensors in the bathroom or bedroom. We defer the multi-user case experiments for future work.

Multi-fault cases. We inserted one sensor fault at a time in our experiment, but multiple faults may occur simultaneously in Smart Homes. Although such a case is much less likely to occur, we randomly selected one to three faulty sensors to generate faults simultaneously to examine the result. In the multi-fault case, we set the value of *numThre* (in Chapter III) to 3. The average precision and

recall for identifying multiple faulty sensors were 79.5% and 63.3%, respectively, which is within a reasonable range.

Impact of different parameters. The precomputation period affected the accuracy of DICE. When the precomputation period was too short, DICE could not extract enough context of sensors and actuators that represent their normal behaviors. When we used the first 150 hours of the datasets as the precomputation data, the precision of the identification decreased by 10%. Thus, the longer the precomputation period, the higher the precision. The size of the segment also affected the accuracy. When we reduced the size of the segment into three hours, the recall of the identification decreased by 6%. The reason is because faults which maintained the correlation did not make any illegal transitions within three hours. Thus, the longer the segment, the higher the recall.

The duration of the sensor state set was also an important factor. When the duration was too short, DICE could not accurately group the correlated sensors due to the time difference of the sensors' reaction. When the duration was too long, DICE grouped the uncorrelated sensors that reacted within the duration together. The optimal duration we empirically found was one minute.

Feasibility. The computation complexity of DICE strongly depends on the number of sensors. A Smart Home with more sensors suffers from heavier overhead than a Smart Home with fewer sensors (Figure 5.3). Nevertheless, our experimental results include datasets that have up to 112 sensors (**hh102**). The computation time to process one minute long data for **hh102** was 50 ms, which is sufficient to meet real-time requirement.

We assumed that a typical Smart Home has a centralized gateway sensor with our system. Although our system only requires bit and matrix comparisons, the computation time of the system that runs in the gateway may take longer time due to limited computation ability. To enhance the performance, the system

can be installed at the Cloud server for fast computation. Almost, if not all, IoT platforms have a Cloud server that manages devices and enable remote access and control. Therefore, we believe that running our system at a Cloud server is not a strong assumption and is a reasonable solution.

Weight of devices. DICE considers all the devices to have equal importance and likelihood of failure. However, failures of some devices such as gas sensors and flame sensors may be more critical than others. Moreover, devices with higher computational power may be less likely to fail than lightweight devices. To consider these implications, we can impose additional weights in the computations – criticality-weight to indicate devices that require early identification, and failure-weight to indicate devices that are more likely to fail. During faulty device identification process, when a device in the probable faulty device list has a high criticality and failure-weight, DICE can fire the alarm of the device even if the number of devices in the list exceeds *numThre*. Higher weight of a sensor enables early detection, but at the same time, increase the false positive rate. Thus, it is important to set an appropriate weight on the sensors. However, it is difficult to devise an automatic method of finding an appropriate weight that can be applied to all devices in general. This is because the weight depends on the type of the device or the intent of the user. However, we defer the issue for our future work.

Expand to security. Correlation analysis helps IoT systems to find malicious IoT devices [9,32]. We also tested attacks against our Smart Home testbed and demonstrated that our system detected the attacks. We considered two cases of attacks. First, we attacked the temperature sensor higher to turn the fan in the empty home, causing economic problems. Second, we attacked the light sensor to raise Smart blind while a user is sleeping in a bed. Our system successfully detects the attack and identify the attacked sensor in both two cases.

VII. Conclusion and Future Work

A system that detects and identifies faulty devices is imperative in Smart Homes to provide reliable services to users. However, prior art has been inapplicable to real-world Smart Homes because of their lack of usability, generality, feasibility, and promptness. In this research, we proposed DICE, a future-oriented, context-based method to detect faulty IoT devices. In the precomputation phase, we extracted the context information based on the correlation among sensors and the transition of the sensor and actuator states without annotating the activities or requesting supplementary information. In the real-time phase, we examined if real-time sensor and actuator data violate the precomputed context and detected faulty sensors. We then compared the problematic context with the most probable context to identify the faulty sensor. Based on this organization, we tested the system on both the publicly available datasets and datasets collected from our own Smart Home testbed. DICE identified faulty devices successfully with an average of 94.9% precision and 92.5% recall. Our system took an average 3 minutes to detect faults and average 28 minutes to identify faulty devices. In optimal conditions, the time was reduced down to 7 minutes. DICE promptly detects and identifies sensor faults with high accuracy, contributing to building a highly reliable Smart Home.

For future work, we plan to expand our work to the security area. DICE has detected any sensors showing uncorrelated activity and modeled sensors as binary active sensors. Therefore, DICE can detect malicious numeric sensors in addition to binary sensors (in Chapter VI), but cannot detect passive malicious sensors or information leaking sensors at this moment. We are currently working on this part.

요 약 문

IoT의 적용 범위가 넓어지면서, IoT 장치의 고장을 안전하고 빠르게 탐지하는 기법은 IoT 환경에서 필수적이다. 하지만, IoT 장치들은 낮은 계산능력이 가져 높은 수준의 보안 기법을 적용하기가 어렵다. 따라서 본 논문에서는 컨텍스트 추출을 통해 자동으로 결함이 있는 IoT 장치를 탐지하고 식별하는 기법 (Detection and Identification solution based on the context extraction), DICE를 제시한다. 시스템은 사전 계산 단계 (Precomputation phase)와 실시간 검사 단계 (Real-time phase), 총 두 단계로 작동한다. 사전 계산 단계에서 시스템은 센서 상관관계 및 센서들의 상태 (sensor state) 간의 전이 확률 (transition probability)를 사전 계산한다. 실시간 단계에서 시스템은 실시간으로 바뀌는 센서 값을 분석하여, 센서 상관관계 및 전이 위반을 찾아 고장을 감지 (detection) 하고 식별 (identification) 한다. 고장을 감지할 때, DICE는 사전 계산 단계 시 얻은 결과값을 바탕으로, 센서 데이터를 분석하여 상관관계에 위반되는 IoT 디바이스를 발견하고, 비정상적인 시퀀스의 존재를 찾는다. 그런 다음 시스템은 문제가 있는 컨텍스트를 가능성 있는 컨텍스트와 비교하여 오류가 있는 장치를 식별한다. 우리는 DICE가 다양한 결함 유형 및 여러 데이터셋에 대한 평가를 통해 결함이 있는 IoT 장치를 정확하고 신속하게 식별한다는 것을 입증하였다.

References

[1] Timothy W Hnat, Vijay Srinivasan, Jiakang Lu, Tamim I Sookoor, Raymond Dawson, John Stankovic, and Kamin Whitehouse. The hitchhiker's guide to successful residential sensing deployments. In *ACM Conference on Embedded Networked Sensor Systems*, 2011.

[2] T Kavitha and D Sridharan. Security vulnerabilities in wireless sensor networks: A survey. *Journal of Information Assurance and Security*, 5(1):31–44, 2010.

[3] G. Padmavathi and D. Shanmugapriya. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *International Journal of Computer Science and Information Security*, 4(1&2), 2009.

[4] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *ACM Transactions on Sensor Networks (TOSN)*, 5(3):25, 2009.

[5] Krasimira Kapitanova, Enamul Hoque, John A Stankovic, Kamin Whitehouse, and Sang H Son. Being smart about failures: assessing repairs in smart homes. In *ACM Conference on Ubiquitous Computing*, 2012.

[6] Palanivel A Kodeswaran, Ravi Kokku, Sayandeep Sen, and Mudhakar Srivatsa. Idea: A system for efficient failure management in smart iot environments. In *ACM International Conference on Mobile Systems, Applications, and Services*, 2016.

[7] Sirajum Munir and John A Stankovic. Failuresense: Detecting sensor failure using electrical appliances in the home. In *IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2014.

[8] Juan Ye, Graeme Stevenson, and Simon Dobson. Detecting abnormal events on binary sensors in smart home environments. *Pervasive and Mobile Computing*, 33:32–49, 2016.

[9] Amit Kumar Sikder, Hidayet Aksu, and A Selcuk Uluagac. 6thsense: A context-aware sensor-based attack detector for smart devices. In *USENIX Security Symposium*, 2017.

[10] Juan Ye, Simon Dobson, and Susan McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, 8(1):36–66, 2012.

[11] Junkil Park, Radoslav Ivanov, James Weimer, Miroslav Pajic, and Insup Lee. Sensor attack detection in the presence of transient faults. In *ACM/IEEE International Conference on Cyber-Physical Systems*, 2015.

[12] Stanislav Rost and Hari Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In *IEEE Communications on Sensor and Ad Hoc Communications and Networks (SECon)*, 2006.

[13] Bor-Rong Chen, Geoffrey Peterson, Geoff Mainland, and Matt Welsh. Livenet: Using passive monitoring to reconstruct sensor network dynamics. In *International Conference on Distributed Computing in Sensor Systems*, 2008.

[14] Ravindra Navanath Duche and Nisha P Sarwade. Sensor node failure detection based on round trip delay and paths in WSNs. *IEEE Sensors Journal*, 14(2):455–463, 2014.

[15] Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. In *International Conference on Embedded Networked Sensor Systems*, 2005.

[16] Ioannis Ch Paschalidis and Yin Chen. Statistical anomaly detection with sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 7(2):17, 2010.

[17] Idris M Atakli, Hongbing Hu, Yu Chen, Wei Shinn Ku, and Zhou Su. Malicious node detection in wireless sensor networks using weighted trust evaluation. In *Spring Simulation Multiconference (SpringSim)*, 2008.

[18] Min Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *IEEE INFOCOM*, 2005.

[19] Jinran Chen, Shubha Kher, and Arun Somani. Distributed fault detection of wireless sensor networks. In *Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, 2006.

[20] Abhishek B Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Transactions on Sensor Networks (TOSN)*, 6(3):23, 2010.

[21] Lei Fang and Simon Dobson. Unifying sensor fault detection with energy conservation. In *International Workshop on Self-Organizing Systems*, 2013.

[22] Saurabh Ganeriwal, Laura K Balzano, and Mani B Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(3):15, 2008.

[23] Shuo Guo, Ziguo Zhong, and Tian He. Find: faulty node detection for wireless sensor networks. In *ACM Conference on Embedded Networked Sensor Systems*, 2009.

[24] David J Hill, Barbara S Minsker, and Eyal Amir. Real-time bayesian anomaly detection for environmental sensor data. In *Congress-International Association for Hydraulic Research*, volume 32, 2007.

[25] Jeffrey Voas. Networks of 'Things'. *National Institute of Standards and Technology (NIST) Special Publication*, 800-183, 2016.

[26] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[27] WSU CASAS Datasets. `http://ailab.wsu.edu/casas/datasets.html`. Accessed: 2017-09-30.

[28] ISLA Datasets. `https://sites.google.com/site/tim0306/datasets`. Accessed: 2017-09-30.

[29] Tim LM van Kasteren, Gwenn Englebienne, and Ben JA Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity Recognition in Pervasive Intelligent Environments*, pages 165–186. 2011.

[30] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas: A smart home in a box. *IEEE Computer*, 46(7):62–69, 2013.

[31] IoTivity Website. `https://www.iotivity.org/`. Accessed: 2017-09-30.

[32] Tamara Denning, Tadayoshi Kohno, and Henry M Levy. Computer security and the modern home. *Communications of the ACM*, 56(1):94–103, 2013.

Acknowledgements

석사 기간 동안 때로는 엄하고, 때로는 부드러운 모습으로 지도해 주신 김종 교수님께 감사드립니다. 교수님께서 변함없는 열정으로 아낌없이 조언해주신 덕분에 2년 동안의 연구를 잘 마무리할 수 있었습니다. 또한 부족한 저에게 연구뿐만 아니라 여러 방면에서 많은 도움을 준 연구실 동료들에게도 감사의 말씀을 드립니다. 저 혼자라면 할 수 없었던 것들을 연구실 선후배님들의 도움을 받아 해결할 수 있었습니다. 마지막으로 언제나 든든한 버팀목이 되어주고 힘이 되어준 가족에게 이 논문을 빌어 감사하고 사랑하는 마음을 전합니다. 자랑스러운 딸이자 누나가 되도록 더욱더 노력하는 사람이 되겠습니다.

Curriculum Vitae

Name : Jiwon Choi

Education

2012. 3. – 2016. 2. Department of Computer Science and Engineering, Pohang University of Science and Technology (B.S.)

2016. 3. – 2018. 2. Department of Computer Science and Engineering, Pohang University of Science and Technology (M.S.)


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교