Preemptive Anomaly Prediction in IoT Components

Alhassan Boner Diallo, Hiroyuki Nakagawa and Tatsuhiro Tsuchiya

Osaka University, Osaka, Japan

Abstract

The Internet-of-Things (IoT) has become a very promising and fruitful area of research. The rapid development of IoT is revolutionizing our daily utilization of technology in every way. The IoT paradigm is that the devices making up an IoT system have resource constraints such as storage, computing and energy consumption. That paradigm makes possible a flexible and pervasive communication between devices that are bound to low resources. These constraints may create a state where there is anomaly occurrence on the component level that may impact the whole system. Some innovative techniques have been proposed to quantify the reliability of these devices for the aforementioned constraints. However, there is a gap between the quantification of the component reliability and the predictive and preemptive maintenance of these components. In this study, we propose an approach combining reliability quantification and reinforcement learning to build a mechanism that can achieve a predictive maintenance for the components of an IoT system such as devices and links. In the approach, a component-level mechanism is built to synthesize the reliability data, and to determine the probability of anomaly occurrence for each component. The approach is being applied to a self-adaptive IoT system for smart environment monitoring named DeltaIoT.

Keywords

self-adaptive systems, IoT, preliability, reinforcement learning, q-learning

1. Introduction

Recently, the Internet of Things (IoT) has been one of the fastest growing fields in the computing domain. Its paradigm has been applied to many critical applications such as early warning systems for earthquake or tsunami, smart home security, traffic management, healthcare, and education systems, etc. Despite a rapid development and improvement in the IoT research area, many challenges remain. The challenges faced in IoT are related mainly to the following properties: scalability, availability, reliability, interoperability, security, mobility, performance, etc.

The IoT infrastructure is made up of low resource devices, meaning that they have low storage and low computing power compared to other devices within the computing domain. This is the result of the desire to accommodate the energy consumption as most of the component rely on battery to power them up[1][2]. Nowadays, the IoT paradigm is applied to many mission-critical systems, such as factory management, personal body sensors in healthcare, surveillance systems in nuclear power plants. These areas of application require a failure free system; otherwise there will be disastrous consequences. We must be able to trust these systems in all conditions as they impact the way we make numerous decisions based on the data they collect and provide. The reliability of the IoT systems depends on the reliability of the components that make up the system. As the IoT devices are constrained by nature, there must be some mechanism in place to ensure their reliability at all time, in order to have accurate decision models based on the data provided by the lower layer of the IoT architecture.

IoT reliability is a critical domain of research that has seen a lot of important contributions over the years. Multiple ways of quantifying the reliability of IoT components have been proposed. However, there is a gap between that quantified reliability and its application in predictive maintenance. In other words, how can we predict an accurate maintenance date for IoT components, based on the reliability measurement? To achieve that, we must build first mechanisms that can synthesize the reliability information from anomalies to determine whether the system has become less reliable from that anomaly occurrence. The ability to reason about the quantified reliability of the IoT system is a valuable step towards achieving predictive maintenance. The idea here is to build a dynamic decision-making process that can collect reliability data in a periodic manner and try to estimate a future failure time.

Fundamentally, we can define reliability as the study of failures. The reliability of a system or a computing device is its quality over a certain period of time. To quantify the reliability of a system or computing device, we use standard metrics all related to time like Mean Time To Failure, Mean Time Between Failures, and Mean Time To Repair, etc. Quantifying reliability is essential to assessing the continued success in the operation of an information system or a computing device.

CASA: 4th Context-aware, Autonomous and Smart Architectures International Workshop, ECSA'21 15-17 September 2021
a-bonerdiallo@ist.osaka-u.ac.jp (A. B. Diallo); nakagawa@ist.osaka-u.ac.jp (H. Nakagawa); t-tutiya@ist.osaka-u.ac.jp (T. Tsuchiya)
0000-0001-5280-4113 (H. Nakagawa)

2. Background

Computing systems require a high degree of performance and availability, but above all, they must be reliable. The appropriate way of assessing the reliability of a computing system depends on the type and mission of the system. In their study, Xie et al. [3] addressed several key metrics for reliability quantification. Some of these key metrics are Mean Time To Failure (MTTF), Mean Time Between Failures (MTBF), failure rate. The MTTF metric quantifies the expected operating time of a system before the occurrence of a failure. The MTBF metric as the name indicates, quantifies the operating time between one failure occurrence to another. The failure rate function helps to quantify the failure of a system within a specified window of time. The maintainability metric quantifies the probability that a system can go back to operating normally after the occurrence of an anomaly or a failure. The availability metric quantifies the probability of the system being expected normally operating.

The methods and techniques to analyze the reliability of computing systems depend on the domains that make up the system. There are mainly four domains or level: system, hardware, software, and network. The assessment of the reliability at a system level is the result of the combined assessment of the hardware, software, and network levels. In the hardware domain, the reliability assessment is related to the decay of the quality over time of the physical components of the computing system. In the software domain, according to the study in [4], there is no concern over a physical decay of the quality over time. As for the network reliability, it may be subjected to a decrease of performance over time due to internal and external factors on the hardware and software that make up the network.

In the case of IoT systems, their reliability can be assessed by quantifying the reliability of the different layers of their architecture. In [5], the IoT functionalities are grouped into the sensing and actuation, the communication, and the end-user application and services. A basic architecture of an IoT system can be divided into three layers: a device layer, a network layer, and an application layer. The device layer is responsible for the sensing and actuation. At the device level, the reliability is constrained by the battery life, the low capacity of both the memory and the CPU which prevent them using complex encryption to protect the transmitted data [6]. The device reliability is further constrained by false reading events that are common for sensors, when they collect and transmit data erroneously after an undetected failure[7].

The network layer is responsible for the communication between the devices of the system. The application layer is responsible for the services and the interactions with the end-user applications. In most cases, its reliability depends on the reliability of the device layer and the network layer. For example the device layer collects and transmits anomalous data, which are sent through the network to the application layer. Beyond being able to reason about the fitness of our IoT devices, we must also be able to attest to the reliability of the network infrastructure that forms the backbone of IoT communication. There are two approaches of network reliability studies which are discussed in this section; studies for enhancing QoS in networks, and studies aimed at quantifying reliability metrics for networks. Some research has also been conducted to evaluate IoT reliability at a system level. These approaches are at a high level and do not capture the individual detail for reliability, such as which devices are responsible for failures, or which parts of the network are responsible for traffic problems.

3. Focus: Anomaly Prediction

In our study, we consider the types of anomalies according to where and how frequent they occur. Anomalies can occur on each layer of the architecture with different degree of frequency. The device layer and network layer of the architecture are where anomalies occur the most, whereas the application layer is less prone to anomalies. As for the occurrence frequency, we consider two main forms of occurrence in the IoT components: *cyclic anomalies* and *random anomalies*. The former type of anomalies are linked to the nature of the component itself. Each component has a starting time and an ending time. The probability of anomaly occurrence is very small when the reliability is quantified closed to the starting time. On the other hand, the probability is great when quantified towards the ending time. The latter type of anomalies, called random anomalies, stem from random external as well as internal factors, like noise, interference, etc.

Our approach combines reliability quantification and machine learning to solve the problem of predictive maintenance from the aforementioned anomalies. Reliability quantification is achieved using the metrics introduced in [3]. Even though the concept of component anomalies is mentioned throughout this paper, detecting anomalies is not the main focus of this study. In their review of IoT reliability and anomaly detection techniques, Moore et al. [8] noted that no study had explored the potential of synthesizing quantified reliability data. The study pointed out that the decrease of reliability of a smart home system has different consequence than a decrease of reliability of a power plant surveillance system. The decrease in reliability of the IoT system increases the probability of anomaly occurrence within the system. As stated in the background, each layer of the IoT architecture has its own way of assessing reliability. In this study, we cover mainly anomaly occurrence at the device

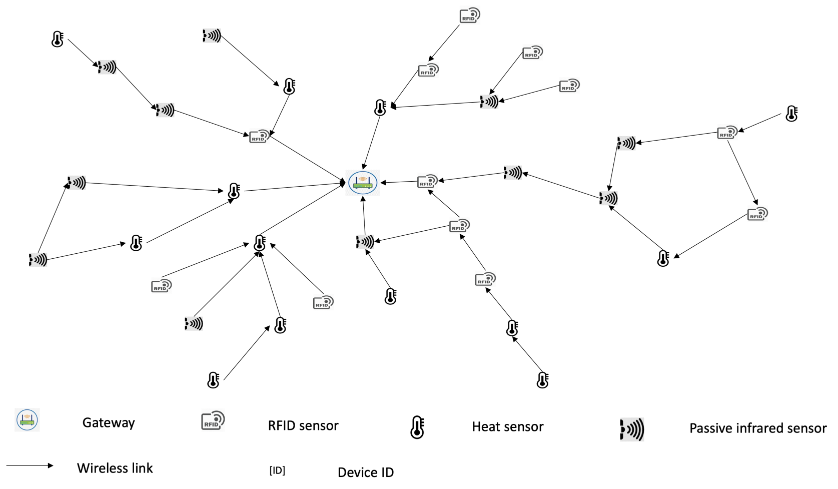

Figure 1: DeltaIoT network structure

layer and the network layer of the architecture. The main goal of our study is to enable the IoT system to achieve predictive maintenance, i.e., predict a probable failure time of one or more components and preemptively apply correction to the components, based on their quantified reliability. Based on this goal, we include in the study components where corrections can be applied after a failure or an anomaly. Therefore, some components of the device layer such as the battery, the memory and the CPU, are out of the scope of this paper. The reason is that they cannot be automatically maintained after a failure or an anomaly occurrence. These components, once the reliability has decreased or a failure has occurred, would require a system where a Human-in-the-loop is placed in for maintenance.

There are components of an IoT system that can be calibrated after the decrease of reliability or occurrence of an anomaly. Such components can be sensors at the device layer or links at the network layer. Therefore, our approach is applied to the sensor devices and the network links in order to achieve predictive maintenance. There are some consequences for undiagnosed anomalous data to be ignored within the different layers of the IoT architecture. Therefore, to decrease the vulnerability of the IoT-centered systems, there is a need to design lightweight solutions that are capable of handling the anomaly detection tasks without impacting the resource constrained systems.

4. Motivating Example: DeltaIoT

In this section, we describe the motivating example of our research which is a self-adaptive IoT system named DeltaIoT [9]. Self-adaptive systems are able to modify their behavior at runtime, in a response to a change in their operating environment, to achieve their goals. In this research, the study is not only about engineering reactive self-adaptivity, rather it is also about designing robust IoT system that are subjected to environmental changes. A typical IoT network system is composed of devices with different types of sensors and actuators, usually linked together wirelessly through the internet[2]. The concept of Internet-of-Things enables devices to operate with the constraints of energy consumption, low computing power and low storage power. The networks connecting the devices are also prone to congestion especially when there is a burst in demand, e. g., during an emergency situation for a system deployed to monitor large geographical areas to detect potential disasters as early as possible[10]. All these constraints make the engineering of dependable and reliable IoT systems more challenging. The next paragraph introduces an IoT system which is used in the case study of applying our approach.

The DeltaIoT system is a platform for smart environment monitoring. The system, introduced in [9], is a self-adaptive system, enabling it to react to environmental changes. The DeltaIoT system “enables researchers to evaluate and compare new methods, techniques and tools for self-adaptation in Internet of Things”. The DeltaIoT system has been built into two versions and they are deployed at the campus of KU Leuven University. The two versions differ in the number of devices present in each network and the geographical deployment of each version of the system. DeltaIoT system is described in Figure 1. DeltaIoT has a multihop communication system in cycles of 570 seconds. The system experiences external and internal stimulations that causes it to change its behavior to achieve its goals. There are two main causes for adaptation. The first cause for adaptation is an interference in the network causing the links to experience delay or packet loss. The second cause for adaptation is the fluctuating load of messages. This results in some or all links to be clogged creating delay and packet loss. There are three quality requirements the system must fulfil. The first quality requirement is about the average packet loss over 12 hours, which should not exceed 10% of the overall messages sent through the links. The second quality requirement concerns the average latency over 12 hours which should not exceed 5% of the cycle time. The third quality requirement concerns the average energy consumption over 12 hours. It has to be minimized during that period.

One of the main mission of the Internet of Things systems is to collect and communicate data about the environment or the people around which they are deployed. DeltaIoT, like many other IoT systems, alternates sensing and actuation during its operation. In many cases, the actuation is performed based on the results of the sensing. Therefore, anomalies during data sensing and during data communication may have a negative effect on the system performance or operation. Collecting anomalous data typically happens on the device level by the sensors. It can be caused by different reasons like noise or defect due to environmental factors. When this happens, the sensors can be calibrated again to perform with a great accuracy. Anomalies occurring on the links of the DeltaIoT system are related to the decrease in the QoS. The packet loss and the latency are some of the manifestations of these anomalies occurring in those links.

We have presented a mechanism for an efficient configuration space reduction [11]. The mechanism focused on the analysis after an anomaly has happened at a component level. In this paper, the main focus of the study is to forecast an anomaly before it happens. It is important to reduce the time between anomaly occurrence and detection. It is equally important to minimize the time from anomaly detection to correction. Moreover, precise anomaly understanding aids in constructing more precise probabilistic model of the system, which helps to find more reliable configuration of the system using probabilistic model checking [12]. Many anomaly detection techniques have been proposed for computing devices in general, each with its advantages and drawbacks. However, techniques for anomaly forecasting are few. In the Internet of Things domain, to the best of our knowledge, our study is the only one that makes use of reliability quantification and machine learning approach to predict anomaly occurrence. As explained in the approach, if the time of anomaly occurrence could be predicted, then corrective measures can be applied in order to prevent the anomaly from happening.

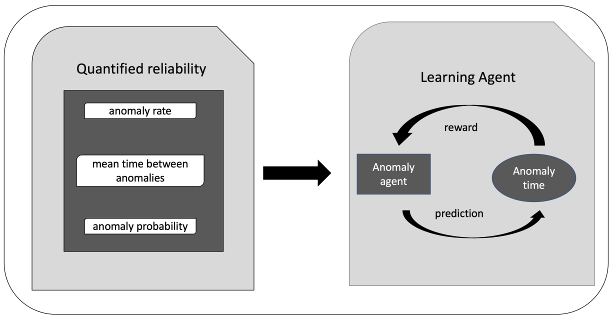

Figure 2: Overview of the component-level mechanism

5. Approach

In this section we describe in details our approach and its practical implementation. The goal of the approach is to determine a high probability failure time or an anomaly occurrence time in order to apply corrective measures. We build two mechanisms. The first mechanism is on the component level, that is the level of devices and links. It captures the behavior of each individual component. The reliability of each component is computed by this mechanism. The second mechanism is on the level of the MAPE feedback loop. The MAPE stands for Monitor, Analyzer, Planner and Executor. The feedback loop is used in autonomic computing to achieve self-adaptation in software systems[13]. The system-level mechanism is connected to the monitor component of the feedback loop.

The backbone of the component-level mechanism is an anomaly agent that is instantiated by each component of the IoT system. The quantified reliability is determined using mainly two metrics: mean time between anomalies, anomaly rate. The function of the anomaly agent is to predict an anomaly time, depending on the quantified reliability of the component. The anomaly agent has to predict an accurate anomaly time. It behaves according to the principles of reinforcement learning. It is rewarded for the accurate prediction of the anomaly time. Figure 2 illustrates the component-level mechanism of the approach. According to [14], “reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward”. The main motivation of using reinforcement learning is to record the different states of the system and their transitions [15][16]. The system has an optimal state in which the probability of each component’s reliability is high. The next state is an in-between state where the component’s reliability is just average. Lastly, the system has a critical state in which an anomaly has already occurred or is very likely to occur. Capturing these different states and reasoning about them can be helpful in discovering an optimal time for predictive maintenance. To implement the anomaly agent, we use an approach that relies on Time Difference Learning [17]. The agent is implemented according to a Q-Learning algorithm [18]. The approach is well suited for situations with great degree of random variables and uncertainty. In the next subsections, we explain the two mechanisms in detail.

5.1. Component-level mechanism

The network of most IoT systems is composed of several heterogeneous devices. These heterogeneous devices possess sometimes different characteristics that can hinder their interoperability. Therefore, when designing a mechanism for anomaly prediction, each individual component of the network must have a self-centered module that captures its unique characteristics. The component-level mechanism is illustrated in figure 2. The mechanism has two main parts. The first part is a reliability quantification algorithm, where the reliability of the module is quantified based on the previously mentioned metrics. The IoT system operates in an environment where the quality of its components deprecates over time. Some components can be calibrated back to normal like the sensors and the network links. However, the physical aspect of the system in most of the cases cannot be calibrated. Therefore, that aspect is out of the scope of this study. We track the component based on the three metrics: the mean time between anomalies (MTBA), the anomaly rate (AR) and the probability of anomaly (PA). First we determine the anomaly rate AR by determining the number of anomalies per cycle of time. It is calculated by dividing the number of anomalies over the cycle of time.

$$AR = \frac{Anomalies}{CycleTime} \tag{1}$$

The MTBA is the time the system or the component is operating normally before an anomaly occurrence. The MTBA is determined by the following formula

$$MTBA = \frac{1}{AR} \tag{2}$$

The probabilbity of anomaly occurrence PA, is determined using the MTBA is the following formula

$$PA = e^{((\frac{-1}{MTBA})*time)} \tag{3}$$

The second part of the component-level mechanism is a Q-Learning agent, where the agent learns the characteristics of the component, based on the quantified reliability and the overall environment of the component. The agent must learn to predict an anomaly time. Therefore, the actions to be taken by the agent are prediction actions related to an anomaly time. We formalize our problem as a Markov Decision Process or MDP. The component, which is the environment interacting with the agent, is modeled as a Markov Process. The Q-learning algorithm used to create the agent, is chosen because it is model-free, off-policy, and value-based algorithm.

The MDP describing the environment for the learning process, contains a tuple of four elements. The first element is a set of finite states S. the second element is a set of finite actions A. the number of states is function of the number of actions. The actions to be performed by the agent are, for each run, adding an integer value to the current time and to check whether the time corresponds to the anomaly time. The third element of the tuple is the reward R to be received after transitioning from state S to state S' as a result of performing an action. The fourth element of the tuple is the probability P related to the performed action.

The Q in Q-learning is a measure of the quality of a state-action combination. When an action is taken by a learning agent, the reward of that action along with the learning rate, the discount factor and the initial condition or previous value of Q, are used to determine the new value of Q for that state.

$$\begin{aligned} Q_t(s,a) = Q_{t-1}(s,a) + \alpha[r+ \\ \gamma * max_\alpha Q(s',a') - Q_{t-1}(s,a)] \end{aligned} \tag{4}$$

5.2. System-level mechanism

The mechanism is implemented on the monitor level of the MAPE feedback loop. The monitor part of the MAPE feedback loop observes the system and the operating environment with which the system is interacting, to check whether there are changes. We leverage this function of the monitor, and append the system-level mechanism on it. The mechanism performs two main tasks. The first task of the mechanism is to check the results from the component during each cycle performed by the IoT system. The second task is to aggregate the results of the from the components over all the cycles performed by the system.

5.3. Learning Process

In the component-level mechanism, our method first determines an accurate quantification of the metrics, that can give a snapshot of the quality of the component at each period of the system operating cycle. This is most required during the time of data sensing and data forwarding. The components of the IoT system, i.e., sensors and links, operate differently in the environment. We have described earlier the kind of anomalies that the components are subjected to. The sensors can have random anomalies like noise but also cyclic anomalies. The links of the network have external interferences or message clogging leading to anomalies. However, most of these anomalies are related to the decrease of accuracy of the device and decrease of power settings of the link. The approach determines the number of anomalies that are occurring during each cycle. Therefore, for each cycle we can observe a different anomaly rate. That observation helps to determine and update the mean time between anomalies during all the cycles. We determine the actions performed by the agent as adding time in seconds to the current time. The reason is that the current time is the time when the agent decides to make a prediction about an anomaly time. The agent decides to make a prediction after getting the anomaly probability for that period. The amount of time in seconds to add to the current time is function of the anomaly probability of that period. If the anomaly probability is high, the amount is small, and on the other hand, if it is low, either no time is added, or a big amount. After each prediction, the Q value of that state-action combination is updated according to the reward obtained.

6. Conclusion

In this research, we are investigating the possibility of preemptive forecasting of anomalies that occur at the device and network layers of an IoT architecture, by implementing an anomaly agent based on the Time Difference Learning method. In the next step, we plan to implement another anomaly agent based on the Monte Carlo method and evaluate the performance of these two agents.

References

[1] D. E. Kouicem, A. Bouabdallah, H. Lakhlef, Internet of things security: A top-down survey, Computer Networks 141 (2018) 199–221.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE communications surveys & tutorials 17 (2015) 2347–2376.

[3] M. Xie, Y.-S. Dai, K.-L. Poh, Computing system reliability: models and analysis, Springer Science & Business Media, 2004.

[4] A. Mavrogiorgou, A. Kiourtis, C. Symvoulidis, D. Kyriazis, Capturing the reliability of unknown devices in the iot world, in: 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, IEEE, 2018, pp. 62–69.

[5] A. Rayes, S. Salam, Internet of things from hype to reality, Springer (2017).

[6] F. A. Alaba, M. Othman, I. A. T. Hashem, F. Alotaibi, Internet of things security: A survey, Journal of Network and Computer Applications 88 (2017) 10–28.

[7] A. Karkouch, H. Mousannif, H. Al Moatassime, T. Noel, A model-driven architecture-based data quality management framework for the internet of things, in: 2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech), IEEE, 2016, pp. 252–259.

[8] S. J. Moore, C. D. Nugent, S. Zhang, I. Cleland, Iot reliability: a review leading to 5 key research directions, CCF Transactions on Pervasive Computing and Interaction (2020) 1–17.

[9] M. U. Iftikhar, G. S. Ramachandran, P. Bollansée, D. Weyns, D. Hughes, Deltaiot: A real world exemplar for self-adaptive internet of things (artifact), in: DARTS-Dagstuhl Artifacts Series, volume 3, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[10] S. Y. Shin, S. Nejati, M. Sabetzadeh, L. C. Briand, C. Arora, F. Zimmer, Dynamic adaptation of software-defined networks for iot systems: a search-based approach, in: Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2020, pp. 137–148.

[11] A. B. Diallo, H. Nakagawa, T. Tsuchiya, Adaptation space reduction using an explainable framework, in: Proc. of the IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC 2021), IEEE, 2021, pp. 1654–1661.

[12] H. Nakagawa, H. Toyama, T. Tsuchiya, Expression caching for runtime verification based on parameterized probabilistic models, The Journal of Systems and Software, Elsevier 156 (2019) 300–311.

[13] J. O. Kephart, D. M. Chess, The vision of autonomic computing, Computer 36 (2003) 41–50.

[14] J. Hu, H. Niu, J. Carrasco, B. Lennox, F. Arvin, Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning, IEEE Transactions on Vehicular Technology 69 (2020) 14413–14423.

[15] M. Wiering, M. Van Otterlo, Reinforcement learning, Adaptation, learning, and optimization 12 (2012).

[16] R. Riveret, Y. Gao, G. Governatori, A. Rotolo, J. Pitt, G. Sartor, A probabilistic argumentation framework for reinforcement learning agents, Autonomous Agents and Multi-Agent Systems 33 (2019) 216–274.

[17] R. S. Sutton, A. G. Barto, Temporal-difference learning, Reinforcement learning: an introduction (1998) 167–200.

[18] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (1992) 279–292.