A Model-based Approach for Self-healing IoT Systems
Position Paper

Franziska Kühn[1], Horst Hellbrück[1,2] and Stefan Fischer[1]
[1]*Institute of Telematics, University of Lübeck, Lübeck, Germany*
[2]*Center of Excellence CoSA, Lübeck University of Applied Sciences, Lübeck, Germany*

Keywords: Self-healing Systems, Model-based Approach, Internet of Things, Sensor Networks.

Abstract: IoT systems become more and more important in our daily life. They will perform monitoring and control tasks which are often safety-critical. Therefore, it is obviously important that IoT systems work reliably, i.e., fulfill their specification. Even if something unexpected happens, it is required that the system moves back into a correct state which we name self-healing. In this paper, we present our idea for a model-based approach for self-healing IoT systems. Based on a formal specification of the system's properties, we derive monitors which observe the system behavior and trigger healing actions when necessary. In IoT systems, the placement of such systems becomes important due to the increased unreliability of single devices. The paper outlines basic ideas where to place monitors and how to assign monitoring tasks to IoT devices.

1 INTRODUCTION

Already in 1990 Mark Weiser described the idea of ubiquitous computing (Weiser, 1999) where computers are replaced by smart things which coalesce with our environment. Today, the idea is becoming reality very quickly. In so called Internet of Things (IoT) environments, the environment, more specifically the things, are equipped with a huge number of small devices, sensors and actuators. In addition, (wireless) sensor networks are often part of IoT environments. In (Evans, 2011) it is estimated that there will exist 50 billion networked devices until 2025. It is expected that the smart things will soon become a natural part of our daily lives, taking over autonomously a lot of tasks that today have to be done by humans.

Such systems must have added value for humans to become accepted by people. They have, for instance, to fulfill tasks better than humans or undertake arduous or unpleasant tasks. For good acceptance it is important that the systems operate reliably. In addition, reliability is a key factor as faulty behavior can have severe consequences for both people and environment.

However, being reliable is difficult to achieve for large systems consisting of many heterogeneous, resource constrained (e.g. energy, power and memory), usually inexpensive and thus often unreliable components. The situation is even worse, because many such IoT systems will operate completely unsupervised and without any human (namely system administrator) intervention. As a result, reliability can neither be based on the reliability of single devices nor on manual repair of faulty behavior.

Therefore, it is essential that the systems have self-healing capabilities to improve their reliability. This means that the systems are able to autonomously detect and diagnose failures (and misbehavior) and autonomously choose and perform appropriate mitigation strategies. Still, self-healing systems pose various challenges (IBM, 2005; Ghosh et al., 2007; Psaier and Dustdar, 2010; Salehie and Tahvildari, 2009), especially the specific characteristics of IoT systems.

In this position paper, a model-based approach for self-healing IoT systems is introduced. The approach allows for transparently attaching the self-healing capabilities to (existing) systems. Thus, the self-healing capabilities must not be added in advance to the systems and existing ones need not to be adapted during runtime. A key aspect of the approach is a formal specification of the system properties that have to be satisfied during runtime (e.g. quality of service aspects). A formal specification facilitates the automatic synthesis of components which observe at runtime whether the systems satisfy or violate the properties. In case of a violation healing actions are triggered to bring the system back into a correct state.

The rest of the paper is organized as follows: In

Kühn, F., Hellbrück, H. and Fischer, S.
A Model-based Approach for Self-healing IoT Systems - Position Paper.
DOI: 10.5220/0006639401350140
In *Proceedings of the 7th International Conference on Sensor Networks (SENSORNETS 2018)*, pages 135-140
ISBN: 978-989-758-284-4

the next section related work is discussed. In Section 3 the model-based approach to add self-healing capabilities to IoT systems is introduced and the challenges are discussed. Afterwards in Section 4 a real-world testbed is presented. Finally, we give an outlook on future work and especially on how to proceed with the project in Section 5.

2 RELATED WORK

Adding self-healing capabilities to IoT systems has been studied before. One of the most frequently applied models to realize autonomous systems is the MAPE-K model (Kephart and Chess, 2003; Kephart, 2005). The model describes four phases in a loop: monitoring, analysis, planning and execution. The phases share a knowledge base, containing for example logs, symptoms and known faults

The MAPE-K model has already been adapted to realize self-healing systems in wireless sensor networks (WSN) (Portocarrero et al., 2014) and IoT environments. The focus of the approaches differ especially on the network protocol layer and thus the types of faults.

In (Gurgen et al., 2013) the MAPE-K model is adapted to realize self-healing capabilities for cyber-physical systems in smart buildings and cities. The approach combines the concept of service-oriented architectures and cloud computing. The focus is to build services for monitoring and processing data and the planning and execution of actions. End users can use the services to equip applications with self-healing capabilities. Anyway, the authors do not pursue a model-based approach, especially to generate the monitors and it may not be possible to transparently attach the self-healing components to (existing) applications.

In (Nguyen et al., 2015) the MAPE-K model is adapted to realize self-healing IoT systems. Nevertheless, the focus is to detect, classify and correct faulty data only. In addition, the approach uses a central component for the knowledge base which might be a bottleneck and single point of failure in such large IoT systems. Furthermore, the detection, classification and correction of failures is performed on the nodes itself. Due to the resource constrained nature of IoT devices this might not be possible in all cases.

In (Bourdenas and Sloman, 2010) the self-healing capabilities are targeted by the reconfiguration of the nodes in case of a failure. In (Fok et al., 2009) the application is moved on a surrogate node in case of a failure.

In (Angarita, 2015; Angarita et al., 2015; Angarita Arocha, 2015) self-healing capabilities are attached to transactional Web Services but are also limited to them.

Using Runtime Verification (RV) for monitoring in WSN has already been studied, e.g. in (Sokolsky et al., 2008; Herbert et al., 2007). The approaches allow for a formal specification of the system properties but not for the mitigation of faults. The idea to use RV techniques for monitoring in a self-healing system is scratched in (Fischer and Leucker, 2013).

In (Decker et al., 2014) a monitoring framework (using RV techniques) for interconnected medical devices based on web services is introduced. The framework allows to automatically generate monitors based on a formal specification. In (Kühn et al., 2017) the framework is adapted for monitoring interconnected medical cyber physical systems. In addition, it is sketched how further components for behavioral exception handling can be added towards a self-healing system. Nevertheless, the framework is not optimized for IoT systems (especially with regard to its resource constrained nature).

3 MODEL-BASED APPROACH

In this section we will introduce our new model-based approach which, based on a formal description of system properties, follows a step by step process from the formal requirements of the application to the implementation and placement of monitoring, diagnosis and mitigation components.

Instead of equipping the applications itself with self-healing capabilities we aim for transparently attaching the required components on arbitrary nodes or dedicated hardware. The advantage of this approach is especially that self-healing capabilities can be attached to existing and / or resource-constrained nodes (on extra hardware or other nodes) and the self-healing components do not crash if the node under scrutiny crashes.

To realize the self-healing capabilities we adapt the MAPE-K model and the self-healing model from our previous work described in (Kühn et al., 2017). Figure 1 shows the three processes in a loop of our self-healing system. It yields the three modules monitoring, diagnosis and mitigation for our self-healing model. In addition, we integrate an information base which may be (partially) shared between the processes.

The *monitoring* module comprises one or several monitors where each monitor is responsible for one safety or system requirement. The requirements have to be specified formally. For that it is valuable to first

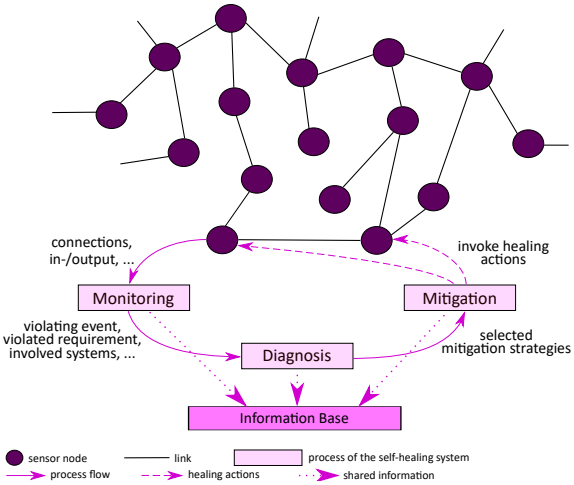

Figure 1: Process of a self-healing system.

create a well-defined failure model for components and connections in IoT systems, taking into account existing models, e.g. (Asim et al., 2010). In addition, without a failure model it becomes more or less impossible to realize reliable self-healing systems. The failure model can for example be stored in an *information base*.

Typical errors that are often investigated in IoT systems are: a component (IoT node) fails completely or a connection (link) between components breaks. However, in large IoT systems there are failure types that are more difficult to detect. Components might show temporarily unexpected behavior or connections might falsify data temporarily due to specific hardware characteristics or software errors. Be reminded that this kind of *misbehavior* is not expected to be malicious but happens due to coincidences. The level of reliability achieved by a self-healing system is always a relative rate and can be described in a formal way by quality of service (QoS) parameter. In contrast to pure data networks in IoT, there are more notions of QoS which include event detection/reporting probability, event classification error, detection delay, probability of missing a periodic report, approximation accuracy e.g, when nodes construct a temperature map, tracking accuracy e.g. difference between true and conjectured position of a mobile object. Especially the lifetime of an IoT system can be described by various states: first node failure, network half-life (how long until 50% of the IoT components died?), partitioning of a multi-hop IoT network, loss of coverage, failure of first event notification.

Our goal is to consider QoS parameter and various failure types to allow the definition of a variety of safety and system requirements the systems have to adhere to. For the specification appropriate formal specification mechanisms have to be provided to the users, such as, e.g., Linear-Temporal Time Logic (Pnueli, 1977) (LTL) which is a well-known specification language and comprehensible formalism. The logic enables to combine boolean and temporal operators allowing for describing behavioral dynamic constraints of IoT systems.

For monitoring distributed systems RV (Leucker and Schallhart, 2009) techniques have been recognized as a valuable solution which we consider as one approach to realize the monitoring component. RV facilitates the automatic synthesis of monitors based on a formal specification such as LTL. The monitors then observe at runtime whether the systems adhere to the specified requirements. For this purpose the monitors analyze at runtime the input and output of the components or the connections during runtime. In case of a failure, necessary information (e.g. violating event, violated requirement, involved IoT devices) is passed to the next module.

In the second module, named *diagnosis*, the extract of the monitoring process is combined and the causing failure is identified. This task is especially challenging. A formal description of the behavior of the diagnosis module might require new language concepts. Furthermore, the second module is responsible to choose appropriate mitigation actions. As a first basic solution the diagnosis could for example be realized using event-condition-action-rules.

The third module *mitigation* invokes the healing actions chosen by the previous module to repair the system which means to transfer the system into a correct state again. The healing actions comprise restart of hardware or software components, triggering actions, relocation of software components just to name a few. It is obvious that only actions can be realized that are provided by the components (on software as well as hardware level).

Common to all aforementioned modules is that they might be placed on resource-constrained nodes. Usually, IoT systems are large distributed systems with many constraints like limited computing power, small link bandwidth and limited energy resources. Especially wireless sensor networks often consist of resource-constrained nodes. The choice, number and the placement of the above described modules needs to be carefully considered. For that reason, it is necessary to implement the modules as resource-efficiently as possible which introduces a trade-off between different strategies consuming different resources. Thus, a formal description needs to include these constraints. Depending on the constraints and requirements the modules adapt by implementing different strategies.

Both the implementation of the modules itself and

the number and placement of the components needs to be considered carefully. Furthermore, the system needs the ability to relocate services dynamically and efficiently, especially if a node under scrutiny crashes or if requirements (e.g. QoS parameter) change during runtime. Particularly, for the number and placement the requirements and constraints have to be taken into account. In the following a few considerations for the number and placement are listed:

- For powerful nodes it might be valuable to execute several complex monitors whereas for small devices this is not feasible. Instead, several small monitors on different nodes or on dedicated hardware is a better choice.
- It is often not useful to put the monitoring component on the node it observes itself but rather to realize a mutual observation of several nodes.
- It has to be evaluated whether it is feasible to use a diagnosis component for several monitors and a mitigation component for several diagnosis components respectively, achieving a hierarchical placement.
- Likewise, a hierarchical arrangement of the monitors themselves have to be taken into account.
- Central components in large distributed systems often do not make sense. Strategies to share and synchronize for example the information base have to be considered.
- In contrast to a hierarchical placement, in some cases a single central component for all self-healing tasks might be advantageous.
- For safety-critical systems it might be useful to deploy redundant components.

A formal description needs to include these and other constraints and characteristics. Based on a formal description, algorithms for the calculation of the number and the automatic placement and relocation of the components have to be developed.

Besides the automatic deployment and placement of the components it is important to develop and implement a resource-saving protocol for "global" agreement of the placement of the self-healing components between the concerning nodes. For the development, amongst others, network problems, lost messages and temporarily unavailable nodes have to be taken into account. Protocols and approaches of other (research) fields, e.g. from mobile communications, have to be evaluated.

For the implementation web services or service-oriented architectures (SOA) tailored to IoT systems and combined with semantic description languages, e.g. (Ankolekar et al., 2002), provide an excellent choice, see (Glombitza, 2011; Kleine, 2016). Furthermore, the use of fog or dew computing might be a valuable approach to improve for example the performance or to reduce the complexity.

In summary, our approach is based on standardized flexible technologies to build large distributed systems like web services or SOA that we tailor to the needs of IoT systems combined with the automatic generation of modules by formal descriptions. A special challenge in our approach is an automatic deployment and relocation of the modules in large heterogeneous IoT systems.

4 A REAL-WORLD TESTBED

We strongly believe that it is very important to include both real-world experiments and simulations in the development process of IoT systems. In other words, one needs experimental environments such as WISEBED (Coulson et al., 2012) or SmartSantander (Sanchez et al., 2014). For our experiments, we borrow the sample scenario and experimental platform from a real-world wireless sensor network project called *Smart Bridge*[1]. Since a few months already, we have been running a sensor network on a Bavarian highway bridge. The goal of this project is to do a long-term monitoring of the bridge in order to detect problems. E.g., changes of cracks over time are monitored with the aim of detecting abnormalities as soon as possible. Figure 2 shows the general architecture of our monitoring solution, consisting of the sensor network itself and the backend where all measurement data is collected. In addition, there is a visualization frontend. It should be noted that all sensor nodes are solely battery-powered and we employ a number of mechanisms in order to save energy and let the sensor network survive as long as possible.

Figure 3 shows a single sensor responsible for measuring certain parameters of the crack (e.g. the width or length of the crack). The sensor node not only consists of the sensor itself (a linear encoder in this case) but also of a computation / communication entity. It is possible to run more or less arbitrary software on this entity which allows to run our proposed runtime monitors on the sensor nodes. The runtime monitors are automatically generated based on a formal description of the properties of interest. At runtime, the monitors observe the system or the sensed data and examine whether the properties are satisfied or violated (i.e. whether abnormalities of the cracks are detected or not). In case of a violation further ac-

[1] http://www.intelligentebruecke.de

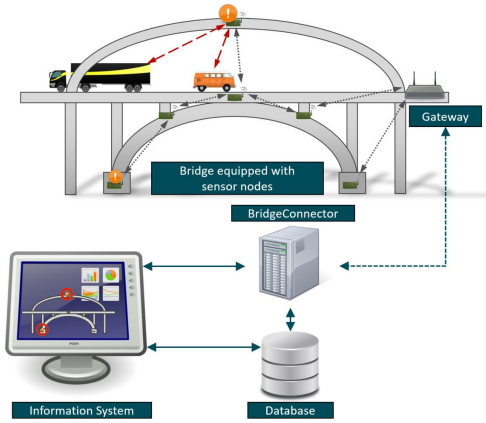

Figure 2: General architecture of the monitoring solution.

tions can be triggered (e.g. the respective traffic office is automatically informed) by the appropriate components of the self-healing system. These components can be deployed for example on the same sensor node, on other sensor nodes or on dedicated hardware.

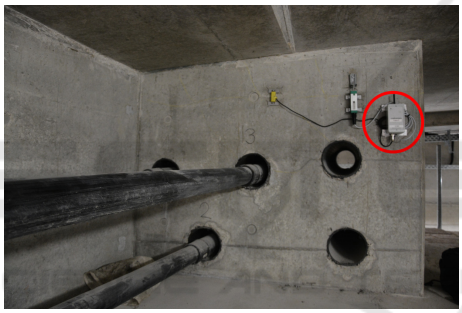

Figure 3: A single sensor node monitoring a crack.

Finally, Figure 4 shows the configuration of all sensors into the overall sensor network along with the strongest links. The network consists of 13 nodes. Their tasks are shown in Table 1. The figure also shows possible placements of the monitors along with the nodes they may observe. Please note that this visualization is a bit misleading, because the monitors rather observe events and check properties of the overall system instead of single or groups of devices.

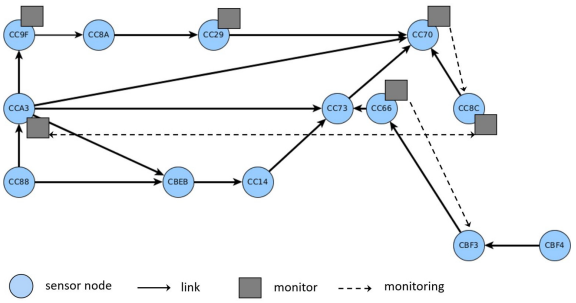

Figure 4: The network of sensors with its strongest links.

Table 1: Sensor nodes with their individual tasks.

ID	MAC suffix	Task
59	CC70	gateway
60	CBEB	surface temperature
61	CBF4	weather station
62	CC14	surface temperature
63	CBF3	outside temperature and humidity
64	CC29	Crack 1 monitoring
65	CC66	repeater
67	CC73	repeater
68	CC8C	Crack 2 monitoring
69	CCA3	Crack 2 monitoring
70	CC9F	Crack 1 monitoring
71	CC88	expansion
72	CC8A	inclination

This scenario / experimental environment is very well suited for our purposes due to the following reasons:

- It offers a sufficient number of nodes in order to evaluate a number of strategies for the distribution of the monitoring, diagnosis and mitigation components.
- Due to the harshness of the environment, it can be expected that sensor nodes show realistic and various failures from time to time.
- The network is multi-hop, allowing to compare strategies which place the monitors for a certain property in one-hop distance to those which place the monitors in multi-hop distance, but on a potentially more reliable node.
- Since we know the functionality of single nodes we can take this information into account when placing the runtime monitors.

5 OUTLOOK

In this paper, we introduced our idea for a model-based approach for self-healing IoT systems, along with a potential and realistic experimental environment. The key challenges are the functionality and the placement of the self-healing components, particularly of the monitors which observe the system behavior and detect violations of properties. The detection of violations makes it possible to trigger appropriate actions trying to bring the system back into a correct state and to prevent harm. In the near future, we are going to develop the concepts in more detail and start the implementation in the next step. Especially, the resulting implementation will be analyzed

and optimized in a long-term performance evaluation on the highway bridge system soon.

REFERENCES

Angarita, R. (2015). Responsible objects: Towards self-healing internet of things applications. In *2015 IEEE International Conference on Autonomic Computing*. IEEE.

Angarita, R., Rukoz, M., and Cardinale, Y. (2015). Modeling dynamic recovery strategy for composite web services execution. *World Wide Web*.

Angarita Arocha, R. (2015). *An approach for Self-healing Transactional Composite Services. (Une approche auto-corrective pour des services composites transactionnels)*. PhD thesis, Paris Dauphine University, France.

Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T., et al. (2002). Daml-s: Web service description for the semantic web. In *International Semantic Web Conference*. Springer.

Asim, M., Mokhtar, H. M., and Merabti, M. (2010). A self-managing fault management mechanism for wireless sensor networks. *CoRR*.

Bourdenas, T. and Sloman, M. (2010). Starfish: Policy driven self-management in wireless sensor networks. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. ACM.

Coulson, G., Porter, B., Chatzigiannakis, I., Koninis, C., Fischer, S., Pfisterer, D., Bimschas, D., Braun, T., Hurni, P., Anwander, M., Wagenknecht, G., Fekete, S. P., Kröller, A., and Baumgartner, T. (2012). Flexible experimentation in wireless sensor networks. *Commun. ACM*, 55(1):82–90.

Decker, N., Kühn, F., and Thoma, D. (2014). Runtime verification of web services for interconnected medical devices. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*. IEEE.

Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything.

Fischer, S. and Leucker, M. (2013). Runtime verification and reflection for wireless sensor networks. In *2013 4th International Workshop on Software Engineering for Sensor Network Applications*. IEEE.

Fok, C.-L., Roman, G.-C., and Lu, C. (2009). Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Trans. Auton. Adapt. Syst.*

Ghosh, D., Sharman, R., Rao, H. R., and Upadhyaya, S. (2007). Self-healing systems—survey and synthesis. *Decision Support Systems*.

Glombitza, N. (2011). *Ein dynamisches, ganzheitliches Geschäftsprozessmanagement in Enterprise-IT-Systemen und drahtlosen Sensornetzen*. PhD thesis, Universität zu Lübeck.

Gurgen, L., Gunalp, O., Benazzouz, Y., and Galissot, M. (2013). Self-aware cyber-physical systems and applications in smart buildings and cities. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. EDAA.

Herbert, D., Sundaram, V., Lu, Y.-H., Bagchi, S., and Li, Z. (2007). Adaptive correctness monitoring for wireless sensor networks using hierarchical distributed run-time invariant checking. *ACM Transactions on Autonomous and Adaptive Systems*.

IBM (2005). An architectural blueprint for autonomic computing.

Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*.

Kephart, J. O. (2005). Research challenges of autonomic computing. In *Proceedings of the 27th international conference on Software engineering*. ACM.

Kühn, F., Thoma, D., Labitzke, D., and Fischer, S. (2017). Monitoring as a service for networked medical cyber-physical systems. In *11th International Workshop on Service-Oriented Cyber-Physical Systems in Converging Networked Environments (SOCNE)*. IEEE Computer Society. In Press.

Kleine, O. (2016). *Semantic Web im Internet der Dinge: Konzeption, Implementierung und prototypische Anwendungen*. PhD thesis, Universität zu Lübeck.

Leucker, M. and Schallhart, C. (2009). A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*.

Nguyen, T. A., Aiello, M., Yonezawa, T., and Tei, K. (2015). A self-healing framework for online sensor data. In *2015 IEEE International Conference on Autonomic Computing*. IEEE.

Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE.

Portocarrero, J. M. T., Delicato, F. C., Pires, P. F., Gámez, N., Fuentes, L., Ludovino, D., and Ferreira, P. (2014). Autonomic wireless sensor networks: A systematic literature review. *J. Sensors*.

Psaier, H. and Dustdar, S. (2010). A survey on self-healing systems: approaches and systems. *Computing*.

Salehie, M. and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*.

Sanchez, L., Muñoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutierrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis, E., and Pfisterer, D. (2014). Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61(Supplement C):217 – 238. Special issue on Future Internet Testbeds – Part I.

Sokolsky, O., Sammapun, U., Regehr, J., and Lee, I. (2008). Runtime verification for wireless sensor network applications. In *Runtime Verification*. IBFI.

Weiser, M. (1999). Some computer science issues in ubiquitous computing. *ACM SIGMOBILE Mobile Computing and Communications Review*.