Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 66 – 73

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Distributed Fault Detection in Smart Spaces Based on Trust Management

Sila Ozen Guclu[a], Tanir Ozcelebi[a], Johan Lukkien[a]

[a] *Eindhoven University of Technology, Department of Mathematics and Computer Science, Den Dolech 2, 5600 MB Eindhoven, The Netherlands*

Abstract

Application performance in a smart space is affected by faulty behaviours of nodes and communication networks. Detection of faults helps diagnosis of problems and maintenance can be done to restore performance, for example, by replacing or reconfiguring faulty parts. Fault detection methods in the literature are too complex for typical low-resource devices and they do not perform well in detecting intermittent faults. We propose a fully distributed fault detection method that relies on evaluating statements about trustworthiness of aggregated data from neighbors. Given one or more trust statements that describe a fault-free state, the trustor node determines for each observation coming from the trustee whether it is an outlier or not. Several fault types can be explored using different trust statements whose parameters are assessed differently. The trustor subsequently captures the observation history of the trustee node in only two evidence variables using evidence update rules that give more weight to recent observations. The proposed method detects not only permanent faults but also intermittent faults with high accuracy and low false alarm rate.

Keywords: distributed fault detection; intermittent fault; outlier; opinion extraction; sensor nodes; trust; smart spaces; reliable; scalability

1. Introduction

In smart spaces, smart objects such as sensor nodes and smart phones are connected over a network in order to enable human-centric applications. By their very nature, smart spaces and smart objects therein are hampered by faulty behaviours of smart objects and communication networks interconnecting these. Furthermore, increment in number of devices also leads to higher failure probability. As this is a danger for the performance and even the correctness of smart space applications, faults must be detected and dealt with, preferably in an autonomous way. Dealing with faults in practice is often through physically replacing, rebooting and reconfiguring faulty parts by itself. If inevitable faults lead to (temporary) erroneous behaviors, a smart space and smart space applications must be able to quickly adapt and return back to a correct operation state without the constant need for human intervention. Fault

* Sila Ozen Guclu, Tanir Ozcelebi, Johan Lukkien
E-mail address: s.ozen@tue.nl, t.ozcelebi@tue.nl, j.j.lukkien@tue.nl

detection accuracy and precision are of huge importance as false alarms and false negatives come with a price, in the form of unecessary adaptations or failures. The focus of this paper is distributed fault detection. Adaptation is beyond the scope of this paper and is left as future work.

Faulty behavior is observed in the form of observations that highly deviate from the average, from here on called *outliers* and does not always take its source in a hardware or software fault. Short-term sequences of outliers can also be a result of fluctuations in the physical environmental conditions. Some examples are wireless signal strength loss due to people passing by, a packet loss due to temporary radio interference and corrupted data in magnetic storage due to cosmic rays. Some smart applications may tolerate short-term outliers without experiencing an application failure or a significant performance drop, while others may not. Obviously, what is 'short-term'must be defined per application. Making a good distinction between outliers that are mostly harmless, and faults that may lead to failures is crucial[7]. For this purpose, a typical approach in the literature is doing time series analysis of observations. This becomes especially challenging in distributed smart space architectures where most of the computation and decision making are done locally, without the computational resources and the global overview of a central server. This is because each smart object observes the environment from its own limited point of view.

Ideally, communication, memory and computation overheads of fault detection should be negligible and allow scaling to large networks[9]. A distributed fault detection method based on trust management is suitable for this purpose as the entire history of observations (information that is relevant) is captured in a single trust variable. A trust relationship between neighbors provides a local view of the network. There are two entities in such a trust relationship: a *trustor* and a *trustee*. The trustor relies on the trustee for the truth of a predicate. Bui et al.[4] states that such a predicate can be, for example, on the integrity, the security or the correctness of the data of the trustee. It can also be on the correct functioning of the trustee or the communication network in between. In the proposed method, the trustor tries to identify whether statements regarding the trustee given by predicates are trustworthy or not. Trust opinions are derived from subjective logic that represents specific belief calculus and considers uncertainty as given by Josang et al.[6].

Sensor nodes send their statistics to neighbor nodes and a base station. The trust is derived from the packet transmissions between sensor nodes. The trustee sends packets that consist of measurements such as communication statistics and sensor readings. Our fault model aims to identify sensor faults and communication faults which might be permanent or intermittent. A permanent fault means that the sensor node generates outliers continuously for the remainder of its lifetime. The source of intermittent faults is typically faulty hardware or faulty software. In this case the faulty component gives a series of outliers at irregular inter-arrival times[8]. Observations of smart objects are monitored for capturing the history of outliers, which may overall indicate a fault.

This paper presents a distributed method that detects both permanent faults and *intermittent faults* based on trust management. In comparison to approaches that employ time series analysis on a recent time-window[1,2,14], the proposed method has much lower memory footprint. It works with very little communication overhead as the only addition to network traffic is the exchange of a couple of variables between neighbor nodes. The entire history of observations is captured by using only two evidence variables, while giving more weight to the recent observations. Nevertheless, the proposed method differentiates outliers and faults with very high accuracy and high precision (or equivalently, low false alarm rate). In simple terms, after establishing which observations are outliers and which are not as we explain later, it works in the following way. Every *normal* observation builds up trust that there is no fault. Every outlier takes away trust, possibly at a different rate. We say that there is a fault if the trust value, whose initial value can be taken suitably depending on the application, falls below a certain threshold. This actually translates to 'there is not enough trust in the trust statement that describes the fault-free state'. Even if the trust value drops dramatically due to temporary outliers, trust can be re-established after observing a number of normal observations. On the other hand, permanent and intermittent faults give rise to a trend in which the dropping rate of trust due to outliers is on average faster than the trust reestablishment rate, resulting in a permanent distrust. Our proposed model investigates outliers in detail. Therefore, the proposed model could cause longer detection time in order to perform low false alarm rate. Mahapatro et al. stated that the number of tests and frequency of tests should be defined initially for detection of intermittent faults[10]. The proposed method tracks the history without the requirement of repetitive tests and parameters for the tests. Parameters of trust management only depend on the failure specification of the application. For example, flipping a bit in the payload of one packet out of every hundred packets is not a very sig-