Journal of Systems Architecture 61 (2015) 511–519

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

A trustworthy, fault-tolerant and scalable self-configuration algorithm for Organic Computing systems

Nizar Msadek *, Rolf Kiefhaber, Theo Ungerer

Institute of Computer Science, University of Augsburg, 86135 Augsburg, Germany

ARTICLE INFO

Article history:
Received 19 September 2014
Received in revised form 27 April 2015
Accepted 1 July 2015
Available online 29 July 2015

Keywords:
Organic Computing
Self-configuration
Trust
Self-organizing systems
Contract Net Protocol

ABSTRACT

The growing complexity of today's computing systems requires a large amount of administration, which poses a serious challenging task for manual administration. Therefore, new ways have to be found to autonomously manage them. They should be characterized by so-called self-x properties such as self-configuration, self-optimization, self-healing and self-protection. The autonomous assignment of services to nodes in a distributed way is a crucial part for developing self-configuring systems. In this paper, we introduce a self-configuration algorithm for Organic Computing systems, which aims on the one hand to equally distribute the load of services on nodes as in a typical load balancing scenario and on the other hand to assign services with different importance levels to nodes so that the more important services are assigned to more trustworthy nodes. Furthermore, the proposed algorithm includes a fault handling mechanism enabling the system to continue hosting services even in the presence of faults. The evaluation indicates that the proposed approach is suitable for large scale and distributed systems.

1. Introduction

The Autonomic [1] and Organic Computing [2] Initiatives have become an important research area for future information processing systems. These initiatives consist of developing computer systems capable of so-called self-x properties (like self-configuration [3,4], self-optimization [5,6], self-healing [7] and self-protection [8]) to cope with the rapidly growing complexity of computing systems and to reduce the barriers that complexity poses to further growth. These properties are achieved by constantly observing the system and initiating autonomous reconfiguration if necessary. An essential aspect that becomes particularly prominent in these systems is trust. In this paper we adopt the definition of trust [9] of the research unit OC-Trust of the German Research Foundation (DFG). In their research, trust covers different facets, as, for example, safety, reliability, credibility and usability. Our investigation focuses on the reliability aspect. Furthermore, it is assumed that a node can not realistically assess its own trust value because it trusts itself fully. Therefore, the calculation of the trust value in this work must be done with our former developed trust metrics [10]:

– *Direct Trust* is based on the own experiences a node has made directly with an interaction partner node. Typically, trust values are calculated by taking the mean or weighted mean of past experiences.
– *Reputation* is based on the trust values of others that had experiences with the interaction partner. Reputation is typically collected if not enough or outdated own experiences exist.
– *Confidence* An estimation about the accuracy of node's own trust is required, before direct trust and reputation can be aggregated. This estimation is done by calculating the confidence of a node. If a node does have a direct trust value but is not confident about its accuracy, it needs to include reputation data as well.

When all the aforementioned values are obtained, a total trust value t_n based on the direct trust and reputation values can be calculated using confidence to weight both parts against each other (See Fig. 1). The value t_n represents the current trust of node n and will always range between 0 and 1. The value of 0 means that n is not trustworthy at all while a value of 1 stands for complete trust. This value can then be used to build trustworthy self-x properties. In this paper, we primarily focus on self-configuration and note that our goal is to develop an autonomously scalable self-configuration algorithm that works in a distributed manner. The algorithm should enhance the self-configuration property of OC systems with trust capabilities to enable building a reliable

* Corresponding author.
E-mail addresses: Msadek@informatik.uni-augsburg.de (N. Msadek), Kiefhaber@informatik.uni-augsburg.de (R. Kiefhaber), Ungerer@informatik.uni-augsburg.de (T. Ungerer).

http://dx.doi.org/10.1016/j.sysarc.2015.07.012
1383-7621/© 2015 Elsevier B.V. All rights reserved.

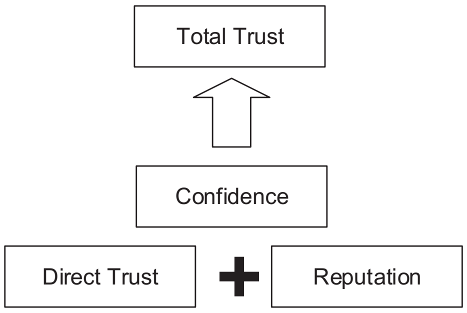

Fig. 1. Aggregating direct trust with confidence and reputation to a total trust value.

system from unreliable components. This is achieved by improving the availability of important services.

The remainder of this paper is structured as follows. Section 2 presents related work on self-configuration including a comparison with our work. Our metrics for enabling a node to host a specific service are presented in Section 3 together with the self-configuration process. In Section 4, we incorporate coordination strategies as enhancement to the self-configuration process in order to make it scalable. The results of the evaluations are shown in Section 5. The paper closes with a conclusion and future work in Section 6.

2. Related work

There are many sophisticated approaches to deal with the allocation problem of services on nodes, either to achieve good load balancing or to minimize energy consumption.

An approach that has become a standard by FIPA[1] is the Contract Net Protocol [11]. It consists of finding an agent that is the most suitable to provide a service. This approach is often adapted and applied in many application domains, for example, manufacturing systems [12], resource allocation in grids and sensor web environments [13,14], as well as in hospitals [15], electronic marketplaces [16], power distribution network restoration [17], etc. Our model is based on the Contract Net Protocol, extended by trust. In this context, trust serves as a mean to give nodes a clue about with which nodes to cooperate.

Bittencourt et al. [18] presented an approach to schedule processes composed of dependent services onto a grid. This approach is implemented in the Xavantes grid middleware and arranges the services in groups. It has the drawback of a central service distribution instance and therefore a single point of failure can occur. Trumler et al. [19] described a scheduling algorithm for distributing services onto nodes based on social behavior. It is implemented in the OCμ middleware. In their model, nodes can calculate a QoS for the services to decide which service is assigned to which node. In this case only resource constraints are used to describe cases when a service should be hosted depending on a specific hardware. In contrast to our approach, this algorithm does not include trust constraints.

In [20], Topcuoglu et al. presented an approach to consider the priorities of tasks. They try to select tasks in order of their priorities and to schedule them to the best machine that minimize their finish time in an insertion based manner. This approach has been shown to significantly improve the schedule computation time. However, a disadvantage is that important tasks might run on unreliable nodes and are prone to fail. Later, in [21], reliability constraints were considered to find a homogeneous allocation of the instances of services. Contrary to this work, our approach is able to work with heterogeneous systems.

[1] FIPA: Interaction Protocol Specifications - [Accessed: April 7, 2015] http://www.fipa.org/specs/fipa00029/.

3. Trust-enhanced self-configuration

The approach of trust-enhanced self-configuration is a crucial part for developing dependable and robust systems using self-x properties. This consists mainly of finding a robust distribution of services by including trust. The services are categorized into important services with a high required trust, and non important services with a low required trust. Important services are those, which are necessary for the functionality of the entire system. E.g., Bernard et al. [22] present a computing grid to solve computationally intensive problems. In their model, trust is incorporated to enable nodes to form Trusted Communities (TCs). The manager, that administrates these TCs is an example for an important service, since its failure deteriorates the entire TC.

The goal is to maximize the availability of important services. Therefore, it is necessary to assign important services to more trustworthy nodes. Trust in this context is expressed by a trust value based on previously developed trust metrics [10]. In addition to trust, resource requirements (e.g., like CPU and memory) should also be considered to balance the load of the nodes.

3.1. Metrics

The self-configuration focuses on assigning services with different required trust levels to nodes which have different trust levels so that more important services are assigned to more trustworthy nodes. Furthermore, the overall utilization of resources in the network should be well-balanced. Therefore, a metric is defined to calculate a Quality of Service (QoS_{total}), i.e., the suitability of node to host a specific service.

$$QoS_{total} = (1-\alpha)\cdot QoS_{trust} + \alpha \cdot QoS_{workload}. \tag{1}$$

The relationship between trust and workload can be set through $\alpha \in [0,1]$. If $\alpha = 1$, the QoS_{total} is only obtained by the current value $QoS_{workload}$, i.e., the suitability of a node to host a specific service with regard to its workload. If $\alpha = 0$, the QoS_{total} is decided only by the actual QoS_{trust} value, i.e., the suitability of a node to host a specific service with regard to its trust value. A higher value α favors $QoS_{workload}$ over QoS_{trust}.

– QoS_{trust} indicates how well the trustworthiness of a node fulfilled the required trust of a service. Fig. 2 visualizes formula 2 to calculate the QoS_{trust}.
t_n represents the current trust of node n calculated based on our former developed trust metrics [10] and will always range between 0 and 1. The value of 0 means that n is not trustworthy at all while a value of 1 stands for complete trust. In this work, it is assumed that t_n is constant at a certain point in time. However, t_n is likely to change over time. This issue was addressed in a subsequent publication of self-optimization [5]. The value t_s represents the required trust value of service s defined by the user according to the importance level of the service. The value of 1 means that the service s is very important and requires to be hosted only on trustworthy node while a value of 0 stands for unimportant service and means that s can tolerate to be hosted on untrustworthy node. If both values are close enough then n has fulfilled the required trust value of a service s. Close enough is defined by the threshold δ_{opt} (optimal area). If the difference between t_n and t_s is more than δ_{opt}, then QoS_{trust} will be gradually decreased until it reaches 0 at $t_n \pm \delta_{tol}$ (tolerance area). If t_s is even beyond $t_n \pm \delta_{tol}$ then the