Science of Computer Programming 80 (2014) 25–49

Contents lists available at ScienceDirect

Science of Computer Programming

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

Formal development of wireless sensor–actor networks

Maryam Kamali [a,b,*], Linas Laibinis [a], Luigia Petre [a], Kaisa Sere [a]

[a] Department of Information Technologies, Åbo Akademi University, Turku, Finland
[b] Turku Centre for Computer Science (TUCS), Turku, Finland

ARTICLE INFO

Article history:
Received 22 January 2011
Received in revised form 30 January 2012
Accepted 5 March 2012
Available online 20 March 2012

Keywords:
Wireless sensor–actor networks (WSANs)
Coordination links
Coordination recovery
Refinement
Pattern development
Event-B
RODIN tool

ABSTRACT

Wireless sensor–actor networks are a recent development of wireless networks where both ordinary sensor nodes and more sophisticated and powerful nodes, called *actors*, are present. In this paper we introduce several, increasingly more detailed, formal models for this type of wireless networks. These models formalise a recently introduced algorithm for recovering actor–actor coordination links via the existing sensor infrastructure. We prove via refinement that this recovery is correct and that it terminates in a finite number of steps. In addition, we propose a generalisation of our formal development strategy, which can be reused in the context of a wider class of networks. We elaborate our models within the Event-B formalism, while our proofs are carried out using the RODIN platform — an integrated development framework for Event-B.

1. Introduction

The separation of computation and control stands at the basis of the software architecture discipline. The control of the *computing* entities as well as the self-coordination of the *controlling* entities are well illustrated by Wireless Sensor–Actor Networks (WSANs), a rather new generation of sensor networks [1]. WSAN nodes can be of two types: *sensors* (the 'computing' entities) and *actors* (the controlling entities), with the density of sensor nodes much bigger than that of actor nodes. The sensors detect the events that occur in the field, gather them and transmit the collected data to the actors. The actors react to the events in the environment based on the received information. The sensor nodes are low-cost, low-power devices equipped with limited communication capabilities, while the actor nodes are usually mobile, more sophisticated and powerful devices compared to the sensor nodes.

A central WSAN requirement is that of node *coordination*. As there is no centralised control in a WSAN, sensors and actors need to coordinate with each other in order to collect information and take decisions on the following actions [1]. There are three main types of WSAN coordination [2]: sensor–sensor, sensor–actor and actor–actor coordinations. The sensor–sensor coordination in WSANs is similar to that of Wireless Sensor Networks, i.e., it defines how sensors route information, how information aggregates among them and which sensors are responsible for which tasks. The sensor–actor coordination prescribes which sensors should send certain data to which actors. Finally, the actor–actor coordination is concerned with actor decisions and the division of tasks among different actors. In this paper, we focus on the latter type of coordination.

To achieve the actor–actor coordination in WSANs, actors need reliable connection links for communicating with each other. These are established upon initialising a WSAN. However, WSANs are dynamic networks where the network topology continuously changes. The changes occur when new links or nodes are added or when the existing links or nodes are removed

* Corresponding author at: Department of Information Technologies, Åbo Akademi University, Turku, Finland.
E-mail address: mkamali@abo.fi (M. Kamali).

0167-6423/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2012.03.002

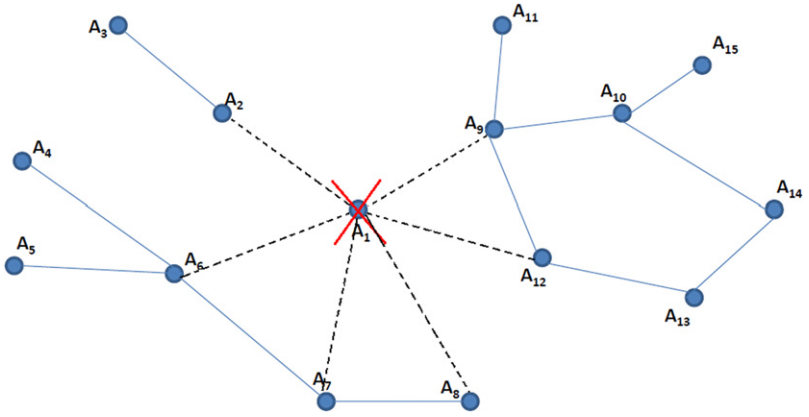

Fig. 1. Three partitions created by the failed actor A_1.

due to failures, typically caused by hardware crashes or malfunctions, the lack of energy, mobility, etc. Consequently, actor nodes can fail during operation of a network. As a result, a WSAN may transform into several, disconnected WSAN sub-networks, thus interrupting the actor–actor coordination. Such a separation is called a *network partitioning* and is illustrated in Fig. 1, where the actor nodes A_1–A_{15} are shown to produce a network partitioning if the actor node A_1 fails.

Another central WSAN requirement focuses on embedding *real-time aspects*. In particular, depending on an application, it might be essential to respond to sensor inputs within predefined time limits, e.g., in critical applications such as forest fire detection. Due to the real-time requirements of WSANs, a failure of an actor node should not affect the whole actor network for too long. To re-establish connectivity of actor nodes, their physical movement towards each other has been proposed in [3,4]. However, during this movement, nodes in different network partitions created by an actor failure cannot coordinate. To shorten the time of recovery, Kamali et al. [5] have proposed an algorithm for establishing new routes between non-failed actors via sensor nodes. This algorithm allows for quick reconnection of the separated partitions, *before* moving actor nodes as proposed in [3,4]. In this paper we focus on studying this recovery mechanism, which alleviates the problems caused by actor coordination failures.

There are several properties that are desirable to verify for this algorithm. First, we need to show that there is *always* a path via sensor nodes that *can be* established for the partitioned actor nodes. Second, it is important to guarantee that the new path is the shortest one for the involved actor nodes. Assuming that the sensor network is sufficiently dense, this also reduces the power consumption of the sensor nodes that are employed to re-establish connection. Third, to shorten the time of recovery as much as possible, it is desirable to re-establish connection as soon as possible. In this paper we address the first and the second property of the algorithm.

The contribution of our paper is threefold. First, we formalise the algorithm for self-recovering actor coordination [5] using a theorem prover tool. This allows us to better understand the functioning of the algorithm and, more importantly, to verify essential properties such as the functional correctness and termination of the recovery mechanism. An important aspect of the recovery is that indirect links between actors are built in a *distributed* manner, thus ensuring *self*-recovering of the network. Second, we prove that the recovery can be done at different levels, via different link types, such as direct or indirect actor links, in the latter case also reusing the WSAN infrastructure of sensors. This contributes to modelling fault-recovery in sensor–actor networks. Third, we explore a generalisation of the described approach to a wider class of networks by using the notions of refinement patterns and pattern-driven formal development. This allows us to facilitate the presented formal development process by identifying the development (design) steps typical for coordinating networks as well as reusing both formal models and proofs.

To model the functional correctness of the recovery algorithm, we use the mathematical concepts of *tree* and *forest*. In graph theory, a tree is a graph whose any two vertices are connected by a non-cyclic path, while a forest is a set of disjoint trees. As special cases, an empty graph (with no nodes) and a discrete graph on a set of vertices (with no edges) are examples of forests. We introduce a special data structure to model a forest and use it to prove correctness properties in the following way. When a node fails, the set of all the neighbours of the failed node is considered as a set of disconnected trees, i.e., a (node) forest. For instance, in Fig. 1 the nodes A_2, A_6, A_7, A_8 A_9, and A_{12} are the neighbours of the failed node A_1. These nodes are considered as disconnected trees, with each tree initially formed of exactly one of the nodes. While the recovery process either finds or re-establishes links, these trees gradually become connected to each other. When the recovery process terminates, we show that all the trees of the forest are connected. This means that, by the end of the recovery, all the neighbours of the failed node are re-connected.

In order to prove correctness of the recovery mechanism, we employ the Event-B formalism for modelling WSANs and the proposed algorithm [5]. Event-B [6,7] is an extension of the B Method [8] for specifying distributed and reactive systems. In Event-B, a system model is gradually specified at several levels of abstraction, always ensuring that a more concrete model is a *correct development* of the previous, more abstract model. The language and proof theory of Event-B are based on predicate logic and the set theory. Correctness of stepwise construction of formal models is ensured by discharging a