Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 29 (2006) 2578–2598
computer communications
www.elsevier.com/locate/comcom

Hazard avoidance in wireless sensor and actor networks

Ramanuja Vedantham *, Zhenyun Zhuang, Raghupathy Sivakumar

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA

Available online 6 March 2006

Abstract

A typical wireless sensor network performs only one action: sensing the environment. Our requirement for intelligent interaction with the environment has led to the emergence of Wireless Sensor and Actor Networks (WSANs), where a group of sensors, actors, and a central coordination entity (sink) linked by wireless medium perform distributed sensing and acting tasks.

In order to provide tight coupling between sensing and acting, an effective coordination mechanism is required among sensors and actors. In this context, we identify the problem of "*hazards*", which is the out-of-order execution of queries and commands due to a lack of coordination between sensors and actors. We identify four types of hazards and show with an example application, the undesirable consequences of these hazards. We also identify and enumerate the associated challenges in addressing hazards. In this context, we discuss the basic design needed to address this problem efficiently. We propose a distributed and fully localized *hazard-free* approach that addresses the problem and the associated challenges based on the design. Through analytical studies and simulations we study the performance of the proposed solution and two basic strategies, and show that the proposed solution is efficient for a variety of network conditions.

Keywords: Wireless sensor and actor networks; Correctness; Sink-to-sensors data delivery; Simulations; Performance evaluation

1. Introduction

Wireless Sensor Networks (WSNs) have a wide variety of applications in civilian, medical, and military applications. However, the nodes in such a network are limited to one type of action: *sensing the environment*. With increasing requirements for intelligent interaction with the environment, there is a need to not only perceive but also control the monitored environment. This has lead to the emergence of a new class of networks capable of performing both sensing and acting on the environment, which we refer to as Wireless Sensor and Actor Networks (WSANs).

The architecture for a WSAN can be seen as an extension of a wireless sensor network where the *sensors* collect information of the environment and report it to the *sink*, which processes the data and issues commands to the *actors* to act on the environment [1]. Thus, in WSANs, the nature of actions performed by the nodes in the network evolves to both *sensing and operating* on the environment. Sensors are typically low-power devices with limited sensing, computation, and wireless communication capabilities [2]. However, since "acting" (e.g., heating) is likely to be a more complex and energy consuming activity than sensing (e.g., temperature reading), it is reasonable to assume that actors are more expensive resource-rich nodes equipped with better processing capabilities, higher transmission powers, large acting range, and longer battery life. For this reason, the number of sensors deployed in a WSAN field will likely be much more than the number of actors.

The evolution from WSNs, which can be thought of to perform only *read* operations, to WSANs, which can perform both *read and write* operations, introduces unique and new challenges that need to be addressed. In this paper, we address one such challenge. Consider a simple

* Corresponding author.
E-mail addresses: ramv@ece.gatech.edu (R. Vedantham), zhenyun@ece.gatech.edu (Z. Zhuang), siva@ece.gatech.edu (R. Sivakumar).

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.02.012

example of a WSAN that uses fire-detector sensors along with water-sprinkler actors. Assume that the sink has issued a *command* directive[1] C to the actors to sprinkle water in response to sensor feedback about a fire. Now, after a certain period of time t, consider the sink to issue a *query* directive Q to check if the fire has been extinguished. If, for a certain region in the WSAN, Q is delivered and executed *before* C by the network, the corresponding response by the sensors – that the fire still exists – will trigger an unnecessary reaction by the sink in the form of more directives to the actors to sprinkle more water. The execution of directives in an order different from what the sink intended it to be has thus resulted in an undesired outcome. While the undesired outcome in the above example is merely the wastage of water, depending upon the nature of the application, such outcomes can even be catastrophic (e.g., poison gas actors where one dose of the gas merely invalidates subject, but two doses can kill).

We refer to such problems where the execution order of directives is different from what the sink intends or expects it to be as *directive hazards*[2]. For brevity, we refer to the problem as simply *hazards* in the rest of the paper. Essentially, the inherent dependency between the actions performed by the sensors, and those performed by the actors, imposes a need for the sink to have control over the order in which directives will be executed, to ensure correctness of operations.

In developing solutions to avoiding hazards, two additional challenges need to be addressed: (i) The rate of execution of the directives, the *directive execution throughput*, has to be maximized in order to serve applications that are real-time. Note that most WSAN applications are likely to have real-time requirements. Revisiting the example introduced earlier, the reporting of the fire and the turning on of the sprinklers have to be done as quickly as possible to ensure effectiveness of the WSAN's application. Hence, a simplistic solution to address hazards that involves the sink waiting to hear confirmation of the previous directive execution from all nodes in the WSAN will be clearly undesirable. (ii) The solution, while avoiding hazards and maximizing directive-execution throughput, should also be designed with consideration to the conservation of communication and energy resources [2] at the WSAN nodes. Specifically, the solution must incur low communication and resource overheads. In this context, we make the following contributions in this paper:

- We first identify the different types of hazards possible in a WSAN, and outline the associated challenges in developing a solution to avoid them.
- We then present a distributed and localized approach called the Neighborhood Clock approach that addresses the hazards and the associated challenges efficiently, and compare its performance against that of baseline strategies using simulations-based analysis.

The rest of the paper is organized as follows: Section 2 illustrates the architecture for WSAN and identifies the problem setting with an example. Section 3 identifies the hazards and describes the associated challenges and goals. Section 4 presents the design that needs to be leveraged for hazard-free operation. Section 5 presents a distributed and fully localized hazard avoidance approach that realizes the basic design. Section 6 evaluates the performance of the proposed approach with two basic strategies for a variety of network conditions. Section 7 discusses some of the issues with the proposed approach. Section 8 discusses related works and Section 9 concludes the paper.

2. Model and example application

2.1. Model

In this paper, we consider an architectural model, where there is a sink to help in the coordination of sensors and actors. This is an extension of the existing architecture for wireless sensor networks, where the sink serves as the coordination entity and issues directives to both sensors and actors. For the above model, we focus on a generic class of applications, where there are *regional events occurring requiring several iterations of directives*. There are other classes of applications, such as applications where the events are point-events, and applications where the events can be addressed by just one round of operation. However, these applications only require a subset of mechanisms to address the problem considered in this work as we explain in Section 7.

Given the above architectural model and class of applications, we now present an example application to explain the problem considered in this work.

2.2. Example application

Consider an automated fire extinguisher system for a large, greenhouse garden application. Let this system be equipped with two types of sensors and two types of actors. The first type of sensor is a temperature sensor that monitors the average temperature in its sensing range and uses this as a trigger to determine the presence of fire. The second type is a humidity sensor, which determines the moisture content in the air. The first of the two types of actors is a sprinkler, which sprinkles water when there is a fire within its acting range, while the second is a fan which removes moisture from the air. One of the goals is to douse any outbreak of fire by activating the sprinklers, without flooding the environment. The second goal is to maintain the moisture level in the air below a certain threshold value by

[1] We use the term *directives* to generically refer to both commands and queries.

[2] The inspiration for the term comes from the data hazards problem when pipelining instructions in a CPU.