Debugging Wireless Sensor Networks Using Mobile Actors

Rajesh K Karmani and Gul Agha
Department of Computer Science
University of Illinois at Urbana-Champaign
{rkumar8,agha}@cs.uiuc.edu

Abstract

Large-scale deployments of sensor networks can potentially serve as infrastructure for multiple, concurrent applications. Realizing this potential requires tools for monitoring, debugging and repairing deployed wireless sensor networks (WSNs). We propose an approach for post-mortem debugging of WSNs using autonomous and mobile actors. By allowing the computation (mobile actor) to move to the nodes where the data is located, we overcome the necessity of moving the data while still providing the flexibility necessary to diagnose errors in WSNs. We define two mechanisms for debugging–namely, forward tracking *and* backward tracking *in which an actor, starting at an error state, tracks the causal events, respectively, forward or backward in time in order to determine the root cause of the error. We show that mobile actors enable both forward and backward tracking, and these may be useful under different conditions.*

1. Introduction

Unlike traditional networks of computers, wireless sensor networks (WSNs) are characterized by unattended operation, low bandwidth, frequent node and communication failures, and large scale deployments. Moreover, each sensor node has limited energy, processing, memory and bandwidth. Errors in such systems may result from the interaction of hardware, software and the environment, requiring energy efficient, network wide debugging support.

We propose the use of mobile actors for debugging WSNs. Note that we use debugging WSNs in the same sense as troubleshooting WSNs. Since mobile actors enable the computation to be moved to where the data is located, mobile code can minimize energy use by reducing the amount of communication necessary. Mobile actors go further and provide the ability to do network wide debugging, where the continuation actor can carry the results of a computation on one node to another. Our approach has been implementated in ActorNet [7]. *ActorNet* is an agent-based framework for dynamically programming and debugging WSNs. End-users (WSN operators but not necessarily programmers) define actors in an expressive, high-level language to specify debugging logic. The framework allows actors to move through the network in order to accomplish their objective.

2 Related Work

Previous research can be divided into work focusing on on debugging individual nodes [9, 3] and that on debugging network wide properties. Obviously, debugging individual nodes is insufficient for determining the source of errors which are detected away from their source, or those that involve the interaction of several nodes. Other proposals for network wide debugging have involved centralized mechanisms which require transmitting logs [6, 5]. However, transmitting logs is expensive and therefore not scalable in WSNs. A proposal which reduces network traffic is to transmit statistics [8] rather than raw logs. Unfortunately, many errors, in particular applications-specific bugs, may not be captured by such statistics. More critically, new types of errors may be detected in a deployed system, requiring different methods for diagnosing them to be used dynamically. Mobile actors can address these difficulties.

3 Motivating Examples

Two mechanisms may be used in debugging. *Forward Tracking* involves tracking causal events forward in time to detect an erroneous computation. *Backward Tracking* involves tracking causal events backward in time to detect root cause of an error. We illustrate their use by means of two examples.

Forward Tracking Example. Consider a token passing protocol executing on the network. Token passing protocol is commonly used as a service in WSN to implement

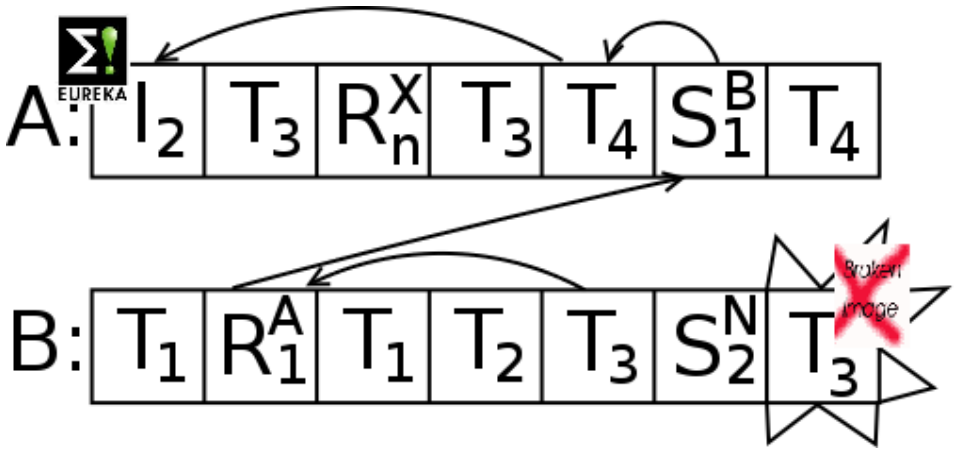

Figure 1. Event logs on two WSN nodes demonstrating distributed backward tracking.

TDMA-based MAC protocols [2, 1]. At some point, the operator of the network observes a breakdown in service. Consequently, the operator would like to find the root of the problem in order to repair the application and/or network. The operator realizes that any token passing protocol obeys the following safety and liveness property:

$$\phi = (\exists i \in N, Token(i)) \wedge (\forall j \in N, j \neq i, \neg Token(j))$$

An intuitive strategy is to grab a consistent system state and forward trace the chain of causal events across the network. In the token passing protocol, a pair of causal events consists of the sending and receiving of a token. By definition, the global state before and after such a pair is consistent. Therefore, each state S_i in the chain $< S_1, S_2, S_3... >$ satisfies the predicate ϕ.

At some point, the chain will terminate at a state, say S_n with the last receive of the token at node p. At this point, the debugging actor returns this information to the operator. Based on the further events at p, the operator can infer the root cause, possibly by querying for a node crash scenario, bad route, intermittent connectivity or others.

Forward tracking in general is important for bugs that lead to complete failure of a node or a portion of network as backward tracking is not feasible in such scenarios. Note that forward tracking is similar in concept to *record-and-replay* mechanism for debugging distributed and parallel programs, but the controlled execution required for record-and-replay has very high memory and communication requirements, making it infeasible in WSNs.

Backward Tracking Example. Consider the problem of finding the malfunctioning sensor that caused an aberrant aggregate value. The aggregate value may be computed at a node that is removed from the node with a malfunctioning sensor. The relevant property specified by the operator could be:

$$\phi = (\forall i, j \in D, |i - j| < k)$$

The actor inspects the data arriving at the aggregating node and finds the source of aberrant data by back tracing data to the message it arrived in. The actor migrates to the sender of this message and performs a similar analysis. The actor continues to trace backward until it arrives at the node which produced the aberrant data. Figure 1 illustrates this process.

The error scenario assumes that the function computing the aggregate is unit tested prior to deployment. Alternatively, if the data satisfes ϕ, fine-grained logging (which includes procedure calls and returns) will enable the actor to mark the aggregating function as root cause.

4 Discussion

Our current approach requires nodes to record all non-deterministic events (such as messages sent and received, interrupts) as well as the node state at regular intervals (checkpoints). Additional events (such as procedure calls/returns and tasks posting/finish) may be logged for fine-grained analysis. Although logs can grow arbitrarily large, we observe that stale data is not often useful in sensor networks and thus usually discarded. Thus, unlike standard distributed systems, it is highly unlikely that errors propagate beyond some time window. However, such storage could be further optimized to reduce the size of the logs. One technique we plan to explore to provide such a reduction is *reversible computations* [4] which by enabling the computation inverses of transformations to figure out inputs (messages, sensor values) would trade computation for storage.

References

[1] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi, and F. Santucci. Seran: a semi random protocol solution for clustered wireless sensor networks. *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, 7-10 Nov. 2005.

[2] S. B. Eisenman and A. T. Campbell. Structuring contention-based channel access in wireless sensor networks. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 226–234, New York, NY, USA, 2006. ACM.

[3] J. Elson, S. Bien, N. Busek, V. Bychkovskiy, A. Cerpa, D. Ganesan, L. Girod, B. Greenstein, T. Schoellhammer, T. Stathopoulos, et al. EmStar: An Environment for Developing Wireless Embedded Systems Software. *Center for Embedded Networked Sensing (CENS) Technical Report, CENS-TR-9*, 2003.

[4] S. I. Feldman and C. B. Brown. Igor: a system for program debugging via reversible execution. In *PADD '88: Proceedings of the 1988 ACM SIGPLAN and SIGOPS workshop on Parallel and distributed debugging*, pages 112–123, New York, NY, USA, 1988. ACM.

[5] M. M. H. Khan, L. Luo, C. Huang, and T. F. Abdelzaher. Snts: Sensor network troubleshooting suite. In J. Aspnes, C. Scheideler, A. Arora, and S. Madden, editors, *DCOSS*, volume 4549 of *Lecture Notes in Computer Science*, pages 142–157. Springer, 2007.

[6] V. Krunic, E. Trumpler, and R. Han. NodeMD: Diagnosing Node-Level Faults in Remote Wireless Systems. Technical Report CU-CS-1017-06, University of Colorado at Boulder, 2006.

[7] Y. Kwon, S. Sundresh, K. Mechitov, and G. Agha. ActorNet: An Actor Platform for Wireless Sensor Networks". Technical Report UIUCDCS-R-2005-2595, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005.

[8] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 255–267, 2005.

[9] J. Yang, M. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks. *SenSys '07*, 2007.