

MEDAL: A coMcompact Event Description and Analysis Language for Wireless Sensor Networks

Krasimira Kapitanova
University of Virginia
krasi@cs.virginia.edu

Sang H. Son
University of Virginia
son@cs.virginia.edu

Abstract—Event detection plays an important role in wireless sensor network (WSN) applications such as battlefield surveillance and habitat monitoring. However, effective approaches for specifying events in a sensor network remain a challenge. In this paper we present MEDAL, a formal event description language. MEDAL is a modified Petri net which provides a more compact formal language than its predecessor SNEDL. As a system analysis tool, MEDAL can capture the structural, spatial, and temporal properties of a complex event detection system, which can be used to assist system designers in identifying inconsistencies and potential problems. MEDAL can also perform case-specific analyses that can make the debugging phase easier. We present a case study as an example illustrating the features and effectiveness of MEDAL. We also describe an approach for simultaneous detection of multiple events in a single WSN.

I. INTRODUCTION

As the demand to explore the physical world increases and the sensing and activation tasks become more sophisticated, wireless sensor networks (WSNs) are expected to support more complicated applications. Regardless of the specific applications that are running on the sensor networks, sensors should be able to individually or collaboratively detect “interesting occurrences”, also called *events*. A number of papers [1], [2], [3] discuss issues regarding event detection and dissemination, as well as support for queries in event-based WSNs. However, the problems of event description and how different event types affect the system design have not seen much publicity.

The need for a description language that can incorporate the knowledge of sensing with event definitions is obvious. A formalized description language can be used as an interface between people who register events (e.g. application semantics experts) and the sensor network designers. Such a language will provide computer scientists with a well-defined interface and more clear requirements for designing the sensor network. Most of the papers that describe sensor network events employ SQL or SQL-like semantics to do this [2], [4], [5]. However, as pointed out in [6], SQL-like semantics are not always suitable for sensor networks because of the lack of collaborative decision making and other necessary features.

Other approaches include M. Worboys’ object-oriented model with integrated timing and location properties for event representation [7]. However, since this approach was initially designed for modeling generic events including social and

economic ones, it does not support characteristics unique to sensor networks, e.g. there is no consideration of different sensor types or interactions. There also exist a number of methods that are based on process algebra and composition logic [8], [9]. However, since these approaches were mainly developed for database systems, some important characteristics of sensor networks such as sensing activities or spatial and temporal properties have not been addressed.

In this paper we present MEDAL - a formalized description language designed for event specification in sensor networks. MEDAL modifies SNEDL [10], the first event specification language to support key features of WSNs that SQL has difficulty handling. By eliminating SNEDL’s unnecessary complexity, MEDAL achieves compactness and ease of use. MEDAL is an integrated spatial, temporal, and stochastic Petri net model which handles collaborative decision making, temporal dependency, geographic control, and other issues related to event-based sensor networks. As it incorporates the capabilities of stochastic Petri nets, MEDAL can also perform probabilistic analysis and evaluation. The case study in section 3 presents a detailed sensor network application which demonstrates the usability and effectiveness of MEDAL.

This paper has two main contributions. First, by removing the SNEDL parameters that are not vital to event description, we provide a less complex and easier to use formal language. Second, we describe an approach for simultaneous monitoring of multiple events in a single network. The remainder of the paper is organized as follows. Section 2 presents the definition of MEDAL, compares it to SNEDL, and describes MEDAL’s temporal, spatial, and probability logic. Section 3 contains an example of a specific sensor network application with a detailed MEDAL description. Section 4 introduces the notion of a *base*. Section 5 illustrates our approach to simultaneous detection of multiple events and section 6 concludes the paper.

II. MEDAL MODEL AND LOGIC

A basic Petri net (Figure 1) consists of places (circles), transitions (rectangles or bars), directed arcs, and tokens (dots inside places). Transitions are used to model various kinds of actions, tokens model instances/objects, and places represent the states in which the objects can be. Arcs model the way in which objects are created or destroyed. They also represent changes between states. A *marking* of a Petri net corresponds to a specific status of the Petri net and is defined

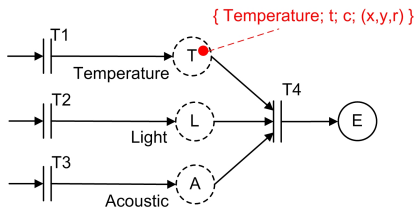


Fig. 1. An example MEDAL Petri net.

as $M : P \rightarrow N$, where P is the set of places and N is the number of tokens.

A. MEDAL Petri Net Definition

SNEDL uses a 10-tuple to represent an event [10]. We consider SNEDL unnecessarily complicated and therefore we have decreased the number of its elements in MEDAL. We have merged SNEDL’s time guard function for transitions δ and the persistency guard for arcs θ into a single time guard function β . In addition, we do not include the quorum/frequency function of transitions Q and the token manipulating function M . M makes the token description needlessly complicated, and the functionality of Q can easily be replaced by properly setting up the time guard function β .

The MEDAL definition of an event description system can be given as a 7-tuple structure: $F = (P, T, A, \lambda, \beta, H, L)$ where P is the set of places, T is the set of transitions, A is the set of arcs, λ is the probability/weight function for the arcs, β is the temporal guard function, H is the threshold function for places, and L is the spatial guard function for transitions. P , T , A , λ , H , and L have the same meaning as in SNEDL [10]. β , the new parameter we introduce, is defined as follows:

β is a temporal guard function. For a transition pre-arc it means that the transition can only fire during the union closure of the given ranges. $\beta(T) = (a1, a2) \cup (a3, a4)$ indicates that transition T can only be fired during interval $(a1, a2)$ or $(a3, a4)$, where $a1 \leq a2$, $a3 \leq a4$. β can also be specified using event incidents: $\beta(T) = (E1, E2)$ means that transition T can only fire between events $E1$ and $E2$. For a post-arc of a transition, β shows how long it takes for this transition to fire. β also acts as a persistency guard, i.e. we can use it to specify the amount of time an arc can hold a token to participate in a transition before this token becomes invalid.

B. Abstracting Sensors Using Sensor Events

Sensors in MEDAL are abstracted by sensor events denoted as places that only take tokens from the environment. Sensor events are the basic entities for composition of higher level events. The number of sensor events is directly related to the types of sensors in the network. For example, if there are three types of sensors - temperature, light, and acoustic sensors, then there are three types of sensor events in the MEDAL model: temperature, light, and sound, as shown on Figure 1. The tokens arriving at each sensor place are associated with temporal and spatial attributes so we can obtain information about when and where the data has been sensed. If we have a

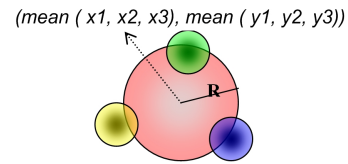


Fig. 2. Enforcing event locality with a spatial guard function.

token with type *temperature*, time stamp t , capacity c , and location attribute (x, y, r) , we can say that a temperature sensor at location (x, y) with sensing range r registered a temperature value c at time t .

C. Temporal Logic

The temporal logic refers to the temporal guard function β which helps specify the temporal concepts “when” and “how long”. β guards the transitions to ensure they fire only during the specified temporal intervals. Using β has practical importance because some events in physical environments can occur only during a particular temporal interval. For example, events that depend on sunshine can only happen during the day. The temporal guard function also allows MEDAL to support complicated temporal logic for event systems.

D. Spatial Logic

The spatial function L is defined to enforce geographic semantics. As a guard function for a transition T , L ensures that the tokens carried by T ’s pre-arcs satisfy the spatial locality conditions. If $L(T) = R$, the effective radius of a higher-level event should be equal to or smaller than R . In other words, there should be a circle of radius R covering all the tokens’ locations in order to consider the sensor readings from different locations as one particular event.

For example, consider a case in which three sub-events (places) have arcs going into transition T . The three types of tokens have location information $((x1, y1), r1)$, $((x2, y2), r2)$, and $((x3, y3), r3)$ as denoted by the three small circles on Figure 2. Under the spatial guard function $L(T) = R$, T can only be fired if the circle centered at the mean of $(x1, y1)$, $(x2, y2)$, and $(x3, y3)$ with radius R overlaps each of the three small circles. After T is fired, the new token’s location is the big circle (Figure 2).

E. Probability Logic

The probability/weight function λ and the threshold function H provide basic probability control for MEDAL. For example, if a sensor network has three types of sensors to support a collaborative decision on detection of an event, the probability function can be used to specify the different weights of each type of sensors in the decision. The probability logic of MEDAL also supports a time-related probability model when the functions are associated with time. In addition, MEDAL is extensible to provide more complicated marking-dependent probability functions, which can employ advanced features of stochastic Petri nets.

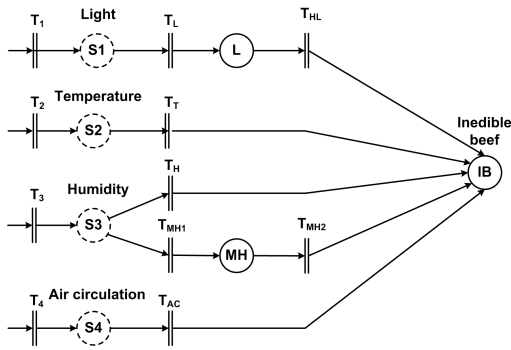


Fig. 3. MEDAL Petri net for a beef monitoring WSN.

III. CASE STUDY - MEDAL FOR BEEF MONITORING

To illustrate the power of MEDAL, we present a sensor network application with a detailed MEDAL description. Suppose that we have a company in the beef storage business. Since beef and meat in general are considered potentially hazardous foods by the US Food and Drug Association (FDA), there are very strict regulations about the conditions in which they should be stored. According to those regulations, the temperature must be maintained within the range from -1°C to 5°C , the mean air speed over the product should be above 0.5 m/s , and the relative humidity must be maintained below 95% or, if the product is stored for longer than 72 hours, below 90% [11]. It is also preferable that beef is stored in the dark, for light accelerates the oxidation of fat with the liberation of free fatty acids and the production of rancidity [11]. The MEDAL Petri net for this scenario is shown on Figure 3. There are four types of sensors in the network: light sensors, temperature sensors, humidity sensors, and air circulation sensors. The structure of the MEDAL Petri net for this system is a 7-tuple $\text{Beef} = (P, T, A, \lambda, \beta, H, L)$ where:

1) P is the set of places. Places $S1$ to $S4$ represent the four types of sensor events, while L , MH , and IB stand for Light detection, Humidity between 90% and 95% , and Inedible beef, respectively.

2) T is the set of transitions.

3) A is the set of arcs. We use $A_{i,j}$ to represent an arc coming from place/transition i to place/transition j .

4) For this scenario, the weight/probability function λ is assigned as follows. For all post-arcs from transitions to states $\lambda = 1$, meaning that once the transition is fired the probability that the token will enter the corresponding state is 1 . All arcs going to sensor transitions $A_{T1,S1}$, $A_{T2,S2}$, $A_{T3,S3}$, and $A_{T4,S4}$ have $\lambda = 1$. The rest of the arcs, except $A_{S3,TH}$ and $A_{S3,THH1}$, also have $\lambda = 1$ since there is a single arc that goes to a single transition. There are two arcs coming out of place $S3$. If we assume uniform probability, the probability that the humidity is between 90% and 95% is the same as the probability that the humidity is between 95% and 100% . Therefore, for both arcs we have $\lambda = 1/2$.

5) β is the time guard function. In order to filter out the sensor reports which claim the presence of sunlight at night,

we define $\beta(T_L)$ to be the time interval between sunrise and sunset. In this way, a number of false alarms can be filtered out. For the rest of the transitions, we can leave β unspecified, which results in the default range $(-\infty, +\infty)$.

For the arcs in this example we specify β in seconds as follows. We assume that the system responds immediately to sensor readings, therefore $A_{T1,S1}$, $A_{T2,S2}$, $A_{T3,S3}$, and $A_{T4,S4}$ have $\beta = 0$. According to regulations, if the humidity is between 90% and 95% the beef is still considered edible as long as it has been less than 72 hours since the humidity has increased above 90% . Therefore, we have $\beta(A_{T_{MH1},MH}) = 2,592,000s$. Also, since the beef is considered inedible if it has been exposed to sunlight for more than half an hour, $\beta(A_{T_{L,L}}) = 1,800s$.

6) H is the threshold function for each place. $H : P \rightarrow v$ means that for a place $p \in P$, $v = H(p)$. If the capacity of a token at place p is over v , that token is allowed to enter place p . The definition of H for this particular scenario is straightforward, since the assigned weights are normalized. We set $H(p) = 1.0 \forall p \in P$.

7) L is the spatial guard function as introduced in section 2 and for sensor events T_1 , T_2 , T_3 , and T_4 the effective radii of the events are their sensing ranges.

IV. THE BASES OF EVENT DETECTION

GEM [10] introduces two concepts - DNA and RNA. A DNA describes the whole structure of a Petri net. Since it contains a considerable amount of data, the DNA is normally stored on the sensor motes before deployment. On the other hand, RNA simply represents the token vector, i.e. the position of the tokens in a Petri net at some specific time. However, neither the DNA nor the RNA can be used to adjust parameter values in the Petri net. If such changes are necessary after deployment, we either have to collect the motes and reprogram them with the new Petri net DNA or pay the cost of sending the DNA over the network. Both of these approaches are too expensive. In order to provide a more suitable solution, we introduce a new abstraction, which we have called a *base* to keep the DNA/RNA reference.

A base is represented by the quadruple $(event, transition, value, time)$, where *event* is the name of the event the base describes, *transition* is the transition to be updated, *value* is the new value that will cause that transition to fire, and *time* is the time period before a token leaves the transition. An example base for our beef monitoring example would be $(Beef, T_T, < 1^{\circ}\text{C}, 0)$. Changing the transitions' bases does not change the structure of the Petri net. It only changes the firing conditions of the transitions which allows us to make modification to the parameters of the model.

The ability to change the initial values of a Petri net does not only help us adjust the system if we have not initially set it up properly. It also allows us to reuse the same WSN if the application requirements change. In terms of implementation, the bases can be stored in a specialized table on the sensor motes. This table is separate from the source code of the MEDAL Petri net, which allows users to easily update and

add entries without having to change the event detection source code.

V. SIMULTANEOUS DETECTION OF MULTIPLE EVENTS

In our beef storage example we are monitoring the storage conditions for a single type of meat. However, a more practical scenario would be storing multiple types of food in the same place. To detect when the storing requirements for any of those foods are violated, we need a sensor network that can simultaneously monitor and identify more than just a single event. A system capable of distinguishing among different events and reporting them accordingly will increase the WSN's efficiency and decrease the event detection cost since otherwise we would have to deploy a separate system per event.

An example two-event scenario could be storing two types of meat (beef and chicken) in the same storage facility. For simplicity, we only consider temperature and air circulation as physical factors affecting the meat quality. There are two approaches we can take. The first one is to build two separate Petri nets for beef and chicken monitoring. The sensor nodes will run these Petri nets simultaneously and events will be detected separately. The structure of these Petri nets, however, will be exactly the same since the events they are detecting are described by the same sensor events. Despite the straightforwardness of this method, if we take into account the scarcity of resources in WSNs, performing identical tasks twice is not desirable. This leads us to the second approach - running a single general Petri net able to detect multiple events.

Once we know the types of sensors that will be used in the WSN, and therefore the sensor events, we can design a general Petri net that can detect all events of interest. The exact conditions describing each event can be specified using the events' sets of bases. As the bases do not require a lot of memory, a sensor node can easily store more than just one event's bases. Since for both the beef and chicken monitoring we are interested in only the temperature and air circulation, the combined Petri net differs from the specific Petri nets only in the definition of the transitions. The transitions will now fire if any of the conditions for either beef or chicken are satisfied. The tokens leaving the transitions can be used to store information about which event's conditions were met.

In the case when the Petri nets of the events we are monitoring do not overlap completely, building the general MEDAL Petri net is not much more complicated. We have shown this process on Figure 4. The two complex events we are interested in detecting are A and B. Their MEDAL Petri nets are displayed on Figure 4a) and 4b), respectively. Event A depends on sensor events X and Y, and sensor events Y and Z describe event B. The two complex events have Y in common and therefore can share a path in the general MEDAL Petri net. The transitions along this path are designed to reflect the constraints specified in the bases of both events. Figure 4c) displays the resulting general Petri net.

When two complex events have nothing in common, i.e. the intersection of their sets of sensor events is empty, combining their MEDAL Petri nets into a single model only requires

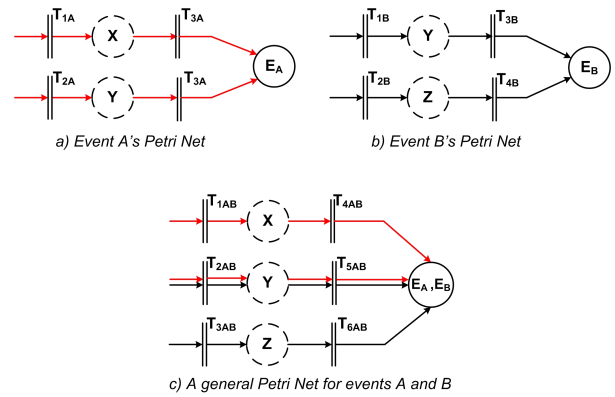


Fig. 4. Building a general event detection MEDAL Petri net.

to merge the final places of the event-specific Petri nets into a single place. This place is the final place of the general MEDAL Petri net and it signals whenever any of the monitored events is detected.

VI. CONCLUSIONS AND FUTURE WORK

We have presented MEDAL, a formalized event description language designed for event-based sensor network systems. MEDAL can model essential features of WSNs such as various sensor types, geographical location, temporal constraints, and probability of events. In addition to formally describing an event-based system, MEDAL can be used as an analysis tool for both system design purposes and system debugging purposes. We have also described an approach for building MEDAL Petri nets that can be used to monitor multiple events simultaneously. Using such a general Petri net instead of multiple event-specific Petri nets will decrease the computational, memory, and communication requirements of event detection.

REFERENCES

- [1] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the physical world," *IEEE Personal Communications*, vol. 7, no. 5, pp. 10–15, October 2000.
- [2] Cornell DB Group - Cougar, "www.cs.cornell.edu/bigreddata/cougar/."
- [3] C. Jaikao, C. Srisathapornphat, and C.-C. Shen, "Querying and tasking in sensor networks," in *Digitization of the Battlespace V and Battlefield Biomedical Technologies II*, vol. 4037, 2000.
- [4] S. Li, S. H. Son, and J. A. Stankovic, "Event detection services using data service middleware in distributed sensor networks," in *IPSN*, 2003, pp. 502–517.
- [5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *SIGMOD 2003*, pp. 491–502.
- [6] M. Franklin, "Declarative interfaces to sensor networks," Presentation at NSF Sensor Workshop, 2004.
- [7] M. Worboys, "Event-oriented approaches to geographic phenomena," *IJGIS*, vol. 19, pp. 1–28, 2008.
- [8] U. Dayal, A. P. Buchmann, and D. R. McCarthy, "Rules are objects too: A knowledge model for an active, object-oriented database system," in *Lecture notes in computer science on Advances in object-oriented database systems*, 1988, pp. 129–143.
- [9] A. P. Buchmann, J. Zimmermann, J. A. Blakeley, and D. L. Wells, "Building an integrated active OODBMS: Requirements, architecture, and design decisions," in *ICDE 1995*, pp. 117–128.
- [10] B. Jiao, S. Son, and J. Stankovic, "GEM: Generic event service middleware for wireless sensor networks," *INSS 2005*.
- [11] J. F. Gracey, D. S. Collins, and R. J. Huey, *Meat Hygiene*. Saunders Ltd., 1999.