FEDL: A Formalized Event Description Language for Wireless Sensor Networks

Binjia Jiao Sang H. Son John A. Stankovic
Department of Computer Science
University of Virginia
{bj3r, son, stankovic}@cs.virginia.edu

Abstract

Event detection plays an important role in sensor network applications such as battlefield surveillance and habitat monitoring. However, effective approaches to specify events in a sensor network remain a challenge. Existing techniques such as natural languages or SQL-like languages have a number of limitations. In this paper, we present a formalized event description language (FEDL) developed especially for sensor networks by extending and modifying Petri Nets. As a description language, FEDL is an extension of Petri-Nets with both diagram and symbolic representation support. A FEDL Petri Net integrates features from color, time and stochastic Petri Nets to tackle problems in specification and analysis. As a system analysis tool, FEDL can capture the structural, spatial and temporal properties of a complex event detection system, which can be used to assist system designers to identify inconsistencies and potential problems. In addition, FEDL can perform case-specific analysis that helps in the debugging phase. A case study is presented as an example to illustrate the features and effectiveness of FEDL.

Keyword: Wireless sensor networks, Petri Nets, spatial and temporal properties, event specification and analysis

1. Introduction

As the demand to explore the physical world increases and the sensing and activation tasks become more sophisticated [5] [7] [9] [15] [12], wireless sensor networks (WSNs) are expected to perform in more complicated applications. Regardless of specific applications that sensor networks are working for, sensors are supposed to individually or collaboratively detect "interesting occurrences". These "interesting occurrences" refer to events, and many papers [5] [9] [24] [29] [42] [45] discuss issues regarding event detection and dissemination, and support for queries in event-based WSNs. However, very few papers have discussed event descriptions and how different event types affect the system design. Most papers that have used event definitions in sensor networks use SQL or SQL-like semantics to describe events [9] [20] [29] [31]. However, as pointed out in [16], SQL-like semantics are not always suitable for sensor networks because of the lack of collaborative decision making and other necessary features.

The need for a description language that can incorporate knowledge of sensing into 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 environmental scientists who know exactly what an event should be) and the sensor network designers (mostly computer scientists, who are responsible for designing protocols to coordinate/sample sensors according to certain temporal and spatial specifications). With such a language, computer scientists are provided with a well-defined interface with clear requirements for designing the sensor network.

While a lot of current approaches use SQL-like languages to describe events, SQL has apparent limitations in describing sophisticated events. The drawbacks of SQL include:

- It cannot elegantly capture data dependency and interactions among different events or sensor types.
- It does not explicitly support probability models.
- It is awkward in describing complex temporal constraints and data dependency.
- It lacks the ability to support collaborative decision making and triggers [16], hence not suitable for events with sensor fusion.
- It cannot give a global picture nor support analysis of the event system.

Consider some examples to demonstrate the above limitations. Suppose there are three types of sensors in the network, which are sound, light, and temperature sensors, to detect explosions. If the explosion event is simply a combination of positive readings from those sensors, then using the notation of [29] an explosion event can be represented as follows:

SQL-Like Representation:

```
INSERT INTO EventList Explosion
      (Event_ID, SubEvent_Set, Spatial Resolution,….)
VALUES
      (0001, SubEventSet,…….)

WHERE SubEventSet is
SubEventSet = (Sound,
                  Light,
                  Temperature,
                  Confidence Function: 0.3*Sound + 0.3*Light + 0.4*Temperature >=1.0,
            …….)
```

The above notation cannot describe a more complex situation where in determining an explosion event, the reading for sound is only valid for 5 seconds, the reading for light only remains valid for 0.5 second, and the reading for temperature can be valid for 10 seconds. For example, if a sensor detects abnormal light, however, within the 0.5 second interval for this light reading, if there is no positive reading from temperature and sound, then this is not an explosion event. This example shows that SQL cannot elegantly handle complicated temporal constraints for events.

Consider the case of probabilistically occurring events. Figure 1 illustrates the scenario of a probabilistic event.

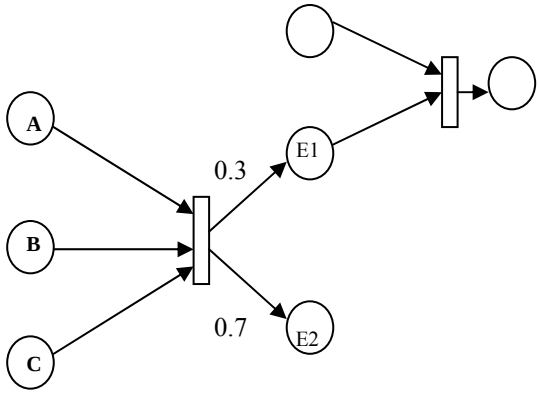

Figure 1: Probabilistic Events

With 30 percent probability, a set of readings from sensor types A, B, and C indicates event E1, while with 70 percent probability, it indicates event E2. Further, E1 and E2 can be sub-events for higher level events. This case is not designed deliberately to overshadow the SQL language; it is much more realistic than simple deterministic model. Since the nature of a sensor network is distributed and noisy, random process and probabilistic models are especially meaningful. Hence, lacking the ability to capture random processes of event systems, SQL-like languages are handicapped as a suitable event description language for sensor network applications.

In this paper, we present FEDL— a formalized description language specially designed for event specification in sensor networks. To the best of our knowledge, FEDL is the first event specification language specially designed to support key features in WSN that SQL has difficulty to handle. FEDL is an integrated spatial, temporal, and sto-

chastic Petri Net model which handles collaborative decision making, temporal dependency, geographic control and other issues found in event-based sensor networks.

Since most events in sensor network applications are in nature concurrent, asynchronous, distributed and non-deterministic, we choose Petri Net as a suitable base model to address the event specification problem in sensor networks. The main contribution of this paper is the development of a formal method, called FEDL, to specify events for WSNs. As a formal method, FEDL is based on Petri Nets, which rigorously specifies events in sensor networks. This rigor prevents ambiguity in event specifications, which is important for the development of a whole rigorous event service system. By integrating features from Color Petri Nets and Time Petri Nets, FEDL can handle different kinds of events including those compound events with complicated temporal and spatial structures.

FEDL is a zero test Petri Net, which has the same power as a Turing machine. This fact makes FEDL immune to suffering from state explosion in a large complex event system, as is the problem for Finite State Machine based formal methods. By including Stochastic Petri Nets property, FEDL can do probabilistic analysis and evaluation. This property is very convenient for a vast number of distributed and individually unreliable sensors. Though not much stochastic analysis is discussed in this paper, we are currently working on performance evaluation using stochastic processes, to be reported in a separate paper. We present a detailed sensor network application in the case study, which visually demonstrates the usability and effectiveness of FEDL.

The remainder of this paper is organized as follows: In Section 2, related research is reviewed. Section 3 begins with the definition of FEDL, followed by structural, temporal, spatial and probability logic. Section 4 presents an example of a specific sensor network application with a detailed FEDL description. Implementation of the FEDL tool is discussed in Section 5, and Section 6 concludes the paper.

2. Related Work

Although a lot of interesting work has been done in sensor networks, not much prior research directly focuses on providing a formalized language to describe events in sensor networks, supporting data dependency and collaborative decision making. The papers that describe events using SQL-like primitives [9] [29] [31] [20] vary a little in semantics. In [9] [20], the authors simply employ general SQL primitives to define events in sensor networks. The limitation of this approach is that the events can only be defined by predicates on sensor readings connected with "AND" and "OR" with very simple temporal and spatial constraints. Madden et al have extended SQL primitives by incorporating streaming support, where the desired sample rate can be included [31]. Li et al proposed defining events using a sub-event list and confidence functions in SQL language [29]. Nevertheless, as shown in the previous example, the inherent limitations of SQL make it limited for a formal description language of events in sensor networks.

Some relevant work attempting to define events in sensor networks can be found in [44]. In that paper, an object-oriented model was proposed to represent events, while some timing and location attributes are integrated. However, the approach was initially designed to model generic events including social and economic ones; hence grafting it directly to sensor networks lacks necessary features to support unique characteristics of sensor networks, e.g., no consideration on different sensor types and interactions.

Although few formal methods have been used for event specification in sensor networks, formalized approaches for event descriptions and compositions are widely studied in other areas. A survey on formal methods for specification and analysis [2] shows that most of the popular approaches are based on the theoretical models such as finite state machine, timed automata, process algebra and Petri Nets. Some widely used methods such as SDL [14] [40], SPIN (Simple ProMeLa Interpreter) [23] and Estelle [13] are based on finite state machines. As pointed out in [48], FSM (finite state machines) approaches have difficulty in dealing with hierarchical modeling, synchronization and specifying control logic. FSM can be augmented to Timed Automata by incorporating a finite set of real-valued clocks. Timed Automata is the mathematical foundations for a lot of specification methods, among which UPPAAL [5], Kronos [49] and HyTech [21] are most well known. Generally speaking, both FSM and Timed Automata are

deterministic finite state machines, and hence they inherit the limitations of finite state machines. For example, because of the limitation in computation power, deterministic finite state machines suffer from state explosions in large and complex distributed systems.

There exist a number of approaches that are based upon process algebra and composition logic. Composition of events was presented together with the concept of Event-Condition-Action rules in [11]. In the paper, the authors proposed HiPAC event algebra for database systems. Some event specification approaches were proposed for database systems in other papers [6] [8] [17], and these methods are all designed for active database systems to define and compose events. Later on, various approaches of composition algebra have been proposed to handle events in a distributed system [30] [38] [46]. However, since these approaches were mainly developed for database systems, the characteristics of sensor networks such as sensing activities, spatial and temporal properties of WSNs have not been addressed in those approaches.

Real-time Maude has been developed as a language and tool for specification and analysis for real time and hybrid systems [35] [36]. It is based on re-writing logic, which has no limitations of FSM in general. Though Real-time Maude may provide some analysis ability, it does not support stochastic models and cannot provide generalized performance evaluation on the system. Also Real-time Maude demands users' knowledge of re-writing logic, and lacks graphical modeling support for users who are not familiar with the theory and syntax.

Since the introduction of Petri Nets [37] in 1962, a large number of papers and books have been published on the topic. Places and transition nets were first formally defined in [25], high-level Petri Nets were introduced in [18] and Color Petri Nets were proposed in [26] [27]. Later on, Time Petri Nets and Stochastic Petri Nets were proposed in [12] [42]. Petri Nets have advantages to describe events in sensor network applications because

- Events in sensor networks tend to be distributed, concurrent, asynchronous and non-deterministic.
- Methods based on Finite State Machine including Timed Automata cannot handle the non-determinism elegantly, and therefore result in state explosion in some cases.
- Extended Petri Nets with zero test (including timed Petri Nets, FEDL Petri Nets) are equivalent to Turing Machines in computation power, thus handle non-determinism [34] and do not suffer from state explosion.
- Geographic information is crucial to events in sensor networks, since a lot of events are related location, area and may have other spatial properties. With features inherited from Color Petri Nets, FEDL can incorporate spatial properties into the token, handling geographical construction for events in sensor networks.
- One advantage of FEDL over other specification methods is the performance evaluation ability due to the features inherited from Stochastic Petri Nets.
- Petri Nets naturally have graphical support, thus it is convenient for those users who are not versed with logic or other complicated theoretical backgrounds.

Although Petri Nets were used in describing discrete event systems in some research [28], they have not been used directly in sensor networks. It might be because that sensor network research has just emerged as a new area and its complex temporal, spatial and probabilistic properties are hard to specify by simple Petri Net models proposed before. FEDL aims to address key aspects of sensor networks such as temporal control, spatial constraints, heterogeneity, and probability issues. In FEDL, a token is associated with type, capacity, time, and location attributes. FEDL provides various guard functions to enforce transitions and arcs (flows) to guarantee the desired temporal, spatial and other constraints defined by the application layer.

3. FEDL Model and Logic

A basic Petri Net 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 to model instances / objects, and places

represent the states in which the objects can be. Arcs represent the way in which objects are created or destroyed; they also represent changes between states [19]. A marking of a Petri Net represents a specific status of a Petri net and is defined as M: P→N, where P is the set of places and N is the number of tokens.

There are many extensions of basic Petri Nets and there is no clear boundary among the variations. There are some famous extensions such as time Petri Nets, color Petri Nets and Stochastic Petri Nets that are widely used. Time Petri Nets associate time related information with Petri Net components. Color Petri Nets distinguish the difference among tokens, and Stochastic Petri Nets associates weights / probabilities with Petri Net components. As Petri Nets were developed in different application domains, a large number of extensions have emerged and most of them cannot be clearly classified into any single category. FEDL is also an extension of Petri Nets to solve various problems in sensor networks, taking advantage of features in time, color and Stochastic Petri Nets.

3.1. FEDL Petri Net Definition

The FEDL description for an event system in a sensor network can be given as an 8-tuple structure
$F = (P, T, A, \lambda, \delta, \theta, H, L)$ where

- P is the set of all places, which includes places for sensor events S and those for higher level events E, and $P=S \cup E$. We will explain S and E in detail later in the section when we discuss sensor event abstraction. Note that in the FEDL diagram, places are represented as circles with sensor event places in dashed circles.
- T is the set of all transitions, which are represented by rectangle bars in the diagram.
- A is the set of arcs/flows, which are represented as arrows in the diagram. Note that $A= I \cup O$, where I is the set of pre-arcs (incoming arcs to a transition), and O is the set of post-arcs (outgoing arcs from a transition).

To this point, the definition is similar to an ordinary Petri Net. FEDL extends this basic Petri Net model to integrate temporal, spatial and stochastic features.

- λ is the probability / weight function for the arcs $\lambda: A \rightarrow [0,1]$. For example, if f is a post-arc from transition T to state B, then $\lambda(f) = p$ means that after T is fired, with probability p the token enters the state B. For a pre-arc, it can also be viewed that a token goes through a channel, resulting in the capacity of c*p when it goes through this arc. More discussion on capacity will be in Section 3.2. With this function, FEDL takes features from a Stochastic Petri Net and can solve probability-involved problems.
- δ is a time guard function for transitions, $\delta: T \rightarrow \cup^{*} (r1, r2)$, where $r1 \leq r2 \in R$. It means a transition can only fire during the union closure of given ranges. For example $\delta(T) = (a1,a2) \cup (a3, a4)$ means transition T can only be fired during interval (a1,a2) or (a3, a4), where $a1 \leq a2$, $a3 \leq a4$. In particular, δ can also be specified using event incidents. For example $\delta: T \rightarrow \cup^{*} (E1, E2)$ means transition T can only fire between event E1 and event E2.
- θ is the persistency guard for arcs. For an incoming arc (pre-arc) of a transition, $\theta: I \rightarrow R^{+}$, it means the arc can only hold the token to participate in a transition for a certain amount of time before this token disappears. This models the fading process of sensed data. For example, lightening can only last about 1 second, while the existence of some chemical may last for 1 day. Hence, you can deem the arcs as leaking pipes that gradually lose water (tokens). θ can model how fast a pipe is leaking. Mapping the semantics onto a sensor network, a θ value for an arc also refers to the valid interval for the tokens in the place, which stands for a sensor event or a higher level event. For an outgoing arc (post-arc) of a transition, $\theta: O \rightarrow R^{+}$, means how long it takes for this event to happen. We will present an example in the temporal logic section.
- H is the threshold function for places only, $H: P \rightarrow R$. For example H(p) = c means that if a token with a certain capacity wants to enter p, and only if its capacity is over c, it can reach place p. Actually, for a sensor network with continuous values for sensor readings, we can simply set H(p)=0 for all p, and the token capacity indicates how strong sensor readings indicate the occurrence of this event. However, for cases that need a binary state (yes or no), the threshold function is of particular use.
- L is the spatial guard function for transitions, $L: T \rightarrow R^{+}$. For example, assume there are three arcs a, b, and c entering T, and one arc d comes out of T. L(T) = r means T can only fire if the locations of tokens carried

by a, b, and c are within radius r. In fact, after T's firing, the new token will have a merged location based on tokens from a, b, c. The details about token merging and firing are discussed later in the spatial logic section. This function is only one particular spatial guard function to guarantee that the sensed data are within the event radius. In FEDL, we can also generalize spatial guard functions to support more complex spatial constructs.

3.2. Token Representation

Tokens are abstract representations of sensed data or occurrences of events. During transition, the values of certain attributes of a token are updated according to the rules described in the previous section. In FEDL, a token is defined as:

Token = Struct { Type tp; Capacity c; Time t; Location l;}

The semantic meaning of the above notation is as follows:

- Type is used to indicate what data or event a token represents;
- Capacity represents the value of the token, which indicates the confidence of the sensed data to identify the happening of a sensor event.
- Time indicates when this token is created;
- Location represents where this event occurs in the sensor network.

In this way, key aspects of sensor readings are encapsulated into tokens and can be utilized while tokens proceed through a FEDL graph.

3.3. Abstraction of Sensors using Sensor Events

A sensor event is an abstraction of sensors in the network. In FEDL, sensor events are denoted as places which only take tokens from outside (environment). Higher level events are constructed using sensor 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 in the network – temperature, light, and acoustic sensors, then there are three types of sensor events in FEDL: temperature, light, and sound. Because the tokens that arrive at each sensor event are associated with temporal and spatial attributes, therefore, the information when and where the sensed data is taken can be obtained. For example, if a token reaches the temperature sensor event with time stamp t, capacity c, and location attribute (x, y, r), then we can say that a temperature sensor at location (x,y) with sensing range r has a positive value c at time t, as shown in Figure 2.

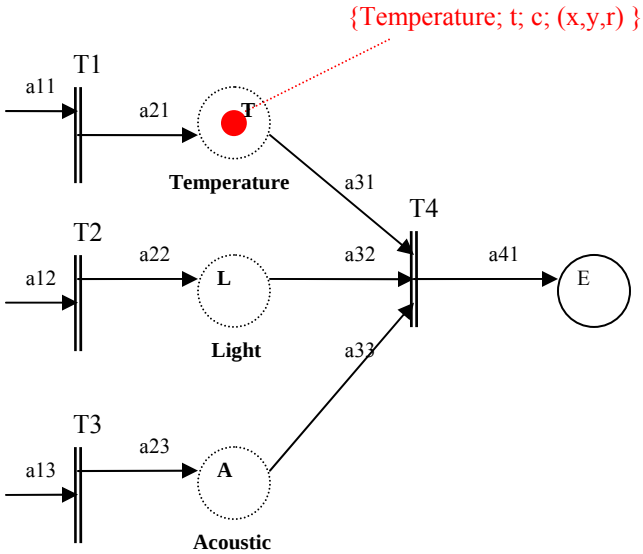

Figure 2: Example of FEDL-depicted Explosion Event Detection System

3.4. Token Path and Firing Rules

We outline the path of a token in FEDL as below:

- A token is initially generated upon receiving sample data from a sensor and encapsulated with the type (sensor event), time (when the data was taken), capacity (the value of the sensor reading) and the location (where the sensor is located and what is the sensing range), as shown by transitions T1, T2 and T3 in Figure 2.
- Tokens in places go immediately through the arcs to get ready to fire transitions, given that firing of a transition should respect all the guard functions.
- A token disappears if the arc carrying it exceeds its persistence value (defined by θ) before the transition can be fired.
- When a token goes through a pre-arc, its capacity is multiplied by the weight on the arc (defined by λ). For example, if $\lambda(a) = p$, when a token with capacity c goes through the pre-arc a, its capacity changes to p*c.
- Before a token enters a place through a post-arc associated with a probability function, the probability of this token entering the place complies with what is specified by the probability function.
- A token can only enter a place if the token's capacity is over the place's threshold value (defined by H).

The token generation and its merging process are performed through transition firing process. We specify the transition firing rules and the token processing procedure as follows:

- A transition T can only be fired if and only if
  - each of its pre-arcs has a positive token,
  - T happens at a time interval satisfying δ.
- During firing, a transition t does the following on the tokens carried by its pre-arcs:
  - Generates a new token for each of the post-arcs
  - The new token has current time as its new timestamp, and the place which it will go next as its new type.
  - The capacity of the new token equals Σpi*ci, where i represents the token from each of the pre-arcs. Note that when the new token is sent to the corresponding post-arc of transition t, its capacity will change again according to λ defined on that post-arc.
  - The location attribute of the new token will become the new center (x,y) and new radius r=L(T). Note that the circle centered in (x,y) with radius L(T) should cover all the original circles of the incoming tokens on the pre-arcs. An approximation algorithm to calculate the new center and radius will be presented in the spatial logic section.

3.5. Structural Logic

Sensor events are the basic entities for composition of higher level events. Events are either atomic events or complex events [29]. Atomic events refer to events which depend only on one sensor event, while complex events depend on multiple sensor events. The example shown in Figure 2 is a complex event, and the example shown in Figure 3 is an atomic one.

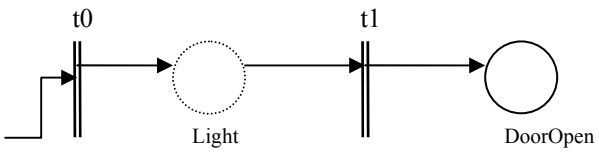

Figure 3: Atomic Event -- Light sensor placed in a dark room to detect if door is open.

To demonstrate that a higher level event is derived from sensor events or sub-events, a transition is constructed between them, and pre-arcs for this transition is from the sensor events and lower level sub-events with the post-arc

pointing to the resulting higher level event. This process is the same as in constructing a place dependent on other places in Petri Nets. Events can be one level events or multi-level events as shown in Figure 4.

In order to provide users with convenience, in FEDL, we not only provide this kind of diagram-based structure construction, but also support a symbol language for specification. For the example in Figure 2, the structure of the event system can be represented as:

S = {T, L, A}

$E = (T4)^{a41}[(T)^{a31} \| (L)^{a32} \| (A)^{a33}]$

Suppose a higher level event E depends on a group of sensor events and lower level sub-events, SE1, SE2, ….., SEn, and transition between them is T, with pre-arcs a1, a2, ….., an, respectively, and the post-arc a. Then the structural logic for this event is:

$E = (T)^{a}[(SE1)^{a1} \| (SE2)^{a2} \| \ldots.. \| (SEi)^{ai} \| \ldots\ldots \| (SEn)^{an}]$

The whole event system can be represented as a list of all the events' structural logic, which means that for $\forall$ e $\in$ E (set of event places), a structural logic for e is defined.

3.6. Temporal Logic

Temporal logic refers to the two temporal guard functions δ and θ. They support to specify the temporal concept "when" and "how long" in FEDL Petri Nets.

- δ guards all the transitions to ensure they fire only during the specified temporal interval. Introducing δ in FEDL has practical importance because some events can only happen during a particular temporal interval in physical environment. For example, some events which depend on sunshine can only happen during the day time.
- θ is the temporal persistency guard for arcs, and it has slightly different meaning for pre-arcs and post-arcs. For a pre-arc, intuitively it reflects how fast the particular sub-event associated with this pre-arc vanishes. For a post-arc, it stands for how long it takes to make the higher level event happen after having satisfied the conditions of its sub-events. A detailed example in Section 4 illustrates the concept.

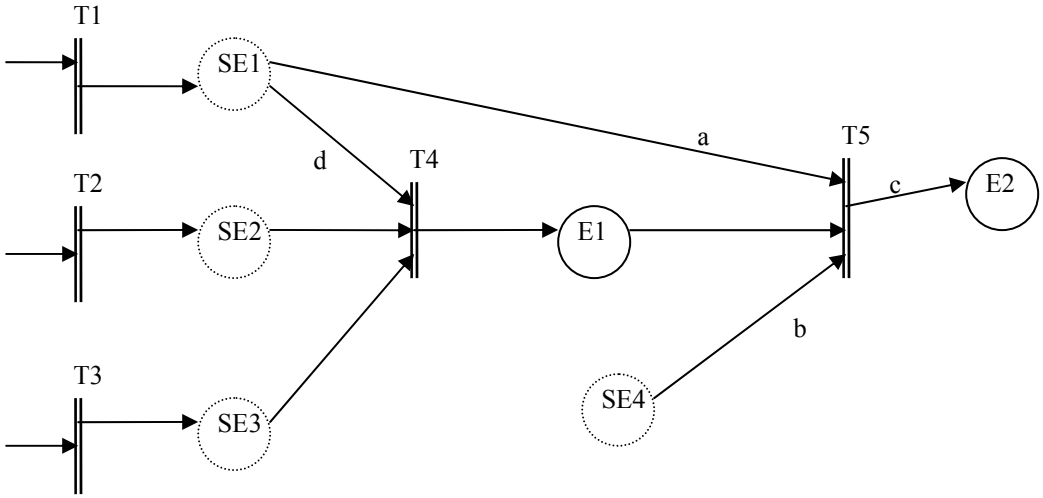

Figure 4: Multi-level Complex Event—E1 has one level, E2 has two levels.

By having these two temporal functions with concepts as "when" and "how long", complicated temporal logic for event systems can be managed in FEDL. More importantly, FEDL can be extended to include other temporal functions in situations where special temporal constraints need to be specified.

3.7. Spatial Logic

A spatial function *L* is defined to enforce the geographic semantics. As a guard function for a transition T, *L* ensures that the tokens carried by T's pre-arcs should satisfy the spatial locality condition. If *L*(T) = *R*, the effective radius of 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 such that the sensor readings from different location can be considered as one particular event.

For example, consider a case in which three sub-events (places) coming into a transition. Suppose 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 as illustrated in Figure 5. 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 overlap each of the three small circles.

In general terms, if *L*(T)=*R*, and transition T has *n* incoming tokens with location attribute *((x1, y1), r1), ((x2, y2), r2), ….., ((xn,yn), rn),* then transition T can satisfy the spatial guard function *L* if

- The circle centered at *(mean(x1,x2,….,xn), mean(y1,y2,….,yn))* with radius *R* should overlap with all the *n* circles, where *Mean(a1, a2, ….., an) = (* Σ*aj)/n,* for $1 \leq j \leq n$. and *Circle(mean(x1,x2,….,xn), mean(y1,y2,….,yn), R)* $\cap$ *Circle(xi, yi, ri)* $\neq \Phi$, where $1 \leq i \leq n$.

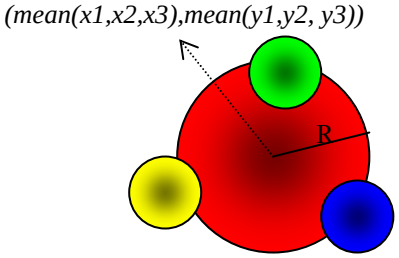

Figure 5: Enforcing Event Locality by Spatial Guard Function

In order to fire, T also needs to satisfy other guard functions. After T is fired, the new token's location is the big circle.

3.8. Probability Logic

The probability / weight function λ and threshold function H provide basic probability control for FEDL. For example, if a sensor network has three types of sensors to support a collaborative decision on detection of a fire event [29], the probability function can be used to specify the different weights of each type of sensors in the decision. In such scenarios, we can assign higher weight to a particular sensor type if the data reported by that type of sensor is more reliable.

The probability logic of FEDL also supports a time-related probability model when the functions are associated with time. In addition, FEDL is extensible to provide more complicated marking dependent probability functions, which can employ advanced features of Stochastic Petri Nets.

4. An Example – FEDL for Mine Monitoring Sensor Network

4.1. Scenario

In order to illustrate the power of FEDL, we present a specific sensor network application with a detailed FEDL description. Suppose we have a discarded underground coal mine that has only a negligible amount of oxygen, but can generate plenty of flammable gases such as methane, and significant CO_2 as well. The mine is big and has some holes at the ground surface levels which results in some sunshine through the holes. The local government is con-

cerned with these underground mines because there are reports that some explosions happen there which can affect nearby villages. Through analysis, the major cause of these explosions are due to some green moss (a kind of botanic plant that can survive very harsh environments) growing in the mine. When the sunshine (gets through the holes on top of the mine) reaches them, they can produce oxygen due to the photosynthesis process. However, it is fatal to mix enough oxygen with methane when there is even a tiny spark of fire (a natural spark is likely when a rock falls off and hits another rock), which can result in a significant explosion.

Certainly, in such a harsh environment, we cannot place people there to monitor or control the whole situation. In fact, it is a good application to set up a wireless sensor network to monitor the mine and send critical information in a real-time fashion before an explosion occurs. Note that we do not want to use oxygen sensors directly to detect O_2, because if we can detect O_2, it might be already too late. What we want to do is to give an alert before enough of oxygen can be mixed with methane by monitoring the photosynthesis process and methane containment. We use FEDL to specify the event system for such application.

Assume there are six types of sensors in the network. They are:

1. Biological sensors to detect the existence of green moss;
2. Light sensors to detect sunshine;
3. CO2 chemical sensors to detect the presence of CO2;
4. Methane chemical sensors to detect high concentration of methane gas;
5. Temperature sensor to detect sudden generated heat;
6. Very sensitive light sensors to detect sparks of light.

4.2. FEDL Description for Mine WSN

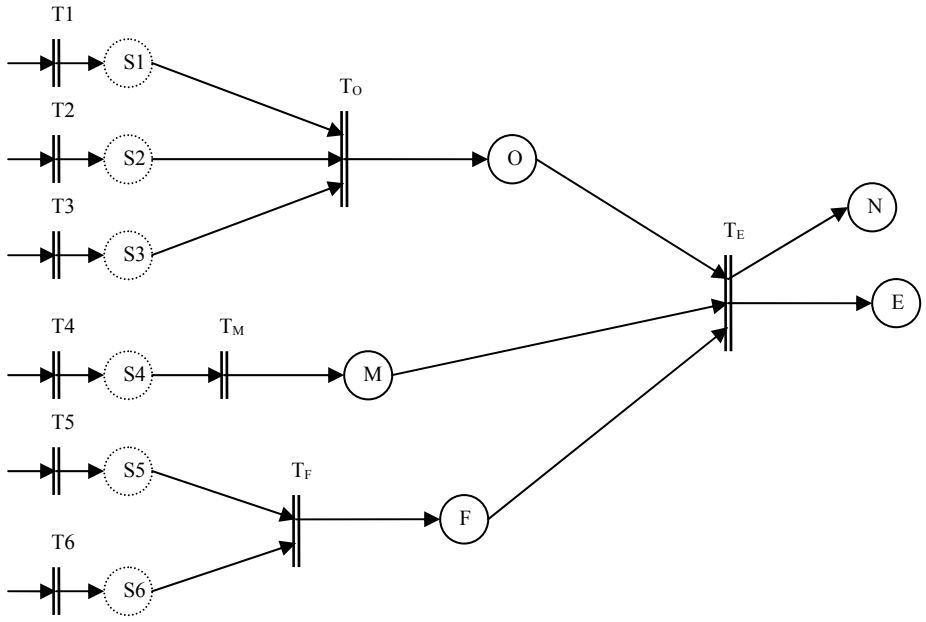

Figure 8: FEDL Diagram for the Mine Explosion Monitoring WSN

The FEDL structure for the system described above is an 8-tuple
MINE = (P, T, A, λ, δ, θ, H, L) Where

- **P** is the set for all the places (circles), P = {sensor events} ∪ {logical events} = { S1, S2, S3, S4, S5, S6} ∪ { O, M, F, E, N}, and hence P= {S1, S2, S3, S4, S5, S6, O, M, F, E, N},

Where S1 to S6 represent the six types of sensor events, and O, M, F, E, N stand for presence of Oxygen, high concentration of Methane detected, presence of sparks of Fire, mine explosion and non-explosive chemical process, respectively.

- **T** is the set of all transitions, T = { T1, T2, T3, T4, T5, T6, T_O, T_M, T_F, T_E}
- **A** is the set of all the arcs, and we use $A_{i,j}$ to represent an arc coming from place/transition i to place/transition j.

$$
\begin{aligned}
A = \{ & A_{,T1}, A_{,T2}, A_{,T3}, A_{,T4}, A_{,T5}, A_{,T6}, \\
& A_{T1,S1}, A_{T2,S2}, A_{T3,S3}, A_{T4,S4}, A_{T5,S5}, A_{T6,S6}, \\
& A_{S1,To}, A_{S2,To}, A_{S3,To}, A_{S4,TM}, A_{S5,TF}, A_{,S6,TF}, \\
& A_{To,O}, A_{TM,M}, A_{TF,F}, \\
& A_{O,TE}, A_{M,TE}, A_{F,TE}, \\
& A_{TE,E}, A_{TE,N} \\
& \}
\end{aligned}
$$

- The probability model is rather complex, which demands the expertise of developers to have deep understanding in the event cause and the processes to give proper weights. Usually this task belongs to the environmental scientists who have such knowledge. For this example, weight /probability function $\boldsymbol{\lambda}$ is assigned as follows:
  All the arcs coming to sensor events have λ=1, and the pre-arcs to T_O have a weight function of 1/3 each, which indicates the three types of sensors S1, S2, S3 have the same weight in determining event O. The weights to be assigned to the pre-arcs to T_F should be determined by the probability of detecting fire. Since the spark of light is more likely to lead to fire, we assign a bigger weight to $A_{,S6,TF}$ compared to that of $A_{S5,TF}$. All three sub-events (O, M, F) should occur to have a transition T_E to fire, and all three pre-arcs to T_E have the same 1/3 weight assigned. After T_E is fired, there is a small probability that it will not lead to an explosion. For the two post-arcs, $A_{TE,E}$, $A_{TE,N}$, we specify that the probability of the transition going to state N is 0.1 and the transition going to E is 0.9. The assignment means that when the conditions of T_E are satisfied, it either goes to non-explosive chemical process with 10% probability, or result in explosion with 90% probability.

- $\boldsymbol{\delta}$ is the time guard function for each transitions in T. $\delta : T \rightarrow (a1,a2)$ means that transition T can only be fired during the time interval (a1, a2), where a1<=a2. In order to filer out the sensor reports which claim a sunshine event at night time, we define the $\delta(T_O)$ to be the time interval that the sunshine is likely to appear. In this way, some unlikely false alarms can be filtered out. For other transitions, we can leave δ unspecified, which results in the default value $(-\infty, +\infty)$.
- $\boldsymbol{\theta}$ is the persistency / delay function for arcs. In this example, we specify θ in seconds as follows:
  - The existence of green moss can last very long, we give a big value 1000000 to $\theta(A_{,T1})$.
  - Since gas such as CO2 and oxygen persist to a certain degree, we assign 80 to both $\theta(A_{,T3})$ and $\theta(A_{,T4})$.
  - The sudden heat and sudden spark of light do not persist, and hence we give small values 2 and 0.3 to $\theta(A_{,T5})$ $\theta(A_{,T6})$ respectively.
  - For a post-arc, θ represents delay from transition to reach of the state. In the example, $\theta(A_{TO,O})=60$ indicates that after conditions of a photosynthesis process is satisfied, it still takes 60 seconds to generate oxygen.
  - θ function for other arcs can be specified according to the nature of the corresponding events, and usually these information are given by personnel with expertise in the application.
- ***H*** is the threshold function for each place. $H: P \rightarrow v$ means for a place p, $v=H(p)$, and if the capacity of the token at place p is over v, then the token can enter the place p. It works as a flag function, which indicates that the event has happened because token capacity is over the threshold. The definition of H for this particular scenario is straightforward, since the weights assigned are normalized. We set $H(p) = 1.0 \text{ for } \forall p \in P$.

- ***L*** is the event size guard function as introduced in Section 3 and it is defined as below in the example:

- For sensor events T1, T2, T3, T4, T5, T6, the effective radius of the events are their sensing ranges.
- For events O and E, since the gas involved are pervasive, the effective event radius is much bigger than event F. Event F can only have a small effective spatial radius since a spark of fire is very small in size. Therefore, we specify $L(T_O)=30$ and $L(T_F)=1$ to specify the spatial characteristics of the events.

4.3. FEDL-Aided Analysis for Mine Application

To demonstrate the power of FEDL, we present a few analyses using the application. With FEDL, we can identify when and why the sensor network may not work as expected. In addition, we can tune the parameters specified to achieve different QoS requirements of the system. In the following sections, we discuss several analyses in detail.

4.3.1. Safe Time to Send Personnel

The local government wants to send personnel into the mine to remove the green moss and/or to perform certain geologic investigations. However, because of possible explosions, it is highly desirable to know in advance when it is safe to enter and the maximum possible safe time interval. Before the sensor network is deployed, this information cannot be determined. From the detected events reported by the sensor network, together with the analysis capabilities of FEDL, we can compute safe intervals.

The events reported by the sensor network at the current time can be reflected by a specific marking for the FEDL Petri Net. For example, consider the case that the sensor network reports event M has just happened, while no sensor events are detected for S1, S2, and S3. The corresponding marking for FEDL is illustrated in Figure 9. In this particular marking, only place M has a token with all other places empty. In this case, through analyzing the reachability delay in different paths to an explosion event, we can conclude that there has to be at least $\max(\theta(A_{TO,O}) + \theta(A_{TE,E}), \theta(A_{TF,F}) + \theta(A_{TE,E}))$. This time is $\theta(A_{TO,O})$ (the time 60 mins to photosynthesize to produce oxygen) plus $\theta(A_{TE,E})$ (the time 0.05 to generate explosion on having O2 + Methane + Fire), which is in total 60.05 minutes. This shows that when the sensor network has detected a methane event, without detecting other events at all, then it is safe to work in the mine for at least 60.05 minutes before a possible explosion can happen.

When the sensor network detects different events, corresponding marking states are indicated in FEDL, and different safe time periods can be computed. For example, suppose there is only an oxygen event reported in the sensor network (reflecting a token in O in the FEDL marking), then the safe time period to work in the mine is $\max(\theta(A_{TM,M}) + \theta(A_{TE,E}), \theta(A_{TF,F}) + \theta(A_{TE,E})) = 5 + 0.05 = 5.05$. It means if the sensor network produces such a report, then the guaranteed time interval during which no explosion can happen is only 5.05 minutes. Therefore, if anyone is in the mine at that time, she should be evacuated.

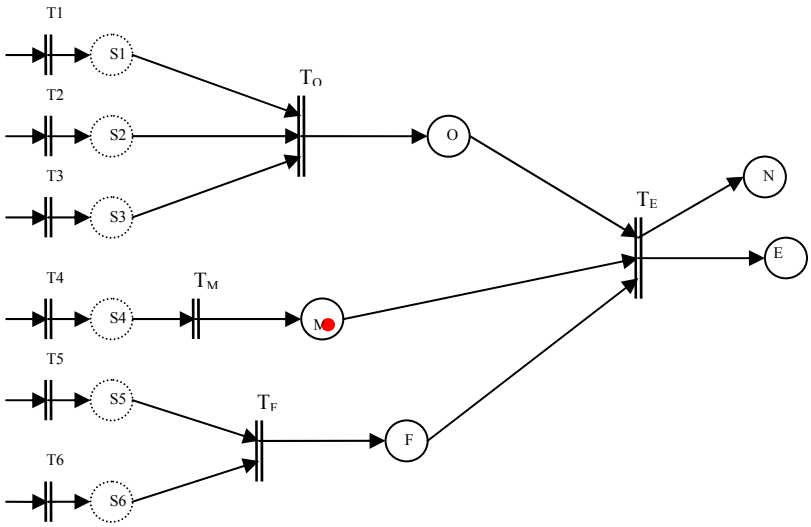

Figure 9: Sensor Network Status Reflected by a Specific FEDL Marking

Generally speaking, this kind of analysis relies on the analysis ability of Petri Nets to compute the delays on different paths to a particular event (a target marking) given a specific marking. This feature allows us to investigate many potential markings to assess various possible system behaviors, even before the system deployment.

4.3.2. FEDL As a Gauge to Monitor WSN Behavior

Because of the unreliable and power-constrained nature of sensor devices, there can be some unexpected behavior of the system. For example, if sensors in a certain area die out, then the events happening in that area can not be detected. Consider the case of a report of an oxygen event. However, before receiving the report, the sensor network has not reported the detection of green moss. From the system specification using FEDL, an oxygen event can only happen if there has been a green moss event as one of its necessary triggers. Hence, we can tell that something has gone wrong in the sensor network, because it is not working as expected.

4.3.3. Tuning System Parameters to Satisfy QoS Requirement

The local government is satisfied with the deployment of a sensor network in the mine because it never misses reporting a real explosion event. However, the rescue squad officers complain that though the sensor network never misses a real explosion, it generates many false alarms. It turned out that the rescue squad is dispatched to the mine after every reports of explosion event, and only 60% of the time there has been a real explosion.

This complaint can be considered as a QoS requirement from the users. The rescue squad requires a false alarm rate lower than the current 40%. The computer scientists working on the project are confident that it has nothing to do with the network itself, since the sensor network has been tested recently and is working as specified by FEDL. Then the high false alarm rate might be caused by improper system parameters specified in FEDL by the environmental scientists. Hence, a debugging scheme to identify the improper system parameters is developed:

- Set 1 month debugging period for the sensor network, during which each sensor's data are sent back to the base station, with all the event reports.

After collecting all data, the analysis process on FEDL works as follows:

- Group all the sensor readings within a local temporal interval for each false alarm. For example, if a false alarm is at 1pm Dec 1st, let t = 1pm@Dec1st, then

[t - max(θ_path), t] would be considered a local time interval, because all the sensor readings that can lead to this false alarm are included. Max(θ_path) refers to the sum of delay on a path from initial sensing transition to the higher level event which is falsely reported.

- Similarly group the sensor readings within a local interval for each real alarm.
- Transform the sensor readings in each group into tokens, and all the tokens from a group form a sample space. As illustrated in Figure 10, the token dispatcher is a transition, which dispatches tokens to their corresponding places (according to the kind of sensor reading) over time (according to the reading's timestamp). By using the token dispatcher, streams of tokens that represent all the sensor readings in the network can be input into FEDL for the MINE system.
- The major purpose of this heuristic scheme is to adjust the system parameters in FEDL to guarantee that the sample space for a real alarm still generates alarms in FEDL, while the sample space for false alarms do not generate alarms or do not generate false alarms as often as before. Thus the new parameters can reduce false alarm rates.
- The FEDL tool can also collect statistics on the components when the Petri Net is running. In order to find a better system parameter to reduce false alarm rates, computer scientists pre-set some statistical metrics when running different samples. These metrics include *average distance* among tokens for transition and *average waiting time* for a specific type of token for each transition.
- Through running 40 sample groups of false alarm and 60 groups of real alarms, computer scientists find that for transition T_E, the average token distance for false alarm is much bigger than the average token distance in real alarms. For false alarms, the average distance of tokens for T_E out of 40 runs is 28, which is quite close to the upper bound defined by $L(T_E)$. On the other hand, the average token distance for real alarms out of 60 runs is only 12, which is less than half of $L(T_E)$. The average waiting time for tokens is very similar for both false alarms and real alarms.
- By this observation, computer scientists conjecture that the high value of $L(T_E)$ may lose the spatial guard for events, thus resulting in higher false alarm rates. Therefore, they change $L(T_E)$ to 15, and run the 100 sample analysis again. The results are that 30 out of 40 false alarms are not reported during this FEDL analysis run, while 5 out of 60 real alarms are not reported either.
- This analysis shows that false alarm rate can be reduced by tuning down the spatial guard in this case; however, it also risks missing reports of real alarms. Therefore, FEDL analysis can provide an analytical testing environment for users to trade off between different metrics. If this tradeoff is not acceptable, then other changes to the system should be tried.

Sample-specific analysis in FEDL is different from simulation study in the sense that FEDL needs no assumption on the system architecture, the protocols, and the data fusion approach employed in the sensor network to do analysis on event occurrences. With massive distributed sensed data input, FEDL can quickly determine the occurrence of higher level events from the low level sensing data.

4.3.4. Analytical Testing Environment for System Feasibility

Assume that the sensors used in the MINE system are very expensive and currently the network density is high. Because there is another mine that also needs to set up a sensor network, the authority wonders if it is possible to reduce the density of sensors in the first mine, while keeping the current performance of the sensor network working correctly according to the specification. However, physically testing this idea is costly as well as dangerous in the situation like this.

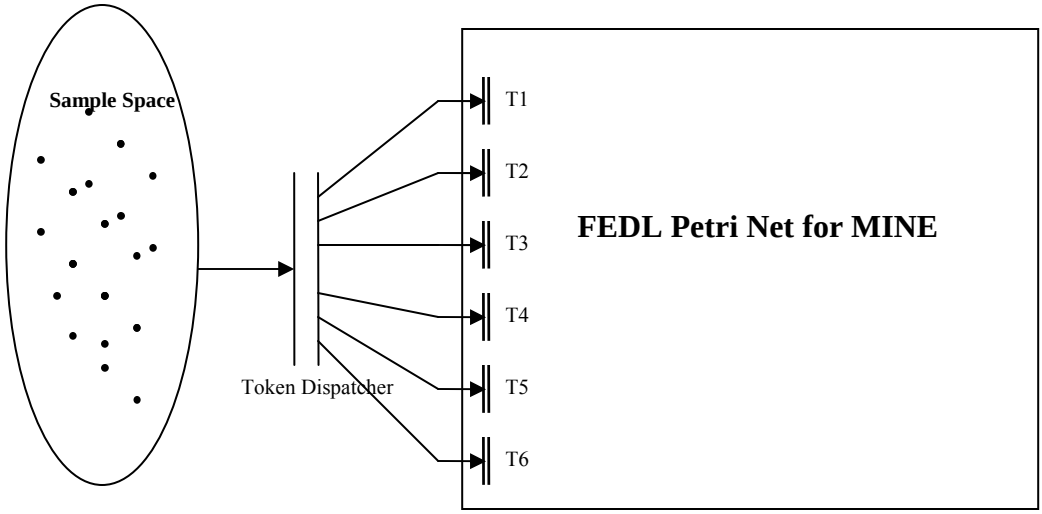

Figure 10: Sample Specific Analysis in FEDL

A sample-specific analysis approach using FEDL is designed to tackle the problem:

- They simulate event occurrences in FEDL system, and in order to make the simulation close to reality, they use the 60 groups of real alarms collected before.
- For each sample space, remove tokens by the geographical attribute, which simulates the process of removing sensors from the network uniformly. Then run FEDL analysis based on this sample space again to check if real alarms are still reported in FEDL.
- After all 60 runs, if the percentage of missing reports is satisfactory, then the newly reduced density is acceptable. Otherwise, the current acceptable density is retained.

5. FEDL Tool Implementation

FEDL has been implemented using GME© [28] toolkit. GME is a meta modeling environment which provides a convenient GUI (Graphical User Interface) interface for end users. FEDL tool exploits this feature of GME, by building a FEDL meta model in GME environment. After FEDL model registration and interpretation, end users are ready to model any sensor event system using the GUI provided.

In the FEDL environment, the users can manipulate entities such as circles (places), arrows (flows or arcs) and bars (transitions) to construct FEDL Petri Net for any sensor network event system, as is illustrated in Figure 11. Since each place, arc and transition in a FEDL Petri Net can have a set of properties, in FEDL implementation, we associate each entity with a set of properties such as threshold, capacity, and types. These properties can be modified easily on the browser window, which makes construction and future analysis process easier. With model checking support from pre-defined FEDL meta model, we can perform model checking for a FEDL Petri Net to verify the validity of a user-defined FEDL Petri Net.

For the analysis part, we have currently implemented reachability analysis module in FEDL. This module is a basic analysis model which reports if a certain event (a place in FEDL Petri Net) can be reached with a given time period. This type of analysis is very useful to do temporal property estimate for the system as illustrated in Section 4.

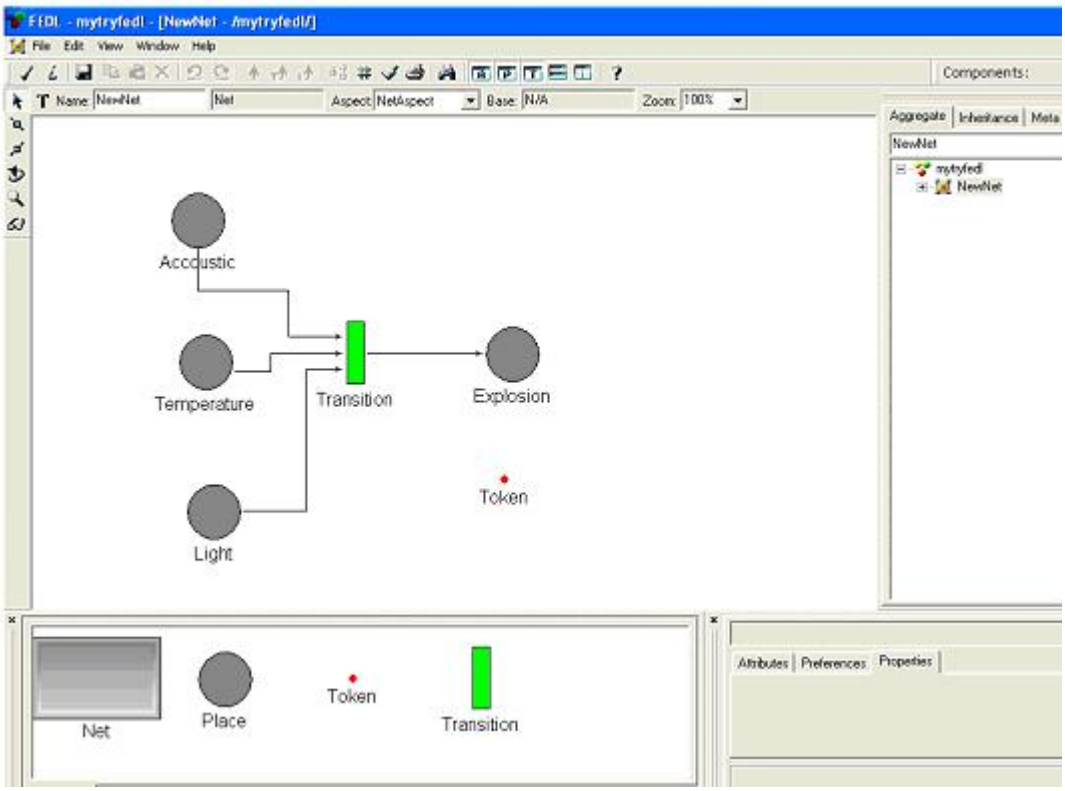

Figure 11: Drawing a FEDL Petri Net

6. Conclusion and Future Work

In this paper we have presented FEDL, a formalized event description language and its tool specially designed for event-based sensor network systems. FEDL is more appropriate than SQL in formally depicting events in sensor network systems owing to its powerful underlying Petri Net model. Essential features in sensor networks such as various sensor types, geographical locations of sensors, temporal constraints and probability of events are modeled. FEDL establishes a clear-defined interface which interprets event expertise knowledge into sensor network semantics. In addition to formally describing an event-based system, FEDL can be used as an analysis tool for both system design purpose and system debugging purpose.

In the future, we will provide event composition algebra in temporal, spatial and probability dimensions with cross-dimension dependencies and interactions. In that way, FEDL can manipulate more complicated scenarios in sensor network applications. In addition, we plan to model power consumption and communication in FEDL to tackle these important issues in sensor networks.

7. Acknowledgement

The authors appreciate Prof. A. Buchmann for suggestions made to the earlier version of the paper. This work was supported, in part, by NSF grants IIS-0208758 and CCR-0329609, and DARPA grant F33615-01-C-1905.

References

[1] M. Adamou, S. Khanna, I. Lee, I. Shin, S. Zhou, Fair Real-time Traffic Scheduling over A Wireless LAN, In Proceedings of the 22nd IEEE RTSS 2001, London, UK, December 3-6, 2001

[2] Fulvio Babich and Lia Deotto. "Formal Methods for Specification and Analysis of Communication Protocols," IEEE Communications Surveys & Tutorials, 2002.

[3] J. Bengsston, K. Larsen, F. Larsson, P. Pettersson, W. Yi, and C. Weise. New generation of uppaal.In Proceedings of International Workshop on Software Tools for Technology Transfer, 1998.

[4] T. Bolognesi and E. Brinskma, "Introduction to the ISO Specification Language LOTOS," Computer Networks and ISDN Systems, vol. 14, Apr. 1987, pp. 92–100.

[5] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the Physical World", *IEEE Personal Communication Magazine*, (7): 10-15, Oct 2000.

[6] A. Buchmann, J. Zimmermann, J. Blakeley, D. Wells. "Building an Integrated Active OODBMS: Requirements, Architecture and Design Decisions," *ICDE 95,* Taipeh, Feb. 1995.

[7] A. Cerp, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. "Habitat Monitoring: Application Driver for Wireless Communication Technology." *In ACM Sigcomm Workshop on Data Communication,* San Jose, Costa Rica, April 2001.

[8] S. Chakravarthy, V. Krishnaprasad, E. Anwar "Composite Events for Active Databases: Semantics, Contexts and Detection", *VLDB 94,* Santiago de Chile, Sept. 1994.

[9] Cougar Project. www.cs.cornell.edu/database/cougar.

[10] R. David, H. Alla. "Petri Nets for Modeling of Dynamic Systems – A Survey", Automatica, 30(2), 1994.

[11] U. Dayal, A. Buchmann, D. McCarthy. "Rules are Objects Too: A Knowledge Model for an Active, Object-Oriented Database System," *Advances in Object-Oriented Database Systems,* Bad Muenster am Stein, LNCS 334, Sept. 1988.

[12] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. "*Practice of Petri Nets in Manufacturing*." Chapman & Hall, 1993.

[13] H. El-Gendy and H. Baraka, "Transformation of LOTOS Specifications to Estelle Specifications," Proc. IEEE Symp. Com-puters and Communications, 1997, pp. 215–20.

[14] J. Ellsberger, D. Hogrefe, and A. Sarma, SDL Formal Object Oriented Language for Communication Systems, Prentice Hall, 1997.

[15] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *In Proceedings of the 5th Annual International Conference on Mobile Computing and Networks,* Seattle, WA, 1999.

[16] Michael Franklin. "Declarative Interfaces to Sensor Networks", *Presentation at NSF Sensor Workshop,* Los Angeles, CA, Feb, 2004.

[17] N. Gehani, H. Jagadish, O. Shmueli. "Specification in an Active Object-Oriented Database," *SIGMOD 92*, June 1992.

[18] H. J. Genrich and K. Lautenbach. "System Modelling with High-level Petri Nets". *Theoretical Computer Science*, 13:109-136, 1981.

[19] C. Girault, R. Valk. "Petri Nets for System Engineering. A Guide to Modeling, Verification and Applications", Springer, New York 2003.

[20] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. "The Sensor Network as a Database." Technical Report 02-771, Computer Science Department, University of Southern California, Sept 2002.

[21] T.A. Henzinger, P. Ho, and H. Wong-Toi. HyTech: the next generation. In Proceedings of the 16th IEEE Real-Time Systems Symposium, pages 56–65, 1995.

[22] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. "System Architecture Directions for Networked Sensors," *Architectural Support for Programming Languages and Operating Systems,* pages 93–104, 2000.

[23] G. Holzmann, Description and Validation of Computer Protocols, Prentice Hall, 1997.

[24] C. Jaikaeo, C. Srisathapornphat, and C-C Shen. "Querying and Tasking in Sensor Networks", In Proceedings of SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V), Orlando, FL, 2000.

[25] M. Jantzen and R. Valk. "Formal Properties of Place/Transition Nets". In W. Brauer, editor, *Net Theory and Applications*, volume 84 of *Lecture Notes in Computer Science*, pages 165-212. Springer-Verlag, 1980.

[26] K. Jensen. "Colored Petri Nets and the Invariant-Method". *Theoretical Computer Science*, 14:317-336, 1981.

[27] L. M. Kristensen, S. Christensen, and K. Jensen, "The Practitioner's Guide to Coloured Petri Nets," Int'l. J. Software Tools for Technology Transfer, vol. 2, Springer Verlag, 1998, pp. 98–132.

[28] A.Ledeczi, et al. "Metaprogrammable Toolkit for Model-Integrated Computing." *In Proceedings of the IEEE ECBS'99 Conference,* 1999.

[29] S. Li, S. H. Son, and J. A. Stankovic, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks,", *2nd International Workshop on Information Processing in Sensor Networks (IPSN'03),* Palo Alto, CA, April 2003.

[30] C. Liebig, M. Cilia and A. Buchmann. "Event Composition in Time Dependent Distributed Systems," *4th International Conference on Cooperative Information Systems*, Edinburgh Scotland, Sept 1999.

[31] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. "The Design of an Acquisitional Query Processor for Sensor Networks," In Proceedings of ACM SIGCMOD, San Diego, CA, June 2003.

[32] José Meseguer "Software Specification and Verification in Rewriting Logic," Lectures at the Marktoberdorf International Summer School, Germany, 2002.

[33] S. Nittel, A. Stefanidis, I. Cruz, M. Egenhofer, D. Goldin, A. Howard, A. Labrinidis, S. Madden, A. Voisard, M.Worboys. "Report from the First Workshop on Geo Sensor Networks", *First Workshop on Geo Sensor Networks,* Portland, Maine, October 9-11 2003.

[34] A. Ohta and K. Tsui, "Turing Machine Equivalence of Time Asymmetric Choice Nets", In IEICE Transactions on Fundamentals in Electronics, Communications and Computers, Vol. #83-A. No. 11, pages 2278-2281. 2000.

[35] P. C. Ölveczky and J. Meseguer. "Specification and Analysis of Real-Time Systems Using Real-Time Maude," FASE 2004, pp. 354-358.

[36] P. C. Ölveczky and J. Meseguer. "Specification of real-time and hybrid systems in rewriting logic," In Theoretical Com-puter Science, Vol. # 285, 2002.

[37] C. A. Petri. "Kommunication Mit Automaten." *Disseration, Technische Universitat Darmstadt*, 1962.

[38] S. Schwiderski. "Monitoring the Behaviour of Distributed Systems", *PhD Thesis, Computer Lab, University of Cambridge*, June 1996.

[39] C-C Shen, C. Srisathapornphat, and C. Jaikaeo. "Sensor Information Networking Architecture and Applications," *IEEE Personal Communication Magazine, 8(4): 52-59*, Aug 2001.

[40] ITU-T, Recommendation Z.100 (11/99) Specification and Description Language (SDL).

[41] ITU-T, Recommendation Z.120 Message Sequence Chart (MSC), Nov. 1999.

[42] M. R. Tremblay and M. R. Cutkosky. "Using Sensor Fusion and Contextual Information to Perform Event Detection during a Phase-Based Manipulation Task." *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* Pittsburgh, PA, 1995.

[43] N. Viswanadham and Y. Narahari. *"Performance Modelling of Automated Manufacturing Systems."* Prentice Hall, Englewood Cliffs, NJ, 1992.

[44] M. Worboys, "Event-based Models of Geosensor Networks," *Invited Talk, First Workshop on Geo Sensor Networks,* Portland, Maine, Oct, 2003.

[45] S. Yan, S. Wang, and Z. Dou. "An Energy Model in Fire Detection and Integrated Analysis on False Alarms," *In Proceedings of the 12th International Conference on Automatic Fire Detection,* Gaithersburg, MD, 2001.

[46] S. Yang, S. Chakeavarty. "Formal Semantics for Composite Events in Distributed Systems," *ICDE 99*, Sydney, Australia, March 1999.

[47] http://www.daimi.aau.dk/designCPN/

[48] Hans Vangheluwe. "State Automata Limitation and Extension," *Lecture Notes in Modeling and Simulation*, Computer Science McGill University, Canada.

[49] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. "The tool kronos. In Hybrid Systems III: Verification and Control," *LNCS 1066*, pages 208–219. Springer-Verlag, 1996.

[50] K. G. Larsen, P. Pettersson, and W. Li, "UPPAAL in a Nutshell," *Springer Int'l. J. Software Tools for Technology Transfer*, vol. 1+2, 1997