Complex Event Processing for Object Tracking and Intrusion Detection in Wireless Sensor Networks

Bhargavi. R and V.Vaidehi

***Abstract*—Complex Event Processing (CEP) has received wider acceptability due to its systematic and multilevel architecture driven concept approach. CEP is an emerging technology in the field of data processing and identifying patterns of interest from multiple streams of events. High levels of integrated self learning applications can be developed. CEP is used in development of applications which have to deal with voluminous streams of incoming data with the task of finding meaningful events or patterns of events, and respond to the events of interest in real time. In this paper a CEP based application for object detection tracking in a Wireless Sensor Network (WSN) environment is proposed. Also the detection of an intruder using semantic query processing is proposed. ESPER, an open source Complex Event Processing engine is used to develop the application.**

***Index Terms*—CEP, Wireless Sensor Network, Semantic Query, Intrusion, ESPER**

I. Introduction

Complex Event Processing (CEP) provides comprehensive solution for building applications to filter, correlate and process events in real-time so that downstream applications are driven by true, real-time intelligence. Any phenomenon happening (or contemplated as happening) in the real world can be treated as an event and normally it is of interest to some group of people. A key stroke, a sensor outputs a reading etc., are couple of examples of an event. A stream is an (almost) infinite sequence of linearly ordered events i.e. {data-tuple, timestamp} pairs. The technology to perform operations on events, (including reading, creating, transforming or abstracting and deleting), out of the event-stream-data is known as Event Processing Technology. A complex event is an abstraction of simple events. CEP primarily focuses on event processing that deals with the task of processing multiple and complex characteristics events with the specific purpose of sourcing and identifying the meaningful stream of events within the defined event cloud. CEP is the crucial key technology element in event processing. CEP allows applications to identify and apply real-time intelligence to streaming data, by making it easy to identify "complex" sequences of events (e.g. "A followed by B, then C") with temporal (e.g. "within 5 seconds") or spatial (e.g. "within 5 miles") constraints. CEP uses techniques such as scanning and detection of complex patterns of many complex events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes. CEP can be used in number of applications like security & surveillance of defense department to detect an unauthenticated object/person in a secured region, inventory tracking, financial trading, etc.

A wireless sensor network is the environment of spatially distributed autonomous sensors, which are simultaneously monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or intruders. Initially Wireless Sensor Networks applications are specially developed for military purposes such as battlefield surveillance [8]. WSN applications are now found in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control.

A sensor network has proven to be useful in cases where automation is required for monitoring and processing of the events. A sensor is a small, low-power piece of hardware with wireless communication and simple processing. Sensor networks are composed of large number of sensor nodes, which are densely deployed either inside the phenomenon they are observing, or very close to it. Sensor nodes can be used for continuous sensing and event detection. The events captured by all the sensor nodes are sent to the base station and then routed to the server via gateway. Thus the sensor networks provide the information about the current events that are happening in an on-line, real-time fashion. These data streams generated by the sensor networks have different characteristics from the traditional data processing [1]. In the server these event streams need to be processed to identify some patterns and take actions if necessary. The events arriving at the server are voluminous and no proper sequence. Processing these continuous event streams in real time to identify the patterns among them is a Herculean task. Conventional process oriented control flow software architectures do not explicitly target the efficient processing of continuous event streams.

In this paper we propose a CEP based solution for object tracking and intrusion detection in a wireless sensor network environment. The objective of the current work is to detect and track an object (here object is considered to be a person) using a multiple sensors (PIR, RFID & Camera). Rest of the paper is organized as follows: Section II gives the related study. Section III explains the proposed model and the implementation. Section IV gives the results of the experiment and section V has the conclusion.

Manuscript received September 30, 2010; revised January 10, 2011.

The authors are with Department of IT, MIT, Anna Univerisity, Chennai,India (email: bhargaviren@gmail.com; vaidehivijay@gmail.com).

II. Related Study

Wireless sensor networks applications like battlefield warfare strategic warning/defense, target acquisition, surveillance, etc., demand automated situation awareness, threat assessment, and response generation. JDL [8] multi source Fusion model is a well known research model. The objectives of data fusion in the JDL model extend beyond merely merging data; the purpose is to achieve dynamic understanding and awareness of situation – including threat impact – and enable adaptation of plans and processes to compensate for such situational threat. The JDL Fusion model first established by the Joint Directorate Laboratories is a functional model. It paints functional goals for fusion at four levels, from raw data collection (Level 0) through entity understanding (Level I), situation awareness (Level II), impact/threat understanding (Level III), and finally re-assigning resources and processes to address gaps between plan/capabilities and threat (Level IV). Data mining and artificial intelligence techniques have been used to achieve the realization of JDL Fusion functional model. Though JDL gives the frame work for multi source data fusion using different levels, the integration of CEP in this frame work has not been proposed in the literature (to the best of our knowledge). In this paper CEP technology is proposed for object detection and tracking. The result of this is used for situation awareness, threat assessment and response generation.

Recently, event-driven architectures (EDA) have been proposed as a new paradigm for event-based applications [2] where streams of events have to be processed. Since wireless sensor networks transmit the sensed data, they can be considered as event-driven systems and hence CEP can be applied in this framework [1]. The idea lies in the processing of events as the central architectural concept. The key concept is to use Complex Event Processing as the process model for event-driven decision support. As mentioned earlier CEP is a technology to process events and discover complex patterns among multiple streams of event data. Through filtering, grouping, aggregating and constructing complex event, Complex Event Processing Engines provides a more meaningful way of identifying events of interest and take necessary action and pave way for complete system automation.

The objective of the research is to develop a system that collect data from numerous sources about raw events on an ongoing basis and apply rules to determine in real time the interconnected trends and patterns that combine them into complex events. A series of queries could look for a pattern of raw events that represents an opportunity, problem, or threat, and pass that alert to a downstream application or person.

The object tracking system using multiple sensors provides a good foundation for a security application and shows the possibilities of sensor networks in this field. In the current research an RFID reader along with a PIR sensor and a camera is used as the sensing unit. Sensor node is used to identify objects and also provide context environment information of these objects. The data or the events generated by the sensor nodes will be sent to the sink (server) where they are filtered, aggregated, processed and analyzed to find the events of interest. The events of interest are given to the system in the form of rules. These rules work on the sensor data and identify the complex events that are of interest in a semantic way. ESPER, a JAVA based tool is an open source Complex Event Processing engine [5] and is used to develop the application. The ESPER engine works a bit like a database turned upside-down. Instead of storing the data and running queries against stored data, the ESPER engine allows applications to store queries and run the data through. The CEP engine gives responses whenever the conditions occur that match queries. The execution model is thus continuous rather than only when a query is submitted.

III. Proposed CEP Based Object Detection/ Tracking System For WSN

Sensor nodes consisting of any proximity sensor like PIR sensor, an RFID reader and a camera are spread throughout the area to be monitored. These sensor nodes detect events (movements of a person). The PIR sensor detects a person in its proximity and at the same time it also triggers the camera to capture the image. The RFID reader on the sensor board reads the RFID information whenever some person wearing a tag comes in to proximity of the reader. Once an event is detected by the node, it generates packet containing the sensor ID, RFID, PIR reading, time and other useful information. Sensors communicate wirelessly to the sink through the best possible route based on the routing algorithm. The captured images are sent to the server in a wired network. Thus the data from the sensor nodes reach the server using WSN and also wired network. Since nodes are widespread and events are random and multiple, the detected data are voluminous. This data continuously streams into the server. The application demands both online and offline processing. Complex Event Processing technique performs the online tracking. CEP takes care of arising alerts in situations that need attention.

A packet is formulated from the data observed from PIR sensor and RFID reader and it is wirelessly routed to the server via the gateway through sink. Images captured from the camera are also sent to the server along with time and sensor ID information. Each wireless packet gives the information about one event detected by one sensor. The information contained in the packet is decoded and given to the event processing module. The images are stored in the database for further processing for intrusion detection.

Object tracking using Complex Event Processing [1] with multiple sensors consists of three aspects:

1) Data cleaning
2) Data aggregation
3) Situation assessment from the events

Large volume raw data (events) captured from RFID readers and PIR sensors arrive at the server from the gateway. This data contains duplicate and redundant information. It is very hard to get useful information quickly from this data. So these raw event data must be cleansed, and preprocessed. After cleaning raw data streams, the implied relationship among events to construct complex events are found.

CEP architectural layers:

CEP system architecture has several layers [9]. ***Event Source Layer*** consists of the event sources. ***Data Collection***

Layer is responsible for collecting the data coming from the various sensors, and filtering the data. Data collection could be distributed or centralized based on the requirement of the application. An important feature of the ***Event Analysis Layer*** is pattern matching and correlation across multiple event streams. In this layer few fixed continuous queries act on the incoming data streams to identify patterns of interest. Comparison of real-time data with historical or static data is also often required for event analysis. ***Application layer*** consists of systems that receive the processed events. For example GUI receives the processed events and displays the information.

Design and Implementation details:

The functional model of the system is shown in Figure 1. RFID, PIR sensor events, visual images are the input streams. Init module, filter module, event processing engine and action engine are the main modules. The following paragraphs explain these modules in detail.

Init Module initializes the internal data structures, caches and other modules using the configuration files and database. Once the sensors are deployed, the static information about the sensors like sensor ID, sensor name, location ID, privilege level of a location etc., are available in the database. All such static information can be updated in the local data structures during the initialization phase for future use.

Filter Module filters out the unwanted information like some events coming without a sensor ID, invalid RFID etc. The filtered data send to the down stream modules for further processing.

Event Processing Engine module has the CEP engine. The incoming streams of data are processed by this engine according to the predefined rules. ESPER Event Processing Engine is used in the current application. ESPER is developed in JAVA. A tailored Event Processing Language (EPL) is used for writing the queries [5]. These queries are initially registered with the engine. The engine stores a number of rather static different (continuous) registered queries, which are then processed against an incoming stream of data which is continuously changing.

Steps involved in event processor are as follows

Step 1: Get an instance of the Engine.

Step 2: Generate fixed queries (Rules). Run the queries continuously on the event streams as and when they arrive.

Step 3: Perform event detection through a rule-based engine or through pattern matching between a pre-specified spatiotemporal pattern and the incoming data streams.

Step 4: Handle the unexpected events by sending the information to the GUI or any other top level application as an event.

Step 5: Develop traditional Data Base interface to store the data for future analysis.

The events arriving from all the sensors are arranged based on the time stamp and are send to the engine for further processing. Next step involved is writing the EPL (Event Processing Language) statements for implementing the stream queries and patterns of interest. All the queries must be registered in the beginning. Corresponding listeners also should be added to receive the results of the posted queries. Once the queries are registered, the queries work on the incoming data streams. Whenever the conditions occur that match the rules then the response is send to the corresponding listeners.

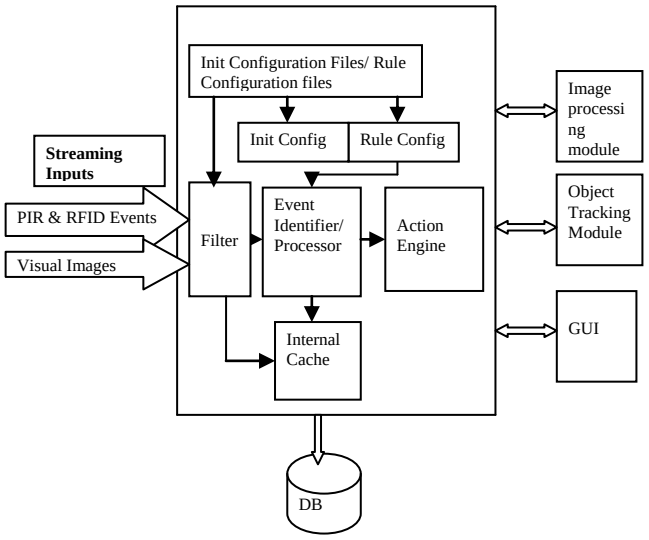

Figure 1. Functional model of the CEP module

Following code snippet is used for getting an Engine instance

```
EPServiceProvider esperEngine;
        config.addEventType("sensorEvent",
sensorEvent.class );
        esperEngine=
EPServiceProviderManager.getDefaultProvider(config);
```

Input to the CEP engine is the event streams. For the current application Java classes are good choice for representing the events. Hence Java class is used to represent the events. The raw events arriving at the base station have the information like sensor ID, RFID, PIR data, timestamp. Sensor Id is a unique Id per sensor node. Using the sensor ID the internal tables are looked up to get the other static information about the sensor node. Now an event that comprises of the raw event details from the sensor and some other static information is send to the Engine where the events are processed. An event can be send to the engine as follows.

```
        esperEngine.getEPRuntime().sendEvent(event);
```

Examples of Queries: Consider the simple example of identifying the people who enter in to some of the privileged locations in to which they are not supposed to enter. RFID tags have a priority associated with them. People are given an RFID based on their privilege.

Now a query is registered with the Engine as follows.

```
stmt = "select sensorId, rfid, sensorPriority from
sensorEvent";
        statement=
esperEngine.getEPAdministrator().createEPL(stmt);
```

Here sensorPriority is the location priority.

Once the Events of interest are identified by the event processing engine the ***Action Engine*** takes the necessary action to be followed. This could be like sending the information to the next level modules like GUI module.

ESPER provides different ways for the application to receive the results of the queries. The engine continuously

indicates results to all listeners as soon they occur. For the current example Update Listener interface provided by the engine is chosen. The application provides implementations of the Update Listener interface to the statement. Listeners receive Event Bean instances containing statement results. Once the query results are obtained, the results can be processed according to the application.

Below mentioned is the code snippet for adding a listener for the query stated above.

```
statement.addListener(new priorityListener());
```

The subscriber class must provide a method by name update to receive insert stream events row-by-row. The number and types of parameters declared by the update method must match the number and types of columns as specified in the select clause, in the same order as in the select clause. The engine posts results events via the new Events parameter to the update method of Update Listener instances listening to the statement. Table 1 shows the code for how the query results can be got in the priority Listener.

By looking at the RFID the priority of the person can be determined. Now knowing the priorities of the sensor Node/location and the object it can easily determined whether the person is authorized to enter in to the location or not by simply comparing the priorities.

At highly secured zones just checking the authorization is not sufficient. There is a possibility for an intruder to wear a valid RFID tag and enter into the secured zone. To identify such intrusion, the captured images are used. When the person enters the zone his RFID, sensor ID and time are captured in the wireless data stream. At the same time PIR sensor enables the camera and the person's photo is also captured with the same time stamp and sensor ID. These images are stored in the data base. For all the authorized people the features are extracted and stored in the data base using BICA algorithm. For every person entering in to the secured zones, RFID from the wireless packet and the image from the wired stream for the same sensor ID and time stamp are taken. This image is compared using BICA algorithm with the available images in the data base with the same RFID. If there is a mismatch between these images then it is treated as intrusion.

Table 2 shows the code for another example where an intruder who does not possess an RFID tag and trying to enter a location just behind an authorized person (Tail gating) can be identified.

TABLE 1. CODE FOR PRIORITY LISTENER

```
public void update(EventBean[] newEvents,
                                EventBean[] oldEvents) {
try {
if (newEvents[0].get("priority") != null)
 sensorPriority = ((Integer)
     newEvents[0].get("priority")).intValue();
            } catch (PropertyAccessException e) {
                System.out.println("priority not there");
            }
   try {
          if (newEvents[0].get("rfid") != null)
            objectRfid = ((Integer)
                        newEvents[0].get("rfid")).intValue();
          } catch (PropertyAccessException e) {
                System.out.println("rfid not here");
            }
    }
```

TABLE 2. CODE FOR TAIL GATING

```
stmt = "select sensorId,rfid,irData from sensorEvent";
 statement =

esperEngine.getEPAdministrator().createEPL(stmt);
statement.addListener(new sensorValidityListener());
public void update(EventBean[] newEvents,
                                EventBean[] oldEvents)
{
 EventBean event = newEvents[0];
 int sensorID;
 int objectID;
 boolean IR;
 sensorID= (Integer) event.get("sensorId");
 objectID= (Integer) event.get("rfid");
 IR = (Boolean) event.get("irData");
 if ((IR) && (objectID == 0))
{
    System.out.println("Possible Intruder in " +sensorID );
}
```

To confirm the tail gating, the images captured by the camera are used. Haar face detection algorithm is used to count the number of persons present in the scene. Thus using the RFID, PIR data and the images tail gating can be identified effectively with the help of CEP and pattern recognition of Images. Figure 3 shows an image captured by a camera where two people are present. Haar face detection notifies that there are two persons present. Thus using PIR, RFID data and camera images, tail gating problem can be identified effectively and alerts can be generated.

Figure 3. Picture depicting tail gating

IV. RESULTS & DISCUSSION

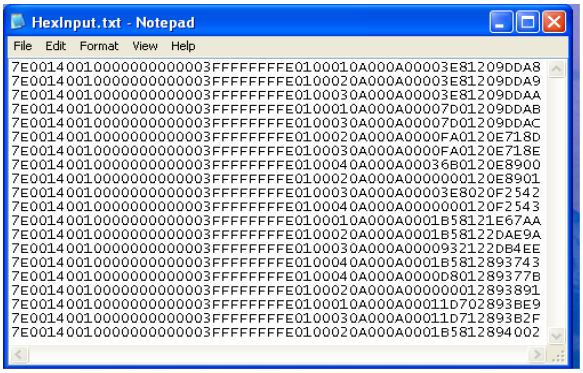

Figure 4. Input from WSN

This paper presents the results of the investigation of using CEP in WSN for object tracking application. The preliminary

results for validating the design are presented in this section. Data packet from the sensor node contains the information about sensor Id, RFID of the object, PIR reading that gives the presence of the object and time of the event in milliseconds. Sample input in hexadecimal notation to the CEP system is shown in Figure 4. Explanation for the input frame format is given in Figure 5.

Hexadecimal position	Explanation
1-2	Start Frame
3-9	Length
10 – 22	Destination Address
23 – 31	Mac Address
32 – 35	Transmission mode
36 -37	Node ID
38 - 43	Node Location
44 - 50	RFID
51	PIR
52 – 58	Time
59 – 60	End Flag

Figure 5 – Input frame format

Results of the priorityListener are shown in Figure 6. The messages are displayed and alarm is generated whenever an object/person enters in to a location where the object is not privileged to enter.

Priority Listener: Object is Not previleged to enter --- obj Priority 1sensor Priority 3
LOG_MSG - Location is – Server Room ObjectId is - 1000 Time is - 34201002

Priority Listener: Object is Not previleged to enter --- obj Priority 2sensor Priority 3
LOG_MSG - Location is - Server Room ObjectId is - 2000 Time is - 34201004

Priority Listener: Object is Not previleged to enter --- obj Priority 0sensor Priority 1
LOG_MSG - Location is - Laboratory ObjectId is - 0 Time is – 34507009

Priority Listener: Object is Not previleged to enter --- obj Priority 1sensor Priority 3
LOG_MSG - Location is - Server Room ObjectId is - 1000 Time is - 34547010

Priority Listener: Object is Not previleged to enter --- obj Priority 0sensor Priority 2
LOG_MSG - Location is – Conference Room ObjectId is - 0 Time is – 34547011

Priority Listener: Object is Not previleged to enter --- obj Priority 2sensor Priority 3
LOG_MSG - Location is - Server Room ObjectId is - 2354 Time is - 36549870

Priority Listener: Object is Not previleged to enter --- obj Priority 0sensor Priority 1
LOG_MSG - Location is - Laboratory ObjectId is - 0 Time is - 42547345

Figure 6. PriorityListener results

Results of the RFID failure check which is done by SensorValidityListener is shown in Figure 7. In this case whenever a person who does not possess an RFID tag tries to enter the location, the PIR sensor detects the presence of the person but the RFID value will be 0. This condition is identified by the SensorValidityListener and the warning messages are given out indicating the sensor Id i.e. location and the time.

RFID VALIDITY FAILED - sensorId is - 2 Time is - 34507009

RFID VALIDITY FAILED - sensorId is - 4 Time is - 34547011

RFID VALIDITY FAILED - sensorId is - 2 Time is - 42547345

Figure 7. SensorValidityListener

Research is in progress for extending the model for tracking more number of targets moving in a randomly deployed WSN.

V. Conclusion

Event driven architecture is more suitable for applications where continuous processing of streaming of events/data is required. Complex Event Processing technology is used for extracting the meaningful data from streaming data which is generated from senor nodes in a wireless sensor network. In this paper Complex Event Processing for object tracking application in a wireless sensor network environment is proposed. Queries for intrusion detection have been implemented and verified using ESPER CEP engine. Data from PIR sensor, RFID and images from camera are used for addressing the security and tail gating issues.

References

[1] Dunkel, J, "On Complex Event Processing for Sensor Networks", invited talk in: IEEE 9th International Symposium on Autonomous Decentralized Systems (ISADS), Athens (Greece), March 23-25, pp. 249-255, 2009.

[2] Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems, Addison Wesley.

[3] Aqeel-ur-Rehman, Abu Zafar Abbasi, Zubair A. Shaikh, "Building a Smart University Using RFID Technology," csse, vol. 5, pp.641-644, 2008 International Conference on Computer Science and Software Engineering, 2008

[4] Asaf Adi, David Botzer, Gil Nechushtai, Guy Sharon "Complex Event Processing for Financial Services". Proceedings of the IEEE Services Computing Workshops (SCW'06), pp: 7-12, 2006.

[5] ESPER TOOL : http://esper.codehaus.org/esper/documentation/documentation.html

[6] Wen-Ming Yuan; Jia Xiao; Dong Wang, "A complex event processing model based on RFID network", in IEEE conference on ACIA'09, December 28, 2009, pages 211 - 214

[7] C. H. Kim, K. Park, J. Fu, and R. Elmasri, "Architectures for streaming data processing in sensor networks," in Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on, 2005, p. 59

[8] Steinberg, A.N., Bowman, C.L. and White, F.E., Revisions to the JDL data fusion model. In: Proceedings of the SPIE. Sensor Fusion: Architectures, Algorithms and Applications, SPIE. pp. 430-441.

[9] Bhargavi.R, V.Vaidehi, P.T.V.Bhuvaneswari, P. Balamurali, Girish Chandra, "Complex Event Processing for Object Tracking in Wireless Sensor Networks", International Conference on Web Intelligence and Intelligent Agent Technology, IEEE/ACM,Toronto,2010.