Unified management of heterogeneous sensors for complex event processing

M. Valdés, I. Nieto, V. Guardiola, D. Gil & A. Gómez-Skarmeta
University of Murcia, Spain

Abstract

The turn of phrase that best summarizes SERKET´s main goal is *to provide security personnel with the right information at the right time*. For this aim, the SERKET security system incorporates mechanisms to collect, fuse, filter and aggregate information coming from heterogeneous event sources and sensors. This *Complex Event Processing* is performed to improve the *global situation awareness* and the management of threats. Additionally, the system is intended to be as open, flexible and scalable as possible at a low cost. In this paper, the mechanisms by means of which scalability and flexibility features are achieved in the particular issue of *sensors management* are shown. Moreover, instead of simply homogenizing basic events and delegating their processing to higher levels of the system, it builds, as far as possible, a picture of the local situation. Therefore, the traffic to the rest of the system can be reduced since this local situation picture is transferred instead of many events that are too basic.
Keywords: public security, situation awareness, complex event processing, service oriented architecture, information fusion, smart sensors.

1 Introduction

SERKET [1] stands for *SEcuRity KEeps Threats away* and its field of application is the security of public events and places. Its main objective is the development of a software system to provide support to the security and surveillance personnel avoiding the so-called *cognitive overload* problem. Moreover, the system is intended to be open, flexible and scalable at a low cost. In the SERKET security system, the intelligence resides in both software and security personnel. By means of the on-line analysis and fusion of complementary, as well as redundant, information coming from different sources – from sensors to

WIT Transactions on Information and Communication, Vol 39, © 2008 WIT Press
www.witpress.com, ISSN 1743-3517 (on-line)
doi:10.2495/RISK080431

human beings – the system is intended to create a *global snapshot of the current situation,* raising alarms whenever an undesired situation is recognized. To make it possible, a Complex Event Processing (CEP) approach is adopted. CEP is an emerging technology comprising the tools and techniques for analysing and controlling the complex series of interrelated events that drive modern distributed information systems [1]. Events in SERKET are supplied by different types of sensors distributed in the target public place. *Sensors,* in the broadest sense, means cameras, microphones or even human beings. In this paper we present a threefold *Sensor Management Module* (SMM), providing:

- A unified management of heterogeneous sensors whose design favours the incorporation of new types of sensors into the system.
- A *local decision module (LDM)* able to generate a *local situation snapshot*. An LDM is in charge of an *area of interest* instead of only a sensor. Then all the sensors in that area are attached to the same LDM. Instead of supplying basic events to the SERKET system, the LDM performs an early and local CEP processing of the local events occurring in its attached area of interest. In other words, a local situation snapshot is generated and ready to be sent to the SERKET system in order to perform the remaining CEP processing for the generation of the global situation snapshot. In this way, the problem of cognitive overload is faced in the lowest level of the system, which prevents the massive and unnecessary flow of basic events to higher levels of the system.
- Mechanisms to request information from the sensors and to reconfigure their settings in order to tune the definition of interesting events. The system could find the necessity of obtaining more precise information about a received event in order to reduce the uncertainty (for example, zoom in the camera for a more detailed image). Once the event is received in the SMM, this kind of decision could be made locally instead of forwarding the event to the higher levels of the system, processing it, making the inference and resending the request to the SMM.

MMS design is based on the adoption of a *Service Oriented Approach* [3], the use of a common OWL ontology [4] and the definition of the XML schemes and protocols [5,6] needed to register new incoming sensors, communicate the detected events and provide the translation between the required information from sensors and their proprietary management language.

2 Overview and architecture

Consider Figure 1. The public place is supposed to be divided into *areas of interest* containing several smart sensors. There exists one SMM managing all the sensors in the area. In fact, the individual sensors are invisible to the rest of the system. The three main entities composing the SMM are shown: the *Registration Module (RM)*, an undefined number of *Event Sources (ES),* and a *Local Decision Module (LDM)*. A SERKET *common ontology* is supposed, containing the concepts involved in the public security domain. However, in

order to give as simple an example as possible, let us consider a basic ontology containing concepts about humidity and temperature changes and let us suppose that they could be signals of a fire situation. When a humidity smart sensor H_A located in certain area A is incorporated into the system, it has to send a registration request to the SMM_A in charge. The request contains information about the *capacities* of the sensor (events that it is able to detect, configuration requests it is able to receive, and so on). They are expressed in the base of the ontology. Concretely, the RM performs the registration. As a consequence this information is saved in its local *repository*. In fact, the RM also supplies a *search service*.

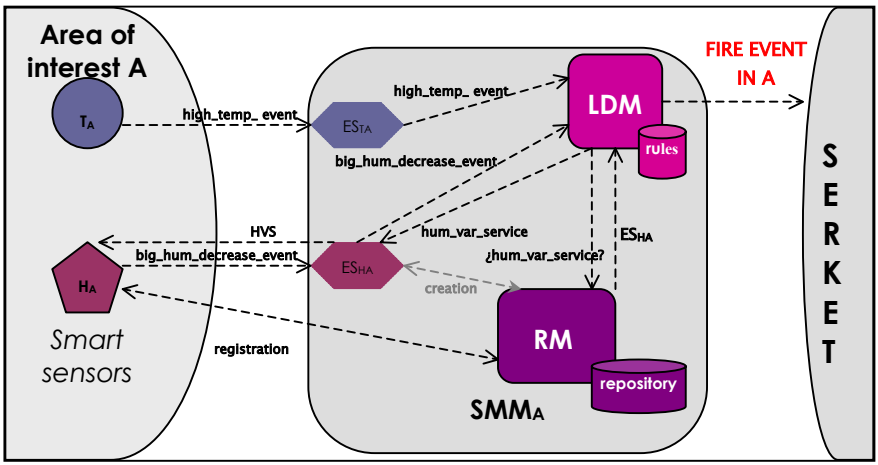

Figure 1: SMM architecture.

Once the RM receives a registration request, this entails the creation of an *event source* ES_{HA} connected to that sensor. From that moment the sensor will be able to send events to the system through its ES_{HA}. The event sources will be in charge of forwarding the events from their respective smart sensors to the correct entity in the SMM. Some types of events will be processed inside the ES_{HA}, for instance, as we will see later, events representing *keep-alive* signals. Others, such as a *high_temperature_event* coming from a temperature sensor T_A, will be forwarded to the LDM in order to perform its local preprocessing.

Sometimes, the LDM decides to discover what sensors in its area of interest detect certain type of events. Those decisions are governed by the rules it was configured with. The LDM has to ask for the needed information from the RM. In the figure, the LDM needs information about the suppliers of humidity change events. The RM answers that ES_{HA} has that capability. At this moment, the LDM requests the humidity variation to ES_{HA}. Then ES_{HA} translates this petition into a sensor H_A *proprietary command*. When ES_{HA} receives the response, it is sent to the LDM. Finally, in this dummy example the LDM rules engine infers that a high temperature and an important decrease of the relative humidity are signals of a fire situation and sends the event just created to the system.

We must remark that for the purposes of simplicity, this example is based on very simple sensors, events and rules. The smart sensors, the ontology as well as the rules that lead the SMM behaviour are capable of being expanded or even completely modified in order to better fit SERKET requirements.

The SMM is implemented in Java and the communication mechanism is JMS (Java Message Service) [7]. JMS is the Java approach to Message Oriented Middleware. JORAM [8] has been selected as the JMS provider. However, in order to obtain independence from the communication mechanism, several proxies have been included, isolating the communications issues from the sensor management. Moreover, these proxies make it possible to use different types of communication in each one of the connections shown in the figure.

The remaining paper is devoted to explaining the processed involved in sensor management. Finally, concluding remarks are given.

3 Registration

This section is devoted to explain more deeply the process by which a smart sensor is integrated into the system. In order to receive events from certain types of smart sensor existing in certain areas of interest, it must send an XML registration request to the RM, based on certain XML schema. This is the so-called *capacities schema.* It defines, basically, the way in which the smart sensor informs of: (1) the type of sensor it is, (2) the events that it can detect (for example, a *high_temperature_event*), (3) the ***configuration services*** it can perform (for example, to modify the temperature threshold that determines when a *temperature_high_event* is produced), (4) the ***information services*** it provides (for instance, it could give information about the current temperature). All this information refers to concepts from the ontology.

3.1 Creation of the event source associated to a smart sensor

As has been said, the immediate consequence of a registration request is the creation of a suitable *event source* in charge of and connected with that sensor. An event source is an instance of a certain class that implements the interface *EventSource*. A class implementing *EventSource* must exist for every type of sensor capable of being part of SERKET.

The channels communicating the event source with the other components of the SMM are created as well. This information is sent to the smart sensor. From that moment, it is able to send events to the system through its event source.

3.2 The RM repository

The registration also entails the creation of a new entry in the *RM Repository*. It contains the information supplied by the sensor in the registration request, the identifier of the event source and the mechanism to communicate with it. Therefore the RM provides a *search service* by means of which the LDM can make queries to find the event sources that provide the concrete service it needs (configuration or information services). Queries and answers must each be

compliant with certain XML schema (*service query schema and service query response schema*). At the moment, only very simple queries can be made but the LDM is ready to be improved with a more sophisticated query language.

3.3 Receiving events from smart sensors

When a smart sensor detects certain events, translation into the SERKET event format must be carried out in order for the event to be CEP-processed. In fact, this *adaptation* is the first phase of CEP. In our approach, it is carried out in the smart sensor. At the moment, certain *event format* schema has been supposed but it is capable of being changed to the definitive SERKET format.

Some of those received events are processed by the event source. These are the *keep_alive_events*. The *keep-alive* mechanism will be explained later. Other types of events are sent to the LDM. According to its rules, the LDM will decide whether they are processed locally or they are forwarded to higher levels of the system.

4 Local processing of incoming events

The LDM is the entity in charge of asking for the information needed, reacting to incoming events, sending CEP events to the higher level of the system, and so on. It is loaded at execution time with a set of *rules.* The rules define the LDM behaviour and they are dependent on the concrete scenario and the concrete ontology of events. Changing the rules of the LDM changes its reaction to the incoming events. The rules are loaded in execution time, and they can be read from a configuration file in plain text, a database or any other mechanism. The rules are handled by the *Esper* engine. Esper is a Java engine for processing CEP that enables the rapid development of applications that process large volumes of incoming messages or events, filters and analyzes events in various ways, and responds to conditions of interest in real-time [9]. When a new incoming event fulfils a certain rule, the LDM performs the tasks written in its right part (querying some information from a smart sensor, requesting some configuration service, or even creating a higher level event to forward to the main CEP processor).

5 Requesting information or configuration services to smart sensors

If the LDM wants to receive an event proactively (LDM decisions depend on its rules), it must ask for the event from a suitable event source. In order to do this, the LDM first uses the RM search service to know what event sources supply the kind of events it wants to receive. Afterwards, the LDM sends the request to the selected event source. It must be remarked that the event source has to convert the received invocation into a proprietary command that the smart sensor can

understand. This is the reason for the design of different classes implementing the *EventSource* interface, each one able to translate to a specific sensor proprietary format.

The LDM may want to change some configuration parameters or give a configuration command to a certain smart sensor. For example, after detecting an intruder, a camera can be asked to track the target to have a log of their activities or to know their destination. Most of these configuration requests will be activated due to an incoming event, or they could be the result of a query from high levels of the system. The sequence of actions is as follows: first, the LDM asks the RM for the entities (event sources) capable of changing their configuration as the LDM needs. The LDM could also state some arguments it wants to specify when invoking this configuration request. Upon receiving this request, the event source will try to fulfil it by communicating with the smart source. In the case of failure, an error event will be generated and sent to the LDM.

6 Smart sensor deregistration

Deregistration is necessary in order to get an up-to-date repository. When a smart sensor is going to power off it has to send a deregistration request to the RM. The effects are the deletion of the smart sensor information from the repository as well as the destruction of the corresponding event source.

On the other hand, a smart sensor could go down without notifying the RM. Then, we need a mechanism to check whether a smart sensor is still running or not. This mechanism is based on *keep-alive* messages. Every *s* seconds -*s* is established by the smart sensor in its registration request – the event source sends an event service request for a *keep_alive_event*. If the required event is not received before a certain timeout, the smart sensor is considered as dead, consequently, it is deregistered from the repository and the event source is destroyed.

7 Concluding remarks

The main issues with service-oriented software (including XML schemes and protocols) for the management of heterogeneous sensors for Complex Event Processing in SERKET have been shown. Far from being just a CEP adaptor, it is also able to carry out local processing of the incoming events. In this way, not all the detected events must be forwarded to higher levels of the system. According to the LDM rules, many of them will be aggregated into more sophisticated ones contributing in this way to a traffic reduction to higher layers of the system. Flexibility has been stressed as one of the more significant features. In fact, it has been achieved in different ways and degrees depending on the particular issue:

- With respect to the incorporation of a new sensor, some things must be done before it can supply events to the system. First, a mechanism to register, deregister and sending *keep-alive* signals must be added to the

sensor software. Additionally, the sensor must provide its events in the SERKET common event format. This is the only issue involving CEP adaptation the sensor is involved in. New incoming sensors can be incorporated in the system on the fly if they are already *known.* That is, if a class implementing the interface *EventSource* for that type of sensor exists. This class is able to translate *information* and *configuration services* to the corresponding proprietary language commands. If not, a suitable class must be designed and included in the system.

- Flexibility has also been achieved with regard to the communication issues. The SMM has been designed to be as independent from the communication mechanism as possible. For that aim, proxies isolating these kinds of issues have been included in every connection between two submodules of the SMM. In this way, although the current SMM is based on JMS in every connection, it would be very easy to change or include a new communication mechanism (sockets, web services,...) adding the necessary proxies.
- Regarding the local CEP processing, the file or data base containing the rules that lead the LDM behaviour can be (and should be) modified as well as the ontology defining the concepts involved, in order to achieve more sophisticated processing and to include new situations of interest.

This flexibility at different levels favours the definitive integration of SMM in the SERKET system as the main future work.

On the other hand, a SMM is in charged of certain areas of interest. Sensors inside the area are *invisible* to the rest of the system. Given that areas of interest could be physically organized into a hierarchy, further works lead us to the design of a hierarchy of SMMs as well, in such a way that lower level SMMs provide events to the next level SMMs. So, lower level SMMs would be considered as sensors supplying events to the next level SMMs. This building-block approach is good practice to reduce the system complexity and to avoid unnecessary traffic to the main SERKET CEP processor.

Finally, as has already been pointed out, another future work will be a more sophisticated RM search service and a more sophisticated query language.

Acknowledgement

This work has been supported by the Spanish Ministry of Science and Technology (MCYT) by means of the project DESEREC IST-2004-026600.

References

[1] ITEA project 04005 – SERKET. http://www.research.thalesgroup.com/software/cognitive_solutions/Serket/index.html
[2] Luckham, D., The Power of Events: An introduction to complex event processing in distributed enterprise systems, Addison Wesley, 2002.
[3] Erl, T., Service-Oriented Architecture. Concepts, Technology, and Design. Edn. Prentice Hall, 2005.

[4] OWL website: http://www.w3.org/2004/OWL/
[5] XML website: http://www.w3.org/XML/
[6] XML Schema website: http://www.w3.org/XML/Schema
[7] JMS website: http://java.sun.com/products/jms/
[8] JORAM website: http://joram.objectweb.org/
[9] Esper website: http://esper.codehaus.org