Using Complex Event Processing for implementing a Geofencing Service

Bogdan Târnauca
Siemens Corporate Technology Romania
Email: bogdan.tarnauca@siemens.com

Dan Puiu
Siemens Corporate Technology Romania
Email: dan.puiu@siemens.com

Septimiu Nechifor
Siemens Corporate Technology Romania
Email: septimiu.nechifor@siemens.com

Vasile Comnac
Transilvania University of Brașov
Email: comnac@unitbv.ro

ABSTRACT

Geofencing is one of the components in the wider spectra of Ambient Assisted Living related applications. Such applications are meant to provide support to persons with disabilities or to those impaired, as well as to their caretakers. Within this context, geofencing targets the safe mobility of such persons. As mobility related data can be translated into events with a temporal dimension, we have developed a geofencing service based on the Complex Event Processing paradigm. This decision was supported also by the fact that Ambient Assisted Living applications, in general, are often considered for integration into event rich environments, such as those resulting from smart cities infrastructure. The ability of Complex Event Processing to handle vast amounts of time correlated events from multiple sources meets the requirements imposed by the geofencing service. This paper presents the architecture of a Complex Event Processing based geofencing service along with the results obtained during the evaluation phase of the first prototype.

I. INTRODUCTION

A geofencing service monitors constantly the position of a person and automatically generates alerts and notifications when the person enters, leaves or moves within a specific geographic area, allowing the detection of being lost and generating appropriate intervention. The service can be addressed to elderly persons, being able to send an alert (such as a SMS message) to a caregiver when the user has averted a predefined distance from a selected location.

This type of service has become more popular in the modern society since it has the ability of increasing the quality of life for the elderly people, many of which are living independently.

Wong et. al., in the paper [1], present a geofencing service application. They provide a complete system for tracking persons, especially elderly people, and their application can be extended to provide a geofencing solution. The system consists of a wearable AGPS (Assisted Global Positioning System) terminal with two-way communication capability and a GPS (Global Positioning System) assistance data server. This approach has a drawback because it forces the end-user to use a dedicated device for GPS based location acquisition and for the data transmission to the server.

In the paper [2], Ryoo et. al. present a geofencing service solution which uses mobile devices for location acquisition and data transmission. They developed an energy-aware proactive framework that uses different communication technologies and sensors based on their energy usage, provided accuracy and availability. The proposed solution tracks the location of a person outdoor using the GPS sensor of the mobile device and indoor using 3G or WiFi interfaces.

Other solutions for geofencing, but at larger scale, are presented in the papers [3] and [4] for tracking fleets of vehicles. The architecture of the system is similar to those presented previously but in this case only GPS technology is used for position acquisition.

Analysing the geofencing service from another perspective one can consider an “event” the reporting of a person’s location or the fact that a person enters or exits a geofencing area. The events can have a strong temporal relationship. For example: if the event generated by a person when he or she exits a geofencing area is followed in less than 5 minutes by an event which announces that the person has returned, then the system should not generate the alarm, otherwise an out-of-geofencing-area event should be generated. If the system contains a lot of clients with different profiles which have to be monitored, then the amount of events as well as the possible conditions to be processed simultaneously increases considerably. In this case a CEP (Complex Event Processing) engine can be used.

The following smart city example illustrates how CEP works. It refers to a city where the vehicles are able to report anonymously their current position. The raw event stream is composed of events with a timestamp (indicating the time when it was generated), a coordinate value (provided by the vehicles’ GPS systems), and current speed. At this stage, the information is by itself of no or limited use but a CEP based solution which would collect the events from most of the vehicles in the city would be able to detect, by counting the vehicles present on a specific road section (considering
their direction and also the speed) that a traffic jam has occurred. This is a complex event composed of the road name, direction and location of the jam together maybe with some aggregated data obtained from the raw events (such as average speed, number of cars, etc.). Also, complex events indicating a possible traffic incident at a certain location can be generated if in a short period of time multiple events that indicate hard break are reported. In this case the hard break event is followed by an event reporting a normal cruise speed, and a full stop.

This paper presents the architecture of a CEP based geofencing service along with the results obtained during the first prototype evaluation of the MOBILE.OLD, AAL (Ambient Assisted Living) European project. The proposed solution makes use of the capabilities provided by CEP in order to increase the flexibility as well as the scalability of a geofencing service.

This paper is structured as follows. Section II presents the key points of CEP and the typical architecture of a CEP application. In section III the scenario description, as resulted from the end users’ requirements, is presented followed by the system architecture presented in section IV. Section V presents some implementation details while the obtained results and conclusions are documented in VI.

II. COMPLEX EVENT PROCESSING

A. Evolution of Event Processing towards Complex Event Processing

CEP refers to the set of methods and tools which have been developed with the main goal of providing business application developers the means of handling vast amounts of near real-time data, often involving operations with a temporal dimension (e.g. aggregation operations over a period of time or pattern matching).

Although in its current form it can be considered a novel set of methods and tools, CEP draws a lot of attention from businesses as well as application developers or academia. This tendency is proved by numerous assessments of the technology such as [5], [6], [7], [8].

Business application have been providing for a long time near real-time event processing in various forms [6], but CEP as now exists in today’s software solutions has been introduced rather recently.

The term, which has been introduced during the work on the Rapide project by David Luckham [9] shifted from the original analysis of hardware oriented discrete-event systems towards distributed software architectures. Its first comprehensive description is provided in [10].

CEP stands for the processing of complex events, where a complex event is the result of one or more operations being applied to raw, basic or simple events or to other complex events.

An event is defined concisely by Chandy and Schulte [11] as anything that happens and more elaborate by Etzion and Niblett [10] as an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain.

The main advantage of a CEP platform, compared to the earlier event processing solutions, is that the former is generic and flexible with no hard-coded event processing functionality, which impacts positively software deployment and maintenance costs.

B. CEP based applications architecture

Even if a number of event processing languages are available, depending on the CEP platform origins, the majority of CEP applications are constructed following a typical architecture. Such an application is build around one or more EPNs (Event Processing Networks) which provide the means of interconnecting processing agents or processing nodes capable of processing basic or raw events as well as other complex events.

Some applications might require input adapters in order to adapt the incoming events format to one which is “understood” by the EPN. This internal format could be POJOs for some Java based CEP engines or hash maps, simple XML objects or some proprietary event representation format.

Output adapters are used in order to expose the resulting processed events so that they can be used by other software components (internal or external to the originating CEP application).

The typical operations provided by the processing nodes are:

- event filtering;
- event translation;
- event splitting;
- event aggregation;
- event composition;
- event pattern detection;

III. SCENARIO DESCRIPTION

The scenario definitions as seen below is the result of analysing the service requirements extracted as part of the MOBILE.OLD project, refined through the feedback obtained from the pilot sites.

A. Requirements Analysis

The requirements of the geofencing service resulted from the refinement of those described in the project proposal. These original requirement were translated into a mock-up of the application which was sent for evaluation, along with a questionnaire to be filled by elderly people from the project’s pilot sites.

Together with some requests which were related to the functionality and usability of the application implementing the geofencing service, a couple of non-functional requirements have been identified. Considering the number of data sources and data rates, their type and distribution, the time dependent actions, the required flexibility in defining the business logic as well as the required scalability and potential for integration with other types of applications (such as the smart
cities category) it became apparent that CEP is highly suitable for the application.

B. Scenario definition

The geofencing service provides mobility assistance for persons suffering from cognitive impairments while performing outdoor activities. The locations of the elderly people, referred from now on as end-users, are constantly monitored and automatic alerts and notifications are sent when any deviation from the planned route is detected. On the first stage a notification is shown only to the end-user if he or she exits a geofencing area, requesting the return to the save area, while in the second stage the alert is sent to the end-user’s family or caretaker if he or she does not return to the geofencing area within a certain time interval (defined by the profile). The geofencing areas can be defined as (see figure 3):

- an area within a predefined radius from a selected location;
- an envelope around a planned route;
- an area defined by any closed polygonal chain shape.

In addition, the service monitors the matching of predefined patterns (sequences of events) which could indicate that the end-user has or might have a problem generated by external physical conditions.

For the end-user’s caretakers or the family the application has two working modes: end-user profile management and end-user monitoring. In the profile management mode the caretaker or a family member has the possibility to create/edit/remove the geofencing areas for each end-user. They can also define hazardous areas which will be applied for all the end-users registered on the application (e.g. temporary construction works on an intersection, crowded events). Furthermore, in this working mode the end-user’s caretaker or the family can edit the patterns for the detection of external physical conditions. Below some possible patterns are presented, where $X$, $Y$, $t$ are configurable parameters:

- temperature higher/lower than $X^{\circ}C$ and an end-user is outside more than $t$ minutes;
- atmospheric pressure higher/lower than $Y$ mmHg and an end-user is outside more than $t$ minutes;
- a variation of temperature larger than $X^{\circ}C$ in a period of time smaller than $t$ minutes;
- a variation of atmospheric pressure larger than $Y$ mmHg in a period of time smaller than $t$ minutes.

In the end-user monitoring mode, the application displays a map where the location of the end-users and the geofencing areas are indicated. In this working mode the caretaker or family can use a set of filters for the information displayed on the map (e.g. all end-users positions, only one end-user position with a specific ID, end-users positions and geofencing areas, only end-users positions outside the geofencing areas, etc.).

The caretakers or the family will receive a SMS message in the following situations:

- the end-user is outside the geofencing area and does not acknowledge the notification message;
- the end-user is outside the geofencing area, he or she acknowledges the notification message but does not return to the geofencing area within a certain time interval specified by the end-user profile;
- the end-user enters a hazardous area;
- an external physical condition is met.

The service requirement is that the end-user carries a smartphone having the application running with the geofencing areas defined by his or her caretaker or family members. When the service detects the end-user being outside the geofencing area, it generates a warning on the end-user’s smartphone (the phone starts to ring or/and vibrate and a notification message is displayed on the screen). The end-user should check the notification message and acknowledge the warning by pressing a button on the smartphone.

The geofencing service workflow for end-users and caretakers/families is presented in figure 1.

IV. ARCHITECTURE DESCRIPTION

The architecture of the geofencing service system is represented in figure 2. The geofencing service contains two main applications: the client application which runs on the end-user’s smartphone and the server application which provides a browser based front-end for the caretakers.

The mobile client application is responsible for monitoring the location and the physical condition of the end-user. The main component of the client application is the android service which is responsible for:

- configuring and reading the sensors of the smartphone;
- sending to the server the data read from sensors (GPS position, temperature, humidity, etc.);
- receiving from the server the geofencing areas and the external physical condition pattern detection parameters.

The server hosts three main components (see fig. 2), as follows:
- the CEP component, responsible for handling multiple events and performing pattern detection;
- the Web Services handling the server side part of the application’s business logic, including user access, user management and part of the geofencing service (e.g. adding/editing/removing geofencing areas, sending the geofencing areas to the client application);
- the Data Persistence component which handles access to recorded data such as user profiles, historical data, system configuration.

The CEP module is responsible for analysing the information received from the android client and generating the complex events (for example: the end-user is outside the geofencing area). The statements presented in section V are executed in this module.

V. CEP BASED GEOFENCING

This section contains a couple of code snippets extracted from the CEP component of the geofencing service application which describe some of operations providing the proposed functionality. The following queries have been slightly modified for the sake of readability.

In order to detect when a person has left the geofencing area and does not return within a predefined time interval, the following set of CEP statements can be used. The client application (the application running on the smartphone) sends every \( t \) minutes to the CEP component (on the server side) an event which contains the GPS position of the smartphone. The input stream is called \( \text{ClientStream} \).

The statement in Listing 1 checks if the last GPS coordinate received is inside a geofencing area or not. The result is considered to be a complex event because it contains all the fields from the initial event (received from \( \text{ClientStream} \)) and a new field \( \text{insideGeofencingArea} \), which holds a boolean value. The field \( \text{insideGeofencingArea} \) is set to true if the point is inside a geofencing area, otherwise it is set to false.

![Figure 2. The architecture of the geofencing service system.](image)

Listing 1. Check if last position received is within a geofencing area or not.

The event generated by the CEP user defined function \( \text{testLocation} \) indicates if the GPS coordinate is inside or outside the geofencing area by using the ray-casting algorithm.

The \( \text{GeofencingPositionStatusStream} \) is monitored and if two consecutive events have the \( \text{insideGeofencingArea} \) field set to false, within a time window of \( T \) length, then a new event is generated in the \( \text{OutsideGeofencingAreaStream} \) (see lst. 2).

\( T \) represents the interval which the end-user is allowed to be outside the geofencing area without the caretaker being informed about the occurrence of this condition. This would allow the end-user to return to the geofencing after he or she receives the notification about this condition on the mobile device.

Listing 2. Check if two consecutive events have the \( \text{insideGeofencingArea} \) field false in a time window with length \( T \).

The event generated by the CEP user defined function \( \text{testLocation} \) indicates if the GPS coordinate is inside or outside the geofencing area by using the ray-casting algorithm.

The \( \text{GeofencingPositionStatusStream} \) is monitored and if two consecutive events have the \( \text{insideGeofencingArea} \) field set to false, within a time window of \( T \) length, then a new event is generated in the \( \text{OutsideGeofencingAreaStream} \) (see lst. 2).

\( T \) represents the interval which the end-user is allowed to be outside the geofencing area without the caretaker being informed about the occurrence of this condition. This would allow the end-user to return to the geofencing after he or she receives the notification about this condition on the mobile device.

Listing 2. Check if two consecutive events have the \( \text{insideGeofencingArea} \) field false in a time window with length \( T \).

In order to prevent repeated reporting of an end-user being outside the geofencing area, a solution had to be provided so that only the first occurrence of such a condition is reported. If a listener is added to the \( \text{OutsideGeofencingAreaStream} \) then its \( \text{update} \) method is invoked every \( T \) milliseconds when the end-user is outside the geofencing area. In conclusion, it is possible to send multiple alarms in this case. To avoid this situation the \( \text{GeofencingTestWindow} \) is created. The test
window keeps all events from OutsideGeofencingAreaStream. The GeofencingTestWindow is reset when an GeofencingPositionStatusStream event is generated with the field insideGeofencingArea set to true. An alarm event is generated when an OutsideGeofencingAreaStream event is detected and GeofencingTestWindow contains one event. The window is reset when the end-user returns to the geofencing area. This mechanism can be implemented using the statements presented in Listing 3 to 6.

Listing 3. Create GeofencingTestWindow.
```sql
create window GeofencingTestWindow win: keep all() as OutsideGeofencingAreaStream;
```

Listing 4. Insert OutsideGeofencingAreaStream events in GeofencingTestWindow.
```sql
insert into GeofencingTestWindow select * from OutsideGeofencingAreaStream;
```

Listing 5. Generates the alarm event when the GeofencingTestWindow contains an event.
```sql
insert into GeofencingAlarmStream select * from OutsideGeofencingAreaStream where ((select count(*) from GeofencingTestWindow) = 1);
```

Listing 6. Reset GeofencingTestWindow when a GeofencingPositionStatusStream event is detected and insideGeofencingArea true, is generated.
```sql
on GeofencingPositionStatusStream(insideGeofencingArea = true) delete from GeofencingTestWindow;
```

Listing 7. Determine the minimum temperature registered in last t period of time.
```sql
insert into MinTemperatureStream select istream min(temperature) as temperature, * from ClientStream .win: time(t minutes);
```

Listing 8. Generates the alarm event when the temperature is bigger than \( X \) \( ^\circ C \) for more than 60 minutes.
```sql
insert into TemperatureAlarmStream select * from MinTemperatureStream (temperature > X);
```

Listing 9. Determine the maximum temperature registered in last t period of time.
```sql
insert into MaxTemperatureStream select istream max(temperature) as temperature, * from ClientStream .win:time(t minutes);
```

Listing 10. Determine the temperature variation in last t minutes.
```sql
insert into TemperatureVariationStream select * from TemperatureVariationStream (temperatureVariation > X);
```

Listing 11. Check if the temperature variation in last t minutes is bigger than \( X \) \( ^\circ C \). The alarm events from TemperatureAlarmStream or TemperatureVariationAlarmStream are generated every time an event is generated on ClientStream and the conditions (in this case temperature conditions) are met. In order to avoid multiple notifications, a mechanism with a test window should be implemented (similar to the mechanisms presented in listings 3 to 6 for geofencing facility). This way only the first alarm event generates a notification to the end-user.

Statements similar to those from Listing 7 to 11 can be used to monitor the atmospheric pressure variation.

The end-user can have different types of geofencing areas for instance waking geofencing areas or public transport geofencing area which can have different footprints on the map. The statement from Listing 12 to 14 determines, based on end-user speed, which type of geofencing area to be used.

Listing 12. Computation of the end-user speed.
```sql
insert into HighSpeedStream select istream max(userSpeed) as maxSpeed from UserSpeedStream .win: length(2);
```

Listing 13. Keep the highest speed from the last two end-user speed events registered.
```sql
insert into VehicleSpeedStream select * from HighSpeedStream (maxSpeed > SPEED_TH);
```

Listing 14. Determine if the end-user walks or uses a vehicle.

The method getDistance is a CEP user defined function which computes the distance between two GPS points. The statement from Listing 13 determines the highest speed from the last two end-user speed events. This is used in order to avoid the situations when the vehicle is briefly stopped (for instance at the bus station) and the end-user speed is 0 km/h. In other words we consider that the end-user is in a vehicle if at least one speed from the last two is higher than \( SPEED_TH \). The parameter \( SPEED_TH \) sets the threshold value for a minimum vehicle speed.

Another aspect which must be monitored is client data connectivity loss. The application from smartphone is configured to send a position event every \( t \) seconds. The geofencing system must notify the caretaker when the data connectivity is lost. This is accomplished by the statements from Listing 15.

Listing 15. Whenever an event is not followed by an event of the same type within \( t \) minutes, the missing event pattern fires.
VI. EXPERIMENTAL IMPLEMENTATION AND CONCLUSIONS

Besides the high flexibility provided for the definition of the geofencing rules, this solution proved to deliver high processing performance with insignificant latencies (less than 100ms) compared to the time constants of the service itself (the lowest one being in the range of tens of seconds). The solution can easily scale-up even on commodity grade hardware such as our test PC which had an Intel i5-2410 2.30 GHz processor, 4GB of RAM, 1GB ethernet adapter, running on a Windows 7, 64 bit operating system. We have developed a set of applications which emulate the behaviour of the android client application (sending to the server a set of random generated end-user positions). The server side application was able to handle more than 20,000 events per second with 1000 queries active in a single CEP instance. These values exceed by far the requirements of the geofencing service, since the typical number of end users as well as the required position tracking sampling rate can not generate such a large rate of events per second. The use of Complex Event Processing in handling the events involved by the geofencing service allows a quick extension of this application for handling events from a smart city environment such as those related to traffic, weather, events, environment or public accessible data sources.

ACKNOWLEDGMENT

This paper is supported by:

- the MOBILE.OLD project (http://www.mobiledotold.eu/), funded by the AAL Joint Programme under the AAL-2011-4 call.

REFERENCES


