USING COMPLEX EVENT PROCESSING TO MANAGE PATTERNS IN DISTRIBUTION NETWORKS

Foued BAROUNI
Eaton – Canada
FouedBarouni@eaton.com

Bernard MOULIN
Laval University – Canada
Bernard.Moulin@ift.ulaval.ca

ABSTRACT

Several efforts have been made in recent years to provide a new generation of data acquisition systems with reliable and efficient communication capabilities. These systems provide real-time data in various formats at multiple application levels. In the power distribution domain, end-users need to handle large amounts of data generated by these acquisition systems. They also need a set of software tools that allow for leveraging existing data and detecting interesting situations. Available commercial solutions are mostly based on relational databases and use SQL queries to offer such functionalities. However, these systems suffer from performance limitations and do not allow real time detection of interesting situations (also called patterns). In this paper, we present a novel approach based on Complex Event Processing for the real-time detection of complex situations involving events. The contribution of the paper is the proposal of an architecture that integrates a pattern detection engine in the distribution grid in order to improve user decision making capabilities.

INTRODUCTION

The emergence of efficient communication systems led to the deployment of large scale software solutions for data acquisition and monitoring involving geographically distributed devices and computers. The notion of computer has changed over the past decade. There is a clear shift from the big tower desktop computer to much smaller devices having the equivalent hardware capabilities (CPU, memory etc.) to any typical computer and that can be easily used in various domains. In particular, Intelligent Electrical Devices (IED) run continuously and report their observations to centralized systems in various formats, generating large databases and leaving end-users with the task to reason about observations and to draw their own conclusions. Actually, users are interested to identify *typical configurations* based on observations provided by distributed devices, enhanced with additional information that they can get from other data sources. Then, users can make inferences and reason about these configurations and make informed decisions. These typical configurations are often called *patterns*. Several approaches and technologies have been proposed to handle the large amount of data generated by devices, as for example data mining techniques. These techniques aim at extracting interesting configurations from databases using statistical techniques and generate new patterns that can be used to query databases. However, these techniques present some drawbacks. First, accessing such large databases to find patterns is time consuming and may not be efficient for real time systems. Second, existing pattern languages are mostly SQL-based and are not expressive enough to represent patterns which are based on the concept of event. Third, current solutions run queries on offline data which is already stored in databases whereas users may be interested to detect pattern instances from real time data.

To overcome the above limitations, we propose a novel approach using Complex Event Processing (CEP). During the past ten years CEP has emerged as a research area which proposes new models and tools to manage event-based patterns. Originally, CEP has been used in the financial industry to predict phenomena such as market development and exchange rate trends. CEP patterns are identified relationships between event attributes such as *time*, *cause* (causal relation between events) and *aggregation* (significance of an event's activity in relation to other events). In this paper, we present a novel approach for pattern detection in power distribution systems using CEP technology. Thanks to this approach, a user will be able to simultaneously correlate events which are reported by IEDs and detect interesting situations. The proposed solution integrates CEP in the Smart Grid communication architecture according to the IEC Standard Group requirements [4]. A set of pattern examples are presented to illustrate the relevance of our approach.

The rest of the paper is organized as follows. After a review of previous works dealing with event processing in the distribution grid, we present a general definition of an event and show how it can be used in power distribution domain. Then, we propose an enhanced system architecture and we present an illustrative example to explain how CEP can be integrated in a substation communication system. Finally, we conclude the paper and we discuss some research outlooks.

RELATED WORKS

Current data acquisition systems used in the power distribution are the Supervisory Control and Data Acquisition (SCADA). They are used to monitor equipment status and to control the IEDs in substations. These SCADA are distributed applications and are built using client-server architecture [2]. Such architectures do not allow managing power systems which evolve and reacts dynamically and unpredictably. It embodies a

centralized authority and provides a persistent infrastructure that applications can reliably depend upon for coordinating the processing of specific tasks. The data acquired and stored by a SCADA does not allow for reacting about complex situations in real time. SCADA systems make use of relational databases so that accessing data is time consuming. Moreover, they offer limited query languages to identify situations of interest and to display them to the end-user. To overcome such limitations, few works attempted to use CEP to provide more powerful tools for real time management in the power industry. Srinivasagopalan et al [2] proposed an agent-based CEP system to manage the smart grid and enable efficient security management of the communication system. Wagner et al [1] used CEP to leverage the rich data sources present in the smart grid to quickly and efficiently react to events and manage complex situations. However, none of these works are concerned with the integration of a CEP in the existing solutions that manage substations. The original aspect of this paper is the discussion about the challenges of CEP integration with current distribution decision tools. In particular, we consider the decision tools available at the substation level. We also provide some examples of patterns that may be of interest to users.

COMPLEX EVENT PROCESSING

Let us first introduce the notion of event and how it can be used in the power distribution domain. Then we present the generic architecture of a CEP system.

Event definition

According to Luckham [7], an event is "*something that happens in reality*". An event occurs in a static background and may or may not change a state (A state is a situation describing stability and things that do not change). Events can be represented by programming entities. When events are similar in terms of structure and meaning, they can be associated with the same event type. Event types are specified using a common general structure, but they can also be customized by the user who can change event attributes. Etzion [3] proposed a general structure of an event type illustrated in Figure 1. In this structure, an event is defined using three main blocks: a *Header*, a *Payload* and *Event Relations*. Meta-information about the event, like its type, occurrence time and detection time is contained in the event header. Specific information about the event itself is contained in the event payload. Event relations are optional. They allow for defining a semantic relationship between different event types such as membership, generalization, and specialization [3].

The temporal attribute of events is of particular interest. Basically, events may have two temporal properties: detection time and occurrence time. *Detection time* is defined as the point in time when the event becomes available for processing. *Occurrence time* is the temporal point when the event occurs in real time. Usually, occurrence time is earlier than detection time. In the rest of this paper and for simplification purposes we assume that detection time is equal to occurrence time.

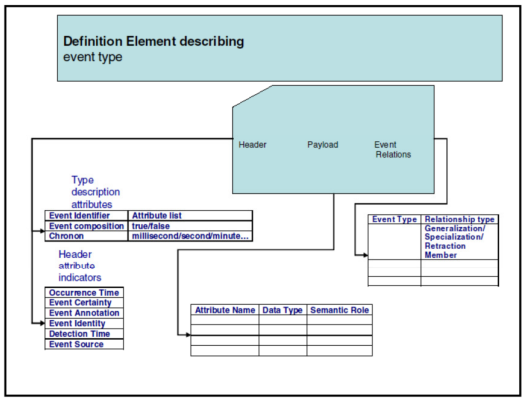

Figure 1: Event type definition [3]

In the Power Distribution domain, events can occur at different levels and in different applications. For example, smart sensors which are deployed in the field to monitor outage status can report events such as *over current*, *peak current*, *battery power failure* and so on. At the substation level, events can be generated from different data sources. They can be classified into three classes: operational data, non-operational data and event messages. Operational data represents the electrical behavior of the grid and includes data such as *voltage* and *current phasors*, *protection devices state change* from *open* to *close* and vice versa. Non-operational data represents asset conditions such as *power quality data* and *reliability data*. Event messages are generated by substation devices. They include data such as *loss of voltage* and *voltage restoration*. Figure 2 illustrates a partial view of an event message generated by the SEL-351 relay where event time attribute and analog inputs (voltage and current) are displayed. CEP systems can be used at the substation level to process such events and create patterns which help users make decision.

CEP: Definition and Architecture

Before presenting the CEP architecture, let us clarify the difference between Complex Event Processing and relational databases. Relational databases and the standard query language (SQL) are designed to manage static data. They can support complex queries. They are optimized for disk access since they mostly store data on disks. A query is used to retrieve data from a

database. If a system needs some data 10 times per second it must trigger the query 10 times per second. This does not scale up well to hundreds or thousands of queries per second [5]. Other solutions have been proposed to overcome these performance limitations such as database triggers which can be activated in response to database update events. However, database triggers tend to be slow and often cannot easily perform complex condition checking and implement the logic used to react to an event. [5]. A CEP can be thought of as a database turned upside-down. The CEP engine allows applications to store queries, run the data through and store the results in databases. Therefore, the CEP engine gives an answer in real-time when conditions occur and match patterns. The execution model is thus continuous rather than only when a query is submitted.
A typical CEP architecture is depicted in Figure 3. Events are generated by *event producers* (input). They are time stamped and have some attributes as described in the previous section. The CEP engine uses temporal and logical operators as well as causal relationships to correlate events and send them to the *event consumer* (output). The correlation between events can be specified by user in a base of patterns. These patterns can be expressed using a scripting language.

```
=>EVE 2 L
2  01/27/14 17:44:20.296 TRIG    100       0   59.99 1   2
eventInfo64=130353182602960000
eventTime=1390844660 296
eventAscii=01/27/14 17:44:20.296 +0
EVE 2 L

Relay ABC           Date: 01/27/14    Time: 17:44:20.296
Station XYZ


FID=SEL-351-7-R312-V0-Z005005-D20030714       CID=3057

                                                                  Out  In
      Currents (Amps Pri)            Voltages (kV Pri)            1357 135
    IA    IB    IC    IN    IG    VAB    VBC    VCA     VS Vdc Freq  246A 246
[1]
 -0.31 -0.14 -0.40 -0.00 -0.84  0.000  0.000  0.000  0.000   0 60.00 .... ...
 -0.25 -0.37 -0.04  0.05 -0.66  0.000  0.000  0.000 -0.003  -0 60.00 .... ...
  0.62  0.18  0.14 -0.01  0.95 -0.003  0.000  0.003 -0.003  -0 60.00 .... ...
 -0.23  0.02  0.30 -0.05  0.09 -0.003  0.000  0.003  0.000   0 60.00*.... ...
```

Figure 2: An SEL-351 event example (partial view)

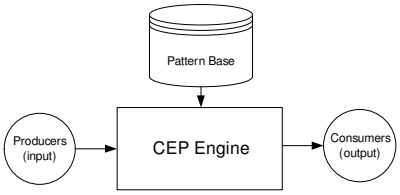

Figure 3: A typical CEP architecture

ESPER is a lightweight CEP system which offers an Event Pattern Language (EPL for short) to represent patterns using an SQL like language. EPL queries are created and stored in the pattern base and publish results to listeners when events are received by the engine or when timer events occur and match the criteria specified in the query [5].

CEP INTEGRATION TO A DECISION SUPPORT SYSTEM

Now we can discuss how a CEP engine can be integrated to a decision support system.
Three data classes (operational, non-operational and event messages) can be generated from the different electrical devices used in a substation. Operational and non-operational data is usually collected from all the devices using a *data concentrator* which converts the data through different communication protocols and forwards the information to the SCADA. For our experiments we used the Eaton's SMP SG4250 data concentrator which supports a variety of communication protocols [6]. An event is triggered when a binary input status changes from 0 to 1 or vice versa. In order to detect the occurrence of such events, a "wrapper" is used between the data concentrator and the CEP engine input. It creates event instances with necessary attributes (such as time stamp) and makes them ready for processing.

Event messages are generated by the IEDs and they are collected by an IED Management System (IMS). IMS is a system that retrieves event files from different devices using industrial protocols (DNP3, IEC61850, etc.) or using device proprietary protocols (such as SEL Events from Schweitzer Engineering Laboratories). For our experiments we used Eaton's IED Management Suite which collects events in COMTRADE format and stores them in a server disk. A wrapper is used between the IMS and the input of the CEP engine to detect the occurrence of event messages and to convert some of the event attributes according to the CEP event type definitions. Figure 4 illustrates an overview of the proposed architecture. The IMS system is located at the substation level but can be located at the enterprise level too. The CEP engine and the Event Converter modules are located at the enterprise level to connect them to multiple substations. The Pattern base is populated by the user who gets the detected patterns from the CEP engine. Other applications may also be connected to the CEP output to use the detected pattern for further processing such as an Outage Management System or an Energy Distribution System.

Some of the pattern instances are presented in the following examples. A pattern that detects the occurrence of a *relay open event* at the transformer relay device for 1 minute can be represented by the following script.

```
Every TransformerRelayOpen
(Device="SEL487") where timer:within(60 seconds)
```

We note the use of the *every* operator which indicates that the pattern sub-expression should restart when the sub-expression qualified by the every keyword evaluates to true or false. Using this operator guarantees that pattern sub-expression does not stop when the pattern sub-expression evaluates to true or false. The *timer:within()* is a pattern guard which acts like a stopwatch. If the associated pattern expression does not turn true within the specified time period it is stopped and returns false.

A pattern that detects the occurrence of a relay open event at the transformer relay device followed by a feeder relay open from one of two distinct devices can be represented by the following script. The "->" sign denotes the temporal operator *followed by* and specifies that the left hand expression must turn true first, and that only then the right hand expression will be evaluated for matching events.

```
Every TransformerRelayOpen -> (
FeederRelayOpen(Device="Relay1") or
FeederRelayOpen(Device="Relay2")) )
```

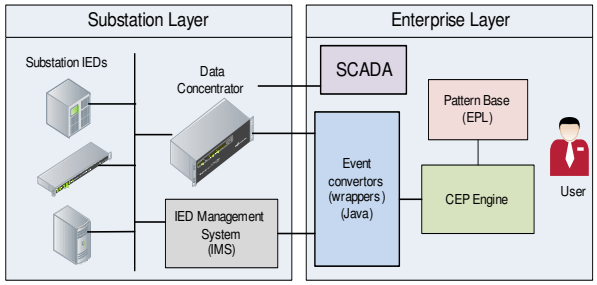

Figure 4: Overview of the proposed architecture

Another pattern instance can be detected using event messages using the temporal operator *followed by*. The event messages have three time attributes: *event time* is the real occurrence time of the event. The *IED time* is the time of the IED when it time-stamped the event and the *scan time* is the time when IMS system received the event from the IED. We use event time to reason about temporal operators in the pattern definition. The *followed by* temporal relation is used to correlate two consecutive events. The CEP engine triggers the pattern at the first occurrence of the *MinorPowerEvent* and waits until the occurrence of the second one which is generated by a different data source. The events are not stored in the data base (DB) before the pattern is evaluated to *true* which makes the pattern detection reactive to real time data.

```
Every MinorPowerEvent Device"SEL321-1" ->
MinorPowerEvent(Device"SEL321-2").
```

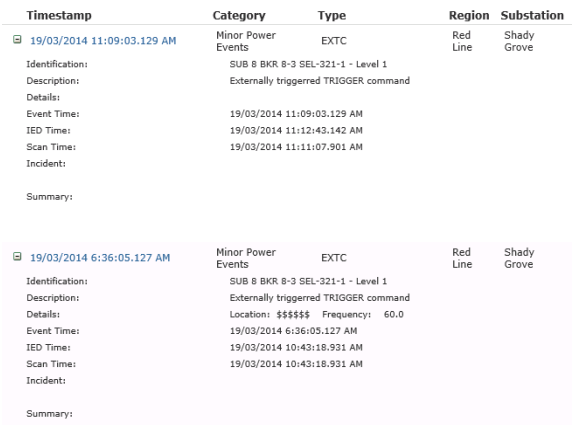

Figure 5: an example of two event messages

From the above examples, two aspects make CEP an efficient pattern detection system for power distribution networks. First, event types allow for associate the different data sources with a sort of "semantics" and make the event classification easy to handle by the end-user. Second, events are not stored in the DB if they are under the pattern engine evaluation. This is an efficient way to deal with real time data compared to traditional relational databases.

DISCUSSIONS AND CONCLUSION

In this paper, we tried to address one of the issues related to big data in the power distribution networks. Using a CEP based architecture, it is possible to leverage the data collected from different data sources to enhance user's decision capabilities in a fast and efficient way. Patterns are defined using an enriched SQL like language and stored in a pattern base. The detection of pattern instances is carried out in real time and is based on the correlation of the incoming events. Using such an architecture allows for reducing the gap between what the user wants to see in terms of complex situations and what current systems offer in terms of data.

Some challenges need still to be addressed to improve the proposed solution. First, event messages have to be converted into a specific event format to be processed by the CEP engine. Current conversion methods only allow to extract event attributes such as *type* and *time*. It would be interesting to develop new algorithms to extract other data from the COMTRADE files and to enrich events with additional information about the observations. Second, some of the complex situations are location-based. Events can be generated from different locations (substations) and may be correlated using spatial operators to form patterns. A spatial extension of ESPER is proposed in [8] and can be used for this purpose. This integration would open the door to

other use cases such as pattern detection in Outage Management Systems where outage sensors are geographically distributed.

REFERENCES

[1] A. Wagner, D. Anicic, R. Stühmer, N. Stojanovic, A. Harth, and R. Studer, 2010, "Linked Data and Complex Event Processing for the Smart Energy Grid", *Proceedings of Linked Data in the Future Internet at the Future Internet Assembly*.

[2] S. Srinivasagopalan, S. Mukhopadhyay, R. Bharadwaj, 2012, "A Complex-Event-Processing Framework forSmart-Grid Management", *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support, New Orleans, LA*

[3] O. Etzion, N. Zolotorvesky,2010, *Spatial Perspectives in Event Processing*, In From active data management to event-based systems and more, Kai Sachs, Ilia Petrov, and Pablo Guerrero (Eds.). Springer-Verlag, Berlin, Heidelberg 85-107

[4] The Smart Grid Standard, http://smartgridstandardsmap.com/.

[5] Esper, http://esper.codehaus.org/

[6] Eaton, www.eaton.com

[7] D.C. Luckham, 2002, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley. USA.

[8] F. Barouni, B. Moulin, 2012, An Extended Complex Event Processing Engine to Qualitatively Determine Spatiotemporal Patterns, *Proceedings of Global Geospatial Conference 2012 Québec City, Canada,*