Towards a Quality of Service Based Complex Event Processing in Smart Grids

Epal Njamen Orleant
University of Grenoble, LIG
Saint Martin d'Hères, France
Email: Orleant.Epal-Njamen@imag.fr

Christine Collet
Grenoble INP, LIG
Saint Martin d'Hères, France
Email: christine.collet@grenoble-inp.fr

Genoveva Vargas Solar
LIG-LAFMIA
Saint Martin d'Hères, France
Email: genoveva.vargas@imag.fr

***Abstract*—This paper investigates distributed event processing in smart grids, considering the following quality of service (QoS) dimensions: event priority, network occupation, memory occupation, and notification latency. It proposes an architecture for a QoS based distributed event processing. Our approach considers that the event processing is implemented as a network of operators that are executed by distributed event processing units. We also investigate on strategies used by event processing units in order to address QoS requirements.**

Keywords–Complex event processing; Quality of service; Smart Grids.

I. INTRODUCTION

Computer systems are more and more distributed (smart grids, sensor networks, cloud based applications, etc). In general, the complexity to manage or supervise distributed systems increases with the number and type of participanting systems (potentially geographically separated), which continuously generate data. Those data can be considered as events that refer to happenings of interest produced within the system environment.

In most cases, the capacity of monitoring and supervising a distributed system relies on the capacity to process low level events, for inferring higher level events, semantically rich for end user applications. This process includes events filtering, aggregations, correlations, windowing, and other computations on events. Infrastructures able to achieve this are referred to as complex event processing systems[1–5].

For example, in a smart grid, smart meters and sensors generate different types of event streams. Let us consider for example CoverOpenAlert and BadVoltage event types, the former being generated each time the cover of a smart meter is opened, and the latter being generated each time a bad voltage is detected by a sensor over an electrical line. An application may be interested in the sequence of CoverOpenAlert and a BadVoltage occurring at the same place, within a two minutes time window. This pattern detects suspicious activities (MeterSuspected event type) on smart meters. The detection of such a high level event includes event filtering (type and attribute based filtering), windowing and temporal correlation.

The production of event streams in distributed contexts, associated with the need to quickly process them to have an aggregated view of a system's state, requires the definition of complex event processing systems like [1–5]. Moreover, those systems should be able to be deployed in distributed architectures. They must efficiently achieve event filtering, correlation, aggregation and composition while adapting to their environment in terms of the multiplicity of data sources (sensors, smart meters, existing databases, etc.) and applications quality of service (QoS) [6].

a) Multiplicity of data sources: Distributed systems like smart grids consist of different types of components that can act as event producers or consumers, with different interaction modes (synchronous or asynchronous, push or pull based style), as illustrated by sensors, smart meters, existing databases. The diversity of interaction modes, coupled with the difference in data formats make it difficult to integrate data from different producers for event processing purposes.

b) Applications quality of service (QoS): The need to detect and notify complex events from basic events is sometimes correlated with some quality of service requirements like latency, memory consumption, network occupancy, event priority, notification latency, etc. Those QoS requirements generally constrain the way the event processing must be achieved. In addition, they are not independent of each other. For example, the reduction of network occupancy generally decreases the notification latency. Therefore, the trade-offs among these QoS metrics has to be done according to application requirements. Existing systems are limited in the sense that they fail to associate QoS preoccupations to event processing.

The problem we address in this paper can be summarized as follows: given smart grid needs in terms of event composition and QoS, how to provide the complex event processing system that best fulfills expected QoS requirements?

Our approach considers an event based abstraction of smart grids functions and services. This abstraction allows to reason on the smart grid in terms of event streams that are generated by smart grid components. In order to identify relevant or critical situations (complex events) among those event streams, we propose a distributed complex event processing architecture. The event processing logic is implemented as a network of operators that have to be executed by distributed event processing units. We also investigate on strategies applicable to event processing units in order to address the following QoS dimensions: event priority, network occupation, memory occupation and notification latency.

The remainder of the paper is organized as follows: Section II presents some related works. Section III presents the overview of our approach for QoS based complex event processing in smart grids, Section IV presents the model and architecture of our approach. Section V discusses the QoS adoption in event processing and finally, Section VI concludes the paper.

 ISBN: 978-1-61208-406-0

II. RELATED WORKS

Many works have been achieved on event streams analysis and composition, and many event processing systems have been proposed so far [1–5], either for centralized or distributed architectures.

In centralized architectures, the generated events are processed by a single node acting as an event processing server [2][4][5][7][8]. This requires event streams to be routed to that server node, which increases the latency of the event processing, overloads the network and the server, which can become a point of failure. Therefore, this approach is not suited for distributed contexts.

In distributed architectures, the event processing logic is performed by a set of distributed communicating nodes, each one achieving a part of the work. This offers a better scalability and availability than centralized approaches. Some distributed event processing systems are [1][3][9][10].

References [11] and [6] identify some QoS dimensions (latency, priority, etc.) relevant for distributed event processing, but they do not propose mechanisms for their adoption. However, some other systems provide QoS support. They optimize the query processing according to a particular objective, and differs from each others by the adopted QoS dimension. For example, [1] has focused on reducing the network traffic whereas [9] studied energy consumption. In wide networking environments, it is not reasonable to expect that all applications share the same objective. In our approach, we identify a set of QoS properties relevant for event processing in smart grids, and we study their adoption by the event processing system.

Reference [12] presents a survey on the QoS requirements of the smart grid communications system. It focuses on the functionalities that have to be provided by the smart grid communication infrastructure in order to address application requirements. References [13] and [14] study the QoS adoption in the smart grid communication network. The former proposes to add QoS by providing differentiated service for data traffic with different priority at the MAC (Media Access Control) layer, while the latter propose GridStat, a publish-subscribe middleware framework that has been designed to meet the QoS requirements for the electric power grid. In our work, we assume the existence of QoS support at the networking layer (e.g, message priority) in order to propose a complex event processing system in smart grid that deals with event priority, memory occupation, network occupation and notification latency.

III. APPROACH OVERVIEW

Our approach to integrate complex event processing technologies into smart grids is presented in Figure 1. It consists in three layers of abstraction, namely the smart grid, event streams, and event processing network layers.

- The smart grid layer consists in the real physical smart grid architecture, which includes telecommunication based devices such as smart meters, sensors, data concentrators, etc. Those devices are connected by communication networks technologies including power line communications, wireline communications or wireless communications [15]. The smart grid is described in terms of information being used and exchange between functions, services and components. This layer of abstraction is referred to as the *Information layer* in the smart grid reference architecture model [16]. In our approach, information is seen as events that happen within the smart grid.
- The event stream layer, which considers that data generated by smart grids components are event streams. In that level, smart grid components act as sources which can generate different types of events in a continuous manner. The event type and event stream models considered in this work are presented in Section IV.
- The event processing network layer consists in a set of distributed event processing units which are connected by event channels. It is created according to complex event subscriptions, and its deployment may be distributed across multiple physical networks, computers and software artifacts. The complex event subscriptions can be tagged with applications QoS requirements such as event priority and notification latency. Those QoS requirements have to be translated into contraints applicable to event processing units at execution time. In addition to those constraints derived from QoS requirements, the smart grid infrastructure itself (processing devices and network technologies) dictates other constraints such as ressources (memory, CPU) limitations and network occupation limitations.

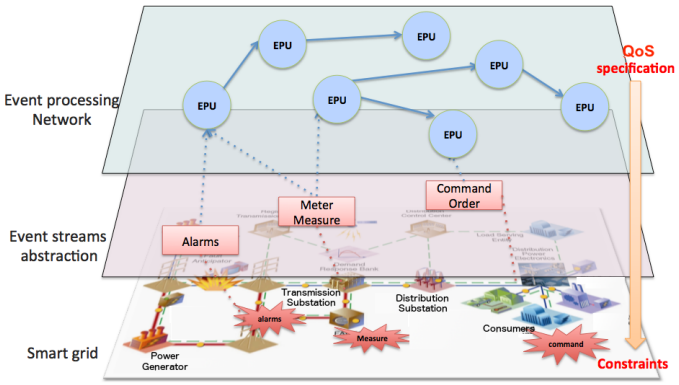

Figure 1. Approach overview

IV. MODEL AND ARCHITECTURE

This section presents the event model (event type and event stream), and the runtime architecture of our approach (event processing network).

A. Event Model

1) Event Type: An event type represents a class of significant facts (events) and the context under which they occur. The defintion of an event type includes the attributes presented in Table I.

The *typeName* attribute refers to the name of the event type. The *producerID* attribute refers to the id of the entity who produced the event occurence. The *detectionTime* attribute refers to the time at which the event occurence has been detected by a source. The *productionTime* attribute refers to the time at which the event has been produced (as a result of a processing on others events) by an event processing unit.

 ISBN: 978-1-61208-406-0

TABLE I. EVENT TYPE ATTRIBUTES

Name	Type
typeName	String
producerID	String
detectionTime	Number
productionTime	Number
notificationTime	Number
receptionTime	Number
priority	Number
context	Set<Attribute >

The *notificationTime* attribute refers to the time at which the event is notified to interested consumers. The *receptionTime* refers to the time at which the event is received by an interested consumer. The *priority* attribute represents the priority value associate to the event occurrence. The context (*context* attribute) of an event type defines all the attributes that are particular to this event type. They represent the others data manipulated by the producer which are relevant to this event type. For example, the context of a *MeterMeasure* event type generated by a smart meter includes the *voltage* and *current* attribute.

An event instance (or simply event) is an occurrence of an event type. An event type can be simple or composite.

Simple event types are event types for which instances are generated by producers (sensors, smart meters, etc.). They are not generated as results of processings on others events. In the example considered in Section I, *BadVoltage* and *CoverOpenAlert* are simple event types. More generally in a smart grid, the event types include *Alarms*, *MeterMeasure* and *SensorMeasure* generated by electric devices and such as smart meters and sensors, and *Command*, *ControlOrder*, *ControlAction* generated by utility applications.

Complex (or composite) event types are event types for which instances are generated as results of processings on others events. Reference [17] includes a set of operators applicable to events. They capture particular situations (relevant or critical) that can be inferred from occurrences of others events. Those situations have to be notified to utility applications so that the system can be automatically or manually controlled. In the same example, *MeterSuspected* is a complex event type. Complex event types can also capture aggregated values, like the daily electricity consumption of a household. This can be generated as result of an aggregation on a one-day window of *MeterMeasure* event instances.

2) Event Stream: An event stream is a continuous, append-only sequence of events. We note $Stream(s,T)$ the stream of events of type T generated by the source s. If S is a set of sources, then $\{\bigcup stream(s,T), s \in S\}$ defnes a stream of events of type T, denoted $Stream(T)$.

B. Event Processing Network

As introduced in Section III, the event processing logic is implemented by the event processing units. The runtime deployment of event processing units with associated event channels is called the event processing network [18][19]. This is illustrated in Figure. 2.

The general vision of our QoS based complex event processing system can be briefly described as follows: applications subscribe to composite events by issuing complex event patterns to the system, with associated QoS requirements. The system then deploys a set of distributed event processing units, which apply different strategies to meet QoS requirements during event processing. The complex events generated by the event processing units are notified to consumers. In a smart grid, such an infrastructure can act as a middleware on which utility applications can rely for detecting interesting or critical situations (sensors errors, alarms, etc.) over the electrical grid, with some QoS guarantees (e.g., priority, notification latency, etc.).

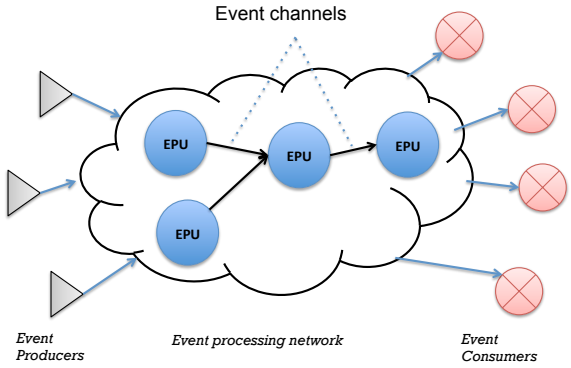

Figure 2. Event processing network

1) Event Processing Unit: An event processing unit can be defined by three types of components (see Figure 3):

- a set of input queues, on which parts of input event streams are maintained.
- an operator, which implements a three step event processing logic: *fetch-produce-notify*. In the first step (fetch), some events are selected from the input queues and marked as ready to be used to produce new composite events. In the second step (produce), the events selected at step 1 are used to produce new composite events according to the operator semantic. The composite events produced are stored in the output queue. In the third step (notify), events in the output queue are notified either to other event processing units or to interested consumers.
- an output queue, which contains events to be notified.

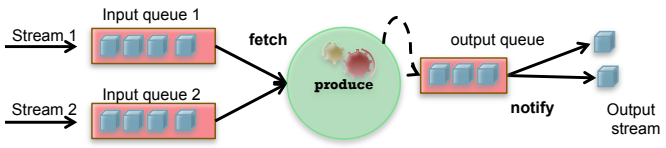

FIGURE 3. EVENT PROCESSING UNIT

2) Event Channel: Event processing units communicate through event channels. Event channels are means of conveying events [20]. This can be done via standard tcp or udp connections, or a higher level communication mechanism like publish/subscribe [21] or group communication [22] provided by a middleware layer.

V. QOS SUPPORT IN EVENT PROCESSING

A. QoS Dimensions

The need to detect and notify complex events from basic events is sometimes correlated with some QoS requirements. The QoS dimensions we proposed to address in this paper are event priority, memory occupation, network occupation and notification latency.

1) Event Priority: Event priority defines a priority order between events. In some contexts, there may exist priorities between events that have to be captured by the event processing runtime. For example in a smart grid, a BadVoltage event can be higher priority than a CoverOpenAlert event. Events that have a higher priority have to be processed and notified earlier than less priority events.

2) Memory Occupation: Differents devices may have different available memory capacity. To adapt the event processing to the memory capacity of each device, it must be a way to specify the maximum memory occupation incurred by an event processing unit at the execution time. The memory occupation constraint gives an upperbound on the number of events that an event processing unit can maintain in its main memory at execution time.

3) Network Occupation: Event processing units can be distributed accross different locations. The communication between those event processing units is achieved by messages sending at the notification step. The underlying network may be overloaded by a high event notification rate among event processing units. The network occupation constraint gives an upperbound on the number of networked event notifications per time unit on a given event processing unit.

4) Notification latency: In the common practice for power device protection, the circuit breaker must be opened immediately if the voltage or current on a power device exceeds the normal values. The notification latency of an event is the time elapsed between its production and its notification to interested consumers (end users or event processing units). The notification latency constraint imposed on an event processing unit defines an upper bound on the notification latency of events produced by that event processing unit.

B. QoS Adoption in Event Processing

Event processing networks are created from application event subscriptions. Those event subscriptions are specifications of event types (simples or complex) that applications are interested in, with their associated QoS requirements. The QoS requirements defined by smart grid applications constrain the way event processing and notification must be achieved. To address those application level QoS requirements, we have to provide:

- an operator placement algorithm that ensures load balance between the available processing devices while minimizing the end to end latency. A first step will be to have a look at works presented in [23][24], and adapt them in our context.
- strategies applicable to event processing units allowing to ensure that high priority events will be processed and notified earlier than less priority events. An idea consists in adopting priority queues for implementing event processing units input/output queues, and provide a mapping function between event priotity and message prioritization, if available in the underlying networking technology.
- strategies to adapt the event processing unit ressources (memory and network) utilisation in a consistent way wich respect to existing notification latency constraints.

VI. CONCLUSION

This paper shows that monitoring of smart grids can be done using an event based approach where event streams generated by distributed sources are processed by distributed event processing units. Such units may produce complex events indicating interesting or critical situations that are notified to interested consumers. Since the invocation of business critical processes is now triggered by events, the QoS of the event processing infrastructure becomes a key issue. We have identified key QoS dimensions relevant to smart grids, and proposed the first step towards a QoS based event processing system.

ACKNOWLEDGMENT

This work was carried out as part of the SOGRID project (www.so-grid.com), co-funded by the French agency for Environment and Energy Management (ADEME) and developed in collaboration between participating academic and industrial partners.

REFERENCES

[1] G. Cugola and A. Margara, "RACED : an Adaptive Middleware for Complex Event Detection," Proceedings of the 8th International Workshop on Adaptive and Reflective MIddleware, 2009, pp. 1–6.

[2] "Homepage of Esper," 2015, URL: http://esper.codehaus.org/ [Accessed: 2015-03-27].

[3] "Homepage of TIBCO StreamBase," 2015, URL: http://www.streambase.com/ [accessed: 2015-03-27].

[4] D. Gyllstrom, E. Wu, H.-J. Chae, Y. Diao, P. Stahlberg, and G. Anderson, "SASE: Complex Event Processing over Streams," 3rd Biennial Conference on Innovative Data Systems Research (CIDR), 2006, pp. 1–5.

[5] "Homepage of Oracle CEP," 2015, URL: http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/index.html [accessed: 2015-03-26].

[6] S. Appel, K. Sachs, and A. Buchmann, "Quality of service in event-based systems," in CEUR Workshop Proceedings, vol. 581, 2010, pp. 1–5.

[7] A. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. White, "Cayuga : A General Purpose Event Monitoring System," CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, 2007, pp. 412–422.

[8] D. Luckham, "Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Orderings of Events," Stanford University, Tech. Rep., 1996.

[9] O. Saleh and K.-U. Sattler, "Distributed Complex Event Processing in Sensor Networks," 2013 IEEE 14th International Conference on Mobile Data Management, Jun. 2013, pp. 23–26.

[10] P. Pietzuch, B. Shand, and J. Bacon, "A framework for event composition in distributed systems," Proceedings of the ACM/IFIP/USENIX, vol. 2672, 2003, pp. 62–82.

 ISBN: 978-1-61208-406-0

[11] S. Behnel, L. Fiege, and G. Mühl, "On quality-of-service and publish-subscribe," in Proceedings - International Conference on Distributed Computing Systems, 2006, pp. 1–6.

[12] Y.-h. Jeon, "QoS Requirements for the Smart Grid Communications System," Journal of Computer Science and Network Security, vol. 11, no. 3, 2011, pp. 86–94. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6038941

[13] W. S. W. Sun, X. Y. X. Yuan, J. W. J. Wang, D. H. D. Han, and C. Z. C. Zhang, "Quality of Service Networking for Smart Grid Distribution Monitoring," Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, 2010, pp. 373–378.

[14] H. Gjermundrø d, D. E. Bakken, C. H. Hauser, and A. Bose, "GridStat: A flexible QoS-managed data dissemination framework for the power grid," IEEE Transactions on Power Delivery, vol. 24, no. 1, 2009, pp. 136–143.

[15] W. Wang, Y. Xu, and M. Khanna, "A survey on the communication architectures in smart grid," Computer Networks, vol. 55, no. 15, Oct. 2011, pp. 3604–3629.

[16] CEN-CENELEC-ETSI Smart Grid Coordination Group, "Smart grid reference architecture," 2012, URL: http://ec.europa.eu/energy/gas_electricity/smartgrids/doc/xpert_group1_reference_architecture.pdf [accessed: 2015-03-27].

[17] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," ACM Computing Surveys (CSUR), vol. V, no. i, 2012, pp. 1–70.

[18] L. Perrochon, W. Mann, S. Kasriel, and D. C. Luckham, "Event Mining with Event Processing Networks," in Methodologies for Knowledge Discovery and Data Mining. Third Pacific-Asia Conference, PAKDD-99 Beijing, China, April 2628, 1999 Proceedings, 1999, pp. 474–478.

[19] G. Sharon and O. Etzion, "Event-processing network model and implementation," IBM Systems Journal, vol. 47, no. 2, 2008, pp. 321–334.

[20] "Event processing glossary version 2.0," 2011, URL: http://www.complexevents.com/2011/08/23/event-processing-glossary-version-2/ [accessed: 2015-03-27].

[21] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," ACM Computing Surveys, vol. 35, no. 2, 2003, pp. 114–131.

[22] G. V. Chockler and R. Vitenberg, "Group Communication Specifications : A Comprehensive Study," ACM Computing Surveys, vol. 33, no. 4, 2001, pp. 427–469.

[23] G. T. Lakshmanan, Y. Li, and R. Strom, "Placement Strategies for Internet-Scale Data Stream Systems," IEEE Internet Computing, vol. 12, no. 6, Nov. 2008, pp. 50–60.

[24] C. Thoma, A. Labrinidis, and A. J. Lee, "Automated operator placement in distributed Data Stream Management Systems subject to user constraints," 2014 IEEE 30th International Conference on Data Engineering Workshops, Mar. 2014, pp. 310–316.

 ISBN: 978-1-61208-406-0