Real-time Congestion Representation using Event Driven Architecture

Yemula Pradeep Kumar[1*], S. A. Khaparde[*], Rushikesh K. Joshi[!], P. Pentayya[#]
[*]Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India.
[!]Department of Computer Science, Indian Institute of Technology Bombay, Mumbai, India.
[#]General Manager, Eastern Regional Load Despatch Center, Kolkata, India.
ypradeep@iitb.ac.in, sak@ee.iitb.ac.in, rkj@cse.iitb.ac.in, ppentayya@gmail.com

Abstract - That event driven architecture (EDA) will compliment the data driven systems in future is evident from the current literature. In this paper, the use of event driven architecture (EDA) as a platform for supporting event oriented applications in power control centers is presented. EDA operates on the semantics of events and event patterns, which are easy to decipher, flexible, extensible and conform to the standard. The concept is explained with real-time congestion management procedure recently introduced in Indian power sector. The semantics of events pertaining to real-time congestion management are defined in terms of states and state transitions of an abstract state machine. The state machine is then used to define a minimal set of basic and complex event patterns, which are implemented using an open source event processing tool. The results show that the congestion state of the system is correctly identified, and can be used as decision support. The case study is indicative of a wider set of event oriented applications that can be supported by EDA at control centers and constituents.

Keywords - Power control centers, event driven architecture (EDA), complex event processing (CEP), state machines, Real-time congestion management

1 Introduction

CONVENTIONAL supervisory data acquisition systems (SCADA) and energy management systems (EMS) are “data oriented”, which operate on the semantics of data. SCADA systems typically provide measurands and alarms to the operators. Limit violation events are detected as alarms and handled at control centers. Alarms result in a notice popping up on operator screen. Automated alarm processing coupled with automated response are available to a limited extent. Operators expertise is needed in inferring and taking decisions on high level events. Commands are passed on-line or sometimes over phone lines to the operators in the field to implement the control decision. Whenever a certain data or visual is needed, corresponding queries are executed and the results are obtained. The information is made available on *pull model*, in which the receiver of the data has to initiate the request for data by sending a query. The application of such system is suitable for reporting purposes. The receiver may only be interested to receive the data when the data has changed. But in pull model systems, there is no way to know when the data has changed without actually querying for the data and receiving the same.[1] Hence, the pull configuration is not optimal for applications like monitoring, event processing and real-time decision support. An *event oriented* system is suitable for such applications. Event oriented systems are based on *push model* wherein, the source of information would initiate the communication and actively send (or push) the data to the receivers. The receivers of the data would register and subscribe with the source for select classes of information. Event driven architecture (EDA) is a software architecture pattern promoting the production, detection, consumption, and reaction to events [1]. Literature on use of EDA in power system operation is limited, vis-a-vis the extensive use of EDA in various other domains such as finance, banking, manufacturing and retail [2–4]. The need for moving from data architecture to event architecture in the EMS environment has earlier been reported in [5]. OLE process control - unified architecture (OPC UA) is a standard for event handling widely used in power industry. OPC UA provides a unified data model for pull-based and event-driven data exchange among automation devices and programmable logic controllers [6]. However, as opposed to device level events, the focus of this paper is to address high level semantic events, which are derived from aggregation of data from multiple sources. A mathematical model of representing and analyzing power system events has been presented in [7]. As event driven systems essentially operate on semantics of *events*, unambiguous and system wide consistent definitions of high level events are to be established [8]. This is achieved by defining an ontology for high level events in the domain, which specifies the event classes, their definitions, relationships and attributes. The event ontology, thus defined, can be used for standardization of event information, similar to the approach followed for standardization of power system data by common information model (CIM). EDA can be used as a flexible and extensible platform for supporting various event oriented algorithms proposed in the literature, having crucial applications in the areas of intelligent monitoring, avoiding cascading failures, prevention of blackouts, system restoration [9–13].

In this paper we describe the design and implementation of an event driven system which originates from the design paradigm of EDA. As a case study, a system to perform functions required by a recent CERC regulation [14] titled “Measures to relieve congestion in real-time operations” is presented. This CERC regulation is chosen for the study because its implementation, by def-

[1]Corresponding Author: Yemula Pradeep Kumar, Address: Field Computations Lab, Department of Electrical Engineering, Indian Institute of Technology Bombay, Powai, Mumbai, 400076, India. Phone: +91 22 2576 4424, Fax: +91 22 2572

inition, has to satisfy certain time based performance criteria that can be conveniently handled by an event driven system. Although the case study is based on congestion representation in Indian power sector, similar concept can be employed in other major power grids such as European gird or US power grid, wherein the semantics of high level events and the context can be driven by respective system operation requirements.

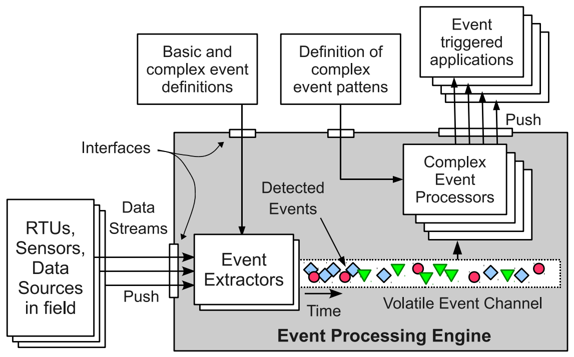
The paper is organized as follows. In section 2, brief description on structure of an event driven system and methodology is given. In section 3, we provide relevant background and overview of the system operation and regulatory scenario in Indian power sector. Section 4, then presents briefly the responsibilities of the regional load despatch centers towards successful implementation of the real-time congestion management system. This establishes the specific requirements to be addressed. In section 5, we then give a state machine representation of the real-time congestion management procedure and establish a minimal set of high level events. In section 6, the implementation of the system is described along with the results, followed by conclusion in section 7.

2 Event driven architecture: Methodology

An event is defined as a significant change in the state of a parameter. An event can be defined in object oriented paradigm as an event class along with its attributes. The *timestamp* is obviously the most important attribute of any event, along with other quantitative or qualitative attributes as required. Complex events are defined by combining basic events using causal, temporal, spatial and logical relations such as *and, or, not, followed by, effected by, within time window* and *state machines*. The complex event definitions are at the center and the real time data stream is passed over these event definitions and monitored continuously. As and when a specific complex event is detected, required services can be triggered. An interesting question here is whether EDA can detect unforeseen events. Unforeseen events are, by definition, not semantically defined a priori. Due to this reason, EDA would not be able to detect unforeseen events. However, as the designer learns about such events, they can be incorporated into the EDA for future detection.

2.1 Components of EDA

The core of EDA comprises of an *event processing engine* (see Fig. 1). The engine performs all the computational tasks involved in processing the events. It also contains a temporary memory to address the representation requirements of event patterns which span over time. The engine also provides interfaces for plugging in data or event sources and event triggered applications. The designer of the system is responsible for specifying the definition of basic events and complex events and their patterns that are needed to be monitored. The engine operates on *Push-Push* model, the data has to be pushed into the engine and after its processing the engine in turn pushes the notifications to the subscribing event triggered services.



Basic and complex event definitions
Definition of complex event pattens
Event triggered applications
Push
Interfaces
Complex Event Processors
Detected Events
Data Streams
RTUs, Sensors, Data Sources in field
Event Extractors
Push
Time
Volatile Event Channel
Event Processing Engine

Figure 1: Block Diagram of Event Driven Architecture (EDA)

2.2 Event processing functions

The events are stored for processing into volatile *event channels* grouped by the class of the event and these memory is made accessible for the event processing functions. Thus, event channels act as temporary memory for the extracted events. An event processing engine performs the following functions on the instantiated objects: Event distribution, event transformation, event pattern matching (transform, aggregate, compare, reject, enrich, project, split), event pattern discovery.

2.3 Event processing language (EPL)

The functions to be performed by the engine are specified by *event processing statements* which are contained in the complex event processors. These statements are written in Event processing language (EPL) [15] which an SQL-like language. Streams replace tables as the sources of data with events replacing rows as the basic unit of data. Thus, with the knowledge of EPL the user can change the behavior of the complex event processor without the need of knowing the implementation details of the engine. This is analogous to the way we interact with databases through SQL queries without having to know implementation details of database. This feature of the event processor, allows for the flexibility and extensibility of the EDA architecture, because changes can be made without having to re-program the system.

2.4 Event processing tools

Various commercial as well as open source tools are available for event processing and EDA implementation. Some of the widely known tools for event processing are Tibco event processor, IBM business events, Apama event processor, Esper, etc [16–19]. A comparative study of capabilities, features of various event processing tools available in the market is carried out in [20]. For this work we have used an open source tool called esper for representation of real-time congestion representation procedure. This is presented in detail in section 6.

3 Background on System Operation and Regulatory Authorities

In this section we provide a brief background of the power sector and regulatory structure in India. Following the electricity act in 2003, the power sector in India has been undergoing rapid restructuring and deregulation. The central electricity regulatory commission (CERC) is the apex regulatory body in Indian power sector [21]. CERC is entrusted with the operation of national load despatch center (NLDC) and regional load despatch center (RLDC) who are vested with the responsibility of promoting competition, efficiency and economy in bulk power markets. Power system operation corporation limited (POSOCO) is the system operator of the country, entrusted with the responsibility of integrated operation, control of the national power grid and carry out the energy scheduling, accounting and commercial settlements thereof among the participating utilities in the national grid [22]. POSOCO operates the national grid through a hierarchy of control centers at national and regional levels. At state level respective state load despatch centers are the system operators. There are about 35 control centers at national, regional and state transmission levels distributed geographically on a well defined operation and control hierarchy with specific roles and objectives at each level [23]. The control centers at state, regional and national levels are known as state load despatch center (SLDC), RLDC, and NLDC respectively. A comprehensive description of information network of Indian power sector as their information exchanges are reported in [24].

Availability based tariff (ABT) introduced since 2002 has led to change in the operating philosophy of the grid from centralized mode to a more participative mode, with commercial incentives and disincentives for achieving grid discipline. Short term open access (STOA) has been introduced for power to be traded from any utility to any utility across the country on a non-discriminatory basis. All these changes have resulted in increased demand on the roles and responsibilities of system operators in power control centers. Traditional SCADA/EMS systems being used in the control centers were commissioned in a centralized fashion by PowerGrid (central transmission utility) in the early part of this century when all the state utilities had not yet unbundled. Vendor dependent non standard legacy SCADA/EMS systems with proprietary software and proprietary communication protocols are not suitable in the fast changing regulatory environment. Owing to the rigid nature of the existing architectures the control centers are finding it increasingly difficult, time consuming and expensive, to incorporate new services into their existing systems.

4 Real-time Congestion Management in India

This section gives a brief description of the structure of the national grid, its control areas and interconnecting corridors, in the context of the real-time congestion management procedure under implementation.

System Description: The national grid is demarcated as five regional grids namely, Northern (NR), Eastern (ER), North-Eastern (NER), Western (WR), and Southern (SR) regions, with six inter-regional corridors. This structure of interconnected regional grids forming national grid is shown in Fig. 2. Operation of such a large interconnected system has unique challenges and opportunities as described in [25]. Each region in turn comprises of multiple generation and load entities or control areas. Each control area is bounded by metering and has a schedule (injection/drawal) issued by SLDC/RLDC under whose jurisdiction the control area falls. The power transactions among the control areas are categorized under long term access (LTA), medium term open access (MTOA) and short term open access (STOA). The STOA transactions include both bilateral (over-the-counter (OTC)) and power exchange transactions. The bilateral STOA transactions have four products, namely, advanced (up to 3 months), first come first serve (up to 1 month), day ahead, and same day. All the above contracts are deemed to be physical and firm. There are no financial contracts in Indian energy markets yet. Therefore, a robust scheduling mechanism aggregating all the above types of contracts has been put in place and drawal schedules or injection schedules of each of the constituents are prepared and published in respective regional load despatch websites by 11pm everyday. These schedules are used as datum line for calculation of deviations known as unscheduled interchange (UI). The role of real-time balancing of active power is fulfilled by UI mechanism under ABT.

Real-time Congestion Management Procedure: The detailed procedure for real-time congestion management is described in the CERC order available at [14]. NLDC/RLDC would assess the inter-regional and intra-regional total transfer capability (TTC) and available transfer capability (ATC) 3 months in advance. They would then display on their websites the actual real time power flow Vis-a-Vis the TTC and ATC declared and the voltages upstream/downstream [22]. The system is deemed to have entered *Congested State* in case of any of the following criteria (i) Grid voltage downstream/upstream moves beyond operating range or (ii) real time power flow exceeds the ATC for one time block (15 min) for the congested corridor or (iii) loading of any transmission line beyond its operating limit. SLDCs on noticing congestion can inform RLDC who in turn will inform NLDC. In case actual flow in inter/intra regional corridor exceeds ATC continuously for two time blocks (30 min), NLDC/RLDCs would issue a *Warning Notice*. In case of continued persistence of congestion and no action from defaulting constituents, event after two time blocks, notice of *Application of Congestion Charges* is issued.

Applicability of Congestion Charges: Congestion charges are applicable simultaneously on all entities upstream and downstream. Charges are payable by entities causing congestion and receivable by entities relieving congestion. Congestion charges may be applied pan India being applicable for entities in one region causing intra-regional congestion in a separate region. Once applied the charges would remain in force for 8 time blocks (2 hours). Congestion charges would not be applied for more than four times in a day to prevent continued flip-flops.

Who causes congestion? An interesting aspect in real-time congestion management is the identification of entity to be charged for causing congestion. Indian grid is allowed operated frequency with in a band of 49.5 Hz to 50.2 Hz (floating frequency rather than fixed frequency). UI is either charged or rewarded as per its tendency to stabilize the frequency. Thus a constituent may choose to overdraw, during a high-frequency regime. This in turn may cause congestion in real-time. Hence a clear signal for relieving congestion is needed which does not conflict with the already existing ABT mechanism. Such a scenario is depicted in Fig. 2. Here A, B, C, D, X and Y are six constituents present in various regions as shown. At the given instant constituents A, C, and X are found to be under drawing (U/D) from the grid, which means the net drawal of the constituent is less than its schedule. At the same instant, constituents B, D, and Y are found to be over drawing (O/D) from the grid. It is also found that the WR-NR corridor and ER-NR corridor are congested. If the frequency of gird is below the rated 50 Hz, we may infer from the flows that A, C and Y are causing the congestion in the system. But if the frequency is above 50 Hz, constituent Y is justified in overdrawal under ABT. Thus, to be consistent with ABT, constituents overdrawing at frequencies above 50 Hz shall be exempted from congestion charges. At frequency below 50 Hz, congestion charge would be levied for over drawal or under-injection in the importing control area and at frequencies above 50 Hz, congestion charge would be levied for under drawal or over-injection in the exporting control area.

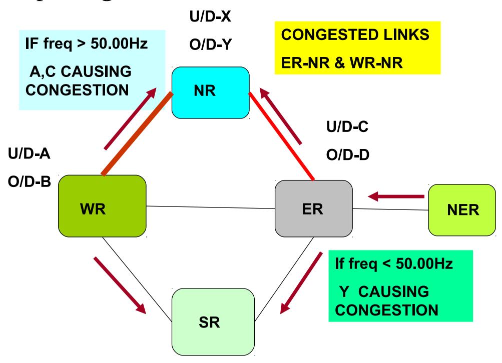


Figure 2: The inter-regional grid in a congested state. The identification of constituent causing congestion and application of congestion charges is dependent on frequency.

Event driven system: Real-time congestion management procedure requires continuous monitoring of the power flows over all the inter-control area tie lines and corridors, keeping track of events such as flow on which line exceeded the available transfer capability (ATC), when and for how long. It is also needed to determine the upstream and downstream control areas responsible for causing or alleviating this congestion, within the time frame prescribed by the regulation. Such an application is highly event driven and thus processing via an event driven system is a suitable solution over manual monitoring. The scope of the event driven system proposed in this paper is limited only to identify and represent the *state* of the system and its changes over time. The automation of this task alone would save a considerable effort on the part of operators, who otherwise need to keep track of states manually by monitoring multiple continuous data streams.

5 State Machine Formulation

In this section we attempt to model the requirements described above using an abstract state machine (ASM) representation [26]. A state machine, is a mathematical abstraction sometimes used to design computer programs. It is a model composed of a finite number of states, transitions between those states, and actions to be performed while transition, entry or exit from a state. To develop the state machine representation for the real-time congestion management problem we follow step wise procedure:

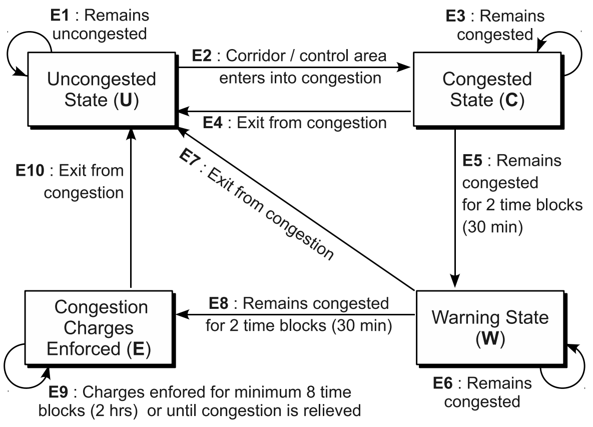

E1 : Remains uncongested
E3 : Remains congested
E2 : Corridor / control area enters into congestion
Uncongested State (U)
Congested State (C)
E4 : Exit from congestion
E10 : Exit from congestion
E7 : Exit from congestion
E5 : Remains congested for 2 time blocks (30 min)
Congestion Charges Enforced (E)
E8 : Remains congested for 2 time blocks (30 min)
Warning State (W)
E9 : Charges enfored for minimum 8 time blocks (2 hrs) or until congestion is relieved
E6 : Remains congested

Figure 3: State Machine of the Real-time Congestion Management Mechanism

Step 1: We begin by identifying a finite set of mutually exclusive states that the system could exist in, from the perspective of real-time congestion. The identified set of states are uncongested, congested, warning, and charges enforced denoted by set S of letters as $S = \{U, C, W, E\}$ respectively. The states are so chosen that at any given time the system would exist in one of these states. We then identify the necessary conditions for transition from one state to other, represented as arrows pointing from current state towards the new state. These are denoted as events $E1$ to $E10$ as shown in Fig. 3. Assuming we begin our analysis from the state U, there is only a possibility of either event $E1$ or $E2$ to occur with effect that we remain in state U or move to state C respectively. Similarly all other state transitions are depicted in Fig. 3.

Step 2: Now, up on further examination we can observe that the events $E1$ to $E10$ above are all deducible from the occurrence or non-occurrence of two fundamental events denoted as $E_A \equiv$ *entering congestion* and $E_B \equiv$ *exiting from congestion*. This step is important because it simplifies the implementation. These events are not the opposites of each other, but have their own negations. Non occurrence of these fundamental events can also be treated as events and are represented as $\neg E_A \equiv$ *not entering congestion* (or remaining uncongested) and $\neg E_B \equiv$ *not exiting congestion* (or remaining congested). The logic for deducing the events identified in Fig. 3 is expressed in the form: initial state $\rightarrow$ fundamental event. The $\rightarrow$ is read as "followed by". Thus we have the following deductions: $E1 = U \rightarrow \neg E_A$; $E2 = U \rightarrow E_A$; $E3 = C \rightarrow \neg E_B$; $E4 = C \rightarrow E_B$; $E5 = C \rightarrow \neg E_B(30min)$; $E6 = W \rightarrow \neg E_B$; $E7 = W \rightarrow E_B$; $E8 = W \rightarrow \neg E_B(30min)$;

$E9 = E \rightarrow \neg E_B$; and $E10 = E \rightarrow E_B$. The occurrence (or the non-occurrence) of the fundamental events can be readily checked from the data streams available. For example events $\neg E_A$ and E_A are defined as below

$$\neg E_A \;=\; \{x_i^{flow} \leq x_i^{ATC}, \quad \forall i\} \qquad (1)$$

$$E_A \;=\; \{x_i^{flow} > x_i^{ATC}, \quad forany \quad i\} \qquad (2)$$

Where x_i^{flow} is the time varying flow on i^{th} corridor or control area and x_i^{ATC} is its corresponding time varying ATC. Hence, these events shall be used as the input source of sequence of events to run the state machine. The set of possible events used as inputs for the state machine is typically represented by Σ. Thus we have $\Sigma = \{E_A, \neg E_A, E_B, \neg E_B\}$. Eqns (1) and (2) form the logic for event extractors which convert data streams to events.

Step 3: The next step is to form a state transition table which is the tabular representation of state-transition function: $\delta : S \times \Sigma \rightarrow S$ where S is a finite, non-empty set of states. This is shown in Table 1.

δ	U	C	W	E
E_A	C	$-$	$-$	$-$
$\neg E_A$	U	$-$	$-$	$-$
E_B	$-$	U	U	U
$\neg E_B$	$-$	C	W	E
$\neg E_B(30min)$	$-$	W	E	$-$

Table 1: State Transition Table: First column is the input event received by the state machine, first row is the current state and the table contents show the new state.

Step 4: A state machine implementation pseudo code is typically represented as a *switch-case* statement. In this section we give the pseudo code for the real-time congestion state machine designed so far. The generalized code for state machine is to use the states as the cases for switching, and the triggers and state changes are coded in each block of the case. Thus the following is the pseudo code

```
+===================================================+
      State Machine implementation pseudo code
+===================================================+
  Switch (state) {
    case "U":
         if (congestion = true){
         state = "C"; increment congestion_timer;}
    case "C":
         if (congestion = true)
         increment congestion_timer;
         else {
         state = "U"; reset congestion_timer;}
         if (congestion_timer > 30) {
         state = "W"; reset congestion_timer;}
    case "W":
         if (congestion = true)
         increment congestion_timer;
         else {
         state = "U"; reset congestion_timer;}
         if (congestion_timer > 30) {
         state = "E"; reset congestion_timer;}
    case "E":
         increment congestion_timer;
         if ((congestion_timer > 120)
         and (congestion = false))
         state = "U"; reset congestion_timer;}
         else state = "E";
  }
+===================================================+
```

6 Case Study and Results

As a case study, we now present an event driven system for identification of real-time congestion in the national grid. We refer the readers to Fig. 1 which shows the block diagram for a typical EDA system. Using the block diagram we describe implementation of various components of Fig. 1 for identifying the congestion states, in following subsections.

6.1 *Data Collection from sources*

The raw data for this study is available on-line at [22]. The event driven system is designed to run on-line in a continuous execution mode. However, for the development and testing purpose we have downloaded a data set for a time period of 21 hours (or 1260 min) and used the CEP engine on the data set in off-line mode. The advantage is that, instead of its regular speed of one data snapshot every minute we can send the data to the engine at much faster rates and study the results without waiting for 21 hours. This is particularly required in the development and testing phase. The problems of loss of data due to the intermittent nature of the website is addressed by simple linear interpolation techniques. The data set has the start time of $2010/11/06 - 07 : 00 : 00$ and an end time $2010/11/07 - 03 : 59 : 00$, with duration of 21 Hrs. The data is collected every 1 min. Hence for the time duration we have 1260 records. The data points include flows on 12 corridors and their corresponding ATC values. Import and export values of 5 corridors, along with their ATC values. Thus the entire data set can be represented as a matrix of size 1260 rows by 44 columns (= 12+12+10+10). All numerical values are in MW.

6.2 *Basic and Complex Events*

As described in section 5. The basic events defined for the system are E_A: entering congestion and E_B: exiting congestion. Complex events can further be expressed in terms of basic events as described in previous section. An open source event processing tool called Esper is used as the CEP engine. Event processing language (EPL) is used to specify the necessary event patterns. Complex event statements written in EPL monitor real time data streams. The use of an event processing language such as EPL, facilitates the description of events and event patterns to be made seperate from the actual programmatic implementation of these patterns in the software code. When a complex event is detected, the required services can be triggered resulting in autonomous response in an asynchronous environment. From the raw data collected above we extract events by checking if at the given time instant the flow on any of the control area exceeds the ATC, as per equations (1) and (2).

6.3 *Event Triggered Application*

For the purpose of this case study the detected events are routed to the output or console window of the coding

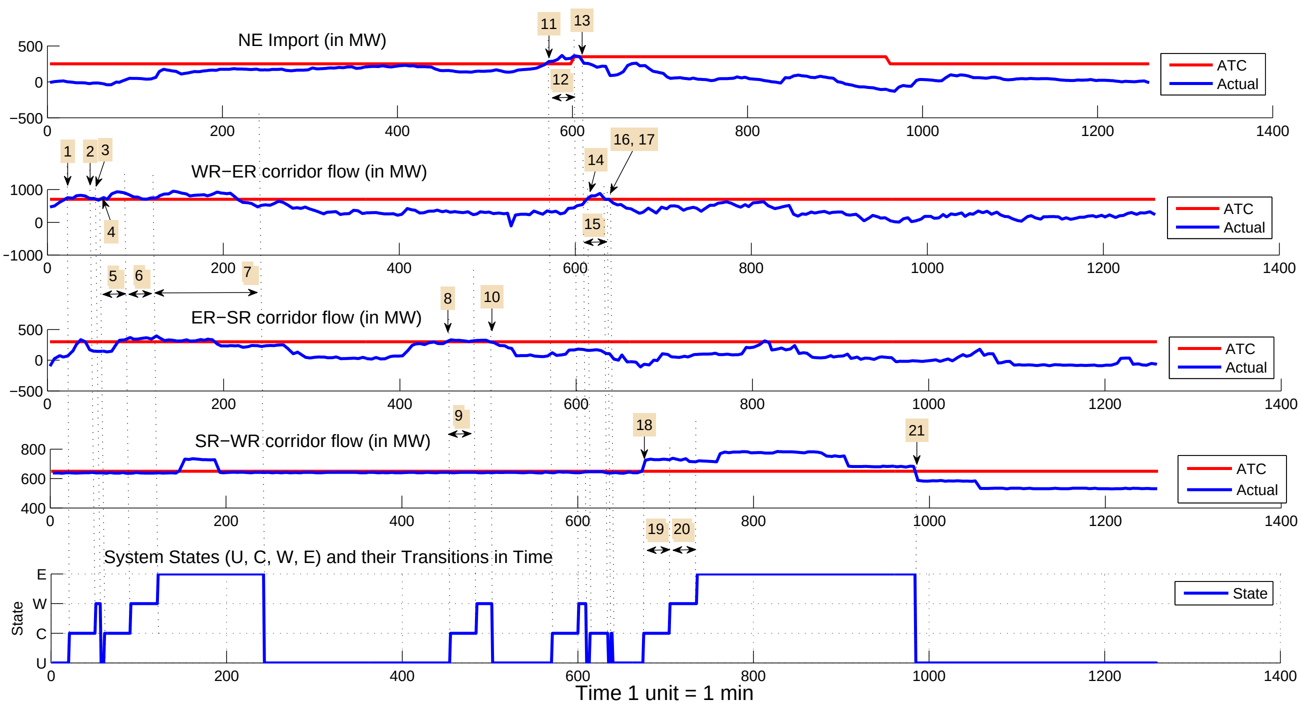


Figure 4: Verification of results: Out of the 22 possible control areas (12 corridor flows, 5 regional imports and 5 regional exports), only flows of 4 control areas are found to be exceeding the ATC limits (see above 4 plots). The resulting states identified by the event driven system are shown in last plot. The numbered markers refer to the event IDs shown in Table. 2.

Time	Event ID	Description of Basic Event	Duration	State Transition (Complex Event)
07:20	1	WR-ER corridor Congested	-	$U \rightarrow C$
07:50	2	WR-ER corridor Warning notice	30 min	$C \rightarrow W$
07:56	3	System Uncongested	6 min	$W \rightarrow U$
08:00	4	WR-ER corridor Congested	4 min	$U \rightarrow C$
08:30	5	WR-ER corridor Warning notice, ER-SR corridor Warning notice	30 min	$C \rightarrow W$
09:01	6	WR-ER corridor Charges Enforced, ER-SR corridor Charges Enforced	30 min	$W \rightarrow E$
11:02	7	System Uncongested	120 min	$E \rightarrow U$
14:34	8	ER-SR corridor Congested	212 min	$U \rightarrow C$
15:04	9	ER-SR corridor Warning notice	30 min	$C \rightarrow W$
15:22	10	System Uncongested	18 min	$W \rightarrow U$
16:30	11	NE Import Congested	68 min	$U \rightarrow C$
17:00	12	NE Import Warning notice	30 min	$C \rightarrow W$
17:09	13	System Uncongested	9 min	$W \rightarrow U$
17:13	14	WR-ER corridor Congested	4 min	$U \rightarrow C$
17:34	15	System Uncongested	19 min	$C \rightarrow U$
17:37	16	WR-ER corridor Congested	3 min	$U \rightarrow C$
17:39	17	System Uncongested	2 min	$C \rightarrow U$
18:14	18	SR-WR corridor Congested	35 min	$U \rightarrow C$
18:44	19	SR-WR corridor Warning notice	30 min	$C \rightarrow W$
19:15	20	SR-WR corridor Charges Enforced	30 min	$W \rightarrow E$
23:24	21	System Uncongested	249 min	$E \rightarrow U$

Table 2: Identified complex events as state transitions.

environment. Each detected state transition is printed at the output as a line of text including the time stamp, state transition, duration, control area, and whether entering or exiting congestion. The results obtained for the entire time window of 21 hours are summarized in Table 2. These results obtained by the event driven system can be verified by manual observation, by referring to the plot of the line flows for the congested corridors.

It is found that out of 22 control areas only 4 have been congested at different spans of times, as seen from the first 4 plots of Fig. 4. This can be verified by the state transitions detected by the system in the last plot of the Fig. 4. In on-line implementation phase, the information of current state of the system can be made available to the operators in real-time, thereby saving efforts on the part of operators.

7 Conclusion

In this paper the use of event driven architecture as a platform for supporting various event oriented applications is explained. In the absence of such high level event platform, applications operating on the semantics of data are needed to obtain event information by hard coded extraction of events from data, thereby leading to rigid system. By operating on semantics of events, EDA offers the flexibility and extensibility by processing events, detecting patterns and generating push based notifications to the applications. Statements written in event processing language (EPL) can be used to configure the working of EDA. As an illustrative case study, we have presented the state machine modeling, representation, and detection of complex event patterns in real-time congestion management procedures being followed in Indian power sector. The results demonstrate that the EDA based solution can automate the process of identification of congestion state of the system, which otherwise needed significant monitoring efforts by operators. The application of event processing systems can be used to automate similar event oriented tasks. Such high level event processors are poised to find wider applications as they are currently in the initial stages of application in the power systems domain.

REFERENCES

[1] P. Dekkers, Complex event processing, Masters thesis, Radboud University Nijmegen, 6500 HC Nijmegen, Netherlands, Sep. 2007. [Online]. Available: http://dist.codehaus.org/esper/CEP_MasterThesis_PaulDekkers_200709.pdf

[2] A. Adi, D. Botzer, G. Nechushtai, and G. Sharon, “Complex event processing for financial services”, IEEE Services Computing Workshop, pp. 7 – 12, Sep. 2006.

[3] C. Zang, and Y. Fan, “Complex event processing in enterprise information systems based on RFID”, Enterprise Information Systems, Feb. 2007, vol. 1, no. 1, pp. 3 – 23.

[4] C. Zang, Y. Fan, and R. Liu, “Architecture, implementation and application of complex event processing in enterprise information systems based on RFID”, Information Systems Frontiers, Nov. 2008, vol. 10, no. 5, pp. 543 – 553.

[5] J. P. Britton, Moving from data architecture to event architecture in the EMS environment, IEEE Power Engg. Soc. General Meeting, vol. 3, pp. 2740 – 2747, 12 - 16, June 2005.

[6] K. Walzer, J. Rode, D. Wunsch, M. Groch, “Event-Driven Manufacturing: Unified Management of Primitive and Complex Events for Manufacturing Monitoring and Control”, Workshop on Factory Communication Systems, 2008.

[7] H. Zhao, Z. Mi, and H. Ren, Modeling and analysis of power system events, IEEE Power Engg. Soc. General Meeting, 18 - 22, June 2006.

[8] Y. Pradeep and S. A. Khaparde, “Complex Event Processing of High Level Events in Multi-area Power Grid: An Indian Perspective”, IEEE Power Engg. Soc. General Meeting, July 2010.

[9] D. S. Kirschen and B. F. Wollenberg, “Intelligent alarm processing in power systems,” Proc. of the IEEE, vol. 80, no. 5, pp. 663 – 672, May. 1992.

[10] T. S. Dillon, “Survey of expert systems is alarm handling,” Electra, Report of CIGRE - Task Force WG38-06-02, no. 139, pp. 133 – 147, 1991.

[11] S. J. Ranade, K. Ramchander, and J. Mitra, “Identification of Chains of Events Leading to Catastrophic Failures of Power Systems”, Circuits and Systems, 2005. (ISCAS). IEEE International Symposium on, vol. 5, pp. 4190 – 4187, May. 2005.

[12] C. C. Liu and J. Li, “Patterns of Cascaded Events in Blackouts”, Summary of Panel Session on Blackouts, in proc. IEEE Power Engg. Soc. General Meeting, July 2008.

[13] M. M. Adibi and L. H. Fink, “Overcoming restoration challenges associated with major power system disturbances”, IEEE Power and Energy Magazine, pp. 68 – 77, Sep. 2006.

[14] CERC, Measures to relieve congestion in real-time operations, June. 2010. [Online]. Available: http://210.212.2.107/Detailed\%20Procedure\%20for\%20Relieving\%20Congestion\%20in\%20Real\%20Time\%20Operation.pdf

[15] Esper Documentation, Chapter 5, Event Patterns Dec. 2010. [Online]. Available: http://esper.codehaus.org/esper-2.1.0/doc/reference/en/html/event_patterns.html

[16] TIBCO Complex Event Processing. Dec. 2010. [Online]. Available: http://www.tibco.com/products/business-optimization/complex-event-processing/default.jsp

[17] IBM Business Events. Dec. 2010. [Online]. Available: http://www-01.ibm.com/software/solutions/soa/business_event_processing.html

[18] Apama CEP Platform. Dec. 2010. [Online]. Available: http://web.progress.com/en/apama/event-processing-platform.htm

[19] “Esper - reference documentation version 2.1.0,” EsperTech, Tech. Rep., 2008. [Online]. Available:http://esper.codehaus.org/esper-2.1.0/doc/reference/en/pdf/esperreference.pdf

[20] A. Medhekar, Event driven architecture for Indian power sector, Masters thesis, Indian Institute of Technology, Bombay, India, June 2008.

[21] CERC, Dec. 2010. [Online]. Available: http://www.cercind.gov.in/about_us.html

[22] NLDC, Dec. 2010. [Online]. Available: http://www.nldc.in/Congestion.aspx

[23] Power Grid Corporation of Indian Limited, “Unified Load Despatch & Communications Scheme”, Nomination for CSI-TCS Best IT Usage Award, National IT Awards, 2003.

[24] Y. Pradeep, C. N. Raghupathi, V. S. Warrier, S. A. Khaparde, “Towards Interoperability Standards in Indian Power Sector,” in Proc. GRID INTEROP FORUM Conference, Chicago, Dec. 2010.

[25] A. Roy, S. A. Khaparde, P. Pentayya, S. Usha, and A. R. Abhayankar, “Operating experience of regional interconnections in india,” in Proc. IEEE Power Engg. Soc. General Meeting, pp. 2528 – 2535, June 2005.

[26] E. Borger and R. Stark, “Abstract State Machines: A Method for High-Level System Design and Analysis”, Springer-Verlag, 2003.