User Interface for Situational Awareness of openPDC

Nick Gellerman, Prakash Ranganathan, Ranganath Vallakati and Anupam Mukherjee
Department of Electrical Engineering, University of North Dakota
Nickolas.gellerman@my.und.edu and prakash.ranganathan@engr.und.edu

Abstract—This paper uses complex phasor data (voltage, current and phase angle) from openPDC for situational awareness of the SmartGrid. Alarm notification features through E-mail and Short message service (SMS) on various parameters of Phasor Measurement Units (PMU) data has been developed in C# programming as an extension to the existing openPDC software functionalities. In addition, real-time display of PMU data based on the geographical locations of PMU is developed for openPDC software and is presented within a Google Maps-based Windows Forms application. The paper also discusses how to process the PMU data using a k-means clustering algorithm.

Index Terms—openPDC, Phasor Measurement Unit (PMU), Phasor Data Concentrator (PDC), k-means clustering.

I. INTRODUCTION

The world is becoming increasingly dependent on electrical power. Everything from transportation to communication is dependent on electricity, and when a brownout could mean life and death in a hospital situation, it is easy to see why electrical stability is so important. Synchrophasors, otherwise known as Phasor Measurement Units (PMUs), are the newest measurement tools that gather key sensor parameters such as voltage, frequency and phase angle information which are being used to monitor the power grid[1,2]. Situational awareness (SA) involves being aware of what is happening in the vicinity of the electric grid, in order to understand how information, events, and one's own actions will impact goals and objectives, both immediately and in the near future. Thus, situation awareness is especially important in smart grid domain, where the information flow can be quite high and poor decisions may lead to serious consequences. Our clustering and email adapter implementations can greatly reduce the event failures caused due to the man made errors. Without these sensors, situational awareness of the electric grid with fine details of grid monitoring is not possible.

PMUs provide time-synchronized measurements of the power system such as frequency, voltage phasors, current phasors and many other parameters which can be used for various applications such as state estimation, transient analysis, capacitor bank’s performance, analysis of load shedding schemes, and Inter-area oscillations [3,4,9]. PMU data can also be used for Generator Black Start, islanding and anti-islanding conditions with Distributed Energy Resources (DER), to measure transmission relay parameters [10,11,12]. A basic PMU can provide a minimum of 30 samples per second up to a maximum of 120 samples per second. With these many samples per second, every fine detail on how the power system is operating can be known. The system and grid operators (SGO) can actively observe abnormalities in the grid, and also respond to them.

This would allow rate of frequently occurring brownouts/blackouts to be prevented. Even though, PMU can provide you the information about the system, the SGO need to know how to gather, classify and analyze all this data (30 samples/ second), so that it can be used to prevent problems even before they occur. This research addresses the visualization of the real-time data collected by PMUs. The software developed in this research was based around openPDC, an open source Phasor Data Concentrator (PDC) [13]. OpenPDC functions by receiving data broadcasted by a PMU and concentrating it, enabling archiving, rebroadcasting, and analysis of the phasor data. The rest of the paper is organized as follows: Section II covers different features that have been added to this openPDC software. Section III deals with data classification algorithm called K-means clustering. Section IV summarizes results and Section V provides an insight on our future goals and concludes the paper.

II. EXTENSION OF OPENPDC FUNCTIONALITIES

The openPDC include a simplified configuration and management system that consists of three primary components[14]. They are:

1. The openPDC Manager
2. The System Configuration Editor
3. The secured remote console monitor

A generic block diagram showing the real-time stream of PMU data to the openPDC with integration of data base (SQL server), and extension functionalities is shown in Figure 1. A Graphical User Interface (GUI) based openPDC Manager Application is installed with openPDC itself which is our point of concern. The openPDC suite provides different options to view the historian data. Some of the options are:
1. Selection of sampling rate.
2. Selection of various parameters such as particular phasor, frequency.
3. Selection of multiple or unique PMU historian data

Generally, openPDC can play the historian data in the form of graph with selected parameters. To make more use of already available alarm adapter, we’ve utilized the adapter to alert any user of openPDC, if something goes wrong with the system. Thus, different types of adapters have been developed based on the alarm feature of openPDC and are discussed in the following sections. In addition, to improvise the user-interface of openPDC, we’ve created a location based data monitoring adapter which assists operators in analyzing the whole grid with multiple PMUs simultaneously. This will identify the location of various PMU on the map based interface. The details of Location Based Monitoring (LDM) implementation is discussed in the later section.

A. E-mail as Alarm Service for openPDC

One feature supported by openPDC is the ability to set alarms. For any type of data being broadcasted by a PMU (frequency, voltage phasors, current phasors, etc.,) an alarm can be set to trip whenever a measurement out of acceptable range is received. At that time, a data point indicating various data related to the alarm and measurement is logged. While these alarm points are archived, openPDC has little built-in functionality to inform the user that the alarm has been tripped. In an effort to address this issue, an add-on adapter for openPDC was developed. This E-mail adapter is programmed to automatically send emails to a specified email address or phone whenever an alarm is tripped.

This adapter was programmed in Visual Studio Professional 2012 in C# and compiled as a dll (dynamic-link library) [15]. The adapter functions by polling alarms that it has subscribed to at a user-defined value (frequency, magnitude, phase angle) and dynamically building an email string whenever an error code is received from one of these alarms. The adapter can be used simply by placing the compiled dll file in the openPDC installation folder. From there, it will be automatically loaded into the openPDC Manager the next time it is opened. It can then be found under Action > Manage Custom Actions.

<table>
<thead>
<tr>
<th>PARAMETERS LIST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FramesPerSecond</td>
<td>The number of data frames per second expected by the adapter.</td>
</tr>
<tr>
<td>LagTime</td>
<td>The Past time deviation tolerance (i.e., how long ago the timestamp can be).</td>
</tr>
<tr>
<td>LeadTime</td>
<td>The future time deviation tolerance.</td>
</tr>
<tr>
<td>PasswordCredential</td>
<td>The password used to log on to the SMTP server handling the email.</td>
</tr>
<tr>
<td>SMTP Host</td>
<td>The name of the SMTP server handling the email.</td>
</tr>
<tr>
<td>UsernameCredentials</td>
<td>The username used to log on to the SMTP server handling the email.</td>
</tr>
<tr>
<td>Dependencies</td>
<td>List of the adapter’s dependencies (should be left blank).</td>
</tr>
<tr>
<td>DependencyTimeout</td>
<td>The amount of time to hold measurements while waiting for dependencies.</td>
</tr>
<tr>
<td>InputMeasurementKeys</td>
<td>List of measurement processed by the adapter. All alarms to be monitored should be entered here.</td>
</tr>
<tr>
<td>OutputMeasurementKeys</td>
<td>List of measurements output by the adapter (should be left blank).</td>
</tr>
<tr>
<td>RecipientEmail</td>
<td>Comma separated list of email addresses to send alerts to (must have the same number of entries as Recipient Name).</td>
</tr>
<tr>
<td>RecipientName</td>
<td>Comma separated list of the names of the recipients to send alerts to (must have the same number of entries as Recipient Email).</td>
</tr>
<tr>
<td>SMTPPort</td>
<td>The port to access the SMTP server.</td>
</tr>
<tr>
<td>SupportsTemporalProcessing</td>
<td>Whether or not the adapter supports temporal processing. Should be set to false.</td>
</tr>
<tr>
<td>UseDefaultCredentials</td>
<td>Whether or not to use default credentials to access the SMTP server (overrides Password Credential and Username Credential).</td>
</tr>
</tbody>
</table>

Then, a new Email adapter can be created by filling out the parameters delineated in Table 1. Once the parameters are set and the adapter is enabled, it will start listening for alarms. Whenever one of the alarms specified by the “Input Measurement Keys” parameter is triggered, the Publish Frame method is called. This method extracts information from the alarm and formats a message to be sent to all of the specified recipients. The messages are formatted as follows:

```
Time: {local timestamp of out-of-bounds measurement}
Name: {name of the alarm that tripped}
Threshold: {threshold being tested}
Operation: {operation used to test measurement}
Severity: {user-defined severity of alarm}
Description: {description of out-of-bounds measurement}
```
Figure 2. A snapshot of an E-mail alert

An alarm has triggered.

openPDC
Thu 7/17/2014 12:37 AM

To: Gellerman, Nickolas

Time: 7/17/2014 5:37:04 AM
Name: TESTALARM
Threshold: 299300
Operation: Greater than or equal to
Severity: Information
Description: Shelby Bus 1 + Voltage Magnitude

Figure 2 show a screenshot of an E-mail alert sent to the subscriber, when the voltage magnitude went beyond the acceptable limits. The value presented in the screenshot (299300) is in volts. The user can keep alarm alerts for all the parameters by defining the acceptable ranges in the openPDC.

B. Text message as Alarm Service

Similar to an E-mail alarm, another adapter was developed which sends out a short message service (SMS) to the user’s cell phone number. An email can be sent to a cell phone if the email address is formatted as follows: {10-digit phone number}@{domain}, where the domain is unique to the service provider. Figure 3 shows a snapshot of the text alert that has been sent to the subscriber’s mobile phone using the text adapter which is also concerned to the voltage magnitude.

Figure 3. Snapshot of the short message service alert

C. Location-based Data Monitoring (LDM)

The next objective of this paper was to develop a more graphical way of analyzing the data. OpenPDC already support Data vs. Time graphs for real-time and historian data, but offers little else for data analysis. To expand upon this goal, a Windows Forms application was developed to allow for real-time observation of the data transmitted by a PMU according to its geographical position.

Since it is a Windows Forms application, this program is entirely event-driven upon startup. It begins by automatically connecting to the data stream sent out by openPDC on the localhost server. Next, it extracts the geographic coordinates of each of the connected devices and builds a Mercator projection centered on the centroid of those coordinates. A Google Static Maps URL is then built from this information, and the resulting image is displayed in the background. Finally, the real-time data is displayed at the proper map location as a circle whose color is related to a numeric value deciphered by using the scale on the right side of the screen[16]. The upper and lower bounds of this scale can be modified by the user through the NumericUpDown boxes on the right side of the window.

Figure 4 shows a screen shot of this location based data monitoring interface. The location of three PMUs with their corresponding heat maps around them is shown in Figure 4. By hovering on a particular PMU, the application will display the details such as frequency, rate of change of frequency (ROCOF), voltage and current phasor values at that particular instant of location. Data relayed by a PMU can also be observed separately by selecting different relayed parameters such as frequency, magnitude of different phases etc. from a dropdown box of the top left of the window. If multiple values of a certain type are collected by a single PMU (i.e., multiple current phasors), the user can cycle through them by clicking on the appropriate data circle on the map.

Figure 4. Screenshot of Location-based data monitoring interface

III. CLASSIFICATION OF PMU DATA USING CLUSTERING TECHNIQUE

Another feature of this application is the ability to process archived data[17]. The classification technique used herein is called as “k-means clustering.”K-means clustering algorithm is one of the most widely used techniques for cluster analysis in data mining community [18]. Simply put,
k-means clustering is a process by which data with similar values are combined into groups. Currently, this analysis is available with frequency and phasor data and the user can only specify use of data from the last 10, 30, or 60 minutes.

Figure 5. Display of Voltage data using K-means clustering

Once the k-means algorithm has executed, the program graphs the data points above their respective cluster in blue while plotting the centroid in red. Figure 5 represents the graph plotted for voltage. The figure shows the cluster of points (in blue) around the centroids (in red).

```csharp
    do
    {
        changed = false;
        // Empty out clusters
        foreach (List<double> cluster in clusters)
        {
            cluster.Clear();
        }
        // Distribute data points through the clusters
        foreach (double datum in data)
        {
            closest = 0;
            // Find nearest centroid for each data point
            for (int i = 1; i < numOfClusters; i++)
            {
                if (Math.Abs(datum - centroids[closest]) > Math.Abs(datum - centroids[i]))
                {
                    closest = i;
                }
                clusters[closest].Add(datum);
            }
            // Find the new centroids for each cluster
            for (int i = 0; i < numOfClusters; i++)
            { if (clusters[i].Any() && clusters[i].Average() != centroids[i])
                { centroids[i] = clusters[i].Average();
                    changed = true;
                }
            }
        }
        while (changed);
    }
```

Figure 6. Pseudo code for centroid computation for k-means clustering

The pseudo code that performs the k-means algorithm is shown in Figure 6. Due to the restrictions on the page limits for this paper, the entire coding structure is not presented here. The algorithm takes the first 35734 samples from openPDC for the last 10 minutes to perform the k-means clustering. Figures 5, 7 and 9 shows the clustering performed on voltage magnitude, frequency and phase angle respectively. X- axis represents the number of cluster that are being formed during each process while the Y- axis represents the parameter that is being clustered. Frequency is typically stable and does not deviate more than a few decimal points during steady state conditions. This is the reason for the formation of clusters mostly around 59.9 Hz as seen in Figure 7.

Figure 7. Display of frequency data using K-means clustering
Figure 8. Flowchart showing PMU data analysis in OpenPDC.

Figure 8 represents data handling processes in the form of a flowchart. At any point of time, OpenPDC will be flooded with streams of data. This data received contains all the parameters from various PMU units that are connected to the openPDC. Therefore, the first decision block will be sorting out PMU data based on the PMU device. Once the data is sorted from a PMU, it is now sorted based on the type of parameter.

Then, the data of a particular parameter (frequency or voltage magnitude) is chosen, processed and with the help of device parameters (latitude, longitude etc.), a location based heat map is displayed. Location based heat map will be of great use to the system operators. At the same time, we can cluster the data from archives by considering different parameters such as number of samples or a particular time frame which can then be used to analyze the data for future predictions.

![Figure 8. Flowchart showing PMU data analysis in OpenPDC.](image)

![Figure 10. Unit circle phase Angle representation](image)

However, phasor data is time-varying and we believe that by using clustering approaches will aid in visualizing and grouping the data, and identifying anomalies with ease as shown in Figure 9. Currently, the authors are working toward representing the phasor data for voltage and current in an unit circle representation for easier visualization as seen in the figure 10 showing three-phase signals.

IV. RESULTS
The adapters developed such as E-mail and Message alerts have been successfully tested, and its response time to the user or system operator is within a delivery time of less than 1 second. However, this delivery time is rather more dependent on the TCP/IP packets speed and the carrier connections. Table 2 provides the information computational performance of k-means clustering algorithm. All the
processes ran and tested on the following computer configuration:

A) CPU–Intel i5-2450M  
B) Cache – 3 MB  
C) System Memory–8GB RAM

<table>
<thead>
<tr>
<th>No. of data (f, V, phase angle ) Samples</th>
<th>Average time (milli seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2.8</td>
</tr>
<tr>
<td>100</td>
<td>4.2</td>
</tr>
<tr>
<td>500</td>
<td>3.1</td>
</tr>
<tr>
<td>1000</td>
<td>4.7</td>
</tr>
<tr>
<td>10000</td>
<td>21.9</td>
</tr>
</tbody>
</table>

To compute the computational performance of the clustering approach, a clock function called as “Stopwatch class” has been included in the algorithm. Column 1 tells the number of samples that have been considered while performing k-means clustering while the column 2 show the average times taken by the algorithm to process the samples respectively with trail run of 10 times.

V. CONCLUSION

The paper provide ways to increase the situational awareness of the grid using the OpenPDC data. Two new functionalities such as E-mail and SMS adapters were added to the openPDC environment and its implementation details have been investigated. In addition, the paper presents a clustering approach using k-means to group multiple data types, and allowing for arbitrary selection of the time-period to examine. As part of future work, the email and short message service adapter will be formatted to the openPDC development standards with improved clustering schemes. We plan to expand the work using other type of clustering schemes such as Support Vector Machine (SVM) [19], and Streaming techniques [20] for further study using different test cases.

ACKNOWLEDGMENT

We thank Professor Ranganathan and the SmartGrid & Data mining group at the University of North Dakota (UND), Department of Electrical Engineering for sponsoring this research activity.

REFERENCES