Detecting Anomalies by Data Aggregation in the Power Grid

Hoang Nguyen  
Department of Computer Science  
University of Illinois at Urbana-Champaign  
Email: hnguyen5@uiuc.edu

Klara Nahrstedt  
Department of Computer Science  
University of Illinois at Urbana-Champaign  
Email: klara@cs.uiuc.edu

Abstract—The August 2003 Blackout event showed that the PowerGrid is vulnerable to the cyber attacks. The event also showed the need for automatic detection of abnormal behaviors in the PowerGrid. In this paper, we propose a solution for the problem of detecting anomaly behaviors such as reporting incorrect status of the devices of the PowerGrid. First, we apply the non-parametric Cumulative Sum algorithm for quickest detection of the value-changing problem. Then, to improve the speed of detection, we explore the use of two data aggregation schemes: average-aggregation and quantize-aggregation. Our analytical and simulation results show that end-to-end detection delay should include not only the semantic computation of the data but also the underlying communication networks. Finally, we show that it is possible to improve the end-to-end detection delay by data aggregation.

I. INTRODUCTION

The August 2003 Blackout event made us all aware of the great vulnerabilities in the current PowerGrid [1]. Although the blackout started at one substation, where overloads on lines were mistaken for faults by the local automated controls, the major issue was the malfunctioning software. The erroneous software prevented the system from properly reporting the status data to the operators which led to inappropriate decisions and ultimating to the major blackout event.

Although the event has been an accident, it is of serious concern that if this event was an attack, as it shows serious failures in security and if not repaired, it could happen again in the future. The event clearly shows that the PowerGrid is vulnerable to cyber attacks since it is indistinguishable if the software is malfunctioning or infected by malicious viruses. Actually, FirstEnergy company in Ohio reported that one of the nuclear power plants was infected by Slammer virus in January 2003. F-Secure company also warned that the Blaster virus might cause the sensors to deliver delayed or wrong data on the status of the grid. All of the cases above show that we need secure solutions that can detect such malicious behaviors for the PowerGrid.

There are several existing solutions to protect the PowerGrid from malicious software behaviors of devices. The simplest solution is to let relays send a signal to open circuit breakers if the measurement falls out of the acceptable range. However, this scheme can only deal with pre-defined local rules. The measurements can be changed arbitrarily as long as they do not violate the local rules.

Another solution is to use the SCADA system (Supervisory Control and Data Acquisition) to monitor the devices, display the historical data on screen and leave the detection task to the operators [2]. This solution does not work very well as shown in the August 2003 blackout event.

Recently, PMUs (Phasor Measurement Units) have been used for wide area measurements [3][4][5]. Data from devices in the field can be sent to a control center synchronously in milliseconds. However, this scheme requires a reference clock, which may be received from the Global Positioning System (GPS) of satellites, for clock synchronization and is an expensive solution because a PMU can cost up to $30,000. Also, there are only a few of them in the US. Furthermore, how to detect the event automatically is not provided in this approach.

Outside of PowerGrid domain, several areas studied malicious software and detection of erroneous data. Quickest change detection problems [6][7][8][9] were studied for long time in the industrial process monitoring area. More recently, these techniques have been applied to sensor networks, but without any underlying communication constrains except [10]. Therefore, it is unclear how well they apply to the PowerGrid.

If one takes the communication constraints in sensor networks into account, it becomes clear that we need techniques that reduce the communication overhead and still allow for erroneous data detection. One of the techniques explored in sensor networks is data aggregation. However, whether data aggregation works in the context of PowerGrid, has not been studied.

In this paper, we explore the two techniques quickest change detection and data aggregation in the context of PowerGrid. We will apply these techniques in the SCADA system for anomaly detection. The proposed anomaly detection process is adaptive, i.e. it can work under various network conditions. This process provides an automatic alarm reporting to the operators, i.e. it automatically and quickly detects abnormal behaviors of devices and alert operators.

Our anomaly detection process will deploy a hybrid approach mixing quickest change detection and data aggregation. Specifically, we will apply Cumulative Sum change detection algorithm as well as two aggregation paradigms: average-aggregation and quantize-aggregation. Furthermore, our anomaly detection process will include a rigorous delay...
analysis which shows that the end-to-end detection delay should include not only the semantic computation of the data but also the underlying communication networks.

The rest of the paper is organized as follows. In section II, we introduce SCADA, the system that governs sub-stations of the PowerGrid, which is the main entity of our anomaly detection process. We present system models and assumptions in section III. In section IV, we show our framework and approach. In section V, we introduce the theory of change-point detection problem and how it can be applied in our solution. In section VI, we give a detailed analysis of delay factors to be used in the detection process. We validate the delay analysis in section VII. Section VIII concludes our paper.

II. SCADA SYSTEMS IN THE POWERGRID

Within a PowerGrid sub-station, the system that controls and monitors all devices is the SCADA (Supervisory Control and Data Acquisition) whose functions are monitoring, gathering data and controlling industrial processes.

SCADA systems have been used not only in the PowerGrid but also in many other critical infrastructures such as Water Systems, Manufacturing Process and Traffic Lights systems. A typical SCADA system consists of four main components:

- **Intelligent Electronic Devices (IEDs)** which measure and report raw data to Remote Terminal Units;
- **Remote Terminal Units (RTUs)** which receive data from IEDs and are responsible for transmitting data to SCADA master or other RTUs;
- **SCADA Masters** which receive and process data from RTUs and trigger alarms if necessary;
- **Communication Networks**, which can be both wired or wireless.

SCADA systems have been well developed and thus, we will not discuss further details here. More details on SCADA systems can be found in [2][11][4][5].

III. SYSTEM MODELS & ASSUMPTIONS

The notation of the entire paper can be found in Table III in Appendix.

A. Network Model

As shown in Figure 1, a SCADA network can be modeled as a hierarchical network where the leaves are represented by the IEDs, the parents to IEDs are the RTUs, and the RTUs report their results to the Control Center. The network connecting IEDs with RTUs may use different technologies from the network connecting RTUs with the Control Center in the context of traditional SCADA systems and the IEC 61850 standard.

In traditional SCADA systems, typical underlying networks connecting IEDs with RTUs have separate communication links and communicate via serial communication ports (RS-232/RS-422/RS-485) [2][11]. RTUs also have separate dedicated links to communicate with the Control Center. These links can be wired (e.g. copper cables, telephone lines) or wireless (e.g. radio)[2][11]. However, we will not use this type of model in our work. Instead, we will use a general type of networks similar to the UCA2 [5] standard and the IEC 61850 [4] standard.

The network connecting IEDs and RTUs is a high-speed LAN (Local Area Network) type of network, while the network connecting RTU and the Control Center is a WAN (Wide Area Network). The reason for LAN is that IED devices are within a sub-station and can be setup to form a LAN, while the Control Center might be far away from RTUs and can only be connected through a WAN.

We also assume that the Internet Protocol runs on top of these networks. In other words, each device will have an IP address configured manually or automatically (e.g. Dynamic Host Configuration Protocol).

We assume each network (LAN/WAN) will have a real-time mechanisms to ensure deadline guarantees to RTU/IED messages. The real-time schemes will have an admission control component, whose job will be to accept new messages only when they can be guaranteed to meet their deadlines. Furthermore, we assume a soft real-time scheduling algorithm over LAN/WAN IP networks to control the message delivery.

B. Data Model

Typically, data reported by IEDs can be voltage or current measurements. Furthermore, the data can be only within a range (min, max). If any data falls out of this range, the IEDs will locally react and alert the Control Center. For example, if voltage goes above a certain threshold, the IED will send a signal to open the circuit breaker and alert the Control Center.

Although the data can be modeled as a deterministic variable, we model it as a normal random variable whose mean is fixed. The reasons are as follows:

- Measurement are subjected to noise error.
- The data is affected by many small effects, which are usually modeled as a normal random variable. For example, the current measurement will vary when any device is turned on/off, and can be modeled as a normal random variable.

In the Power Grid, typically IEDs report data deterministically. Furthermore, they must meet the deadline. For example, the deadline for reporting data from IEDs to IEDs is 4ms [4]. The deadline for reporting data from RTUs to the Control Center is 5s-10s, depending on each sub-station.

We model data reporting tasks as aperiodic tasks whose arrivals follow a Poisson process. The reasons for this general model are as follows:
• Under an unknown attack, data may not follow a deterministic (periodic) pattern. Therefore, an aperiodic task model is necessary.
• The Poisson process assumption relaxes the periodic requirement. On average, the arrival rate is still equal to that of periodic tasks. Therefore, if one requires periodic reporting behavior, our model is still valid.

C. Attack/Failure Model

There are two types of attacks/failures that can happen under our data model. The first type is mean-change attack/failure where the mean value of data is shifted arbitrarily. However, we assume that the operators are only interested in attacks where the mean-shift amount is greater than $D_{\min}$.

The other type is the attack on data reporting patterns (e.g. Denial-of-Service attack) where the data can be delayed or flooded arbitrarily. Although our framework enables the detection of both types of attack, we only consider the first type in this paper.

D. Aggregation Model

As mentioned above, data aggregation is a technique compacting or converting data to another form for reducing communication overhead in sensor networks. This technique can be applied to the PowerGrid due to two reasons:

• Most of data aggregation techniques in sensor networks work in hierarchical manner. Since the PowerGrid also has hierarchical network, these techniques can be applied to the PowerGrid.
• Since the scale of IEDs of the PowerGrid can grow large (e.g. thousands of IEDs), real-time communication requirements could be violated. Aggregating data at the RTUs will significantly reduce the communication overhead, which makes it easier to meet the real-time requirements.

Next, we present our aggregation model. Then, we show the definition of average-aggregation and quantize-aggregation methods and explain why we use them.

1) Model: An aggregation function $g$ takes a tuple of $n$ values $(x_1, x_2, ..., x_n)$ as inputs, then outputs the aggregated value $y$, i.e.

$$ y = g(x_1, x_2, ..., x_n) $$

Given the network is a hierarchical system, the aggregation can be performed at intermediate nodes. Figure 2 shows an example of a hierarchical aggregation where $y_1 = g(x_1, x_2, x_3)$, $y_2 = g(x_4, x_5, x_6)$ and $y_3 = g(x_7, x_8, x_9)$. $y_1, y_2, y_3$ will be sent to the root node for further processing.

2) Average Aggregation: The average aggregation is defined as follows.

$$ g(x_1, x_2, ..., x_n) = \frac{x_1 + x_2 + ... + x_n}{n} $$

It is easy to see that hierarchical aggregation for average aggregation can be done in a straightforward manner. Each intermediate node calculates the average value of its children and reports directly to the Control Center.

As we can see that, there are two reasons why we use average-aggregation in our solution.

• $average()$ function is simple and can be calculated quickly without incurring much computation overhead for RTUs.
• $average()$ function maintains enough information of abnormal values but it is not too sensitive to those values as $min()$ and $max()$. Furthermore, if aggregation scheme uses $min()$ as the aggregation function, then the attacker could easily find the nodes (by listening) that produce the min value, and attack those nodes by changing their values. The $average()$ makes it harder to detect which nodes have that values and which nodes to attack.

3) Quantize Aggregation: An analog signal (e.g. current signal) is sampled for digital representation. Each sample being a real number is quantized and represented as a bit string for digital processing. The more bits are given to represent the real sample, the more accurate its digital form is. Our quantize-aggregation technique compresses further the digital form by keeping only most significant bits and leaving out the rest.

A $b$-bits quantize means we take $b$ left-most significant bits of the value. For example, 2-bits quantize of 1101 is 11.

A $b$-bits de-quantize means we will recover the original value by adding 0s to the quantized value. For the above example, b-bit dequantize process will return 11000.

A $b$-bits quantize aggregation of a tuple $(x_1, x_2, ..., x_n)$ will take $b$-bits quantize of each element and merge the bits results into a single value. For example, 2-bits quantize aggregation of the tuple (1010, 1101, 0101) will return 101101.

To perform a $b$-bits hierarchical aggregation, each intermediate node performs $b$-bits quantize aggregation of its children and reports the results to the Control Center. Then, the Control Center can split the value into multiple $b$-bits quantize value and perform $b$-bits de-quantize to get values for further processing.

The reasons we use quantize-aggregation are as follows.

• Quantize Aggregation is very light-weight in terms of computation for RTUs.
• Similar to average aggregation, it keeps enough information of abnormal values but it is not too sensitive to those values as $min()$ and $max()$.
• We can control the degree of aggregation by adjusting the number of allocating bits $b$.

4) Accuracy of Aggregation: Data aggregation can be considered as a type of lossy compression. Thus, it is important...
to understand factors affecting its accuracy. Based on the definitions of the two aggregation schemes, we have following observations.

- For average-aggregation, number of variables \( n \) is the main factor that affects its accuracy. The more variables are in the set, the less accuracy of the aggregated value is observed.
- For quantize-aggregation, number of bits \( b \) allocated for aggregation is the main factor affecting its accuracy.

The accuracy of the aggregation plays an important role for mean-change attack detection and is discussed in section V.

IV. FRAMEWORK

A. Goals

The ultimate goal of our system is monitoring and detecting anomalies at IEDs as soon as possible, under communication and false alarm rate constraints.

As mentioned in section III-C, we only consider mean-change attacks. In the subsequent sections, first we will show our approach, including the architecture and protocols for the problem. Then, we will introduce the theory of change-point detection to see how the problem can be solved theoretically. Because this theory does not consider any communication model, we will analyze how it works in our scheme by giving a detailed end-to-end detection delay analysis.

B. Approach

1) Architecture: Figure 3 shows the proposed architecture. It includes three components: The semantic (data) aggregator, topology aggregator and detector.

Data aggregators (DA) are placed at RTUs. They sample data from IEDs, aggregate and send the result to the Control Center. DAs are used to detect mean-change attacks.

Topology aggregators (TA), also residing at RTUs, monitor, aggregate and store QoS (Quality of Service) information, such as available bandwidth and delay, about the topology among IEDs and RTUs. In addition, TAs are also responsible for clustering IEDs based on QoS metrics and locations. TAs are used for detection of attacks on the data reporting patterns.

The last component, Detector, placed at the Control Center, takes information either directly from DAs or TAs, analyzes and generates alarms to the operators if needed.

In this paper, we will mainly focus on Data Aggregators and how Detectors detect mean-change attacks. Other components will be addressed in the future work.

2) Protocols: Figure 4 illustrates how data goes from IEDs to RTUs and is aggregated and used for detection.

- At IED level: When an IED wants to send a packet, it follows the realtime scheduling algorithm, which includes an admission control test and a priority assignment procedure, to ensure that its packet can meet the deadline without affecting other IEDs and RTUs.
- At RTU level: An RTU collects, aggregates and sends data from its IEDs to the Control Center in rounds. Different RTUs have different rounds. RTUs start a new round when the system is booted. An RTU ends the current round when it receives data from each of its IEDs in that round\(^1\). After aggregating data and sending aggregated data to the Control Center, the RTU starts a new round.

Similar to IEDs, RTUs must follow the realtime scheduling algorithm when sending their packets to ensure that all packets can meet the deadlines without interfering with other IEDs and RTUs.

- At the Control Center level: The only task of the Control Center in our scheme is receiving data from RTUs and performing the mean-change attack detection algorithm, which is introduced in the next section.

V. QUICKEST CHANGE-POINT DETECTION PROBLEM

Mean-change attack detection is similar to the quickest change-point detection problem, whose goal is the quickest detection of the change-point under a fixed false alarm rate. While there are various algorithms for this problem, the Cumulative Sum algorithm (CUSUM), besides its simplicity, has been shown to have advantages over other detection methods under small disorder scenarios (see [9], Chapter 4). This advantage is very well suited for our data model of the PowerGrid. Therefore, we use the CUSUM algorithm as the mean-change detection algorithm. It is important to stress that solutions for quickest change-point detection problem do not consider any underlying communication. Therefore, all the terms in this section that express the delay have the same unit: number of samples.

Before discussing the CUSUM algorithm, we briefly introduce the background of the general change-point detection

\(^{1}\text{We do not consider the attacks that delay the reporting of data. We also assume the transmission is reliable.}\)
problem and the mean-change detection problem.

A. General Change-Point Detection Problem

Let us consider a sequence of observed random variables $X = \{x(n)\}_{n=1}^{\infty}$ with conditional density $p_{\theta}(x_k|x_{k-1},...,x_1)$. Before the unknown change time $\tau$, the conditional density parameter $\theta$ is constant and equal to $\theta_0$. After the change, the parameter is equal to $\theta_1$. The problem is to detect the occurrence of change as soon as possible, with a fixed rate of false alarms before $\tau$.

An online change detection algorithm will define a stopping rule, which usually has the form

$$ T = \inf\{n : g_n(x_1,\ldots,x_n) \geq h\} $$

where $h$ is a threshold, and $(g_n)_{n\geq1}$ is a statistic, calculated from observations $(x_1,\ldots,x_n)$. Function $g$ can be a log-likelihood function or a cumulative function as defined below. The intuition is that if the result of the statistic function (e.g. log-likelihood) is greater than a certain threshold $h$, the change-point is declared.

Metrics for the change detection algorithm are usually average detection delay and false alarm rate. Formally, the average detection delay is defined as

$$ ADD(\tau) = E_{\theta_1}(T - \tau | T \geq \tau) $$

and false alarm rate is

$$ FAR(T) = \frac{1}{E_{\theta_0}[T]} $$

B. Mean-Change Detection Problem

The above change detection formulation requires the knowledge of the distribution of observed random variables $X$. In other words, they are parametric methods. In practice, especially in our case, it is undesirable to require such knowledge because when an attack occurs, the observed random variables may not follow any distribution. This necessitates non-parametric methods.

The non-parametric change detection problem also has received attention. Due to the lack of distribution of observed random variables, non-parametric change detection problems usually look at changes in mean, variance or correlation values. In our case, we are only interested in mean change. The mean-change problem can be defined as follows.

Let us consider a sequence of observed random variables $X = \{x(n)\}_{n=1}^{\infty}$ such that

$$ x(n) = \mu + A_n I(n < \tau) + (D + B_n) I(n \geq \tau) $$

where $A = \{A_n\}_{n=1}^{\infty}$, $B = \{B_n\}_{n=1}^{\infty}$ are random sequences such that $E[A_n] = E[B_n] = 0$, $D$ is the change amount of the mean, $\tau$ is the change-point and $I(\cdot)$ is the indication function. Note that without the loss of generality, we assume $D > 0$.

Similar to the parametric change detection problem, stopping rule and metrics are defined in the same manner for the mean-change detection problem. The only difference is the definition of the statistic function $g$ for quickest detection as we will show now.
C. Cumulative Sum (CUSUM) Algorithm


Let us define the statistic function \( g \) as

\[
g_k = (g_{k-1} + s_k)^+, g_0 = 0
\]

where \( x^+ = \max(0, x) \) and \( s_k = \log \frac{p_{\theta_0}(x_k)}{p_{\theta_1}(x_k)} \).

The stopping rule is

\[
T = \min\{ k : g_k \geq h \}
\]

where \( h \) is the threshold and is normally set to \( \log \frac{1}{1 - \alpha} \).

The key idea of CUSUM algorithm is to exploit the nature of the log-likelihood ratio \( s_k \), which is negative before the change and positive after the change. CUSUM cumulates the positive part and declares a change when the cumulation exceeds the threshold \( h \). The negative is out of interest and is set to zero.

2) Non-parametric version: The statistic function defined above obviously requires the form of distribution. A non-parametric version was proposed in [9][8]. The statistic function is now defined as

\[
g_k = (g_{k-1} + s_k)^+, g_0 = 0
\]

and

\[
s_k = x_k - \mu_0 - \epsilon E[\mu_1|x_k, ..., x_0]
\]

where \( \mu_0 \) is the mean before change and is supposed to be known, \( \mu_1 \) is the unknown mean after change, \( \epsilon \) is the sensitive factor.

The key idea for the non-parametric version is to replace the log-likelihood ratio \( s_k \) in the parametric version with another term that shows the difference between the mean before change \( \mu_0 \) and the mean after change \( \mu_1 \). Because the mean \( \mu_1 \) is not known, we need to predict it based on historical data \( \{x_0, ..., x_k\} \). The estimator can be an average or a similar linear estimator. However, we want an estimator that can forget the observations that are far in the past. Therefore, we use an adaptive exponentially weighted estimator. This estimator predicts based on sequential inputs as follows.

Let \( \hat{\theta} \) be the current prediction of the mean \( \mu_1 \). If \( s_n = 0 \), we just set \( \hat{\theta}_n = \mu_0 \) because there is no change occurring. If \( s_n > 0 \),

\[
\hat{\theta}_n = \frac{1}{\beta_{n-1} + 1}(\beta_{n-1} \hat{\theta}_{n-1} + x_n)
\]

where the weight is \( \beta_n = 1 + \beta_{n-1} \) and is reset to zero when \( s_n = 0 \). This weight is similar to the weight of the exponential moving average. The only difference is that it depends on historical observations.

In this paper, due to the effectiveness and simplicity of non-parametric CUSUM algorithm, we will use it to provide the quickest mean-change detection. The following property of CUSUM will be used for our analysis in the next section.

\[2\] The moving average has the form \( \hat{\theta}_n = \alpha x_n + (1 - \alpha)\hat{\theta}_{n-1} \). \( \alpha \) is similar to \( \frac{1}{\beta_{n-1} + 1} \) as in (1).

Lemma 1: Asymptotically, the non-parametric version of CUSUM for mean-change detection problem has the average detection delay

\[
ADD_e(T) = \frac{h}{(1 - \epsilon)\mu_1 - \mu_0}, h \to \infty.
\]

Proof: See [8].

Note that the unit of the delay is number of samples of observed variables.

3) Non-parametric CUSUM algorithm under data aggregation: Let us consider a system which has an RTU \( r \) and \( n_e \) IEDs reporting to RTU \( r \). Let us assume that there is an attack at IED \( e \). We have following lemmas.

Lemma 2: Let ADD\([\ast]\) be the asymptotic delay to detect the attack without any aggregation. It will take \( n_e \cdot ADD\([\ast]\) samples for average-aggregation to detect the attack.

Proof: Omitted due to limited space.

Lemma 3: Let \( b_{min} \) be the smallest number of bits that is necessary to detect the attack for quantize-aggregation. The asymptotic detection delay is equal to \( ADD\([\ast]\) \). Furthermore, any \( b \)-bit quantize aggregation where \( b > b_{min} \), will have the same asymptotic detection delay.

Proof: Omitted due to limited space.

Lemma 4: The number of bits representing the data. For example, if 4 bits are used for representation, \( D_{min} = 4 \) and \( \mu_0 = 0 \), then \( b_{min} = 2 \).

Lemma 3 shows that once the Control Center can detect the attack with current bit-allocation, allocating more bits will not help to decrease the detection delay. This property might suggest that quantize-aggregation will outperform average-aggregation. However, if the number of IEDs per RTU increases linearly, the number of bits to represent the aggregated value is also linearly increased, which will take more communication overhead. Therefore, if one takes into account underlying communication, the quantize-aggregation method will expose its weakness under certain conditions. We will validate this claim in section VII.

VI. Delay Analysis of Value Attack Detection

Since our goal is to detect mean-change attacks as soon as possible, we need to understand factors contributing to the end-to-end detection delay. In this section, we will decompose the end-to-end detection delay into individual component delays and show the relationship among them. Then, we will show the bounds for those individual delays. This analysis will not only help to understand factors affecting the end-to-end detection
delay but also make it possible to optimize system design for a faster detection.

Figure 4 shows four main components contributing to end-to-end detection delay.

1) **LAN Delay**: This delay is the time when an IED sends the packet until corresponding RTU receives the packet. It includes realtime scheduling, sending, propagation and queueing delay. We denote this delay as $D_1$.

2) **Collection Delay (Round Length)**: This delay is the time that the RTU waits for all of its IEDs to send the packet in that round. This delay is equal to the round length. We call this delay as $D_2$.

3) **WAN Delay**: Similar to LAN delay factor, this delay factor is the time between an RTU sending a packet and the Control Center receiving the packet. This delay is denoted as $D_3$.

4) **Detection Delay**: This is the delay incurred by the mean-change detection algorithm in section V performed at the Control Center. We call this delay as $D_4$. As mentioned in section V, the unit of this component delay is the number of samples reported by IEDs.

Briefly, the key ideas of this delay analysis is 1) finding out how many samples that the Control Center needs from the attacked IED to detect the attack ($D_4$) and 2) calculating how long it takes for each sample to transmit from an IED to the Control Center ($D_1 + D_2 + D_3$).

In subsequent sections, we will show the bound of these delay factors.

A. **LAN Delay - $D_1$**

As we assume that IEDs generate data aperiodically, the realtime scheme at LAN will execute aperiodic tasks. Although the realtime guarantee for aperiodic tasks has been considered [13] [14], what we really want is a bound for realtime communication delay in LAN. Fortunately, this problem can be derived from the work in [14]. The result in [14] can be summarized as follows. (Note that in this section the terms “tasks”, “messages” and “packets” are used interchangeably).

Let us now focus on a LAN that has $n_e$ IEDs. Also, let $C_i$ be the time to transmit a message generated by $i$th IED and $D_i$ be the relative deadline (e.g. 4ms after generated) of the message.

At any given time $t$, let $M(t)$ be the number of messages that have generated but whose deadlines are not expired yet. Abdelzaher et al. [14] define $U(t)$, the synthetic utilization, as follows

$$U(t) = \sum_{T_i \in M(t)} C_i / D_i$$

They proved that using an optimal time-independent scheduling policy, all messages will meet their deadlines if $\forall t: U(t) \leq UB(n_e)$, where

$$UB(n_e) = \frac{1}{2} + \frac{1}{2n_e}, n_e < 3$$

$C_i$ includes service time, propagation delay and queueing delay of the message and

$$UB(n_e) = \frac{1}{1 + \sqrt{\frac{1}{2} (1 - \frac{1}{n_e})}}, n_e \geq 3$$

Because all IEDs are identical, $C_i = C$ and $D_i = D \forall i = 1..n_e$. Thus, (2) in our system will be

$$U(t) = \frac{n_e C}{D}$$

Substitute (4) to (3), we get

$$D \geq \frac{2n_e^2 C}{n_e + 1}$$

and

$$D \geq n_e(1 + \sqrt{\frac{1}{2} (1 - \frac{1}{n_e})})C, n_e \geq 3$$

(5) shows that even in the best case where the best time-independent scheduling is used, the smallest guaranteed delay for a message to transmit from an IED to corresponding RTU is the right-side of (5). Therefore,

$$D_1 = n_e(1 + \sqrt{\frac{1}{2} (1 - \frac{1}{n_e})})C$$

B. **Collection Time (Round length) - $D_2$**

As mentioned above, the aggregation at an RTU requires that data from all of its IEDs has to be ready at the time of aggregating. The time between two consecutive sending of an RTU to the Control Center is called a round. Now, we are interested in calculating the average round length.

Let us consider an RTU $r$ and its IEDs $1...n_{er}$ where $n_{er}$ is the number of IEDs of RTU $r$. Let $X_i(t)$ be the Poisson process generating values of $i$th IED. According to the property of Poisson process, the time between two consecutive values generated by an IED is an exponential random variable. Let $Y(t)$ be the random process of round length.

At the time $t_0$ of beginning of a new round, IEDs will generate a new value following an exponential distribution. Because RTU $r$ must wait for values from all of its IEDs and because of the memoryless property of the exponential random variable, we have

$$Y(t) = \max (X_1(t), X_2(t), ... X_{n_{er}}(t))$$

Thus,

$$E[Y(t)] = E[\max (X_1(t), ..., X_{n_{er}}(t)))]$$

Due to limited space, we omit the proof and show the final result.

$$E[Y] = E[\int_0^{n_{er}} e^{-\lambda x} (1 - e^{-\lambda x})^{n_{er}} - 1 x dx]$$

(6) shows how to calculate the average round length $E[Y]$. Thus, we have $D_2 = E[Y]$. 

Substitute (4) to (3), we get
C. WAN Delay - D3

The analysis of WAN delay is exactly the same of LAN delay. The only difference for WAN delay is the network parameters and realtime requirements, which are not relevant for this analysis. Therefore, we will not discuss it further.

D. Mean-Change Detection Delay - D4

As shown in Lemma 1, the asymptotic average detection delay of the non-parametric CUSUM algorithm for mean-change detection is

\[ D_4 = \text{ADD}_r(T) = \frac{h}{(1 - \epsilon)\mu_1 - \mu_0}, h \to \infty. \] (7)

Note that the unit of this delay is the number of samples.

E. Formula for End-To-End Detection Delay

When a mean-change takes place at an IED, it will take \(D_4\) number of samples to detect the change. Each sample takes \((D_1 + D_2 + D_3)\) time unit (e.g. seconds) to be transmitted from the IED to the Control Center. Therefore, the formula for end-to-end detection delay is:

\[ E2DED = D_4(D_1 + D_2 + D_3) \] (8)

VII. VALIDATION

In this section, we validate our scheme in ns-2 simulator with analysis results obtaining from Matlab.

A. Experimental Setup

1) Setup: We have implemented our scheme in the ns-2 (version 2.29)[15]. We place fixed number of 50 IEDs and vary the number of RTUs from 1 to 50. For each number of RTUs in the system, we distribute evenly number of IEDs to RTUs. For example, if the number of RTUs is 10, then each RTU will be responsible for 50/10 = 5 IEDs. Intuitively, the more we have RTUs in the system, the smaller number of IEDs that each RTU must handle.

We use the voltage measurements as the status values we simulate. The normal value of the voltage measurement that the system expects under normal condition is 120V. The percentage of the variance of the measurement introduced by noise is 2\% (\(\approx 2V\)). Please note that these measurement values are just for simulation, not the fixed values. We provide a summary of the parameters of our experimental setup in Table I.

2) Attacks: For each run of the simulation, at random time we uniformly pick an IED to change its mean value between the range specified in Table I.

B. Scenarios

We want to see how our scheme performs under different network scenarios. Specifically, we validate the mean-change detection process under low-bandwidth networks and high-bandwidth networks scenarios. Table II shows the parameters for the low-bandwidth and the high-bandwidth network scenarios.

C. Metrics

The first metric that we use for validation of the anomaly detection process is the average end-to-end detection delay of the attack (in seconds), which the system wants to minimize. As we vary the number of RTUs we want to measure how the system performance is affected.

The second metric is the number of exchanged messages between RTUs and the Control Center. This metric will measure the effect of data aggregation on the overhead, in terms of computation, at the RTUs and the Control Center. \(^4\)

D. Results

Figure 5(a) shows the results for the low-bandwidth scenario. The x-axis shows the increasing number of RTUs in the system. The y-axis shows the average end-to-end detection delay (in seconds). The graph shows that the analysis and simulation results closely match with each other. It also shows that the quantize-aggregation performs better than average-aggregation under low-bandwidth networks. This result makes the claim in Lemma 3 clearer: under low-bandwidth networks when the communication cost is expensive and because the average-aggregation must need more samples than that of quantize-aggregation to detect the attack, the quantize-aggregation will outperform average-aggregation.

Figure 5(b) shows the results for high-bandwidth scenario. As we can see in the graph the quantize-aggregation exposes its weakness under high-bandwidth networks. The reason is that when the time to transmit an aggregated sample of average-aggregation is much less than that of quantize-aggregation, average-aggregation can transmit much more samples and thus has better end-to-end detection delay.

We can see that the end-to-end detection delay without data aggregation, which is represented by the thick line in Figure 5(a) and Figure 5(b), is larger than that of average-aggregation and quantize-aggregation (especially in Figure 5(b)) when the system has enough number of RTUs. This observation confirms that data aggregation helps to improve the speed of mean-change attack detection delay.

Figure 6 shows the messages exchanged between RTUs and the Control Center. We can see in the figure that average-aggregation and quantize-aggregation also help to reduce the number of processing messages at RTUs and the Control Center while still be able to detect the mean-change attacks.

E. Discussion of Results

The results show that data aggregation provides a chance for faster end-to-end detection delay. However, neither average-aggregation nor quantize-aggregation outperforms the other in all scenarios. Depending on the communication cost i.e. low-bandwidth networks or high-bandwidth networks, the scale of the system i.e. number of IEDs and RTUs, we can choose the suitable data aggregation scheme for faster detection.

\(^4\)We do not measure the number of exchanged messages between IEDs and RTUs because the data aggregation has no effect on it.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAR</td>
<td>$10^{-6}$</td>
<td>Acceptable false alarm rate of CUSUM algorithm</td>
</tr>
<tr>
<td>#IEDs</td>
<td>50</td>
<td>Number of IEDs in the system</td>
</tr>
<tr>
<td>#RTUs</td>
<td>[1,50]</td>
<td>Number of RTUs vary from 1..50</td>
</tr>
<tr>
<td>IED Packet Size</td>
<td>1Kb</td>
<td>The size of an IED packet</td>
</tr>
<tr>
<td>RTU Packet Size</td>
<td>1Kb</td>
<td>The size of an RTU packet</td>
</tr>
<tr>
<td>Normal Value</td>
<td>120V</td>
<td>The mean of IEDs' value in normal condition</td>
</tr>
<tr>
<td>Mean-change range</td>
<td>[200V, 300V]</td>
<td>The range of the mean-change attack</td>
</tr>
<tr>
<td>Attack time</td>
<td>50th seconds</td>
<td>The time the attack starts at</td>
</tr>
<tr>
<td>Simulation duration</td>
<td>5000s</td>
<td>The time simulation stops</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.4s</td>
<td>Data-Generating rate of IEDs</td>
</tr>
<tr>
<td>Variance of Noise</td>
<td>2% ($\approx 2V$)</td>
<td>The percentage of variance of IED measurement introduced by noise</td>
</tr>
</tbody>
</table>

**TABLE I**

PARAMETERS FOR EXPERIMENTAL SETUP

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Network from IEDs to RTUs</th>
<th>Network from RTUs to the Control center</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-bandwidth network</td>
<td>100Kbps</td>
<td>100Kbps</td>
</tr>
<tr>
<td>High-bandwidth network</td>
<td>100Mbps</td>
<td>100Mbps</td>
</tr>
</tbody>
</table>

**TABLE II**

PARAMETERS FOR LOW-BANDWIDTH AND HIGH-BANDWIDTH NETWORK SCENARIOS

![Graphs showing End-to-End Detection Delay](image)

(a) End-to-End Detection Delay under Low-bandwidth networks  
(b) End-to-End Detection Delay under High-bandwidth networks  

Fig. 5. Average end-to-end detection delay

**VIII. CONCLUSION**

We have looked at the use of data aggregation for detecting mean-change attack in the PowerGrid. We have derived the relationship and the importance among delay factors contributing to the end-to-end detection delay. Our analysis has shown that end-to-end detection delay should not only involve with the semantic of the data but also should involve with underlying communication networks.

Then, we explore two aggregation schemes: average-aggregation and quantize-aggregation and apply non-parametric CUSUM method for quickest mean-change detection. We have shown that it is possible to reduce end-to-end detection delay by data aggregation. Finally, the analysis and simulation results show that neither average-aggregation nor quantize-aggregation outperforms the other in all scenarios. We suggest that depending on the system parameters such as network bandwidth, we could choose an appropriate aggregation method for faster detection.

**IX. APPENDIX**

Table III shows the notations used in this paper.

**REFERENCES**

Fig. 6. Number of Exchanged Messages between RTUs and the Control Center under Low-bandwidth networks

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_e$</td>
<td>Number of IEDs in the system</td>
</tr>
<tr>
<td>$n_r$</td>
<td>Number of RTUs in the system</td>
</tr>
<tr>
<td>$n_{er}$</td>
<td>Number of IEDs per RTU i.e. $n_{er} = \frac{n_e}{n_r}$</td>
</tr>
<tr>
<td>$(\text{min}, \text{max})$</td>
<td>The acceptable range of IEDs value</td>
</tr>
<tr>
<td>$D_{\text{min}}$</td>
<td>the value which the mean-shift amount must be greater than for a noticeable attack</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Rate of data-generating Poisson process of IEDs</td>
</tr>
<tr>
<td>$ADD_T(T)$</td>
<td>Average end-to-end detection delay with the stopping time rule $T$ and the change-point $\tau$</td>
</tr>
<tr>
<td>$h$</td>
<td>The threshold for CUSUM algorithm</td>
</tr>
<tr>
<td>$D_1$</td>
<td>LAN Delay</td>
</tr>
<tr>
<td>$D_2$</td>
<td>Collection Delay (Round length)</td>
</tr>
<tr>
<td>$D_3$</td>
<td>WAN Delay</td>
</tr>
<tr>
<td>$D_4$</td>
<td>Detection delay of CUSUM algorithm for mean-change detection</td>
</tr>
</tbody>
</table>

TABLE III

COMMENTS