ABOUT THE PROBLEM OF DATA LOSSES IN REAL-TIME IOT BASED MONITORING SYSTEMS

Prof. Alieksieiev V. PhD.[1], Prof. Gaiduchok O. PhD.[1]
Lviv Polytechnic National University, Lviv, Ukraine [1]

Vladyslav.I.Alieksieiev@lpnu.ua

Abstract: *Fast growing market of IoT devices revealed a number of complex problems. Among these problems, there is a problem of data losses caused by data package losses or delays while its transition from sensor to server. As anticipated, there are a number of businesses relying on easy opportunity to build real-time monitoring systems using modern software and IoT hardware solutions. Although the growing reliability of contemporary communication networks one can find the problem of making decision about lost or delayed data packages. Current research is dedicated to building an algorithm for compensation of gaps in data series to support real-time monitoring systems with appropriate artificially generated values. Cases of applicability of the algorithm were also studied and discussed.*

Keywords: DATA LOSSES, DATA SERIES, TIME SERIES, IOT, COMPENSATION ALGORITHM

1. Introduction

An IoT growth allowed designing some new platforms for supporting business both with surveillance tools and analytics software applications. Three key features of such platforms are:

- Device – a remote computer like Raspberry Pi or any of its alternatives [1] or some custom hardware device that may include a set of sensors (business solution may consist of a whole network of such devices).
- Internet – any kind of Internet wireless connection via Wi-Fi, GPRS, 3G/4G or anything else supplied with mobile network [2] (business solution may combine different types of Internet service providers to establish connection).
- Cloud – any popular IoT platform [3] to store and process big data.

If one adds some software solution to provide data analytics to that IoT platform than it becomes a powerful business tool supporting real-time monitoring. There is a number of companies developing their own platforms or exploiting powerful cloud services to provide their client with such platform as a reliable business solution.

However, practical use of such system reveals some serious problems. One of most important problem is the problem of lost or delayed data packages. It is obvious that real-time monitoring faces vulnerability to data losses. The data losses or at least delays in package deliveries via Internet over mobile networks is an ordinary problem. Such data losses may cause fake alerts about hardware failures or, otherwise, hide a real failure or a critical state. This means the correct functioning of the whole solution requires some reliable analytical decision about current state of remote device.

2. Prerequisites and means for solving the problem

Let us assume that data packages should come to server within a fixed period of time t_p. This can be any value appropriate for particular business logic. For example, it can be a quarter of an hour – receiving packages every 15 minutes. Again, depending on observed business area we can define a critical time for monitoring and accuracy of alerts. Now we will not discuss the accuracy of data and the actual method or frequency of sensors gathering some values. Thus, we assume that each one sensor gives us a single value across the period of t_p. This value is sent to our server within a single data package (there can be a set of values – one for each sensor in the remote device).

As we analyze some value, than we can assume a range of "good" and "bad" values for normal state and failure state. For example, the range $v_{\min} \le v \le v_{\max}$ can be defined to indicate normal state and values outside of this range can be considered as failure state. For the purpose of analysis of the need of invoking an alert, we can simplify these values to Boolean value: TRUE for normal state, FALSE for failure state. We also need one more value to indicate missing value (package loss) – NULL can be an appropriate one for that state.

Next, we are to define the number of lost packages to consider device to go offline and the size of a "window" to display the current state in the real-time monitoring system. The exact values depend on peculiarities of observed processes and can be defined experimentally. For the purpose of our research, let us assume these values ($N_{offline}$ and N_{window}) to be interdependent:

$$N_{offline} / N_{window} = k \in (0, 1] \quad (1)$$

The k value can describe system sustainability or vulnerability to data losses (with consider to the "window" width). k=1 means sustainable system, and $k \to 0$ means vulnerable system. To be clear we can consider the k value as a measure of customer requirement of data losses vulnerability.

The easiest conditions to make decision about system state (normal or failure) are in case with no data losses. We can use simple probability calculation to find whether system should indicate normal state (n_{true} and n_{false} are the numbers of TRUE and FALSE values among all N_{window} values in the "window" of observation, so that $N_{window} = n_{true} + n_{false}$):

$$n_{true} / N_{window} = 1 - n_{false} / N_{window} > 0.5 \quad (2)$$

Condition "> 0.5" is expected to be strict to be sure that most values indicates the normal state. Table 1 lists some common examples for definition of state indicator according to analysis of sensor values within predefined "window" (N_{window} = 8).

Table 1*: Examples of state definition in case without data losses (N_{window}=8).*

State indicator	Set of values							
Failure	F	F	F	F	F	F	F	F
Failure	F	F	T	F	T	F	F	T
Failure	F	F	F	F	T	T	T	T
Normal	T	T	F	F	T	T	T	F
Normal	T	T	T	T	T	T	T	T

In case of data losses, it looks unclear how to make a reliable decision about current state. On one hand, uncertainty can be ignored, but this will yield mistakes in indication the state. On the other hand, replacement of lost data with artificially generated value is possible, but we cannot be definitely sure about accuracy of the result. Table 2 shows examples of situations, when data loss obstructs ability to indicate system state.

Table 2*: Examples of uncertainty caused by data losses (N_{window}=8, F – FALSE, T – TRUE, N – NULL).*

State indicator	Set of values							
Unknown	F	N	T	N	F	N	T	F
Unknown	N	N	N	N	N	N	F	T
Unknown	N	N	N	N	T	T	T	T
Unknown	N	N	F	F	T	T	N	N
Unknown	T	T	F	F	N	T	T	F
Unknown	T	T	F	T	F	F	N	N

There is a number of algorithms based on calculation of some kind of average values – arithmetic mean or moving average [4, p.153]. However, such approach usually uses only previous data and does not "cover" gaps. Considering trends in data series with moving average "on-the-fly" can be a good method to replace data losses of the last package. Nevertheless, we cannot rely on several artificially generated values to cover new loss. The best way is to store gaps (for example, as a NULL values) and use some other algorithm to cover gaps later.

Another way to fill in the gap is to use some regression model or prediction technique [5, p.279]. For example, one may use simple linear regression or logistic regression model to cover gaps in our time series. Trying to make it better, we can even use polynomial or spline regression models [6, p.166]. Considering these methods, one can mention a high complexity for its implementation. This means that these methods will not fit the requirement of "on-the-fly" data processing, particularly in case of "big data" volumes.

Finally, we can define a couple of requirements for constructing an appropriate algorithm to cover gaps in time series caused by data losses:

- Simplicity – the algorithm should be easy to implement and fast enough to be used "on-the-fly".
- Reliability – the algorithm should give a reliable approximation for each lost value as the most probable value.

3. Solution of the examined problem

First, we are to determine key concept for the algorithm, to fit all above mentioned requirements and limitations: *a lost package value can be replaced with a dominant value of neighbouring packages*. Both with a common sense of this concept we should assume to keep the quantitative majority concept of eq. (2). In the case of missing values presented with its number n_{null} we can rewrite (2) as

$$n_{true} / (N_{window} - n_{null}) = 1 - n_{false} / (N_{window} - n_{null}) > 0.5 \qquad (3)$$

Here, the number of values in the "window" fits the condition $N_{window} = n_{true} + n_{false} + n_{null}$. We should also mention, that $n_{null} < N_{offline}$ is the rule to consider device staying online. Now we can search the way to determine that dominant value to cover gap.

Second, we are to determine known patterns of data losses presented in terms of Boolean values and most appropriate replacement for each missing value. Table 3 lists examples of patterns containing three packages (these cases are obvious according to eq. (3)).

***Table 3**: Example of 3-package pattern (F – FALSE, T – TRUE, N – NULL).*

Replacement	Set of values (X for F or T)						
N→X	N	X	X	…			
N→X	…		X	N	X	…	
N→X	…				X	X	N

Next, in Table 4, we list examples of patterns containing four packages (these patterns were formed excluding the subset of patterns similar to smaller patterns in Table 3).

***Table 4**: Example of 4-package pattern (F – FALSE, T – TRUE, N – NULL).*

State indicator	Set of values (X for F or T and Y for *Not*{X})						
N→X	N	X	Y	X	…		
N→X	N	Y	X	X	…		
N→X	Y	N	X	X	…		
N→X	…		X	N	Y	X	…
N→X	…				X	X	N	Y
N→X	…				X	Y	X	N
N→X	…				X	X	Y	N

Here and in further patterns, we use a general extension to the right rule – "*first step right, next step left*" – and in case of reaching the border of the "window" we can use extension to the free border (left or right accordingly). The rule of extension allows us find easily the dominant value to fit the majority concept. It is important here to remember, that we place packages like time series – from the left to the right. This assumption means we have latest values closer to the right border of the "window" and this is why we should use extension to the right rule. Therefore, this is how we accomplish not only implementation of majority concept, but also consider the influence of the latest state (unlike to most of moving average algorithms).

In some cases a "collision" can happen – the state of emerging new gap instead of value during extension procedure and reaching both borders of the "window". The collision state means repetitive need of extension while gap cover is impossible. To avoid such difficulties in case of collision we use propagation rule – "*replace all nulls with major value (if can be found) or last value (if there is no major value), when number of meaning values exceeds number of nulls*" – and in case of majority of nulls, we can consider device to be offline.

4. Results and discussion

Now let us formalize description of the algorithm for compensation of gaps in data series:

1. Define a "window" width N_{window} with respect to $N_{offline}$ and transition to "offline" state.
2. Use (2) to make decision when there are no gaps.
3. Following extension to the right rule – "*first step right, next step left*" – use (3) to make decision when there are gaps.
4. Use propagation rule – "*replace all nulls with major or last value, when number of meaning values exceeds number of nulls*" – in case of a "collision".

Such kind of algorithm allows covering all the gaps. It is simple enough to be used the same easily both at back-end or front-end solutions to supply appropriate values in customer's real-time IoT based monitoring system. Certainly, it is more likely to utilize the algorithm at front-end to lower the load of servers.

For the purpose of greater accuracy, one can choose between different kinds of moving averages (as a common solution for considering missing values), regressions models (as a common predictive solution) and various approximation techniques (like polynomial or spline).

5. Conclusion

The described algorithm for compensation of data losses fits all requirements of implementation simplicity and reliability. The level of computation efforts and complexity of the algorithm are relevant to arithmetic mean calculations. Simultaneously, the algorithm considers latest values (unlike the moving average techniques). The influence of latest values is crucial for indicating current system state. Considering the initial conditions for the problem and a specific time series, the algorithm appears to be the most adequate and reasonable solution.

6. Literature

1. 10 Best Raspberry Pi and Pi 2 Alternatives. // Beebom.com. – April 18, 2017. – https://beebom.com/raspberry-pi-and-pi-2-alternatives/
2. Global State of Mobile Networks (February, 2017) // OpenSignal, Inc. – February 6, 2017. – https://opensignal.com/reports/2017/02/global-state-of-the-mobile-network
3. Top 20 IoT Platforms in 2018./ Santosh Singh // Internet of Things Wiki – March 8, 2016 – https://internetofthingswiki.com/top-20-iot-platforms/634/
4. Allen B. Downey. Think Stats. — O'Reily, 2015. – 225 p.
5. Head First Data Analysis. — O'Reily, 2009. – 488 p.
6. Peter Bruce and Andrew Bruce. Practical Statistics for Data Scientists. — O'Reily, 2017. – 317 p.