A Data Distribution Model for Large-Scale Context Aware Systems

Soumi Chattopadhyay[1], Ansuman Banerjee[1], and Nilanjan Banerjee[2(✉)]

[1] Indian Statistical Institute, Kolkata, India
{mtc1203,ansuman}@isical.ac.in
[2] IBM Research - India, New Delhi, India
nilanjba@in.ibm.com

Abstract. Very large scale context aware systems are becoming a reality with the emerging paradigms such as machine-to-machine communications, crowdsensing, etc. Scalable data distribution is a critical requirement in such large scale systems for optimal usage of computing and communication resources. In this paper, we present a novel theoretical model for middleware design for such large-scale context aware systems that distributes only relevant data based on its *effective utility*. We also present extensive experimental results to validate the efficacy of our proposed model.

Keywords: Context-aware computing · Data distribution model

1 Introduction

We are witnessing an unprecedented instrumentation of our environment due to the rapid advancements in computing and networking technologies over the last decade. Large scale sensor network deployment, numerous devices connected over the Internet and the tremendous development and adoption of sensor enabled smartphones are enabling us to build context aware systems capturing physical contexts and interpret our personal objective world like never before. Context aware systems studied so far have been limited to small scale systems like smart homes, smart offices. Recent advances in cost, form and performance factors of computing and networking technologies are making the design and the deployment of very large scale, distributed context aware systems a reality. Machine-to-machine (M2M) or Internet of Things (IoT) is an emerging paradigm that is facilitating the development of such large scale distributed context aware systems. In M2M, powerful devices are connected to each other so that they can communicate among themselves to perform certain tasks. For example, efficient energy management is now possible with the large scale deployment of connected smart energy meters at home. Consider the following situation, where a large energy company provides a service to regulate the home temperature of its customers based on total energy load at the customer premises, the ambient temperature and the energy price. In order to provide such a service, the

I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 615–627, 2014.
DOI: 10.1007/978-3-319-11569-6_48

energy company collects the relevant data from various sources including the smart energy meters to derive a real-time context that determines whether to turn the air conditioners at the customer homes, on or off. Another contemporary phenomenon enabling large scale distributed context aware system is the rapid development and proliferation of smartphones equipped with sophisticated sensors. The new emerging paradigm of *crowdsensing* [1] leverages this truly pervasive, mobile sensor network of smartphones to provide a cheaper and more pervasive alternative for large scale sensor network based monitoring of physical environment. In crowdsensing, people with smartphones contribute data either voluntarily or driven by certain personal objectives or incentives. For example, in the absence of a dedicated sensor network, public authorities can monitor temperature, pollution level at a particular location by collecting the temperature and pollution data from volunteers who are carrying smartphones equipped with appropriate sensors and are present at that location.

The role of context data distribution in any such large scale distributed sensing network is very crucial to ensure the availability of the context data with the right quality, in the right place and at the right time i.e. with acceptable Quality of Context (QoC). Context data distribution consumes costly and scarce network bandwidth resource, making it a challenging task especially for very large scale distributed systems generating dynamic context data in huge volumes. Context data distribution is typically done by the context management middleware that transparently manages and routes huge volume of context data between the producers and the consumers, while ensuring the desired QoC. While different aspects of data distribution by context management middlewares have been explored in existing body of research works [2], the significant overhead of context data distribution for desired QoC that leads to system scalability and reliability issues have relatively been less explored.

In this paper, we address this particular problem of managing the overhead of context data distribution and propose a theoretical model for middleware design for large scale distributed context aware systems. Our approach stems from the observation that not all the data generated in a system is relevant for context computations and hence does not need to be distributed always, thereby saving network resources resulting in improved system resource utilization and efficiency. In our model, we propose a formal framework for the context management middleware that can identify the set of context data that is relevant to the currently defined context rules in the system for distribution, thereby reducing redundant context data dissemination. Simulation experiments show our proposed framework is capable of saving network resources significantly for very large scale systems.

Context data distribution, particularly in large-scale ubiquitous systems has been an important and fertile area of research work in the last decade or so. A comprehensive survey on the wide range of data distribution systems can be found in [2]. A number of proposals have been made to solve various issues with context data distribution in purely distributed context management systems [3] as well as in centralized context management systems [4]. The fundamental

difference with this body of work on context data distribution and our proposed approach is that the prior works primarily used subscription matching in the context management middleware for efficient data dissemination, whereas our work proposes a novel approach of efficient data dissemination based on actual context data utility. In our proposed scheme, the producers disseminate only the context data that impacts the context consumed by the context-aware applications in the runtime.

2 A Motivating Example

We explain the motivation behind our work on a simple example of a monitoring system. Consider a scenario with 3 sensors, communicating with a central context management server. The first sensor records the water storage level along with the information on whether it is raining or not. The second sensor records the temperature and checks if it is favourable (in a certain range) and the third sensor measures a number of environmental parameters like CO2, chemicals, CO, hydrocarbons etc., to check if their values are within certain limits. The context management server implements the following context rules:

- *Context Rule 1:* Send a grain protection alert if any of the following holds:
  - It is raining and the water level > threshold (t_1) and the temperature is unfavourable.
  - If temperature is favourable, it is raining and water level > threshold (t_1).
  - The temperature is unfavourable and it is not raining and the water level < threshold (t_1).
  - The temperature is favourable, it is not raining and the water level < threshold (t_1).
- *Context Rule 2:* Send an air pollution alert if any of the following holds:
  - CO_2 measurement > threshold (t_4).
  - Chemical in the atmosphere > threshold (t_5).
  - Quantity of burning fossil fuels > threshold (t_6) and the emission of carbon monoxide, oxides of nitrogen, hydrocarbons and particulates > threshold (t_7).

The example presented here may appear to be non-intuitive and dry to the reader, but in reality, a general context system designer is equipped with very limited exposure to sophisticated logic and specification formalisms to be able to write context rules that are easily amenable to automated analysis. The example presented here is representative of the formal training (or lack of it) available to the designer. The context rules are usually expressed by domain experts (and not computer scientists/engineers/logicians) who model the system in natural language forms. We show here how a formal foundation can lead to remarkable efficiency in system performance as far as the transmission profile is concerned.

We now analyse the situation from the perspective of the sensors, which are entrusted with the responsibility of observing the parameters and communicating the values to the server, whenever there is a change in the value of the

parameter. Values received from the sensor are used by the server to evaluate the two rules and accordingly decide the next course of action on the alert signals. We define a few Boolean variables in Table 1 to model some predicates on the sensor inputs. These predicates are used in the context rules. We also assume the sensors are able to evaluate the values of the predicates (in other words, the Boolean variables), before sending the update to the server. The sensors communicate the Boolean variables, and the server can act accordingly, since the predicate-to-Boolean map is available there as well. The communication is done using the Boolean variables, which takes fewer message bits, as opposed to transmitting the exact value of the observed sensor inputs. This could have equivalently done using some encoding scheme for the predicates directly. However, we use the Boolean encoding just for the sake of simplicity of illustration.

For the purpose of illustration, we consider random changes of the variables and depending on those changes we observe the number of transmissions. Table 2 contains a 0 or 1 depending on whether the value of the predicate (or equivalently the corresponding Boolean variable) is true or false. For example, if the water-level in the storage $>$ threshold (t_1) then the value of the Boolean variable $v_1 = 1$, otherwise $v_1 = 0$.

Table 1. Definition of the Boolean variables

v_1 : Water-level in the storage $>$ threshold (t_1)

v_2 : Raining or not

v_3 : The temperature lies between lower threshold (t_2) and upper threshold (t_3) value

v_4 : CO_2 measurement $>$ threshold (t_4)

v_5 : Chemical in the atmosphere $>$ threshold (t_5)

v_6 : Burning fossil fuels $>$ threshold (t_6)

v_7 : The emission of carbon monoxide, oxides of nitrogen, hydrocarbons and particulates $>$ threshold (t_7)

How Will Standard Transmission Methods Work? Consider a synchronous transmission scheme. In this case, each transmission involves the values of the variables, either periodically or aperiodically (when something new happens). In the case of a change-driven synchronous transmission scheme, whenever there is a change in the value of any sensor observed variable, a transmission message is triggered, with the message carrying the latest valuations of all the variables (irrespective of whether they have changed or not). Quite evidently, this is not quite an efficient dissemination scheme. For the purpose of illustration, we have considered in Table 2 a window of 10 instants, with each slot showing the valuation of each variable. In the synchronous scheme, a total of 70 (in each of the 10 slots, all the 7 variables are transmitted) transmissions are required. We now consider an asynchronous transmission scheme, i.e. transmission only occurs when there is a change and only the change is transmitted and not the entire snapshot. In this case, 57 transmissions are required as shown in Table 2.

Table 2. Random changes of variables with respect to time

Time Instant	v_1	v_2	v_3	v_4	v_5	v_6	v_7
1	1	0	0	1	0	0	1
2	0	1	1	1	1	1	0
3	1	0	0	1	0	0	1
4	0	1	1	1	1	1	0
5	1	0	0	1	0	0	1
6	0	1	1	0	1	1	0
7	1	0	0	1	1	0	1
8	0	1	1	0	1	1	1
9	1	0	0	0	0	1	1
10	0	1	1	0	1	1	1
Total changes	10	10	10	4	8	8	7

Table 3. Transmission of variables with respect to time in our method

Time Instant	v_1	v_2	v_3	$v_4 + v_5$	$v_6 . v_7$
1	1	0	X	1	0
2	0	1	X	X	X
3	1	0	X	X	X
4	0	1	X	X	X
5	1	0	X	X	X
6	0	1	X	X	X
7	1	0	X	X	X
8	0	1	X	X	X
9	1	0	X	0	1
10	0	1	X	X	X
Total changes	10	10	0	2	2

Our Method at Work: If we analyse the context rules carefully, we can reduce the number of transmissions. If we observe context rule 1, we see that when it is raining and also the water level > threshold (t_1), the server sends the alert for protection of grain, irrespective of the temperature because in this case flood may occur. Again, if it is not raining and the water level < threshold (t_1), then also the server sends the alert without verifying whether the temperature is favourable or not because of the dryness in weather. Combining all the four cases, we can see the server does not need to know about the temperature for sending the protection alert. Hence, for rule 1, the sensor does not need to transmit the observations of temperature at all, since the server does not need to know about v_3 at all. Further, if we analyse the remaining two conditions [whether it is raining and whether the water level > threshold (t_1)], we see that the server sends an alert whenever both of them are true or both of them are false. So, the server changes its decision if any of them changes. Let us suppose, at a specific time instant, both of them are true. If one of them changes, the server has to change its decision. The same happens if both of them are false. Again, in the present instant, if one of them is true and the other is false and subsequently, if any of them changes, then either both of them become true or both of them become false. In this case as well, the server has to change its decision. So, knowing the values of both the variables v_1 and v_2 is *critical* to the server, since every change affects the server's decision. Our method can compute this critical set of variables in an efficient way.

If we observe the second rule, we see all the four variables are required to send the air pollution alert. However, as long as the CO_2 measurement > threshold (t_4), without knowing the status of the other three variables, the server can send the air pollution alert. The same happens if the chemical in the atmosphere > threshold (t_5). If the quantity of burning fossil fuels < threshold (t_6), the status of the emission of carbon monoxide, oxides of nitrogen, hydrocarbons and particulates does not affect the server's decision. However, if both the predicates are true at the same time instant, the server sends an air pollution alert irrespective of the status of the CO_2 measure and the chemical in the atmosphere. These observations lead to further reduction in the number of transmissions.

It is also possible to reduce the transmission by analysing the sensor's observations at the current snapshot. We can see that the 3^{rd} sensor observes CO_2 measurement and Chemical in the atmosphere. If any of the predicates v_4 and v_5 is true, the server sends an air pollution alert. Hence, instead of sending the individual status of these predicates, the sensor can send true if either of them is true and false if none of them are true. The server does not need to know exactly which of the predicates is actually true. This saves message bits.

Based on the above observations, we define the *effective utility* of a sensor input, based on the relevance of transmission of the predicates appearing in the context rules. With our proposed optimizations, 24 transmissions are required. The transmissions are described in Table 3, where each cell entry denotes the value transmitted to the server (X indicates no transmission). In this work, we put forward a formal framework for reducing the number of update transmissions, thereby, contributing to the overall objective of sensor network design.

3 Detailed Methodology

We consider the following simple system setting in this work.

- A set of n distributed sensors, with no communication among themselves.
- Each sensor observes a set of environment parameters, in their sensing zone. The parameters can be of arbitrary data type. We further assume the sensing zones to be unique, in other words, no parameter is observed by more than one sensor. We assume this for simplicity of analysis.
- The sensors communicate with a central server $\mathcal{S}$ via message passing.
- The server $\mathcal{S}$ has a set of context-triggered actions.

At the server end, before the network is initialized, the server pre-processes the context rules using our method and communicates to the sensors, which observations are needed and at what instants. The overall objective of our scheme is to save the sensors from needless transmissions. Based on the values received from the sensors, the server evaluates the context-triggered actions and takes necessary steps. At the sensor end, the necessary set of predicates at the appropriate instants (as communicated to them by the server) are evaluated based on the observed values, and transmitted if needed. We illustrate our proposal in the following subsections. We begin with the definition of a data predicate.

Definition 1. Data Predicate: *A data predicate is an expression of the form* $< lexp > \bowtie < rexp >$*, where* $\bowtie \in \{=, \neq, <=, >=, <, >\}$*,* $< lexp >$ *and* $< rexp >$ *are data variables (e.g. temperature, humidity,* CO_2 *level etc.) or constants.* □

Given a valuation to its variables, a data predicate evaluates to *true* or *false*. Sensors transmit the Boolean values of relevant data predicates to the server.

Example 1. Consider a variable u which denotes the temperature of the system. A data predicate $\mathcal{P}$ is defined as: $\mathcal{P} : u < 20\,^{\circ}\text{C}$. If the temperature is less than $20\,^{\circ}\text{C}$ at any instant, $\mathcal{P}$ is evaluated as true, else it is false. □

Definition 2. Context-Triggered Action: *A context action is an expression of the form Rule: Action, where Rule is a Boolean combination (using Boolean operators) of data predicates over data variables and Action is the resulting actuation performed by the server.* □

The intuitive semantics of a context-triggered action is as follows: the *Action* will be executed only if the variable values at the current snapshot makes the *Rule* evaluate to *true*. For the sake of simplicity, we assume that the rule base is free from contradiction, if not, a consistency check does the job.

Example 2. Following is an example context triggered action: Context Rule: Room temperature $> 20\,^{\circ}$C *and* AC is off. Action: Server turns on AC. □

3.1 Transmission Control Mechanism

We now explain the optimizations in message transmission proposed in this paper. We begin with a few definitions, which help us build the foundations of our analysis framework. We refer to the data predicates appearing in the context rules using Boolean variables with the obvious interpretation (the predicate-variable map in Table 1) as in Sect. 2. In the following discussion, we use the term variable to mean a Boolean variable, which stands for a Boolean-valued predicate defined over the sensor-observed variable (arbitrary type).

Definition 3. Co-factor: *The positive (negative) co-factor of a context rule $\mathcal{C}$ defined over a set of Boolean variables $\mathcal{V} = \{v_1, v_2, \ldots v_n\}$ with respect to a variable $v_i \in \mathcal{V}$ is obtained by substituting 1 (true) or 0 (false) in $\mathcal{C}$.* □

The positive co-factor, denoted by $\mathcal{C}_{v_i}$ is obtained as $\mathcal{C}_{v_i} = \mathcal{C}(v_1, v_2, \ldots, v_i = 1, \ldots, v_n)$. Similarly, the negative co-factor, denoted as $\mathcal{C}_{\bar{v_i}}$ is obtained as $\mathcal{C}(v_1, v_2, \ldots, v_i = 0, \ldots, v_n)$. We now define the concept of decomposition of a context rule. This follows as a straightforward application of Shannon's expansion of Boolean functions [5]. The co-factors are independent of the variable v_i with respect to which they are computed.

Definition 4. Context Rule Decomposition: *The decomposition of a context rule $\mathcal{C}$ with respect to a variable $v \in \mathcal{V}$ is obtained as: $\mathcal{C} = v.\mathcal{C}_v + \bar{v}.\mathcal{C}_{\bar{v}}$, where $\mathcal{C}_v$ and $\mathcal{C}_{\bar{v}}$ are respectively the positive and negative cofactors of $\mathcal{C}$ with respect to v.* □

Example 3. Consider a context rule $\mathcal{C} = v_4 + v_5 + v_6.v_7$. The positive co-factor of $\mathcal{C}$ with respect to v_4 is $1 + v_5 + v_6.v_7 = 1$. The negative co-factor is $0 + v_5 + v_6.v_7 = v_5 + v_6.v_7$. The decomposition of $\mathcal{C}$ with respect to v_4 is $\mathcal{C} = v_4.\mathcal{C}_{v_4} + \bar{v_4}.\mathcal{C}_{\bar{v_4}}$. $= v_4.1 + \bar{v_4}.(v_5 + v_6.v_7) = v_4 + v_5 + v_6.v_7$. □

Optimizations Based on Co-factors: We begin our proposal of reducing message transmissions with a simple observation in terms of co-factors. Consider, the current value of $v = 1$ for some variable $v \in \mathcal{V}$ and also consider for some context rule $\mathcal{C}$, $\mathcal{C}_v = f(v_{i_1}, \ldots, v_{i_l})$. In this case, to evaluate $\mathcal{C}$, the server does not need to know the status of the variables in $\mathcal{V} \setminus \{\{v_{i_1}, \ldots, v_{i_l}\} \bigcup \{v\}\}$ until v gets changed, where $\setminus$ is the set difference operation. So, the sensor does not transmit the status of those variables to the server accordingly. In the special case, when $\mathcal{C}_v$ or $\mathcal{C}_{\bar{v}}$ equals to a constant, i.e. 0 or 1, that means the change of the state of other variables are not reflected in $\mathcal{C}$ as long as the state of v remains same. So to evaluate $\mathcal{C}$, the server does not need the status of any other variable other than v until it changes. Hence, the sensor can keep on observing v, but transmit only when it changes. At the initialization stage, the server analyses the context rules and gathers the set of variables whose positive or negative co-factors evaluate to a constant. This information is passed on to the sensors, and the sensor precisely knows when to transmit the valuations of the other variables it observes. Depending on the current snapshot, the server informs the sensors which variables are required.

Example 4. Consider Context rule 2 in Sect. 2. We analyze the context rule in the following way: $\mathcal{C}(v_4, v_5, v_6, v_7) = v_4 + v_5 + v_6.v_7$, then positive co-factor of $\mathcal{C}$ w.r.t. v_4, $\mathcal{C}_{v_4} = 1 + v_5 + v_6.v_7 = 1$, negative co-factor of $\mathcal{C}$ w.r.t. v_4, $\mathcal{C}_{\bar{v_4}} = 0 + v_5 + v_6.v_7 = v_5 + v_6.v_7$. So, when $v_4 = 1$, the server does not require to know the status of v_5, v_6, v_7 until v_4 becomes 0. The same happens for v_5: as long as $v_5 = 1$, the server does not need any information about v_4, v_6, v_7. □

Context rule decomposition can be extended to multiple variables as well. The decomposition of a context rule $\mathcal{C}$ with respect to two variables $u, v \in \mathcal{V}$ is obtained as: $\mathcal{C} = uv.\mathcal{C}_{uv} + u\bar{v}\mathcal{C}_{u\bar{v}} + \bar{u}v.\mathcal{C}_{\bar{u}v} + \bar{u}\bar{v}\mathcal{C}_{\bar{u}\bar{v}}$, where, $\mathcal{C}_{uv}$, $\mathcal{C}_{u\bar{v}}$, $\mathcal{C}_{\bar{u}v}$, $\mathcal{C}_{\bar{u}\bar{v}}$ are Shannon co-factors with respect to multiple variables. The co-factors are obtained as follows: $\mathcal{C}_{uv}$ by substituting the value of $u = 1$ and $v = 1$ in $\mathcal{C}$, $\mathcal{C}_{u\bar{v}}$ by substituting the value of $u = 1$ and $v = 0$ in $\mathcal{C}$, $\mathcal{C}_{\bar{u}v}$ by substituting the value of $u = 0$ and $v = 1$ and $\mathcal{C}_{\bar{u}\bar{v}}$ by substituting the value of $u = 0$ and $v = 0$.

In certain cases, co-factor analysis involving multiple variables helps us reduce the transmission. But in this case, if we want to analyze the co-factors for a context rule defined over a set $\mathcal{V}$ of n variables, we have in all an exponential number (2^n) of combinations, considering all subsets of all cardinalities of $\mathcal{V}$. In our work, we restrict ourselves to two variable co-factor analysis. Again, instead of looking at all the variables in the variable set $\mathcal{V}$ for *multiple variable co-factor analysis*, we can even concentrate on a smaller set, containing the variables whose positive and negative co-factors are not constant.

Example 5. Consider context rule 2 in Example 1 and the current snapshot $v_4 = 0, v_5 = 0, v_6 = 1, v_7 = 1$. According to *single variable co-factor analysis*, the sensor needs to transmit the status of all the four variables, since none of the co-factors are constant. *Multiple variable co-factor analysis* reveals that $\mathcal{C}_{v_6 v_7} = v_4 + v_5 + 1.1 = 1$. So as long as v_6 and v_7 remain 1, the server does not need the information of v_4 and v_5. But if any of them changes, the server requires the information of v_4 and v_5. □

For each context rule, we first analyze single variable co-factor and create a decision table which contains two fields: the value of the variable and depending on that value of the variable which variables are not required to evaluate the context rule. This is followed by a two variable co-factor analysis using the variables whose positive and negative co-factors are not constant. If the set contains v number of variables, then we have $\binom{v}{2}$ many combinations, and for each combination, there exists four different values (corresponding to the four co-factors). The results of the two variables co-factor analysis is stored in the decision table in the same format.

Optimizations Based on Unateness Analysis: We define a few more concepts that further reduce the number of transmissions.

Definition 5. Unateness: *A context rule $\mathcal{C}$ is positive unate in v_i if changing the value of v_i from 0 to 1 keeps $\mathcal{C}$ constant or changes $\mathcal{C}$ from 0 to 1.* □

$\mathcal{C}(v_1, v_2, \ldots, v_{(i-1)}, 1, v_{(i+1)}, \ldots, v_m) \geq \mathcal{C}(v_1, v_2, \ldots, v_{(i-1)}, 0, v_{(i+1)}, \ldots, v_m)$.
Negative unateness is defined similarly.

The unateness criterion leads to an interesting observation. If for every $v_i, \mathcal{C}$ is either positive or negative unate in v_i, then $\mathcal{C}$ is said to be unate function otherwise $\mathcal{C}$ is called binate function. Suppose a context rule $\mathcal{C}$ is evaluated to 1 at some point of time according to the current snapshot. *Then the value of $\mathcal{C}$ does not change with the change of all those variables whose current value is* 0 *and in which $\mathcal{C}$ is positive unate.* In other words, these variables are not needed to be transmitted. As earlier, the server decides on these conditions in the preprocessing stage and informs the sensor about its transmission requirements.

Example 6. Consider one context rule $\mathcal{C} = v_1.v_2.v_3.v_4 + v_5.v_6 + v_7$. Also consider the current snapshot $(v_1 = 1, v_2 = 1, v_3 = 1, v_4 = 1, v_5 = 0, v_6 = 0, v_7 = 0)$, and according to this snapshot $\mathcal{C}$ evaluates to true. We can see that $\mathcal{C}$ is positive unate in $\{v_1, \ldots, v_7\}$. So as long as $\mathcal{C}$ remains 1, the change of all the variables whose current value is 0, i.e. $\{v_5, v_6, v_7\}$, does not affect the value of $\mathcal{C}$. Accordingly the sensors are informed not to transmit these variables.

It is interesting to note that unate analysis takes care of the situation where co-factor analysis with respect to more than two variables are involved.

Optimizations Based on Effective Utility: Analyzing the criticality of a variable with respect to a context rule leads to further reductions. It is important to know which of the variables in the rule can actually influence the evaluation result of the rule. This is the effective utility of the variable with respect to the context rule. There may be some variables whose changes are not reflected in the context rule at all and there may be some variables for whom every change is crucial for the evaluation of the context rule. This is handled using *derivative analysis.*

Definition 6. Derivative: *The derivative of a context rule $\mathcal{C}$ with respect to a variable v_i is defined as the exclusive-or of the positive and negative co-factors of $\mathcal{C}$ with respect to the variable v_i, $\partial\mathcal{C}/\partial v_i = \mathcal{C}_{v_i} \oplus \mathcal{C}_{\bar{v_i}}$.* □

The intuitive idea is as follows. v_i can only change from 0 to 1 or 1 to 0. Consider the present value of v_i as 1, then $\mathcal{C} = (1.\mathcal{C}_{v_i} + 0.\mathcal{C}_{\bar{v_i}}) = \mathcal{C}_{v_i}$. When the value of v_i changes to 0, $\mathcal{C} = (0.\mathcal{C}_{v_i} + 1.\mathcal{C}_{\bar{v_i}}) = \mathcal{C}_{\bar{v_i}}$. $\mathcal{C}$ changes with change in v_i if the values of the positive and negative co-factors are different. This is expressed using $\mathcal{C}_{v_i} \oplus \mathcal{C}_{\bar{v_i}}$. Hence, $\partial\mathcal{C}/\partial v_i$ denotes the change of the context rule $\mathcal{C}$ with respect to the variable v_i.

Analyzing the derivative leads us to interesting observations as below.

- $\mathcal{C}_{v_i} \oplus \mathcal{C}_{\bar{v_i}} = 0$, implies $\mathcal{C}$ is independent of v_i. This is because $\mathcal{C}_{v_i} = \mathcal{C}_{\bar{v_i}}$, hence $\mathcal{C}$ is not be affected with change in v_i. This means that the sensor does not need to send the status of v_i to the server at any time.
- $\mathcal{C}_{v_i} \oplus \mathcal{C}_{\bar{v_i}} = 1$, implies the variable v_i is *critical* to the server since every change in v_i is reflected in the context rule $\mathcal{C}$. Hence, all updates to a critical variable have to be transmitted to the server.
- $\mathcal{C}_{v_i} \oplus \mathcal{C}_{\bar{v_i}} = g(v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_m)$, implies g is free from v_i. In this case, the server requires to know the status of v_i at least once.

Example 7. Consider the first context rule in Sect. 2. We can analyze the context rule in the following way: $\mathcal{C}(v_1, v_2, v_3) = v_1.v_2.\bar{v_3} + v_1.v_2.v_3 + \bar{v_1}.\bar{v_2}.\bar{v_3} + \bar{v_1}.\bar{v_2}.v_3$. The derivative of $\mathcal{C}$ w.r.t. v_3, $\partial\mathcal{C}/\partial v_3 = \mathcal{C}_{v_3} \oplus \mathcal{C}_{\bar{v_3}} = 0$, Since no change of v_3 is reflected in $\mathcal{C}$, the server does not require to know the status of v_3 at all. Consider the derivative of $\mathcal{C}$ with respect to v_1 and v_2, $\partial\mathcal{C}/\partial v_1 = \mathcal{C}_{v_1} \oplus \mathcal{C}_{\bar{v_1}} = 1$; $\partial\mathcal{C}/\partial v_2 = \mathcal{C}_{v_2} \oplus \mathcal{C}_{\bar{v_2}} = 1$, so all the changes of v_1 and v_2 are reflected in $\mathcal{C}$. Therefore, the server requires to know every change of v_1, v_2. □

Optimizations Based on Sensor's Data Analysis: The number of transmissions can be reduced further by analyzing the sensor's data in the current snapshot. Instead of sending the individual value of all variables, it can merge the value of some variables either by ANDing or by ORing, depending on the context. The variables sent by a sensor should have the following properties:

- The derivative of the variables with respect to some context rule must not be constant.
- Either all of them are part of a context rule or none of them are part of a context rule.
- One of the co-factors of each variables is constant.
  - If either of the co-factors of the variables is 1, then instead of sending their individual value sensor can send only one value by ORing them.
  - If either of the co-factors of the variables is 0, then instead of sending their individual value, sensor can send only one value by ANDing them.

Example 8. Consider the second context rule in the previous example. If a sensor needs to send the status of v_4 and v_5, then instead of sending their individual

values, it is better to send the value of $v_4 + v_5$, because the context rule is true if any one of them is true and if both of them are false then the decision of the context rule depends on the value of v_6, v_7. □

4 Putting Everything Together

We are given a set $\mathcal{C}$ of context rules over a set of Boolean predicates encoded as Boolean variables $\mathcal{V}$. The server performs the following pre-processing steps.

- Step 1: The server performs derivative analysis. $\mathcal{V}_1 = \{v \in \mathcal{V}$ *such that* $\partial c/\partial v = 1, \exists c \in \mathcal{C}\}$, $\mathcal{V}_2 = \{v \in \mathcal{V}$ *such that* $\partial c/\partial v = 0, \forall c \in \mathcal{C}\}$; Based on the result, the server informs the sensors to stop monitoring the data variables corresponding to the data predicates belonging to $\mathcal{V}_2$ and also to transmit all changes of the data predicates belonging to $\mathcal{V}_1$.
- Step 2: Evaluates the positive and negative co-factors of all context rules with respect to each variable belonging to $(\mathcal{V} \setminus \mathcal{V}_2)$ in the cases where $\partial c/\partial v \neq 1$.
- Step 3: Analysis of the two variable co-factors for each context rule $\mathcal{C}$ using the set of variables such that $\mathcal{C}_v$ and $\mathcal{C}_{\bar{v}}$ are not constant for all v in that variable set.
- Step 4: Sensor Data Analysis: (variables belonging to $\mathcal{V}_3 = (\mathcal{V} \setminus (\mathcal{V}_1 \cup \mathcal{V}_2))$), the server can instruct the sensor which variables can be transmitted by ORing or ANDing. Let the new set be $\mathcal{V}_4$, where some variables belonging to $\mathcal{V}_3$ are combined into one. Though they are different data variables while observing, they are combined while transmitting.
- Step 5: For each context rule($\mathcal{C}$), find two sets of variables in which the context rule is positive and negative unate, such that $\forall v$ belong to that sets $\partial c/\partial v$ is not equal to constant.

The following are the execution time computations:

- Find the set of variables ($\mathcal{V}_c$) to evaluate each context rule ($\mathcal{C}$) depending on the current snapshot. Then the server informs the appropriate sensors to stop transmitting variables belonging to $\mathcal{V}_3 \setminus (\cup_{c \in \mathcal{C}} \mathcal{V}_c)$.

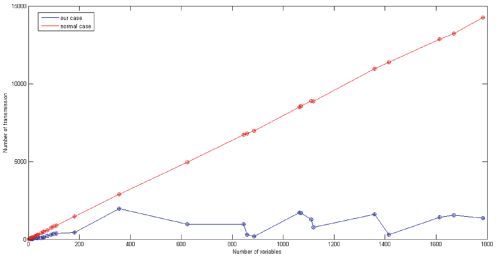
5 Implementation and Results

We implemented the proposed framework in Java. We tested our method on two published context system descriptions (entries 5 and 12 of Table 4 corresponding to the descriptions in [6,7]) and random environments, where the number of variables, the context rules and the number of sensors and their observation sets were randomly generated. The results are shown in Table 4. We have compared our method with the asynchronous transmission scheme with respect to the number of transmissions. A graphical view of the comparative transmission numbers is shown in Table 5, showing the transmissions in usual case denoted by red line and the number of transmissions in our case denoted by the blue line, with increase in the number of variables (horizontal axis).

Table 4. Test-case statistics

No. of variables	No. of formulas	No. of sensors	Avg. observation of sensors	Tx in usual case	Tx in our case
3	1	1	3	24	14
4	3	2	2	26	21
6	1	2	3	52	29
6	9	2	3	51	30
6	1	1	6	49	9
8	8	3	3	65	35
9	5	3	3	62	34
10	15	3	3	80	27
12	19	4	3	100	20
15	10	7	2	131	46
16	3	8	2	120	23
16	5	3	5	129	35
19	5	5	4	134	40
26	29	12	2	213	68
26	140	13	2	206	159
33	15	10	3	251	79
37	19	12	3	296	81
38	48	20	2	301	104
39	11	20	2	303	84
51	8	16	3	408	79
58	29	6	9	458	123
60	44	8	8	489	117
74	40	20	4	566	209
90	68	24	4	728	289
100	138	38	3	836	341
109	80	33	3	887	366
182	48	4	50	1478	429
357	1022	2	150	2898	1972
624	143	11	57	4949	956
845	93	6	141	6727	959
858	22	6	143	6815	297
885	14	4	227	6961	187
1065	183	2	523	8495	1717
1070	148	15	72	8550	1689
1110	97	4	278	8888	1259
1119	64	19	59	8862	767
1358	116	17	80	10964	1614
1414	20	7	202	11385	284
1614	110	13	124	12856	1423
1670	136	14	120	13216	1537
1783	99	19	94	14256	1367

Table 5. Number of variables Vs Transmission of variables



6 Conclusion

In this paper, we have proposed a new scheme for data dissemination in a large-scale context aware system. Experimental results show promising improvements in the number of transmissions. With context aware systems gaining widespread popularity in recent times, we believe our work will have interesting applications. We are currently working on generic data types and predicates and more complex context rules for more realistic applications of this work.

References

1. Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Commun. Mag. **49**, 32–39 (2011)
2. Bellavista, P., Corradi, A., Fanelli, M., Foschini, L.: A survey of context data distribution for mobile ubiquitous systems. ACM Comput. Surv. **44**, 1–45 (2012)
3. Macedo, D.F., Santos, A.L.D., Nogueira, J.M.S., Pujolle, G.: A distributed information repository for autonomic context-aware MANETs. IEEE Trans. Netw. Serv. Manag. **6**, 45–55 (2009)
4. Li, F., Sanjin, S., Schahram, S.: COPAL: an adaptive approach to context provisioning. In: Proceedings of the IEEE 6th International Conference on Wireless and Mobile Computing, Networking, and Communications (WiMob10), pp. 286–293 (2010)
5. Hachtel, G.D., Somenzi, F.: Logic Synthesis and Verification Algorithms, 1st edn. Kluwer Academic Publishers, Norwell (2000)

6. Yao, W., et al.: Using ontology to support context awareness in healthcare. In: Proceedings of the 19th Workshop on Information Technologies and Systems (2009)
7. Wang, X.H., et al.: Ontology based context modeling and reasoning using OWL. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18–22 (2004)