A Collaboration Mechanism Between Wireless Sensor Network and a Cloud Through a Pub/Sub-based Middleware Service

Mohammad Hasmat Ullah[1,3]
Sung-Soon Park[1,3]
[1]Department of Computer Science and Engineering
Anyang University,
Anyang, Korea
e-mails: {raju, sspark}@anyang.ac.kr

Jaechun No[2]
[2]Dept. of Computer Software,
Collage of Electronics and Information Engineering
Sejong University,
Seoul, Korea
e-mail: jano@sejong.ac.kr

Gyeong Hun Kim[3]
[3]Gluesys Co., Ltd.
Anyang, Korea
e-mail: kgh@gluesys.com

***Abstract*— Cloud computing has emerged as a new paradigm of computing platform. It covers almost every area of computing and provides platform to most of the data services. On the other hand, Wireless Sensor Networks (WSN) has gained attention for their potential supports and attractive solutions such as environment monitoring, bio-median acknowledgment, healthcare monitoring, industrial automation, etc. Additionally, our virtual groups and social networks are in main role of information sharing. However, this sensor driven data is not available to community groups or cloud environment for general purpose research or utilization yet. If we reduce the gap between real and virtual world by adding this WSN driven data to cloud environment and virtual communities by providing censor driven contents to general researchers, and it can gain a remarkable attention from all over, by giving us the benefit in various sectors. Collaboration between WSN and the cloud environment can achieve this. We have proposed an integrated Publish/Subscribe (pub/sub)-based middleware service for the cloud platform to collaborate with WSN. This collaboration will provide resource, service, and storage with sensor driven data to the community. Furthermore, we have proposed a content-based event matching algorithm to analyze subscriptions and publish proper contents easily. We have evaluated our algorithm which sows better performance comparing with previously proposed algorithms.**

Keywords-Cloud computing; WSN; middleware service; event matching; pub/sub

I. INTRODUCTION

Interests are increasing about WSN for their essentiality. Multiple small sensing nodes gather information and monitor events to provide data processing, which couples the digital world with physical environment. It has been gaining importance for their contribution by sensing processing and communicating in vast areas like environmental monitoring and forecasting, medical, military, transportation, crisis management, bio-median acknowledgment, industrial automation, etc. They allow the interaction between users and physical environment. Although a WSN has unlimited potentiality for numerous application areas, it contains sensor devices with limited sensing capability, low processing power, and poor communication power.

Besides, cloud computing provides unlimited resource, processing power, storage and reliable services. Cloud computing provides access to applications and data from anywhere and anytime. The applications are hosted as "Software as a Service". Only cloud computing can provide unlimited resource, computing power, bandwidth, storage, dedicated servers to access from anywhere anytime to use application like software. If we can utilize both powerful platforms together, we may get benefitted by all means.

Super computer may provide resource and power to process sensor data, but it is not easily available for general use and needs much overhead. Cloud computing can analyze, process and store the vast amount of data collected by sensors and these sensors can be shared by applications and users easily, which is the main reason to collaborate WSN to the cloud. Not only cloud provides powerful computation but also serves with huge amount of storage to store processed sensor data for further use.

We propose an integrated pub/sub-based middleware for cloud platform to collaborate with sensor network. It will monitor the subscriptions for sensor driven data through cloud and will receive sensor produced data, also will encapsulate those data as event and will provide them to appropriate subscribers. This middleware will deliver information to the subscribers, who has subscribed for the sensor driven data through cloud-based application.

To accomplish this, we need an algorithm for event matching, which will provide sensor driven data to subscribers. Our proposed middleware will simplify the integration of sensor network with cloud-based community centric applications. The middleware provides an efficient event matching algorithm to bring appropriate sensor driven data to appropriate users.

[1]This research is supported by WBS (World Best S/W) Development Project, Grants No. 10040957, funded by Ministry of Knowledge Economy Korea, 2011 and by Global IT Development project, Grants No. 10043026, funded by Ministry of Knowledge Economy Korea, 2012.

[2]This work is also supported by the 2008 Sabbatical year project from Anyang University.

 ISBN: 978-1-61208-285-1

In Section II, we review the previous work in this field. Section III illustrates the content-based middleware and describes our system overview, Section IV presents our proposed algorithm, Section V provides experimental methodology and experimental evaluation of content-based event matching algorithm for sensor cloud middleware, and Section VI states the conclusion of our work.

II. RELATED WORKS

So far, no many efforts were taken to address the issue of integrating sensor networks to cloud computing-based networks. SGIM [4] addresses the opportunity and challenges for sensor-cloud framework only for analyzing the healthcare sensor data for range predicate case only. Sensor-Grid [5] architecture is already proposed, but grid computing is not same as cloud computing [6] and setting up the infrastructure is not easy. Grid focuses on High Performance Computing (HPC) related applications, whether cloud focuses on general purpose applications, which is easily accessible from anywhere anytime for general users.

Our proposed middleware contains a content-based pub/sub model to deliver sensor driven processed data to subscribers, facilitating exchange between sensor networks and cloud-based networks. Pub/Sub system encapsulates sensor data into events and provides the service of event publications and subscriptions for asynchronous data exchange. The most notable pub/sub systems implemented in recent years are:

The MQTT-S [7] is a topic-based pub-sub protocol that hides the topology of the sensor network and allows data to be delivered based on interests rather than individual device addresses. It allows a transparent data exchange between WSNs and traditional networks and even between different WSNs. Mires [8] is a pub/sub architecture for WSNs. Basically sensors only publish readings if the user has subscribed to the specific sensor reading. Subscriptions are issued from the sink node which then receives all publications. Subscriptions are made based on the content of the desired messages in Distance Vector/Dynamic Receiver Partitioning (DV/DRP) [9]. Though subscriptions are flooding over the network, but DV/DRP only publishes data if there are some subscriptions for the specific data.

Several event matching algorithms are proposed to deliver published sensor data or events to subscribers. In Sequential and sub-order [10] algorithm, according to each predicate, searching space is gradually reduced by deleting unsatisfied subscriptions. The second algorithm, sub-order, reduces the expected number of predicate evaluations by analyzing the expected cost differences when subscriptions are evaluated in different orders. If two predicates are same and trying to create a chain in range predicate case, it is difficult to make chain in such scenario. So, it creates heavy overloads while inserting and deleting subscriptions as it has to maintain a complete graph.

III. MIDDLEWARE ARCHITECTURE

A. Pub/Sub Middleware

Our current environmental data monitoring and analyzing system does not provide real-time auto generated data when sensor gets such information about natural calamities just started to take place by passing sensor driven data to cloud environment through some collaborating middleware to share with the community. On the other hand, the researchers who are trying to solve some complex problems need data storage, computational capability, security at the same time to process vast amount of real time data. For example, assume that a team is working on the unusual environmental situation. They plot sensors on some specific regions to monitor the magnitude continuously and use this data for large multi-scale simulations to track the natural calamities along with providing auto generated forecast to the end-users, who has subscribed to know the forecast. This may requires computational resources and a platform for sharing data and results that are not immediately available to the team. Traditional HPC approach like Sensor-Grid model [5] can be used in this case, but setting up the infrastructure as mentioned above is not easy in this environment. Cloud data centers, such as Amazon EC2, can provide resource and platform to keep many copies in a data center and to provide them when needed. Though, they did not address the issue of integrating sensor network with cloud applications, and thus, have no infrastructure to support this scenario. Here, the subscribers need to register their interests to get various environmental states (magnitude, temperature of ionosphere, electromagnetic field, etc.) from sensors for large scale parallel analysis and to share this information with each other for finding useful solutions for their research related problem. So, the sensor data needs to aggregate first, then process and, lastly disseminate based on user's subscriptions.

B. System Overview

In our proposed system, we have a pub/sub-based middleware to make interaction between cloud and a WSN to provide appropriate data to appropriate subscribers. WSN generates real-time data and needs to be processed at the same time. Our proposed middleware connects to such WSNs and receives real-time data, then processes them and prepares those data as events. The sensor data come in many forms, such as raw data and that raw data must be captured, filtered and analyzed in real-time, and also sometimes it should be stored and cached for further use. Pub/Sub-based middleware also has registry, analyzer and disseminator. Subscribers can request for sensor data through cloud API (Application Programming Interface). There may be two kinds of subscription: i) general purpose for end-users or community-based users to get processed data like forecast about earthquake or natural calamities, ii) special purpose for encapsulated data as event for further research.

 ISBN: 978-1-61208-285-1

Simple architecture of our proposed middleware is shown in Fig. 1

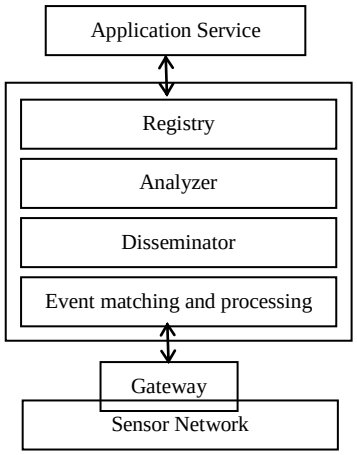

Figure 1. Simple middleware architecture

Interested subscribers can subscribe through cloud application; this subscription will be stored and categorized. The Pub/Sub middleware receives sensor driven data from the gateway between WSN and the middleware, then event matching and monitoring section encapsulates these data as event and passes to the analyzer. The analyzer analyzes subscription types and the disseminator provides corresponding data to subscribed users by matching the registry through the cloud API. The cloud environment may manage these data, process them and may also keep to the repository for further utilization as needed. General user will be able to get user friendly output of these complex data by matching its predicates and by normalizing it.

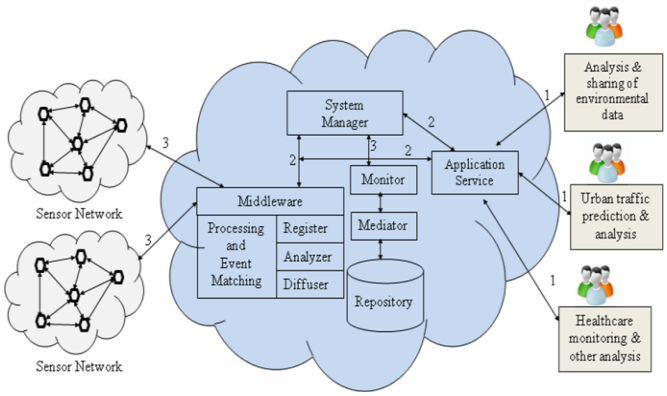

Figure 2. Middleware service integrating WSN and cloud environment.

Figure 2 shows the overview of proposed system. Our proposed middleware service is integrated with cloud platform joining the WSN with cloud. Cloud service provides the application for users to subscribe based on their interests as needed. The proposed event matching algorithm will provide appropriate data efficiently to the subscribers.

IV. EVENT MATCHING ALGORITHM

We need an efficient event matching algorithm for our system to deliver published data to appropriate subscribers. Our target is a cloud-based environmental data monitoring and analyzing system, where researchers can express their interests into attributes, and also general end-users can request for easy to understand outputs. First, we have implemented to support range predicates to cover multi range data only; then, we have extended the algorithm to support overlapping predicates also.

A. Event Matching

In our system, a subscription S is expressed by a pair (ID, C_i, P_i), where ID is the subscriber's ID, C is subscription category and P is a set of predicates specifying subscriber's interest.

Here is an example of a subscription and an event in the system. Subscription: S [magnitude, 7(+), ionosphere temperature, 300K(+)] contains two predicates that are joined together to specify a discrete value predicate; here, magnitude 7(+) represents 7 and more alternately ionosphere temperature 300K(+) indicates from 300K to max, i.e., P_1 = magnitude >= 7 and P_2 = ionosphere temperature >= 300K. We also can express it as 6.9 < magnitude < 8 and 299K < temperature < 500K. Let event e be; e: [magnitude = 7.6, ionosphere temperature = 350K].

```
1.  C is set of indexes {C1, ,….Cn-1,Cn} where n is no of indexes
2.  Each Ci points to a set of category index or single category S'
3.  P is set of predicates {p1, p2,….pm-1,pm} where m is number of
    predicates in a subscription
4.  Initialize pj = searching predicate
5.  Event E containing set of predicates P' = {p'1, p'2,….p'm-1,p'm}
6.  Procedure Search (pj, C, E, C_out)    search event E in C where
    C_out is output subscription set   ⊳
7.  S_tmp is a temporary set
8.  for each Ci in C    check each category for desired subscription
9.      if (Ci contain⊳ E) then
10.         if (j ≠ m) then
11.             Procedure Search(pj, Ci, E, C_out)
12.         else then    already found
13.             Initializ⊳S_tmp = S'
14.             C_out = C_out U S_tmp
15.             for each p'j in P'
16.                 for each s'r in S_tmp
17.                     if (s'r. p'j doesn't match E. p'j) then
18.                         Delete the subscription from output set
19.                         Delete the subscription from temporary set
20.                     end if
21.                 end for
22.             end for
23.         end if
24.     end if
25. end for
```

Figure 3. Pseudo code for event matching algorithm

An event satisfies a subscription only if it satisfies all predicates in the subscription. Here, the event [magnitude = 7.6, ionosphere temperature = 350K, 375K] satisfies the subscription S as our proposed method supports discrete predicate values also. So, the matching problem is: Given an event e and a set of predicates in subscription set S. We need to find all subscriptions in set S that are satisfied by e. Our middleware supports various expressions of predicates. First, "(data >= LV || data <= UV)" [here LV = lower value and UV = upper value] is used when consumers want to know normal patterns of sensed data. Second, "(LV > data || UV < data)" is used when consumers need to receive unusual states of the situation such as natural calamities.

B. Proposed Method

Here, we describe the Category Matching Algorithm (CMA). This algorithm operates in three stages. In the first stage, it preprocesses subscriptions by categorizing them through the predicates corresponding to the relevant properties of events. The basic categorizing idea from statistics is employed to decide the number of category. In the second stage, matching subscriptions or predicates are derived sequentially. All predicates stored in the system are associated with a unique ID. Similarly, subscriptions are identified with subscription ID. Finally, it will store the sensor driven data to knowledgebase for future analysis.

Suppose S is a set of subscriptions, $S = \{s_1, s_2, \ldots s_{n-1}, s_n\}$, where n is total number of subscriptions and P is a set of predicates in S, $P = \{p_1, p_2, \ldots p_{m-1}, p_m\}$, where m is the total number of predicates in a subscription. In our system, we have two predicates in a subscription (i.e., data > LV and data < UV) and these two predicates are used to categories the subscriptions. We define a set S' that contains all the subscriptions of S sorted by LV value in ascending order. Then, we define a categorizing sequence (mC_1, $mC_2, \ldots mC_g$). The categorizing space, denoted by SP (S', c), is defined as the set containing all such category sequences over S' and c. Now, each $mC_{i=1\ldots c} \in SP(S', c)$ contains k = n/c subscriptions; that are why category index is created for each $cI_i \in mC_{i=1\ldots c}$. Here, this categorizing sequence is called almost balanced categorizing sequence since every category contains same number of subscriptions except the last one which may or may not contain the same number of subscriptions. It depends on the value of c and n.

When categorizing of subscriptions is done in the above way, first predicate of an event is compared with category index $cI_1 \in mC_1$ and, if any match found then second predicate is compared with category indexes $hI_i \in MC_{i=1\ldots h}$. This way all categories are found that matches with event data. Finally, sequential matching is done in the selected categories to find the subscriptions that are satisfied by all predicates in the event.

V. EVALUATION

Our experimental methodology and simulation results are presented in this section. We have compared our proposed method with sequential sub-order [10], forwarding [14], and naïve [10] algorithms. Naive, a baseline algorithm, evaluates the subscriptions independently. Specifically, it evaluates the subscriptions one by one and for each subscription, evaluates its predicates until either one of them becomes false or all of them are evaluated.

A. Experimental Methodology

Due to the lack of real-world application data, it is not easy to evaluate this kind of pub/sub system. Previous works show that in most applications, events and subscriptions follow either uniform or Zipf [10] distribution. We have used both distributions to evaluate our proposed algorithm. We used subscription evaluation cost, which is the average number of predicates that the system evaluates to match an event for the subscription. This is only a rough estimation of the absolute time that the matching process may take, because different operators may have different complexity and even the same operator may take different time slots for different parameters. However, in a long-term average sense, we believe the number of evaluated predicates can well reflect the efficiency of the evaluation process.

B. Experimantal Results

We have compared Naïve, Sequential and sub-order algorithms with our CMA using a uniform distribution. The experiment results of evaluation are given below:

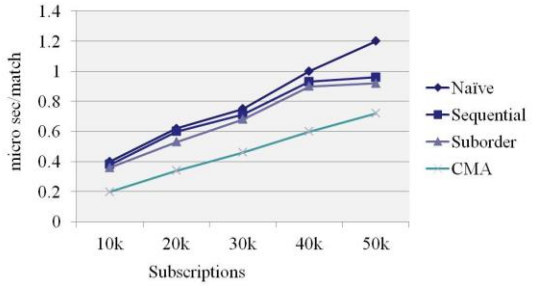

Figure 4. Matching time vs. number of subscriptions.

From first comparison, we can observe that CMA performs better than all other algorithms. For example, with 10K subscriptions and 5000 events, the naïve, sequential, sub-order and CMA evaluate predicates in 0.4, 0.38, 0.36, 0.2 micro sec respectively. Thus, CMA reduces the evaluation cost by 50%, 42%, and 38% as compared to naïve, sequential, and sub-order algorithms, respectively.

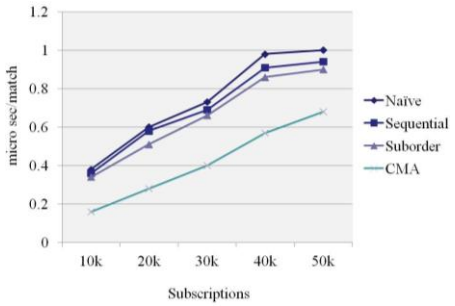

Figure 5. Matching time vs. number of subscriptions (Zipf distribution)

Again, we repeated the experiments with the same parameter settings except the distribution follows Zipf rather than the uniform distribution. The experiment results exhibit similar trends as in first comparison.

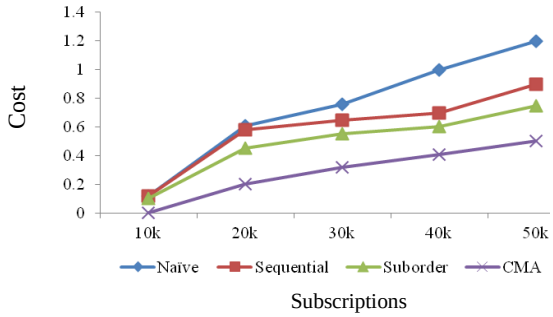

Figure 6. Evaluation cost for multi range predicates

Figure 6 shows that for multiple ranges of predicates, our algorithm performs much better than others. For example, beginning from 10k subscriptions and 5000 events; Naïve, sequential, and sub-order event matching performed 35% ~ 55% poorer than CMA. The cost is evaluated in micro seconds.

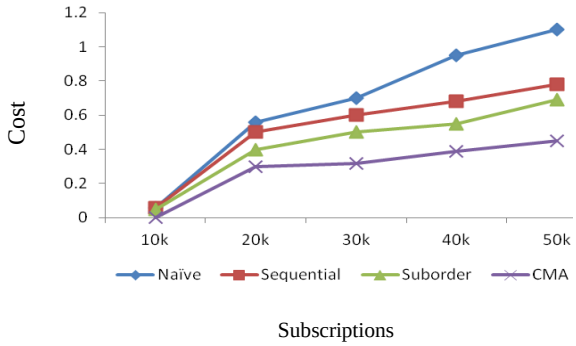

Figure 7. Evaluation cost for overlapping predicates

Figure 7 shows the comparison result for the overlapping predicates. As the subscription increases, CMA shows better and better performance than others. So, it will outperform if the subscriptions are larger.

The above experiments clearly show that our CMA algorithm performs better (in case of uniform and Zipf distribution) than the existing ones in terms of efficiency and scalability.

VI. Conclusions And Future Work

In this paper, we have proposed a pub/sub-based middleware service for the collaboration between sensor networks and the cloud environment for utilizing the ever-expanding sensor data for various next generation community-based sensing applications. For the computational tools needed to launch this exploration, it is more appropriate to build them in the data center of "cloud" computing model than the traditional HPC approaches or Grid approach. We proposed a middleware to enable this by content-based pub/sub model. To deliver published sensor data or events to appropriate users of cloud applications, we also have proposed an efficient and scalable event matching algorithm. We evaluated its performance and also compared it with existing algorithms in a cloud based environment analysis scenario. In the future, we will study further to make the middleware more efficient for distributing sensor driven data to appropriate subscribers and will try to simplify the communication overhead between WSNs and cloud environment.

References

[1] R. Buyya, C. S. Yeo, and S. Venugopal, "Market Oriented Cloud Computing: Vision, Hype and Reality for Delivering IT Services as Computing Utilities," Proc. of 10th IEEE Conference on HPCC, Dalian, China, Sep 2008, pp. 5-13.

[2] K. K. Khedo and R. K. Subramanian, "A Service-Oriented Component-Based Middleware Architecture for Wireless Sensor Networks," International Journal of Computer Science and Network Security, vol. 9, no. 3, Mar 2009, pp. 174-182.

[3] A. Weiss, "Computing in the Clouds," netWorker magazine, ACM Press, vol. 11(4), Dec 2007, pp. 16-25, doi: 10.1145/1327512.1327513.

[4] M. M. Hassan, B. Song, and E. N. Huh, "A framework of sensor-cloud integration opportunities and challenges," Proc. ICUIMC'09, ACM, 2009, pp. 618-626, doi: 10.1145/1516241.1516350.

[5] H. B. Lim, et al., "Sensor Grid: integration of wireless sensor networks and the grid," Proc. of the IEEE Conf. on Local Computer Networks, Nov 2005, Sydney, Australia, pp. 91-98.

[6] D. Harris, "The Grid Cloud Connection (Pt. l): Compare and Contrast," http://www.hpcinthecloud.com/hpccloud/2008-10-08/the_grid-cloud_connection_pt_i_compare_and_contrast.html, retrived: July, 2013.

[7] U. Hunkeler, H. L. Truong, and A. S. Clark, "MQTT-S – A publish/subscribe protocol for Wireless Sensor Networks," IEEE Conf. on COMSWARE, Bangalore, India, Jan 2008, pp. 791-798, doi: 10.1109/COMSWA.2008.4554519.

[8] E. Souto, et al., "Mires: a publish/subscribe middleware for sensor networks," ACM, Personal and Ubiquitous Computing, vol. 10(1), Dec 2005, pp. 37-44, doi: 10.1007/s00779-005-0038-3.

[9] C. P. Hall, A. Carzaniga, J. Rose, and A. L. Wolf, "A content-based networking protocol for sensor networks," Department of Computer Science, University of Colorado, Technical Report, Aug 2004.

[10] Z. Liu, S. Parthasarthy, A. Ranganathan, and H. Yang, "Scalable event matching for overlapping subscriptions in pub/sub systems," Proc. DEBS'07, ACM Press, 2007, pp. 250-261, doi: 10.1145/1266894.1266940.

[11] M. Gaynor, et al., "Integrating wireless sensor networks with the grid," IEEE Internet Computing, vol. 8(4), Jul-Aug 2004, pp. 32–39, doi: 10.1109/MIC.2004.18.

[12] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, "The many faces of publish/subscribe," ACM Computing Surveys, vol.35(2), June 2003, pp. 114–131, doi: 10.1145/857076.857078.

[13] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "TinyREST – A Protocol for Integrating Sensor Networks into the Internet", Proc. of Real-World Wireless Sensor Networks (REALWSN), Stockholm, Sweden, June 2005.

[14] A. Carzaniga and A. L Wolf, "Forwarding in a content-based network," Proc. SIGCOMM, ACM Press, 2003, pp. 163-174, doi: 10.1145/863955.863975.

 ISBN: 978-1-61208-285-1