International Journal of Advanced Smart Convergence Vol.4 No.2 109-119 (2015)
http://dx.doi.org/10.7236/IJASC.2015.4.2.109

IJASC 15-2-16

Pub/Sub-based Sensor virtualization framework for Cloud environment

Mohammad Hasmat Ullah[a,c], Sung-Soon Park[a,c*], Jaechun No[b] and Gyeong Hun Kim[c]

[a]*Department of Computer Science and Engineering, Anyang University, Korea*
[b]*Dept. of Computer Software, Sejong University, Korea*
[c]*Gluesys Co., Ltd., Korea*
{raju, sspark}@anyang.ac.kr, jano@sejong.ac.kr, kgh@gluesys.com

Abstract

The interaction between wireless sensors such as Internet of Things (IoT) and Cloud is a new paradigm of communication virtualization to overcome resource and efficiency restriction. Cloud computing provides unlimited platform, resources, services and also covers almost every area of computing. On the other hand, Wireless Sensor Networks (WSN) has gained attention for their potential supports and attractive solutions such as IoT, environment monitoring, healthcare, military, critical infrastructure monitoring, home and industrial automation, transportation, business, etc. Besides, our virtual groups and social networks are in main role of information sharing. However, this sensor network lacks resource, storage capacity and computational power along with extensibility, fault-tolerance, reliability and openness. These data are 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, then it can gain a remarkable attention from all over, along with giving us the benefit in various sectors. We have proposed a Pub/Sub-based sensor virtualization framework Cloud environment. This integration provides resource, service, and storage with sensor driven data to the community. We have virtualized physical sensors as virtual sensors on cloud computing, while this middleware and virtual sensors are provisioned automatically to end users whenever they required. Our architecture provides service to end users without being concerned about its implementation details. Furthermore, we have proposed an efficient content-based event matching algorithm to analyze subscriptions and to publish proper contents in a cost-effective manner. We have evaluated our algorithm which shows better performance while comparing to that of previously proposed algorithms.

Keywords: *Sensor Network ; Cloud computing; Pub/Sub middleware;*

1. INTRODUCTION

Wireless sensor networks do much more than just reducing wiring cost or even just connect sensors to other nearby sensors and devices. They open the realm of cloud computing to include Sensors as a Service and also can be utilized in several areas like environmental monitoring and forecasting, medical, military, transportation, crisis management, bio-median acknowledgment, industrial automation, etc. These sensors

Manuscript Received: Sept. 14, 2015 / Revised: Oct. 12, 2015 / Accepted: Nov. 4, 2015
Corresponding Author: sspark@anyang.ac.kr
Tel: +82-31-463-1347
Department of Computer Science and Engineering, Anyang University, Korea

may provide various useful data that are closely attached to each of their relevant applications and services directly, causing several other services to be unused. Thus, huge number of valuable resources become unused and waste. Multiple small sensing nodes gather information and monitor events to provide data processing, which couples the digital world with physical environment. 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. If we can integrate these sensors by sharing each other's data through unlimited services, it can help to accelerate service creation.

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 of these two 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 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.

In our previous work we have proposed an integrated pub/sub-based middleware for cloud platform to collaborate with sensor network which monitors subscriptions for sensor driven data through cloud and receives sensor produced data, also encapsulates those data as event and provides them to appropriate subscribers. This middleware delivers information to the subscribers, who has subscribed for the sensor driven data through cloud-based application. In this stage, we have studied several cloud-sensor architectures, and tried to improve our previously proposed system by enabling data to be categorized, stored and processed in such a way that it becomes cost effective, timely available and easily accessible.

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

In Section 2, we reviewed the previous work in this field. Section 3 illustrates the content-based middleware and describes our system overview, Section 4 presents our proposed algorithm, Section 5 provides experimental methodology and experimental evaluation of content-based event matching algorithm for sensor cloud middleware, and Section 6 states the conclusion of our work.

2. RELATED WORKS

The concept of sensor-cloud is a new paradigm for cloud computing that uses the physical sensors to accumulate its data and transmit all sensor data into a cloud computing infrastructure. Sensor-Cloud handles sensor data efficiently and then used for monitoring several applications. Sensor cloud infrastructure [16] has been proposed which is the extended form of cloud computing to manage our valuable sensors scattered around the network (sensor). This infrastructure would provide the service instances (virtual sensors) automatically to users as and when requested in same way as these virtual sensors are part of IT resources (like disk storage, CPU, memory etc.) to the end users. These service instances and its appropriate sensor data can be used via a user interface through the web crawlers. Before generating the service instances; the IT

resources (like CPU, Storage devices etc.), sensor capable devices, service templates (that has to be used to create virtual sensors) should be prepared first. SGIM [5] addresses the opportunity and challenges for sensor-cloud framework only for analyzing the healthcare sensor data for range predicate case only. Sensor-Grid [6] architecture is already proposed, but grid computing is not same as cloud computing [7] 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.

There exist no application that can make use every kind of physical sensors all time; instead each application required pertinent physical sensors for its fulfillment. To realize this concept publish/subscription [17] mechanism is being employed for choosing the appropriate physical sensor [18]. Our proposed middleware contains a content-based publish/subscription 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 [8] 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 [9] 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) [10]. 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 [11] 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.

3. VIRTUALIZATION FRAMEWORK

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 [6] 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 WSN to provide appropriate data to appropriate subscribers. WSN generates real-time data and needs real time processing. 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.

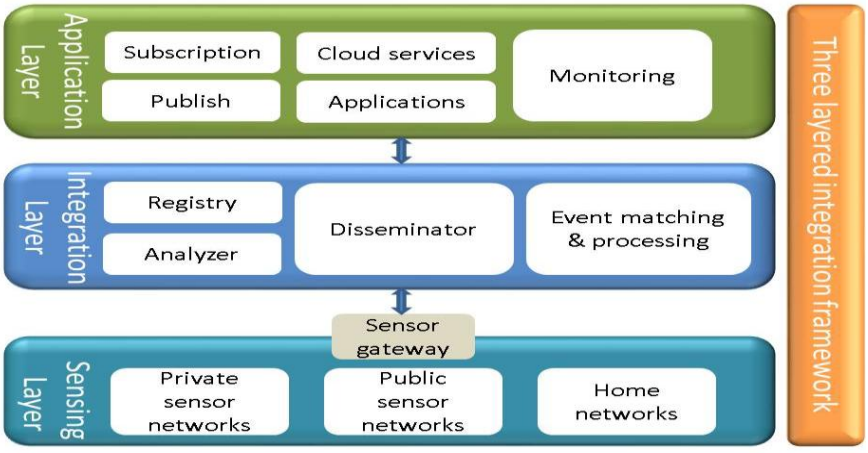

Figure 1. Three layered integration framework

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.

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.

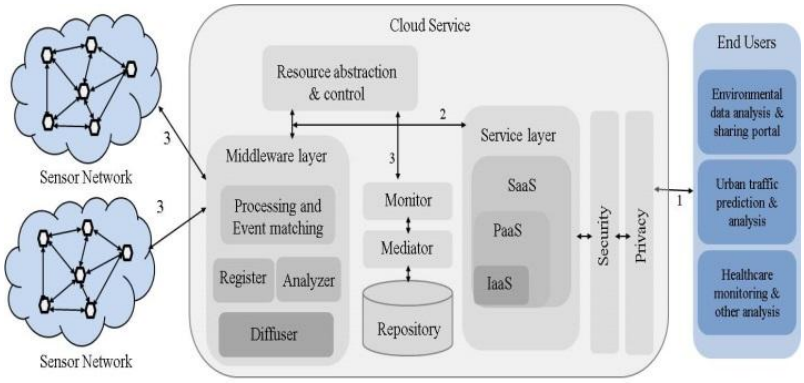

Figure 2. Middleware framework for Sensor virtualization to Cloud environment

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.

4. 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 contains E) then
10.         if (j ≠ m) then
11.           Procedure Search(pj, Ci, E, C_out)
12.         else then ⊳ already found
13.           Initialize 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.

5. EVALUATION

Our experimental methodology and simulation results are presented in this section. We have compared our proposed method with sequential sub-order [11], forwarding [15], and naïve [11] 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 [11] distribution. We have used both distributions to evaluate our proposed algorithm. We have evaluated the algorithm in a Linux machine with Xeon 3.5GHz and 4GB memory. 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.

Table 1. Workload parameters in the experiment

Parameter	Description	Value
N_s	number of subscription	10k~50k
Model	Popularity distribution	Uniform or Zipf
Min_s	min predicate in a subscription	3
Max_s	max predicate in a subscription	5~11
N_e	no. of events	1000
$ratio_{match}$	Matched subscription ration	0.3%

We have conducted the experiments using the set of subscriptions and events as previously implemented algorithms has been evaluated[11] so that we can carefully examine various aspects of our system. The parameters used in synthesizing the workload are listed in Table 1. We feed these events into the system and evaluate the performance of different algorithms. For each parameter setting, we repeat the experiments for multiple times and the results reported in the paper are based on the average over these runs.

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:

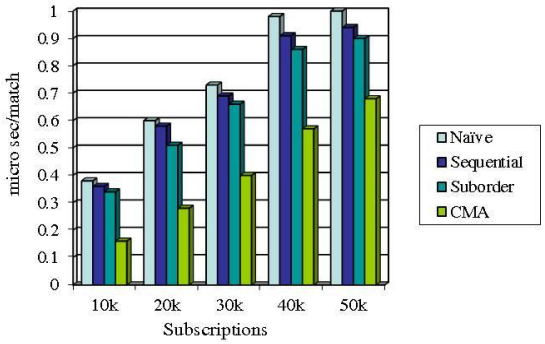

Figure 4. Matching time vs. number of subscriptions.

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.

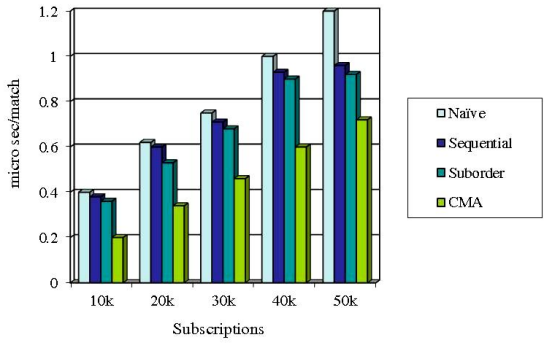

Figure 5. Matching time vs. number of subscriptions (Zipf distribution)

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.

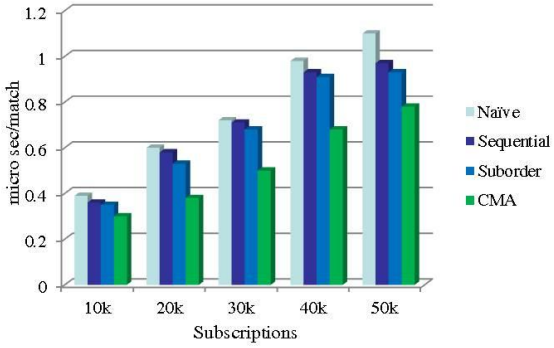

Figure 6. Matching time vs. number of subscriptions (Exponential Distribution)

Figure 6 shows similar performance if we use same parameter settings with exponential distribution (p = 2). The experiment result exhibits similar trend as in above comparisons except this time performance of each algorithm is little closer to each other.

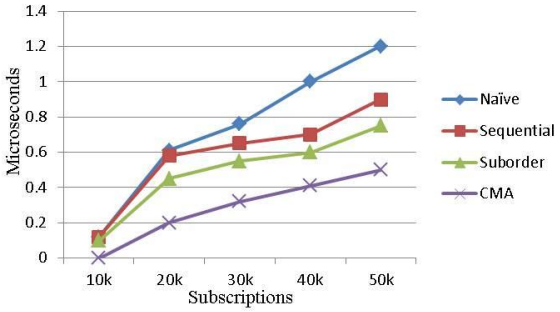

Figure 7. Evaluation cost for multi range predicates

Figure 7 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.

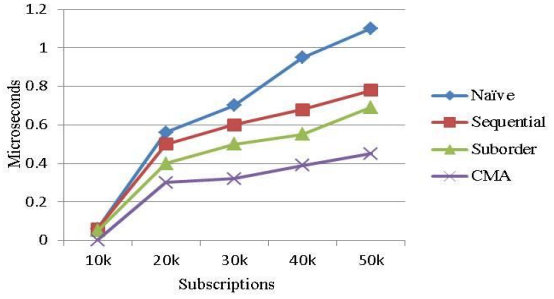

Figure 8. Evaluation cost for overlapping predicates

Figure 8 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.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a pub/sub-based sensor virtualization framework to integrate sensor networks and the cloud environment for utilizing Sensors as a Service. 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 future, we will study further to make the virtualization framework more efficient for distributing sensor driven data to appropriate subscribers and will try to simplify the communication overhead between WSNs

and cloud environment. Further research work will lead us to utilize IoT devices through the cloud securely as it needs proper authentication to be accessed and controlled.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R0101-15-0159, Development of Responsive RRUI Technology based on Device/Environmental/Emotional/Cognitive Information for Information Technology Devices).

REFERENCES

[1] M. H. Ullah, et al., "A Collaboration Mechanism Between Wireless Sensor Network and a Cloud Through a Pub/Sub-based Middleware Service," Proc. of 5th INTERNET Conference, Nice, France, pp. 38-42, July 2013.

[2] 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, pp. 5-13, Sep. 2008.

[3] 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, pp. 174-182, Mar. 2009.

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

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

[6] 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, pp. 91-98, Sydney, Australia.

[7] 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.

[8] 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, pp. 791-798, Jan. 2008.
DOI: 10.1109/COMSWA.2008.4554519.

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

[10] 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.

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

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

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

[14] 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.

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

[16] MadokaYuriyama and Takayuki Kushida, "Sensor Cloud Infrastructure: Physical Sensor Management with Virtualized Sensors on Cloud Computing," 13th International Conference on Network-Based Information Systems, IEEE, 2010.

[17] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzerand M. Welsh, (2004), "Hourglass: An Infrastructure for Connecting Sensor Networks and Applications," Harvard Technical Report TR-21-04, 2004.

[18] M. Gaynor, M. Welsh, S. Moulton, A. Rowan, E. LaCombe, and J.Wynne, "Integrating Wireless Sensor Networks with the Grid," IEEE Internet Computing, Special Issue on Wireless Grids, 2004.