Multi-dimensional Sensor Data Aggregator for Adaptive Network Management in M2M Communications

Kenji Yoi[1], Hirozumi Yamaguchi[2], Akihito Hiromori[2], Akira Uchiyama[2], Teruo Higashino[2]
Naohisa Yanagiya[3], Toshikazu Nakatani[3], Atsuo Tachibana[4], and Teruyuki Hasegawa[4]
[1]Nara Institute of Science Technology, yoi.kenji.yc6@is.naist.jp
[2]Osaka University, {h-yamagu,hiromori,uchiyama,higashino}@ist.osaka-u.ac.jp
[3]KDDI corporation, {na-yanagiya,to-nakatani}@kddi.com
[4]KDDI R&D Laboratories, Inc. {tachi,teru}@kddilabs.jp

***Abstract*—This paper proposes a method of aggregating tempo-spatial data generated by sensors deployed in buildings or houses. The size of each sensor data such as temperature is usually small, but it often involves many additional data to represent its attribute values like time, location, data type and data precision. This would often increase the traffic volume between sensor gateway at building/home side and service providers at server side. In our method, such sensor data are packed into multi-dimensional matrices indexed by those attribute values for more compact representation, and the compressed sensing technique is adaptively applied to further reduce the data size. The method was applied to a field trial with KDDI corporation to collect data from 29 community facilities, and the traffic volume was reduced to 50% with reasonable precision of data restoration.**

I. Introduction

Machine-to-Machine communication (M2M) is a promising architecture where not only legacy computation and communication devices but also machines and sensors, which have been formerly offline, are now being online and becoming new information sources that generate real-time information to be delivered to the rest of the world of interest. In particular, nowadays buildings and houses have more sensors being deployed in their inside than ever, for remote environmental indoor sensing to capture temperature, humidity, power generation and consumption and so on, as well as for remote human tracking to better understand contexts in those buildings and houses. Using such information, optimal strategies to control HVACs, devices, home appliances and machines can be designed and situation-aware services such as recommendation to advice healthy and eco-friendly life, urgent medical care and monitoring, and human-centric demand-response services can be provided to buildings and home residents.

Due to such features of dealing with a lot of (or a variety of) sensors and due to requirements for real-time monitoring, the sensor gateways must sufficiently be capable of handling and packing such a number of sensor data and transferring data to service providers (simply called providers hereafter) efficiently. However, there are several issues and constraints on realizing such communications in M2M applications and services. Firstly, each sensor is not often optimized for data aggregation; it usually works as a standalone device and accompanies several attribute values such as time, location, data-type and precision to EACH small measured data such as temperature. Therefore, in order to send N temperature data, we need to send N data tuples, where each tuple consists of a single measured data with multiple attribute values. Nevertheless, link bandwidth between a sensor gateway and a provider is often limited. This likely happens in mass housing of urban areas where a number of families share a single public connection to the server. It also happens with elderly people who are living in suburb areas. For real-time elderly people care, we need remote monitoring of such houses, and the wireless broadband (3G/LTE) is promising alternative which is easy to be deployed. Nevertheless, such a wireless service is often unavailable or severely limited in suburb areas. In particular, Japan has a lot of mountain areas, and most people living in such areas are elderly people. In order to assure the remote sensing services, an efficient aggregation and transfer mechanism is mandatory. Finally, the computational complexity of data aggregation and compression at sensor gateways should be low as such functions are implemented and executed on resource-constrained computers such as STBs. As a consequence, we need a lightweight and flexible sensor data aggregation mechanism that can adaptively aggregate a number of sensor data and compress them to fit to bandwidth limitation.

For adaptive network management in such M2M data sensing architectures and contexts, in this paper, we design and develop an adaptive Multi-dimensional Sensor Data Aggregator called *MSD Aggregator* (or *MSDA* in short). MSDA takes the following two steps. Firstly, it aggregates a set of tempo-spatial sensor data (*e.g.* temperature data on the same floor over an hour) to remove redundant attribute values. As such data have similar attribute values like two timestamps “10:30:29 on July 1, 2014” and “10:30:30 on July 1, 2014” but they are slightly different from each other, we need to send both attribute values if pursuing accuracy. Meanwhile, depending on sensor types and situations, such difference is ignorable in real operations. Therefore, those attributes should be aggregated allowing

some deviation from the true values and such sensor data are packed into multi-dimensional matrices indexed by those attribute values after pruning redundant attribute values for more compact representation. Secondly, the data aligned into a matrix are further compressed by the compressed sensing technique, which only requires linear transformation of data sequence at sensor gateways. As it is lossy compression, the decompressed data at a provider side may contain some errors. However, we have developed a technique to keep the errors as small as possible in the l_1-reconstruction process, which results in only small errors.

We have implemented our method on a small Linux box and conducted a large field experiment supported by the Ministry of Internal Affairs and Communications of Japan. It was conducted in Toyota city and its surrounding region, which include both well-populated urban areas and depopulated suburb areas. In the experiment, KDDI corporation has collected data from 29 community facilities by applying our MSDA and the traffic volume necessary to send all the sensor data to a single server was reduced to 50% with reasonable precision of data restoration.

II. RELATED WORK

Many applications have been designed or developed to monitor and collect multiple sensor data. One of the most common data structures is *Extensible Markup Language (XML)*, which is a simple and flexible data format based on *Standard Generalized Markup Language (SGML)*, to transmit and handle the data from sensors. Similarly, *JSON [1]* is also widely used for web applications because it can represent data in more concise human-friendly ways. Google has presented a serialization format called *message pack* [2], in which fast coding and decoding can be performed and data size was small. However, these formats do not consider data compression. A compressive sampling technique is widely used to handle and compress such huge data. For example, it is well known that image, audio and movie data can be compressed drastically by choosing appropriate vectors to represent most elements in the data matrix as 0 through DCT or wavelet techniques. The multi-dimensional data from environments are compressed by choosing vectors according to sparse characteristics for environmental data such as temperature and humidity. Ref. [3] has investigated several techniques for networked data in WSNs through considering the distributed data sources and their sampling, transmission, and storage.

In home and building management systems, a gateway is usually introduced to deal with heterogeneous sensors that do not have sufficient computational power [4][5]. On the other hand, MQ Telemetry Transport (MQTT) [6] is designed to enable low-cost and low-power sensors to send their data to the server directly over bandwidth-constrained WSNs such as ZigBee or TinyOS based networks. MQTT is a topic-based pub/sub protocol and also facilitates the subscription to multiple topics. It supports basic end-to-end Quality-of-Service (QoS) according to application requirements, *e.g.* the highest QoS level in MQTT ensures that messages are

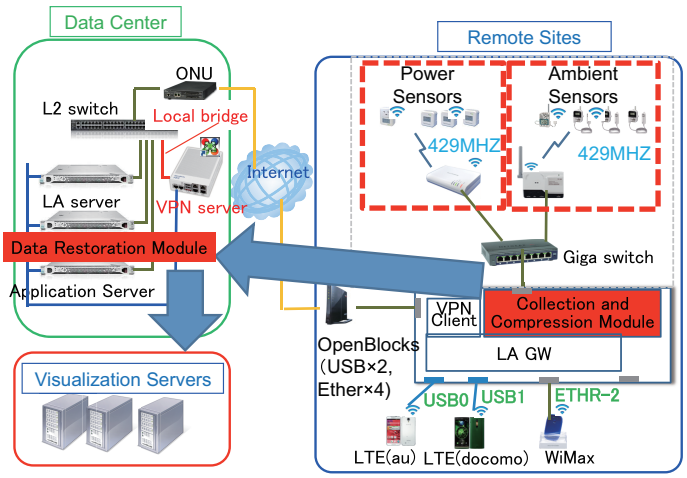

Fig. 1. System Architecture

delivered only once to the receiving entities. Although an MQTT server can handle about ten thousands sensors, it does not support a data compression function.

Compared with the existing techniques, our method focuses on efficient collection of data from sensors and their transmission to a server. These sensing data contain data values with a variety of attribute values such as sensor names, precision, geometry, frequency, and privacy policies. In our method, we try to prone and compress them drastically by the compressive sampling technique with a low error rate. In addition, our important contribution is that we have developed the system and deployed it in the real facilities in Toyota city where the effectiveness of our method has been confirmed.

III. MSD AGGREGATOR: BASIC DESIGN

In this section, we describe our MSD (Multi-dimensional Sensing Data) aggregator that collects data from a set of sensors, compresses them efficiently, and send them to a server via a sensor gateway deployed at a remote site.

A. System Architecture

Figure 1 illustrates the system architecture. Basically, a gateway at a remote site (usually houses and buildings, but is not limited to them) collects sensing data from the sensors deployed at the site, and transmits the data to the server.

More specifically, MSD aggregator works as follows. Firstly, (i) each sensing data value and its attribute values embedded in XML or other formats are extracted. Secondly, (ii) the values extracted by step (i) are packed into a *Multi-dimensional Sensing Data (MSD) matrix* indexed by the attributes after pruning redundant meta data. Thirdly, (iii) a sub-matrix from the matrix of step (ii) is selected and the compressed sensing technique is applied to transform the data part of the sub-matrix into a compressed data sequence. (iv) The data sequence is transferred by a cellular or wired connection to a server. Finally, (v) the data sequence is decompressed to restore the original data.

We note that steps (i), (ii) and (iii) are performed on a sensor gateway, while (v) is done on a server.

TABLE I
MULTI-DIMENSIONAL SENSING DATA (MSD) MATRIX

		1	2	3	4
	Location / Timestamp	Living	Bathroom	Bedroom	Dining
1	2013:07:03:08:00:00	125	125	250	252
2	2013:07:03:08:00:30	126	125	250	253
3	2013:07:03:08:01:00	126	126	251	N/A
4	2013:07:03:08:01:30	126	126	251	N/A

B. Multi-dimensional Sensor Data Matrix

This section describes steps (i) and (ii) of the above procedure.

Sensors periodically generate sensing data, *e.g.*, for every second or every tens of seconds. Each sensor data value such as temperature usually accompanies some *attribute values* such as sensor ID, timestamp, location and precision, which should also be delivered with the data. However, they are usually wrapped by XML or some other structural languages to pack the attribute and sensing values into a single data block. Therefore, we assume that we have extracted those sensing data and attribute values from a block generated by each sensor by XML parser tools.

Formally, we assume there are p types of attributes that are common to all the sensing data, and denote i-th attribute as A_i. Suppose $|A_i|$ is the number of values for attribute A_i. We also assume that the values of each attribute are uniquely indexed by sequential numbers 1, 2, ..., $|A_i|$. Assuming each pair of an attribute index and an attribute value is known at the gateways and the server, the MSD matrix is consisted of $|A_1| \cdot |A_2| \cdot \cdots \cdot |A_p|$ elements which correspond to sensor values.

A simple example of an MSD matrix is shown in Table I with two attributes A_1 (timestamp) and A_2 (sensor ID). The attribute indexes are given in the first row and column, and the sensor values are given in each element. "N/A" in the matrix represents empty data (no values) due to sampling intervals, sensor reading errors, and/or some other reasons.

The problem here is how to send such a matrix efficiently while there may be some (or many) N/A elements in the matrix. In particular, obviously, the MSD matrix is likely to be sparser with larger p and/or finer attribute values.

1, time, ID = (8:00, 0) = 0 2, time, ID = (8:00, 1) = 1 ...

4, time, ID = (9:00, 0) = 5, time, ID = (9:00, 0)

For a spatio-temporal block with n sensor IDs and timestamps with intervals of t seconds, we let a bit sequence $X_n X_{n-1} \dots X_1$ denote the availability of sensing data. The sensing data is available when X_i is 1 and not available when it is 0. If the same bit sequence is repeated m times starting from the time stamp YYYYMMDDhhmmss (i.e Year, Month, Day, hour, minute and second), we define the meta data of the block as

$$(X_n ... X_2 X_1)_{10}, YYYYMMDDhhmmss, t, m. \quad (1)$$

In this manner, our method prunes redundant attribute values from the meta data. For example, the meta data of Table I is

$$15, 20131207080000, 30, 2,$$
$$14, 20131207080100, 30, 2, .$$

The proposed method repeats transmission of the meta data and the sensing values of the blocks. We apply compressed sensing described in the following section to the sensing values to further reduce the amount of data size.

C. Compression and Restoration by Compressed Sensing

Compressed sensing [7] is a signal processing technique which can probabilistically recover original signals from fewer samples by leveraging sparseness of signals. The sparseness is the characteristic to represent the number of zero (N/A) elements in a matrix.

In this paper, we regard sensing values of a spatio-temporal block as signal and apply compressed sensing. We convert N sensing values to the relative values to the mode among the N values to maximize sparseness. The vector of the converted N sensing values is called the original vector and represented as N-dimensional vector $\boldsymbol{x}_0 \in \mathbb{R}^N$. An original vector is called "K-sparse" if the number of non-zero values is at most K.

Data compression is conducted by multiplying an $M \times N$ matrix A ($M < N$) called a sensing matrix and an original N-dimensional vector $\boldsymbol{x}_0$. The result of the above matrix multiplication is a compressed M-dimensional vector $\boldsymbol{y} \in \mathbb{R}^M$, i.e.

$$\boldsymbol{y} = A\boldsymbol{x}_0 \quad (2)$$

The original vector $\boldsymbol{x}_0$ is restored from the compressed vector $\boldsymbol{y}$ and the sensing matrix A. However, $\boldsymbol{x}_0$ cannot be uniquely identified since N variables are unknown while M equations are given where $M < N$. Therefore, we use l_1-minimization [8] to estimate the original vector. Due to space limitations, the details of l_1-minimization are omitted. Please refer to Ref. [8] for precise procedures.

IV. MSD AGGREGATION SYSTEM DEVELOPMENT

A. Overview

We have developed a sensor data compression and collection system which collects sensing data from multiple sensors at remote sites such as houses, apartments, and office buildings.

We deployed sensors that monitor temperature, humidity, and CO2 density in some rooms at each site. The monitored data is transmitted to a gateway in the site at the interval of seconds to tens of seconds. The gateway applies the meta data pruning to the collected sensor data, compresses the MSD matrix by compressed sensing described in Sec. III, and transmits the compressed data with attribute values to the server in the data center. The server decompresses the received data to restore the original data, which are provided to applications for visualization.

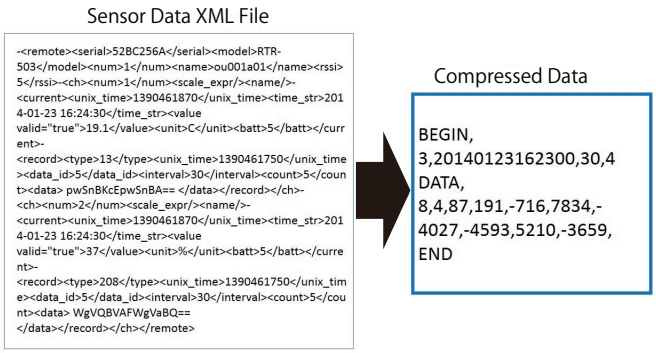

Fig. 2. Example of Compressed Data Format

B. Compression and Collection Module

A compression and collection module runs on a gateway and applies the meta data pruning and the compressed sensing. The size of a data matrix and a compression rate are specified to determine the size N of the original vector and the size M of the compressed vector. Then, the $M \times N$ sensing matrix is determined based on a random matrix.

Figure 2 shows an example of the compression of a sensor data XML file. In the compressed data, the pruned meta data and the compressed MSD matrix are described in the `BEGIN-DATA` block and the `DATA-END` block, respectively. For the restoration, the compressed data also contains the size N, the number K of non-zero elements, the mode and the mean of the original vector. For the evaluation, the original vectors were stored at gateways. The module is implemented by Python.

The module at a gateway sends the compressed data to the server in the data center via sockets. When transmission fails, it retries at a specified interval. The module at the server listens on the port, accepts connections from gateway modules and receives the compressed data. The received data is restored by the restoration module. The restoration module restores the compressed data by l_1-minimization and add the meta data for each restored element. We use OpenOpt and FuncDesigner Python modules to derive solutions of l_1-minimization. The Python-based l_1-minimization function is called by the restoration module, which is implemented by Ruby. We also implemented a visualization module by JavaScript to visualize the sensor data collected at the server.

C. Link Aggregation Based on CMT-SCTP

We use the Link Aggregation (LA) communication system [9] developed by KDDI R&D Labs based on CMT-SCTP (Concurrent Multipath Transfer Stream Control Transmission Protocol) for communication between gateways and the data center. LA aggregates multiple physical communication links to provide virtual bandwidth expansion and high redundancy. We assume data collection from a vast number of sensors. In such an environment, burst data transmission may occur due to unstable communication links. Then, the amount of data may be huge although we reduce the amount of data by the proposed method. Therefore, LA is efficient in our environment by aggregating not only wired links but also wireless links such as LTE and WiFi. By combining LA and the proposed method, we aim to achieve large-scale and continuous sensor data collection with a little bandwidth.

V. Real Experiment

Fig. 3. Experimental Sites in SmartCity

A. Environment

We developed and deployed the proposed system with KDDI corporation in Toyota city as a part of Toyota City Low-carbon Society Verification Project (Smart Melit) [10]. This project is funded by The Ministry of Internal Affairs and Communications, Japan, and focuses on integration and control of next-generation vehicles by HEMS as well as home appliances, enabling users to enjoy a pleasant and convenient low-carbon lifestyle without waste and effort. In real experiments, we deployed 29 gateways and 201 sensors to monitor temperature, humidity, and CO2 density at 29 sites such as smart houses, community centers, and nursery schools across the city. Figure 3 shows a smart house located in Toyota city, and a gateway and sensors installed in the house. The top-right figure shows sensors which measure temperature, humidity and CO2. The bottom-left figure shows a gateway in a site, which not only collects the sensing data from the sensors but also aggregates and compresses the data, so that it can send the data to the server with reasonable size.

Each sensor records corresponding values every 30 seconds and transmits the recorded values to its gateway every 5 minutes. The size N of the data matrix transmitted from gateways was set to 12, 15, 20, 24, or 30. We used temperature and humidity monitored by the sensors for evaluation. The row and the column of the matrix corresponds to the timestamps and the sensor identification numbers (IDs), respectively. We used $N \times 1$ matrix size in the experiments except in Sec. V-E, which indicates compressed sensing is applied to N temporally consecutive values transmitted from a single sensor. In the following evaluation, we used two-days sensor data of which size is approximately 100MB collected from the sites.

TABLE II
COMPRESSION EFFICIENCY FOR PRUNED DATA

N	Data Size after Compression (KB)	Reduction Rate
12	7221	63.9%
15	6525	67.4%
20	6177	69.1%
24	6061	69.7%
30	5858	70.7%

B. Compression Efficiency

We evaluated the effect of the meta data pruning. We compared the data size after pruning with the original data size without XML tags. The original data size was 100MB, reduced to approximately 20MB after removing XML tags.

Table II shows the *reduction rates* for different data matrix size N. The reduction rate is defined as the rate of the reduced data size compared to the original data size without XML tags. The result shows that the meta data pruning becomes more efficient as the size N of the data matrix increases. This means the meta data pruning is efficient when we deployed many sensors since N becomes large with the increase of the number of sensors. Overall, the compressed data size is less than 10MB which is 10% of the original data size and the reduction rate is 60% to 70% excluding the effect of the meta data pruning. From the results, we confirmed our method could efficiently reduce the data size.

C. Restoration Precision

TABLE III
RESTORATION RATE

N	# of Elements	# of Correct Restorations	Restoration Rate
12	501,120	489,636	97.7%
15	501,120	487,113	97.2%
20	498,916	482,357	96.7%
24	498,916	477,485	95.7%
30	598,916	473,657	94.9%

The high compression rate does not straightforwardly indicate the effectiveness of our method since the compressed data is required to be restorable to the original data. To confirm the effectiveness of our method, we evaluated a *restoration rate* defined as the rate of correctly restored elements to the number of the original elements. The total number of elements was approximately 0.5 millions. The compressed data is transmitted from a gateway to the data center after the gateway receives N elements from the sensors. Hence, the number of the original elements depends on the matrix size N.

Table III shows the restoration rates for the different matrix size N. The restoration rate becomes worse with the increase of N. On the other hand, if we select smaller N, the meta data size becomes larger while the realtimeness of the data transmission increases since gateways do not need wait for the reception of a large number of sensor values. Considering the trade-off between the realtimeness and the data size, we conclude 20, 24, and 30 are appropriate values for N.

D. Effect of Compression Rate

To see the effect of the specified compression rate M/N, we evaluated the reduction rate and the *approximate restoration rate*. The specified compression rate is the ratio of the number M of elements in the compressed data compared to the original data with N elements. For example, when the number of elements is 20 and the compression rate is 0.6, the data is compressed so that the number of elements becomes 12. The approximate restoration rate is the rate of the number of restored elements within an error θ to the original. This metric is used because in practical use of M2M applications, the restored elements are not necessary equivalent to the original as long as the values in the restored elements are "close enough" to the original. In the following evaluation, we use $\theta = 0.3$. Actually, the effect of θ is very small since tenfold sensor readings are transmitted to the data center in our deployment. For example, only $\pm 0.03^{\circ}$C error is allowed for 10°C original data when $\theta = 0.3$. We note that the errors of almost all restored elements were within ± 1.0 in the collected data corresponding to $\pm 0.1^{\circ}$C error.

Table IV shows the reduction rates and the approximate restoration rates for the different data compression rate M/N where $N = 24$. The data compression rate was specified from 0.3 to 0.9 at increments of 0.1. The reduction rate increases with the decrease of the compression rate from 0.9 to 0.4, and starts decreasing from 0.4 to 0.3. This result indicates 0.4 is the lower bound of the compression rate. The lower bound exists because the compression rate by compressed sensing cannot exceed the sparseness of the original data. Therefore, the sparseness of the original data in the experiment is estimated to be approximately 60%-70% for $N = 24$. From the above results, we can see that the proposed method can adjust the compression rate as needed, which is essential for adaptive network management.

TABLE IV
SPECIFIED COMPRESSION RATE VS. RESTORATION/REDUCTION RATE

Specified Compression Rate (M/N)	Approx. Restoration Rate	Reduction Rate
0.9	96.7%	20.2%
0.8	96.4%	24.4%
0.7	96.0%	28.8%
0.6	95.8%	33.4%
0.5	95.5%	38.0%
0.4	95.5%	38.9%
0.3	95.4%	33.7%

E. Spatially vs. Temporally Compression

In addition to the data matrix size itself, the row and the column sizes are important because the effectiveness of the compressed sensing depends on the sparseness of the data matrix. Table V shows an example of data matrices constructed to enable our method to apply compressed sensing for sensor values in the same room. In the table, sensor IDs T1, T2, and T3 are temperature sensors, and H1, H2, and H3 are humidity sensors. To see the effect of row$\times$column sizes, we compared the restoration rates and the reduction rates of 4×6, 8×3, 24×1, and 8×6. We note that 8×6 is to see the effect of

temporal similarity by compressing 8 temporally consecutive values from 6 sensors.

From the results shown in Table VI, we can see that the reduction rate is high when 4×6 or 8×6 is selected. The column size 6 indicates that each row is consisted of sensor values from 6 sensors in the same room at each time. Therefore, the sensor ID bits in the meta data are always 1111, which leads to high reduction rates including the meta data after compression. However, the sparseness of 4×6 and 8×6 is 20%-30%, which is much lower than 60%-70% in the case of 24×1. The low sparseness results in low effectiveness of compressed sensing. Nevertheless, the result in Table VI reveals that the effect of meta data pruning more than compensates for the lack of the high sparseness.

TABLE V
DATA MATRIX OF SENSORS IN THE SAME ROOM

sensor ID / min:sec	T1	T2	T3	H1	H2	H3
56:00	221	201	201	460	470	460
56:30	221	201	201	460	470	460
57:00	221	201	201	460	480	460
57:30	221	200	201	460	480	460
58:00	221	200	201	460	480	460
58:30	221	200	201	470	480	460
59:00	220	200	201	470	480	460
59:30	220	200	201	470	480	460
...	...	...	...	...	...	...

TABLE VI
COMPRESSION EFFICIENCY OF DATA BLOCK SELECTION

Block Pattern (SensorID×timestamp)	Data Size after Compression (KB)	Reduction Rate
4×6	4669	76.6%
8×3	5597	72.0%
24×1	6061	69.7%
8×6	4640	76.8%

F. Analysis through Visualization

Finally, we show a snapshot of the visualization tool. In Fig. 4, each pie chart represents the total power consumption in each site and each color in the pie charts corresponds to the power consumption measured by each power sensor. Figure 5 also shows the changes of sensor data over time. This visualization tool helps analyzing relationships between power, temperature, and humidity, which is useful for demand response in a smart grid.

VI. CONCLUSION

In this paper, we have proposed a method for tempo-spatial data aggregation generated by a large number of sensors in buildings or houses. Our method prunes redundant meta data such as timestamp in the raw sensor data and leverages compressed sensing for further data compression considering the tempo-spatial similarity of the sensor data. The method is applied to a field trial with KDDI corporation conducted over 29 community facilities. The results have shown our method efficiently reduce the traffic volume by 50% with reasonable precision.

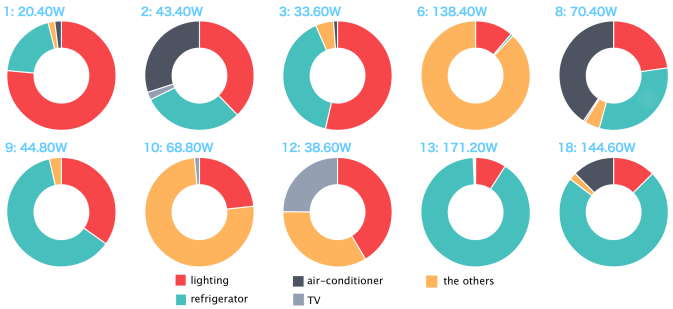

Fig. 4. Snapshot of Visualization Tool: Power Consumption at Remote Sites

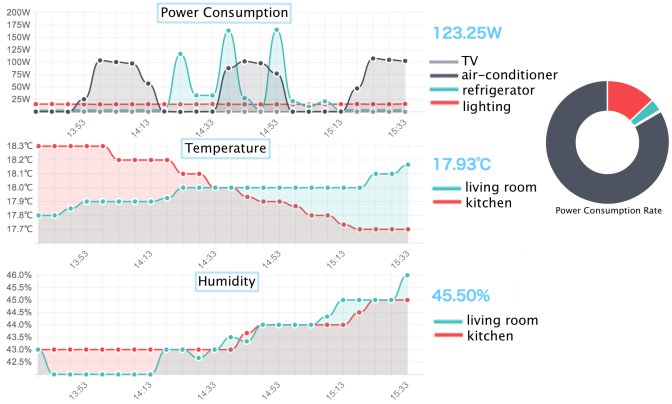

Fig. 5. Snapshot of Visualization Tool: Sensor Data over Time

ACKNOWLEDGMENT

This work is supported by the demonstration project for innovative information and communication technologies in smart grid by the Ministry of Internal Affairs and Communications, Japan.

REFERENCES

[1] Internet Engineering Task Force (IETF), "The javascript object notation (JSON) data interchange format," 3 2014, request for Comments 7159.

[2] S. Furuhashi, "Messagepack," http://msgpack.org/.

[3] J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, 2008.

[4] T. Schwartz, S. Denef, G. Stevens, L. Ramirez, and V. Wulf, "Cultivating energy literacy: Results from a longitudinal living lab study of a home energy management system," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. ACM, 2013, pp. 1193–1202.

[5] A. Biselli, E. Franz, and M. P. Coutinho, "Protection of consumer data in the smart grid compliant with the german smart metering guideline," in *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, 2013, pp. 41–52.

[6] "MQ telemetry transport (MQTT)," http://mqtt.org/.

[7] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[8] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008.

[9] R. R. Stewart and Q. Xie, *Stream Control Transmission Protocol (SCTP)*. Addison-Wesley Professional, 2001.

[10] "Toyota city low-carbon society verification project (Smart Melit)," http://jscp.nepc.or.jp/article/jscp/20140403/391255/ (in Japanese).