Smart Computing Applications Using BLE and Mobile Intercloud Technologies

Yik Him Ho, Yerkezhan Sartayeva, Chak Pang Chiu and Henry C. B. Chan
Department of Computing
The Hong Kong Polytechnic University
Hong Kong

***Abstract* — With the advent of mobile computing and mobile cloud computing, many smart computing applications can be developed. In this paper, we present three smart computing applications using BLE and mobile Intercloud technologies. The first application is a smart roll call application using BLE, enabling teachers to easily take attendance while at the same time ensuring privacy. The second application is an ad hoc positioning application with a new BLE positioning paradigm. In this case, a user can determine the positions of other users through an ad hoc network. Experiments were conducted to evaluate the accuracy of this positioning method. The third application is a mobile Intercloud application, which allows mobile terminals to transfer data through their associated clouds (i.e., using an Intercloud protocol for data transfer). Experimental results are presented to evaluate performance.**

Keywords—BLE, smart computing, mobile Intercloud

I. Introduction

In recent years, there has been considerable interest in mobile computing and mobile cloud computing. One increasingly popular technology trend has been the use of Bluetooth Low Energy (BLE) devices. BLE is currently one of the most commonly used wireless technologies for the Internet of Things (IoT) in particular, as BLE works well for ad hoc and short-range communication (i.e., even without the Internet). Furthermore, as most smartphones are equipped with BLE, it is well suited for supporting a wide range of mobile applications. For example, Collotta and Pau [1] employed BLE to predict energy consumption in homes, based on an artificial neural network. Tosi *et al.* [2] used BLE for human motion tracking purposes. BLE beacons can also be deployed to museums, where exhibit-related information is sent to visitors according to their location [3]. Similarly, in another location-based application using BLE, customized and location-based information can be sent to visitors in a smart shopping mall [4]. In addition, there are other studies/applications related to BLE positioning, typically using fixed beacons. For example, one study employed iBeacons to help visitors find places of interest at airports and monitor congested areas and facilities (e.g., availability of charging facilities) [5]. BLE positioning technology can also be used in smart homes, for example to control lights and lamps, based on the user's location [6].

In addition, mobile cloud computing is becoming increasingly popular [7]. Apart from various offloading applications (e.g., [8]), mobile cloud data management is another popular application (e.g., [9]). In relation to mobile cloud computing, mobile Intercloud provides a new way for mobile terminals to interact with one another through their associated clouds [10]. This can be viewed as an integration between the physical domain and cyber domain as well. For example, as managed by mobile terminals in the physical domain, data and virtual machines can be transferred among clouds in the cyber domain. This enables the development of various smart computing applications (e.g., for mobile data management).

To contribute to the development of smart computing applications, we present three applications with analysis/evaluation through the use of BLE and Intercloud technologies. The remainder of this paper is outlined as follows. Section II presents the smart roll call application. Section III presents the ad hoc positioning application. Section IV presents the mobile Intercloud data transfer application. Finally, Section V provides the conclusion.

II. Application 1: Smart Roll Call Application

A. Overview

The first smart computing application is a smart roll call application using BLE broadcasting and scanning capability. Traditionally, teachers use a pen and paper to record student attendance records. With the popularity of smartphones/tablets/notebook computers in the education sector, attendance taking has taken on a more digital form, but teachers continue to manually record attendance on their devices. This mode of attendance taking is inefficient, especially if the number of students is large. Some schools have begun using smart card readers and have asked students to record their attendance using their student ID card. However, this is not very efficient, as it requires the installation of additional hardware (i.e., a card reader and a network). Furthermore, such an attendance-taking mode implies long queues when the number of students is large.

The proposed smart computing application uses BLE scanning and broadcasting capability to communicate, and does not require an Internet connection when taking attendance. Compared to other BLE-based positioning applications, this application does not require the use of BLE beacons to locate students. Instead, teachers' and students' phones will scan and broadcast BLE packets for communication purposes (i.e., without the need to set up connections).

There are two types of devices, namely the scanner (i.e. teacher) and the broadcaster (i.e., student) based on each person's login information. Fig. 1 shows an overview of the roll call scenario. Note that before attendance taking can start, the teacher must import the list of students to the app from the

Internet or locally (this only needs to be done once). The app will hash student IDs with the SHA1 algorithm, and student IDs coming in from advertisers will be compared to database records.

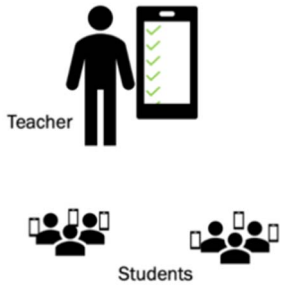

Fig. 1. An overview of the smart roll call application

Fig. 2 and Fig. 3 demonstrate the attendance-taking process in detail. First, the teacher opens the app and starts scanning nearby BLE devices. A BLE service with a customized BLE service UUID is included, so that the phone ignores other unrelated BLE devices. Next, students open the app, enter their student ID and start broadcasting. Note that for privacy, student IDs are hashed in the advertising packets. At the same time, a student's phone also starts BLE scanning. When the teacher's phone receives a packet, it extracts a hashed student ID and matches it against database records imported beforehand. If there is a match, attendance is recorded with the current timestamp. Then the teacher's phone broadcasts acknowledgment packets to notify students that their records were saved. Finally, students can stop broadcasting and close the app. Thus, attendance taking can be conducted effectively and efficiently.

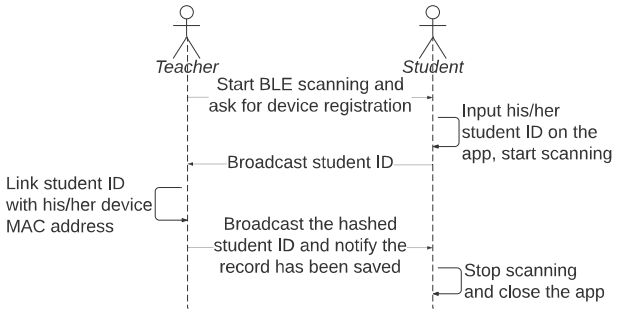

Fig. 2. Flow chart of first-time registration

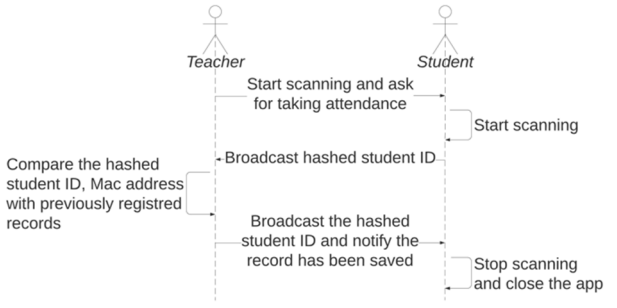

Fig. 3. Flow chart of taking attendance

Exposing raw student IDs and MAC addresses poses a major privacy concern. We address this issue by hashing student IDs on the broadcaster's side. Hashing is not computationally expensive and provides an effective solution for the privacy problem. While implementing a student ID hashing, we were constrained by the limits on advertising packet size, which is 24 bytes. Defined by the Bluetooth standard, the available size of Bluetooth advertising data is 31 bytes. A single packet can hold multiple pieces of advertising data (AD). The first AD (namely AD_1) is reserved as the 'Flag' AD, which is used to indicate its connection capability to other devices, for example, 'BR/EDR Not Supported', 'LE Limited Discoverable Mode', etc. As the Flag AD data is 3 bytes long, the remaining space constitutes 28 bytes. In order to broadcast customized data, 'Manufacturer-Specific Data' is used. As 4 bytes are reserved for AD length (1 byte), AD type (1 byte), and manufacturer ID (2 bytes), the available size for our custom data is 24 bytes (i.e., 31 – 3 – 4 bytes). Note that as student IDs are not as sensitive as passwords, a basic hashing algorithm such as SHA1 should be adequate, due to the limited data fields available.

III. Application 2: Ad hoc positioning

A. Overview

In a traditional positioning system, anchor nodes (namely reference nodes or broadcasters) are typically pre-deployed at fixed positions. The target nodes (namely scanners) scan signals from the anchors to estimate the positions. For example, in a BLE-based indoor positioning application, BLE beacons are installed at different locations. Users' mobile phones then scan for the BLE signals, and estimate the proximity and/or physical distance from the beacons based on the RSSI. Fig. 4 shows an example of this positioning architecture.

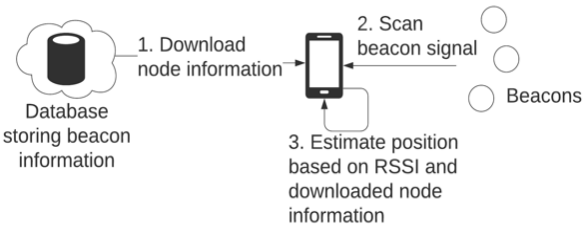

Fig. 4. Architecture of a typical BLE-based positioning system (e.g., iBeacon)

Here we consider a different positioning paradigm called ad hoc positioning, in which no beacons are used. The scanner can move to different positions (at least three points) and scan for signals from the broadcasters. This operation simulates the pre-deployed beacons. Also, the target node becomes a broadcaster instead of a scanner. By estimating the physical distance between the target node and scanner positions, the scanner can estimate the relative position of the targets. Fig. 5 shows an overview of the ad hoc positioning method.

Assuming that the targets are relatively stationary, the scanner can estimate the distance to each target. Basically, the scanner scans the BLE signals emitted by a target and hence the RSSI within a certain time period. As the signals are likely affected by interference and noise, a Kalman filter can be used for filtering purposes. Based on the RSSI measurements, distances can be estimated and finally the positions can be estimated (see the following subsections).

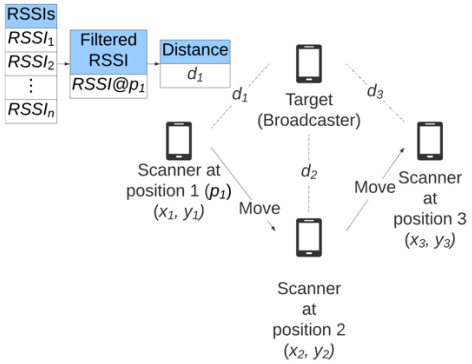

Fig. 5. Overview of the ad hoc positioning method

B. Distance estimation

A received signal strength indication (RSSI) in BLE transmissions is commonly used to estimate the distance between two nodes [11]. However, RSSI measurements are unstable and change constantly due to environmental factors, causing noise in the incoming data. Therefore, obtaining a specific distance is a challenge. Another inconvenience is that at least three beacons with known positions are required to estimate the position of a node [12]. As was mentioned previously, to address this issue, we propose a new distance estimation methodology using just two smartphones, i.e., with no need for beacons. To estimate distance, we use a formula based on the logarithmic attenuation model [13] as below:

$$[P_r(d)]_{dBm} = [P_r(d_0)]_{dBm} - 10nlog\left(\frac{d}{d_0}\right) + X \qquad (1)$$

$[P_r(d)]_{dBm}$ refers to the RSSI of the receiver and d is the distance between the receiver and the sender. $[P_r(d_0)]_{dBm}$ is the reference RSSI, where the reference distance is d_0. n is the attenuation factor whose value is contingent on the environment and the devices used. X is a Gaussian random variable with zero mean. The following formula can be derived from (1) to obtain the RSSI-to-distance formula:

$$d = 10^{\frac{RSSI_{d_1} - RSSI + X}{10n}} \qquad (2)$$

where d is the estimated distance, $RSSI_{d_1}$ is the sampled RSSI at 1m and $RSSI$ is the measured RSSI.

In our experiment, we used two Android phones, one as the sender and the other as the receiver, to collect measurements from 1, 2, 3, 4 and 5 meters for exactly one minute. Measurements were collected in an open space with minimal obstructions. This was done to estimate the attenuation factor n. A constant transmission power of -60 dBm was used. Here is a graph showing raw RSSI values for each distance.

Based on this data, two approaches were taken in developing a formula for RSSI-based distance estimation, which will be discussed next. To address the noise issue, the Kalman filter was used in both approaches.

1) Approach 1: Distance Estimation Based on the Most Frequent RSSI and the Logarithmic Attenuation Model

The first approach is to take the most frequent RSSI value for each distance, use the logarithmic attenuation model and find an attenuation factor n that would approximate this function best. Here is a graph illustrating the real RSSI versus distance curve for 1, 2, 3, 4 and 5 meters along with corresponding estimated distances based on (2).

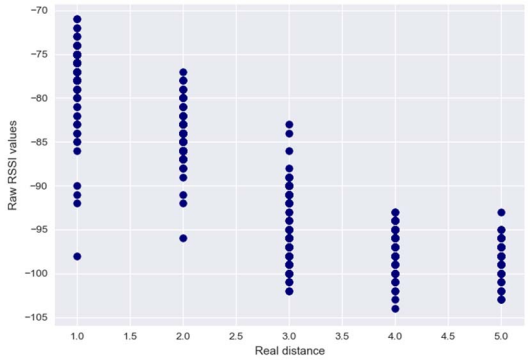

Fig. 6. Raw RSSI measured at different distances

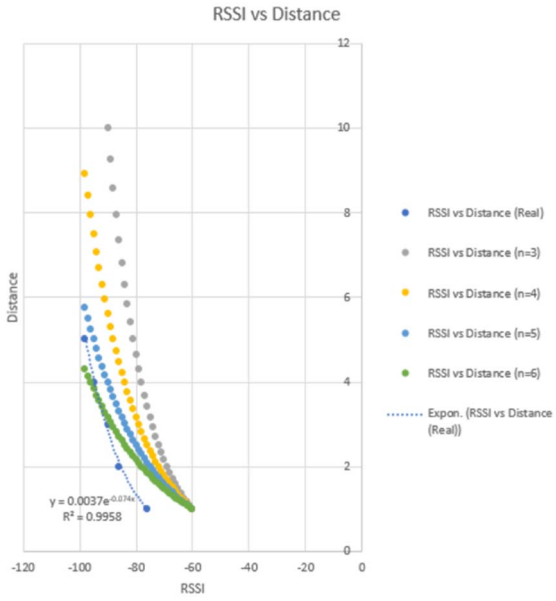

Fig. 7. Relationship between RSSI vs. estimated distance with different values of n

The curve with real measurements did not fit the general growth pattern of the RSSI-to-distance formula, so a Python script was written to find n to minimize the mean error. With a step of 0.01 in each iteration, the optimal attenuation factor was found to be 5.81, with an average error of 0.49m. Then, the cumulative distance estimation error was calculated for different distances, and histograms for 1 and 5 meters are illustrated here.

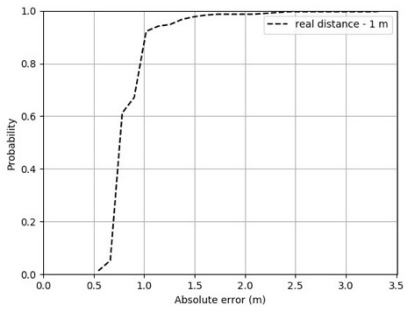

Fig. 8. Cumulative distance estimation error when the real distance is 1 meter

It can be observed that in approximately 90% of cases, the error is lower than 1.5m, but the figure goes up as the distance between receiver and sender rises.

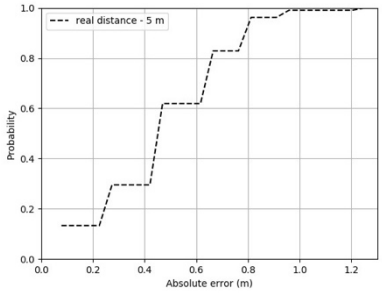

Fig. 9. Cumulative distance estimation error when the real distance is 5 meters

2) Approach 2: Distance Estimation Based on Mean RSSI and Linear Regression

To explore the possibility of further reducing the error rate, another approach based on linear regression was tested. First, a mean RSSI of every five measurements was taken, and the negative logs of all means (i.e., -log($|\sum_{i=1}^{5} RSSI_i|$)) along with the logs of real distance values, were plotted as explained below:

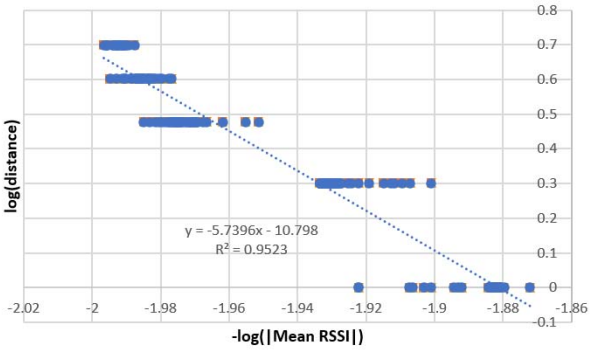

Fig. 10. Finding the parameters using linear regression

A line of best fit was found with $R^2 = 0.9523$, whose equation is presented below:

$$\log(distance) = -5.7396(-\log(|mean\ RSSI|)) - 10.798 \quad (3)$$

This can be further reduced to the following RSSI-to-distance formula:

$$distance = \frac{|RSSI|^{5.7396}}{10^{11}} \quad (4)$$

The average error of this approach was found to be 0.33m, an improvement compared to the first approach. Thus, the linear regression formula was used in the experimental part, which is discussed next.

C. Position estimation

After estimating the distance between the targets and the known points (i.e., scanner's positions), trilateration can be used to calculate the position. Given n known position $P = p_1, p_2, \dots, p_n$ and the coordinate (x_n, y_n) and the distance d_n, the position of the target (x_t, y_t) can be found by solving the following equations:

$$(x_t - x_{p_1})^2 + (y_t - y_{p_1})^2 = {d_1}^2$$
$$(x_t - x_{p_2})^2 + (y_t - y_{p_2})^2 = {d_2}^2$$
$$\vdots$$
$$(x_t - x_{p_n})^2 + (y_t - y_{p_n})^2 = {d_n}^2$$

However, because RSSI measurements are highly affected by noise and interference, distance estimations are subject to error. As a result, the above equations cannot be solved exactly. Fig. 11 illustrates two cases of inaccurate distance estimation (i.e., the circles do not intersect at one point). Instead, least square optimization can be used. One of the ways to achieve this is to start at a random point (e.g., the middle of three beacons), and iteratively find a point that gives the best approximation to the actual position based on the measured distance.

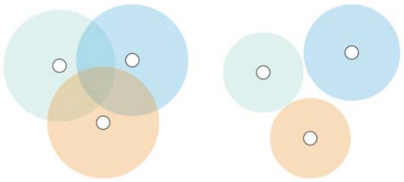

Fig. 11. Two cases of inaccurate distance estimation

D. Experiment setup

An Android application was developed to test the above method. Fig. 12 shows the screenshots of the scanner application. A user first scans and records the signals from the targets at a position, namely (0, 0). After receiving a large enough number of measurements, he/she will move to another position and update the coordinates (e.g., move one meter right to (1, 0)). In order to calculate the position, users should repeat these steps at least three times (e.g., move one meter forward from the first position to (0,1)).

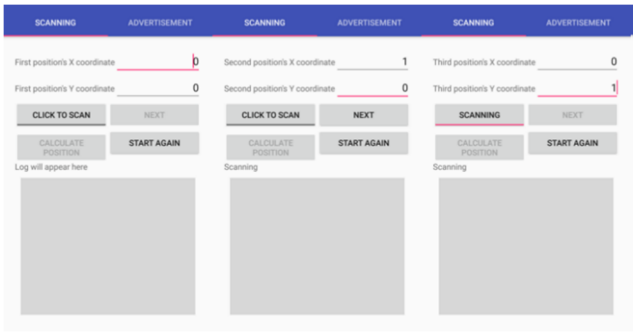

Fig. 12. Application screenshots when measuring target device signals

After measuring signals from at least three positions (i.e., simulating three different beacon positions), the application can calculate the estimated positions and show the results on the screen, as shown in Fig. 13. Note that the coordinate used in the application is a relative position defined by the users.

We have conducted experiments in different indoor environments. Several target devices were placed in the testing area, and a user carried a scanner to measure and calculate the position.

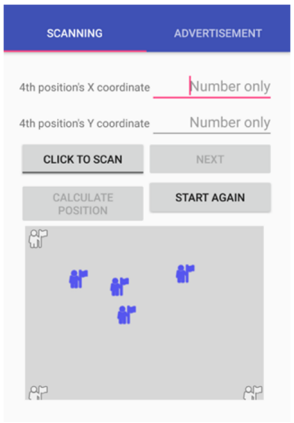

Fig. 13. Screenshot of estimated positions (Blue points: estimated positions for targets. White points: positions where the scanner measured RSSI)

E. *Experimental results*

Multiple sets of experiments were conducted to investigate the effect of different settings and parameters, including the scanning time used in each position, the number of positions of the scanner, and the number of target devices in the testing area. Specifically, we compared the positioning accuracy of different scanning times (3, 6 and 12 seconds) that the scanner used at each point. Also, we investigated the effect of the number of scanning points (3, 4 and 5 points) on position accuracy.

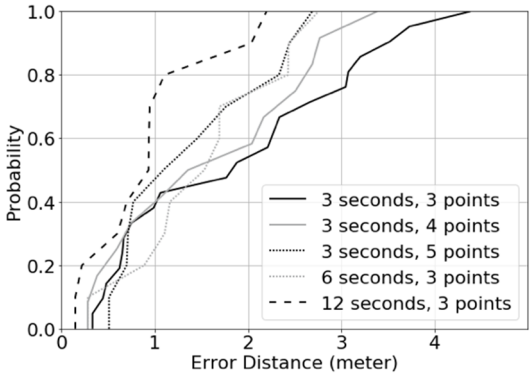

Fig. 14. Cumulative distribution function (CDF) of positioning errors

The cumulative distribution function of positioning errors is shown in Fig. 14. The lines in the graph show the cumulative distribution function of the positioning errors, where a line placed at left is better. For example, with the experimental setting of 12 seconds of scanning time and 3 scanning points, around 75% of the experimental points achieve an error of 1m or below, and that the maximum error of all experimental points is 2.1m or below. As shown in the figure, we can see that the length of scanning time and number of scanning points can help improve positioning accuracy. One of the reasons is that with a shorter scanning time, the scanner may not be able to receive signals from all of the targets, or the measurements are affected by interference and noise. With a longer scanning time and more measurements, the effect of noise can be minimized, particularly when using a filtering technique. Also, when the number of scanning points increases, it can improve accuracy or reduce extreme errors. For example, when one of the three scanning points suffers from interference, the error of the positioning estimation may be very large. If the scanner adds one more scanning point (i.e., moves to another location and scans again), the positioning result improves, as the fourth point can help correct the positioning result if it is not suffering from interference. In general, based on the experimental result, a scanning time of 3-6 seconds and 4-5 points is suggested for effective and efficient ad hoc positioning.

IV. Application 3: Mobile Intercloud Data Tranafer

A. *Overview*

The third application is a mobile Intercloud data transfer application. We envision that a mobile terminal is associated with a private or public cloud for data storage (e.g., object storage). Mobile terminals can send data objects to one another directly through their clouds (i.e., Intercloud data transfer as directed by the mobile terminals). In fact, we have developed a number of App Inventor blocks to support the development of mobile Intercloud applications [10]. As shown in Fig. 15, a mobile terminal first instructs its cloud/gateway to send a data object. The gateway then sends an ICCP PUT Request to the destined gateway(s). The data objects are also sent accordingly. The destined gateway(s) then upload(s) the data object to the corresponding cloud. Note that a private cloud (e.g., MinIO) can be co-located with the gateway. For public clouds, the gateway needs to upload the data object to the corresponding cloud over the Internet (i.e., data transfer over the Internet is required). Finally, each destined gateway sends an ICCP Respond to notify the successful uploading.

We have conducted experiments to evaluate mobile Intercloud data transfer (see Fig. 15). A mobile phone is associated with a private cloud in Japan (Japan (MinIO)). Data are sent to the associated clouds of four mobile terminals: one private cloud in Japan (Japan (MinIO)), one public cloud in Japan (Japan (AWS)), one private cloud in the U.S. (US (MinIO) and one public cloud in the U.S. (US (AWS)) through the Intercloud gateways.

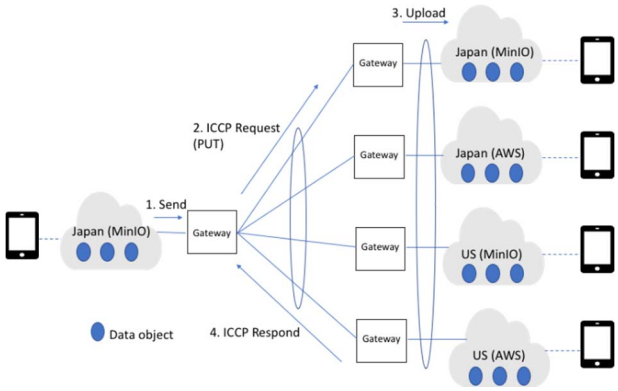

Fig. 15. Experiment setup overview

Fig. 16 shows that processing time increases linearly with file/data size in general. The processing time for Japan (MinIO), Japan (AWS) and U.S. (MinIO) can be maintained at low levels. However, the processing time for U.S. (AWS) is much higher, as it involves more data transfer over the Internet. Note that for U.S. (MinIO), as a private cloud is involved, the gateway and the MinIO server can be co-located in the same cloud.

As mentioned above, each experiment was repeated 10 times to compute the average processing time. Fig. 17 shows that when the file/data size is small, the average processing time for all cases does not vary significantly. However, when

the file/data size is large, the average processing time for U.S. (AWS) varies more significantly, due to more data being transferred over the Internet, as shown in Fig. 18. This indicates that if the gateways and cloud servers are properly configured (e.g., with sufficient resources), processing time can be maintained at a reasonably low level. Furthermore, Internet transfer time exerts the most significant effect on average processing time.

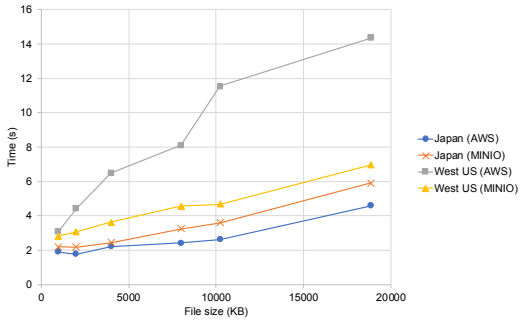

Fig. 16. Average processing time for different file sizes

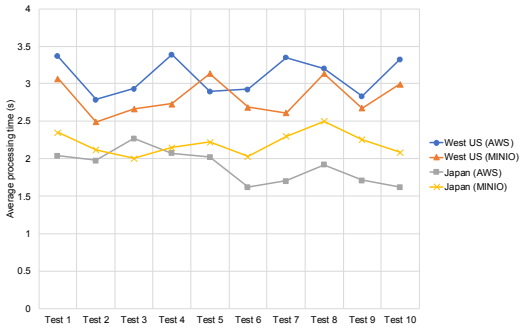

Fig. 17. Average processing time when the data/file size is 1,000KB

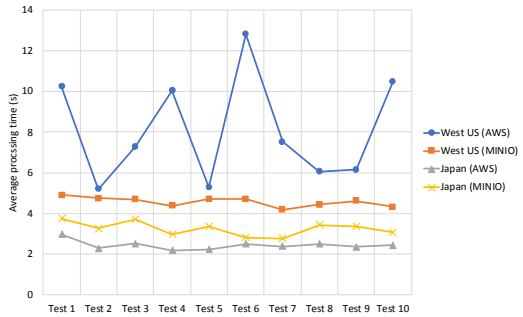

Fig. 18. Average processing time when the data/file size is 8,000KB

V. Conclusion

In this paper, we have presented three smart computing applications. The first is a smart roll call application based on BLE. The second is an ad hoc positioning application using BLE. Unlike other BLE-based positioning, it is a new positioning paradigm through a BLE-based ad hoc network. The third is a mobile Intercloud data transfer application. Data objects can be transferred through clouds associated with mobile terminals. The applications and results demonstrate the effectiveness of mobile computing through the use of BLE and Intercloud technologies.

Acknowledgments

This work is supported by The Hong Kong Polytechnic University.

References

[1] M. Collotta and G. Pau, "An innovative approach for forecasting of energy requirements to improve a smart home management system based on BLE," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, pp. 112-120, 2017.

[2] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino and D. Formica, "Throughput analysis of BLE sensor network for motion tracking of human movements," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 370-377, 2019.

[3] P. Spachos and K. N. Plataniotis, "BLE beacons for indoor positioning at an interactive IoT-based smart museum," *IEEE Systems Journal (Early Access)*, pp. 1-11, Feb. 2020.

[4] P. Shende, S. Mehendarge, S. Chougule, P. Kulkarni and U. Hatwar, "Innovative ideas to improve shopping mall experience over E-commerce websites using beacon technology and data mining algorithms," *Proc. 2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2017.

[5] B. Molina, E. Olivares, C.E. Palau and M. Esteve, "A multimodal fingerprint-based indoor positioning system for airports," *IEEE Access*, vol. 6, pp. 10092-10106, 2018.

[6] T. D. Schepper, A. Vanhulle and S. Latré, "Dynamic BLE-based fingerprinting for location-aware smart homes," *Proc. 2017 IEEE Symposium on Communications and Vehicular Technology (SCVT)*, 2017.

[7] D. Huang, T. Xing and H. Wu, "Mobile cloud computing service models: a user-centric approach," *IEEE Network*, vol. 27, no. 5, pp. 6–11, September 2013.

[8] S. Deng, L. Huang, J. Taheri, and A.Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, Dec 2015.

[9] Z. Jiang, Z. Zhenfeng and G. Hui, "Towards secure data distribution systems in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3222 - 3235, 2017.

[10] Y. Deng and H.C.B. Chan, "Development of mobile Intercloud applications," *Proc. IEEE COMPSAC 2019 – The 2nd IEEE International Workshop on Smart Computing & Applications.*

[11] J. Zhu, H. Luo, Z. Chen and Z. Li, "RSSI-based Bluetooth low energy indoor positioning," *Proc. International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, p. 526–533, 2014.

[12] J.-G. Lee, B.-K. Kim, S.-B. Jang, S.-H. Yeon and Y. Woong Ko, "Accuracy enhancement of RSSI-based distance estimation by applying Gaussian filter," *Indian Journal of Science and Technology*, vol. 9, no. 20, 2016.

[13] M. Hata, "Empirical formulae for propagation loss in land mobile radio service," *IEEE Transactions on Vehicular. Technology*, vol. 29, no. 3, pp. 317-325, 1980.