2009 International Conference on Advanced Information Networking and Applications

Name Resolution Middleware Using Relative Positional Relationship to Support Wireless Visible Area Communication

Atsushi Noda
Graduate School of
Information Science and
Electrical Engineering,
Kyushu University
744 Motooka Nishi-ku,
Fukuoka 819-0395, JAPAN
+81(92)802-3644
noda@f.csce.kyushu-u.ac.jp

Teruaki Kitasuka
Graduate School of
Science and Technology,
Kumamoto University
2-39-1 Kurokami,
kumamoto-shi, Kumamoto
860-8555, JAPAN,
+81(96)342-3898
kitasuka@cs.kumamoto-u.ac.jp

Shigeaki Tagashira,
Tsuneo Nakanishi and Akira Fukuda
Faculty of Information Science
and Electrical Engineering,
Kyushu University
744 Motooka Nishi-ku,
Fukuoka 819-0395, JAPAN
+81(92)802-3644
{shigeaki,tun,fukuda}@f.csce.kyushu-u.ac.jp

***Abstract*—In this paper, we propose name resolution middleware that involves positional awareness to identify target hosts in wireless visible area communication (WVAC). WVAC is a wireless communication network that enables users to exchange information with nearby hosts, especially within the users' visible area, wherein the underlying network can be locally and temporarily constructed without the aid of any centralized administration. In such communication, the proposed middleware assists the user to intuitively identify a target host by presenting the relative positional relationship among the neighbor hosts. The middleware possesses the following characteristics: (1) it provides simple APIs to be easily utilized by various applications, (2) a dedicated server preliminarily located in the network is not required to operate it, and (3) it adopts a positioning method that could estimate the relative locations of neighbor hosts without any pre-configuration. Finally, we construct a prototype system and evaluate the middleware using this system. The results indicate that this middleware can sufficiently work even on resource-limited mobile devices.**

***Index Terms*—Wireless Local Communication, Location-based Host Selection, Real-time Locating System (RTLS), Location Estimation.**

I. Introduction

Due to recent advancements in mobile communication technologies, many consumers enjoy various network services through wireless communication devices. In particular, a spread of infrastructure-less and short-range communication devices, such as Bluetooth, ad hoc mode in IEEE 802.11, and so on, has rapidly increased the demand for wireless local communication, especially with nearby (visible) networked terminals or equipments; however, a complicated procedure is required to establish such local communication when conventional communication techniques are used. This motivates the study of a simple way to realize wireless communication covering a user's visible area, called wireless visible area communication (WVAC), which enables the users to readily exchange information with visible hosts.

In conventional computer networks, users are required to specify (or identify) a remote host by its ID to establish a connection. For example, fully qualified domain name (FQDN) is introduced as a name-based approach, wherein a specified name is resolved to its associated ID (e.g., an IP address) using DNS. This approach is designed in a user-friendly manner; i.e., the unique name of a host hierarchically consists of human-memorable names. As another approach to specify a remote host, a browsing mechanism is used as a list-based approach in a local area network. The browsing mechanism automatically generates a list of hosts' names sorted in a lexicographic order and then the target can be identified by the user's selection from the list without typing the name. The name- and list-based approaches work effectively in conventional computer networks; however, these approaches are inadequate to support WVAC, since a user is required to memorize the exact name of a communication partner or look it up in a list lacking positional awareness, although the user can obviously view the partner. In particular, these issues become critical if the communication with a partner is rarely established and a large number of neighbor hosts exists in the network.

In this paper, we propose name resolution middleware that involves positional awareness to identify target hosts to support WVAC. The proposed middleware assists a user to intuitively identify a target host by presenting relative positional relationships among the neighbor hosts. The middleware possesses the following characteristics: (1) it provides simple APIs to be easily utilized by various applications, (2) a dedicated server preliminarily located in the network is not required to operate it, and (3) it adopts a positioning method that can estimate the relative locations of the neighbor hosts without any pre-configuration. More specifically, we enhance the positioning method, which has been proposed by wireless LAN indoor positioning system (WiPS) [3], [4], to estimate the relative locations in WVAC. Moreover, we evaluate the effectiveness

1550-445X/09 $25.00 © 2009 IEEE
DOI 10.1109/AINA.2009.55

of the prototype system by developing an example application providing a graphical user interface to display the relative locations, and examine the impact of errors for the distance measurement of wireless LAN devices. The results indicate that this middleware can sufficiently work even on resource-limited mobile devices.

To better understand our proposed middleware and the results, the remainder of this paper is organized as follows. Section II provides an overview of WVAC and its host selection issue. Section III describes our proposed middleware. An evaluation of our proposed middleware is given in Section IV . Section V investigates location-aware systems and positioning systems. Section VI concludes the paper and discusses future work.

II. WVAC

In this section, an overview of WVAC and its host selection method is described.

A. Overview

WVAC is a wireless network that enables users to exchange information with visible target hosts, wherein the underlying network can be locally and temporarily constructed without communication infrastructures. In other words, a pair of hosts can be directly connected to each other in WVAC. For WVAC, we can find many useful situations. For example, a user wants to control only the TV without performing any complicated operations. As another situation, when there are three printers in an office, the user would like to print a document from the nearest printer.

In a conventional computer network, users are required to follow a unified procedure to communicate with a host, regardless of whether it involves remote or local communication. This requirement leads to complicated operations even in local communications. In particular, traditional ways to specify hosts lack the advantage of positional awareness in real-world environments; for the above useful situations, it is complicated to specify the TV and the nearest printer, although the user can obviously view them. In this paper, we consider well-suited host selection for the WVAC environment.

B. Host Selection

Host selection is the user action in which a target host is selected through the user interface of the user's host. Here, we consider that the user would like to control the TV in front. A naive solution for selecting the TV is a name-based approach, i.e., the TV is selected by its associated name. However, the exact name should be known by the user and furthermore it is annoying to type the name. Another solution is a list-based approach for host selection. A list of automatically generated target hosts is displayed on the user's host. The user can select the target host from this list. However, the list is generally presented in a lexicographic order and therefore it is expensive to look up the name of the TV from such a list. Furthermore, the user also has to know the partial name for selecting the TV.

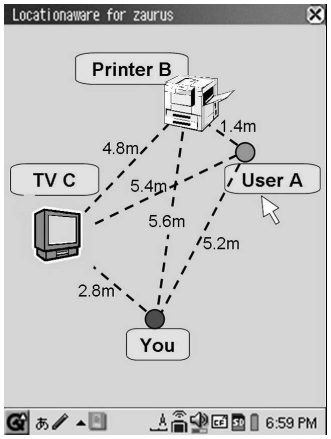


Fig. 1. Illustration for specifying a target host using relative location.

In this paper, we propose location-based host selection for WVAC. Figure 1 shows an illustration for location-based host selection. The user can select the target host more intuitively, i.e., the locations of the icons for neighbor hosts keep the relative positional relationship on the real world on the display. The user can easily find the target host on the display as compared to the actual user's view. Moreover, we introduce the notion of positional domain names in the location-based host selection. By using this domain name, users can easily select a group of hosts concerned with a specified domain, e.g., nearby domain or a specific domain. Although the proposed mechanism realizes a intuitive host selection, it requires additional calculation cost and management cost for the location estimation. Therefore, we carefully design a low-cost location sensing and management mechanism in the location-based host selection. We aim at providing location-based host selection as middleware so that various applications can utilize it easily.

III. Proposed Middleware

In this section, the proposed middleware for location-based host selection in WVAC is described.

A. Basic Design

The basic function of the proposed middleware is to provide translating/binding between relative locations and names, as well as that between names and identifiers. The requirements on designing the middleware could be summarized as follows:

- to prepare simple APIs that can be easily utilized by various applications,
- to run without a dedicated server preliminarily located in the network, and
- to adopt a positioning method that could estimate the relative locations without any pre-configuration.

After designing the middleware satisfying the above requirements, we carefully consider a particular positioning method for estimating the relative locations. Often, existing positioning methods require several reference points with predetermined

locations and pre-calibrations [1], [2]. Therefore, we focus on the method proposed by WiPS [3], [4] as a basic positioning method, because this method can accurately maintain the relative positional relationships, and estimate the relative locations in a short computation time without pre-calibration. However, several issues are involved when using this method in the WVAC environment. In the following section, we first explain the APIs and the software architecture of the middleware and then describe how to apply the WiPS method to our proposed middleware.

B. Application Interface

The APIs of the proposed middleware provide the translation between names and identifiers, as well as that between the relative locations and names. The APIs for the former translation follow traditional APIs in the C language, i.e., *gethostbyname* and *gethostbyaddr*:

WVAC_gethostbyname:
: *WVAC_gethostbyname* returns the *hostent* structure for a given name. The returned structure contains the name and identifier of the host.

WVAC_gethostbyaddr:
: *WVAC_gethostbyaddr* returns the *hostent* structure for a given identifier.

As for the latter translation, we introduce the notion of relative domains to translate the relative locations into their corresponding names. More specifically, a relative domain is specified instead of a relative location and the names of hosts that exist in the specified relative domain are returned. On the other hand, the APIs for translating from the names to the relative locations simply return the relative location that corresponds to the specified name. The relative domains are defined as follows:

ALL:
: This domain represents the area that is within the wireless communication range.

NEARBY domain:
: This domain represents the area that is within $r(>0)$ [m] around the user's host.

SPECIFIC domain:
: This domain represents the area that is within $r(>0)$ [m] centering the point (x, y).

PINPOINT domain:
: This domain represents the host existing at the point (x, y). If there is no host at this point, the nearest host is returned.

The APIs for the translation between the relative locations and names are described as follows:

WVAC_gethostlocbydomain:
: *WVAC_gethostlocbydomain* returns the *hostloc* structure for hosts that exist in a specified domain. The returned *hostloc* structure contains the name and relative location of each host.

WVAC_gethostlocbyname:
: *WVAC_gethostlocbyname* returns the *hostloc* structure that contains the relative position for a specified host name.

WVAC_updatelocation:
: In our proposed middleware, each application uses the relative location information stored in its own buffer. *WVAC_updatelocation* updates the buffer to the latest one.

WVAC_rotateLocation, WVAC_mreverseLocation:
: Details of these APIs are explained in section Ⅲ-D3.

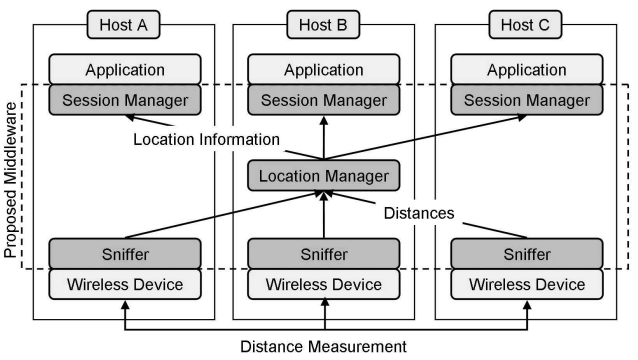


Fig. 2. Software architecture of the proposed middleware.

C. Software Architecture

The proposed middleware consists of three subsystems, namely, *sniffer*, *location manager*, and *session manager*, as shown in Figure 2. The middleware runs on all of the hosts of users and all of the target hosts. The proposed host selection system consists of all the hosts that run this middleware. A sniffer subsystem and session manager subsystem should be run on each host participating in the system and the location manager subsystem should be operated on a selected host. The selection scheme of the location manager will be described later.

Each sniffer subsystem is responsible for measuring the distances to the neighbor hosts. It sends the pair list of measured distances to the neighbors and their identifiers (e.g., IP addresses) to the location manager. The sniffer subsystem does not depend on a specific wireless communication device; any device that has the capability of distance measurement can be accepted by the sniffer subsystem. For example, wireless LAN devices compliant with the IEEE 802.11 MAC, which is one of the most popular wireless communication devices and normally installed in notebook PCs and PDAs, can realize the distance measurement based on the received signal strength indicator (RSSI). Note that RSSI attenuates in proportion to the square of the distance between sender and receiver. Moreover, the reliability of the distance measurement is low due to reflection, attenuation, multipath, and so on, i.e., the estimated distance includes error of an order of several meters.

The location manager subsystem aggregates the measured distances from the sniffer subsystems of all the hosts. By

using the aggregated distances, the location manager calculates the relative locations of all the hosts. The location manager informs the calculated relative locations to each session manager. The relative locations are expressed by two-dimensional coordinates for each host.

Each session manager subsystem manages the received relative locations centering on the host operating the session manager. It answers the requests from applications, i.e., the requested relative location is resolved to the corresponding identifier. The middleware provides a library that has an application programming interface to the session manager. Application engineers can simply develop the location-based application by using the session manager library.

D. WiPS-based Positioning Method

The location manager subsystem estimates the relative locations of all the hosts from the aggregated distance information. In this subsection, we describe an overview of the positioning method adopted by WiPS and enhance the positioning method for our proposed middleware.

1) Overview of WiPS: WiPS [3], [4] is a positioning system that can estimate the absolute locations based on the relative ones using several reference points. The positioning method determines the relative locations, while minimizing the cumulative errors between the estimated distances and the measured ones of all the pair of hosts by using the steepest descent method. The major features of the method are as follows: (1) the accuracy of the location estimation becomes higher as the density of hosts increases and (2) the computation time for location estimation is relatively shorter than that using other existing methods, e.g., a least-squares method. However, the method may yield not an optimum solution but a local optimum one. The basic process of the location calculation is described below.

1) Aggregate the list of distances of each pair of hosts.
2) Determine the initial position of each host.
3) Iterate the modification of the positions of hosts, until convergence.
4) Notify the location to each host.

Note that the initial position is the key to improve the computation time of the estimation due to fast convergence. To reduce the computation time, the result of a previous convergence is used as the initial position for the next cycle.

There are two difficulties to use this method in WVAC. First, the above process is operated on a dedicated server. To operate without a dedicated host, we introduce a selection mechanism of the host, instead of a dedicated server, as the location manager for the middleware. Second, user orientation cannot be determined using this method. However, the orientation is essential for estimating the relative locations. To overcome this problem, we include a mechanism to correct the orientation of the relative locations.

2) Selection of Location Manager and Location Estimation: A location manager is responsible for calculating the relative locations described in Section Ⅲ-D1, and it is automatically selected among all the hosts participating in WVAC. The basic idea of selection is to make the host starting the middleware at the earliest time as the location manager. In cases where the manager becomes down or multiple managers start at the same time, this system cannot operate correctly. We will address these issues as interesting directions of the future work. The detailed procedure for location estimation is described as follows:

Step.1 Each host invokes the middleware.
Step.2 The middlewares broadcast a discover message for a specific port to find the location manager that has been already started.
Step.3 If the middleware gets no response, it starts as the location manager.
Step.4 Otherwise, the middleware registers its identifier and name to the found location manager and then receives a list of identifiers and names for the hosts, which have already been registered, from the location manager.

The process for location estimation is shown in Figure 3, and the procedure for location estimation is described as follows:

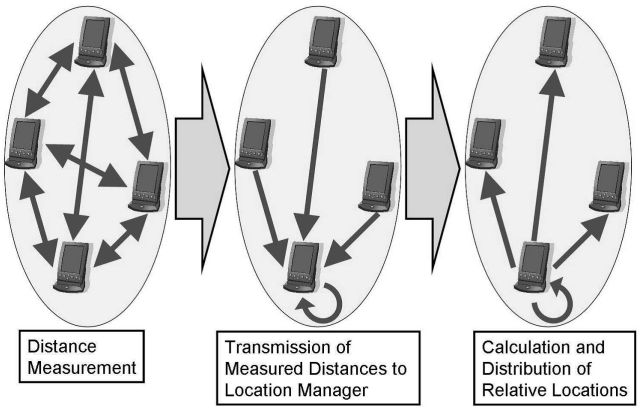
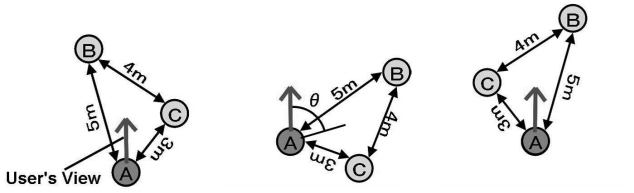


Fig. 3. Procedure for estimating relative locations.

Step.1 Each sniffer measures the distance to each host included in the received list.
Step.2 The sniffers send the measured distances to the location manager that runs on the bottom host, as shown in Figure 3.
Step.3 The location manager calculates the relative locations from the received distances and distributes the relative locations to the session manager running on each host.

3) Modification of Relative Locations: The location manager can calculate the relative locations of the participating hosts in an infrastructure-less environment. However, it is difficult for the session manager to provide orientation of the relative locations matching each user's view. Moreover, although the location manager collates the distance between any two hosts, it cannot acquire the direction information from each user, which induces a mirror-reversed error of relative



(a) Relative Locations

(b) Estimated Relative Locations of Rotational Error

(c) Estimated Relative Locations of Mirror Reversed Error

Fig. 4. Errors of estimated relative locations.

locations. To resolve these issues, the middleware provides two operations to modify the relative locations. In the following, we explain the details of these issues and their solutions.

Figure 4 (a) shows the relative locations for hosts A, B, and C in an actual environment, which is centered on host A. In this figure, the upper arrow represents the direction of a user's view. Figure 4 (b) shows the relative locations estimated in the same situation in Figure 4 (a). We observe the difference between the two figures; the estimated relative locations are rotated by an angle θ as compared to that on the actual view. Note that in this middleware, the initial direction of the user's view is set as a random direction. The main idea for resolving this issue is to provide a user interface to input a right angle of the user's view, and store this angle in the session manager. The API to deliver this angle θ to the session manager is the *WVAC_rotateLocation*, as described in section Ⅲ-B. More concretely, when the rotation is noticed by a user, the orientation is modified by a predetermined angle through the rotation operation until the orientation is matched to the actual one. The session manager records the total angle of rotation. Further, this will automatically provide corrected relative locations modified by the rotation for future requests.

Figure 4 (c) shows the mirror-reversed error of the relative locations. In this case, it is evident that the distances between the hosts are the same as that in Figure 4 (a) and the direction of the user's view is also matched to the actual one; however, the relative directions differ in the two figures. For resolving this issue, the proposed middleware provides the API *WVAC_mreverseLocation* that reverses the relative location. If a user notices the reversed error, the user employs the reverse operation using this application and *WVAC_mreverseLocation* is invoked. After this modification, the session manager will provide the corrected relative locations for future requests.

IV. Evaluation

In this section, we evaluate the performance and operability of the proposed middleware. In this evaluation, we first construct the prototype system of the middleware and implement an application example using the prototype system. Next, we evaluate the computational resources required for estimating the relative locations and examine the impact of errors for the distance measurement of wireless LAN devices.

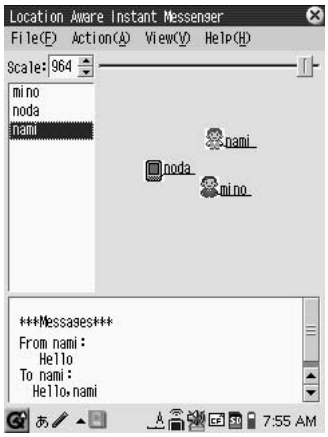


Fig. 5. Screenshot of location-aware instant messenger.

A. Location-aware Instant Messenger

We implement a location-aware instant messenger as an application example using the proposed middleware. Figure 5 shows the screenshot of the application. This application displays the relative locations provided by the middleware on the user's screen. The application has a naming mechanism that can share the handle names of users participating in the application group. The handle names are also displayed. The application supports the graphical operations for modification in terms of relative locations as described in Section Ⅲ-D3, i.e., the user can operate the rotation and mirror-reversing of the displayed location information by dragging the screen. The user selects a target by graphically selecting its corresponding icon on the screen. Internally, the relative position is acquired from the selected icon and then translated to its identifier. The user can input messages to a form prepared by the application, and the application sends the messages to the selected target.

We exhibit the pseudocode used in this application using APIs provided by the proposed middleware, as shown in Figure 6. This pseudocode shows only a part of this application.

B. Computational Resources

We evaluated the amount of memory required for operating the proposed middleware. In this evaluation, the location manager calculates the relative locations by varying the number of hosts, i.e., 5, 10, 20, and 40 hosts. We use SHARP Zaurus SL-C1000 as the location manager; it has Intel(R) XScale (TM) (PXA270 416 MHz) and 64 MB RAM. The other hosts are generated by evaluation software and randomly placed in the $100m \times 100m$ field. They are located within the wireless communication range of each other. The result excludes the amount of memory required for the evaluation software. In this evaluation, the distance measurement using wireless communication devices contains 20% error in a normal distribution. We measure the amount of memory for 10 s during the estimation and the result shows the average value.

Table I shows the result. First of all, we observe that the memory usage does not increase too much as compared to an

```
if ( WVAC_updatelocation() ) { /* updates relative locations */
  /* gets names of neighbor hosts with WVAC_getnamesbydomain */
  for ( repeats for number of surrounding hosts ) {
    /* gets relative locations with WVAC_getlocbyname */
    /* displays hosts with names, while keeping relative positional relation */
  }
}
```

Fig. 6. Pseudocode for messenger software using the proposed middleware.

TABLE I
RESOURCES REQUIRED FOR CALCULATING RELATIVE LOCATIONS.

Num of Hosts	Memory Size [KB]
5	3700
10	3704
20	3716
40	3764

increase in the number of hosts. The increase is only about 64 KB and the total amount is about 3.7 MB. This is sufficiently small as compared to the size of the memory equipped on today's standard mobile terminals.

We calculate the amount of communication required for this middleware. The size of one packet of distance information is $(n-1)\times$ 8 B, where n denotes the number of hosts. In this prototype system, each sniffer sends the distance data to the manager every second. When the number of hosts is 10, the amount of communication between the sniffer and location manager becomes $(n-1)\times$ 8 $\times n$ = 9 $\times$ 8 $\times$ 10 = 720 B/s. The size of one packet for the location information delivered to each session manager is $n\times$ 76 B. When the number of hosts is 10 and the location information is delivered every second, the amount of communication between the location manager and session manager is $n\times$ 76 $\times n$ = 10 $\times$ 76 $\times$ 10 = 7600 B/s. The total amount of communication for 10 hosts is 720 + 7600 = 8320 B/s. The amount of communication increases with the square of the number of hosts; for example, when the number of hosts is 40, the total amount of communication becomes about 135 KB/s plus the protocol overhead. By compressing the data packet, we can decrease the total amount of communication.

C. Computation Time

Next, through a evaluation, we determine the computation time required for estimating the relative locations. This evaluation runs on two types of computers: (1) a notebook PC as a high-performance computer and (2) a PDA as a low-performance computer (i.e., resource-limited device). The detailed specifications are listed in Table II.

The evaluation environment is the same as the previous one. The estimation is repeated 1000 times for different positions and errors for all the hosts. At each time, two trials are conducted. The first trial uses random values as the initial positions for all the hosts. The second one uses the result of the first trial as the initial positions and the measured distance containing another error. From the viewpoint of a running system, the first trial represents the immediate computation just after the location manager starts, and the second trial implies the continuous computation of the steady location manager. In this situation, we measure the average computation time and the worst one by varying the number of hosts.

TABLE II
TWO COMPUTERS USED IN THE EVALUATION.

(a) Notebook PC

Model	Panasonic Let's note CF-R4
CPU	Intel(R) Pentium(R) M 1.20GHz
RAM	1GB

(b) PDA

Model	SHARP Zaurus SL-C1000
CPU	Intel(R) XScale(TM) (PXA270 416MHz)
RAM	64MB

TABLE III
COMPUTATION TIME FOR ESTIMATION [MS]. PC1: FIRST TRIAL FOR NOTEBOOK PC. PC2: SECOND TRIAL FOR NOTEBOOK PC. PDA1: FIRST TRIAL FOR PDA. PDA2: SECOND TRIAL FOR PDA.

Num of hosts	PC1	PC2	PDA1	PDA2
5	0.43	0.35	147	110
10	2.5	1.8	755	433
20	15	9	2443	1440
40	98	53	-	-

Table III lists the results of this evaluation. From the table, we observe that the average computation time for the notebook PC is less than 100 ms in the entire range of the number of hosts. As for the PDA, the computation time can be decreased to less than 600 ms when the number of hosts is less than or equal to 10, whereas it is more than 2500 ms when the number of hosts becomes 20.

As described in Section Ⅲ-D, the proposed positioning method can reduce the computation time if the quality of the initial positions is improved. To clarify this effect, we present a comparison between the computation times for the first and second trials. Figure 8 and Table III show the results of this comparison. From the figure, we can see that the computation time can be improved especially when the number of hosts increases. In particular, the second trial for the PDA improves the time consumption by 40 % over the first trial when the number of hosts is 20; however, the computation time is over 1000 ms even during the second trial. The average computational time t is approximated as $t = an^{2.4}$, where a is 0.01 for PC, or 2.11 for PDA, and n is the number of hosts.

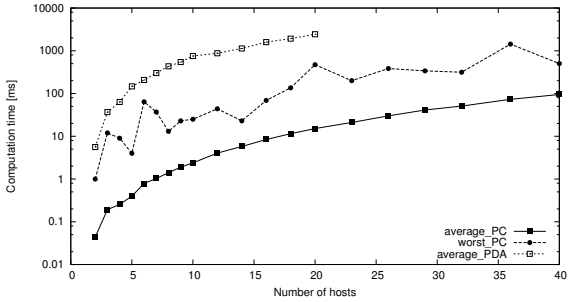


Fig. 7. Computation time of the positioning method.

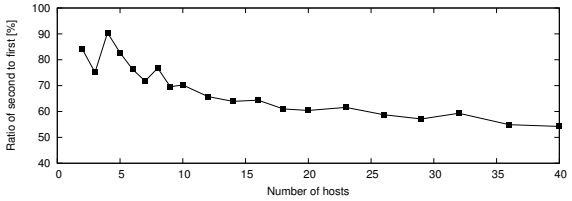


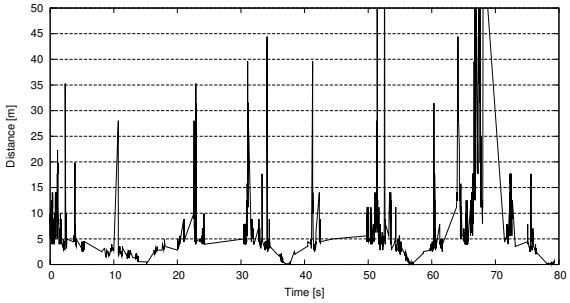
Fig. 8. Computation time ratio of second trial to first trial.

The above results indicate that the location estimation using resource-limited mobile devices is available for limited scenarios where high scalability or real-time solutions are not required. Otherwise, a notebook PC should be used as the location manager.

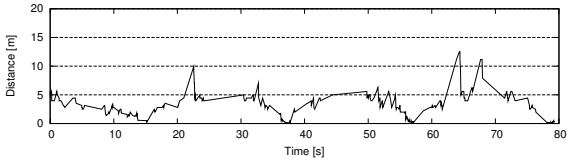
D. Effect of Noise Reduction Mechanism

The prototype system uses wireless LAN devices compliant with IEEE 802.11 as the distance measurement device. Recall that the distance measurement is based on the attenuation of RSSI acquired by the devices. The accuracy of the estimated distance is low, which leads to low system reliability. Therefore, the prototype system incorporates a filtering mechanism of the distance measurement into the sniffer subsystem to reduce the impact of errors. Here, we describe the filtering mechanism and evaluate the effectiveness of the mechanism through experiments.

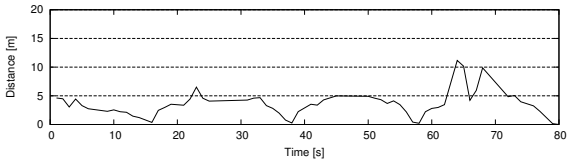
The main idea of the filtering mechanism is to cancel the sharp transition of the estimated distances as the measurement error. Figure 9 (a) shows the result of the measured distances using wireless LAN devices in an actual environment. In this figure, we plot the estimated distances while varying the actual distance between the two devices from 0 m to 5 m. From this figure, we observe that the curve of the estimated distances contains noise. Next, we apply the filtering mechanism to the sniffer subsystem. More concretely, we introduce a threshold for filtering the sharpness of the transition, i.e., if the difference between two successive distances is larger than the threshold, it is cut off. We assume that the average walking speed is less than 5 m/s; therefore, the threshold is set as 5 m/s in this experiment. Figure 9 (b) shows the result of the filtered distances. From this figure, we can see that the filtering



(a) Transition of raw distances.



(b) Transition of filtered distances.



(c) Transition of filtered and averaged distances.

Fig. 9. Effect of noise reduction for measured distances.

mechanism reduces noise in the estimated distances.

Further, the sniffer subsystem adopts the averaging of the estimated distances to significantly reduce the measurement error. Figure 9 (c) shows the result of the filtered and averaged distances. In this experiment, the interval for averaging is 1 s. From Figure 9 (c), the line of the estimated distance is smoother as compared to Figures 9 (a) and (b). We confirm that averaging can improve the stability of the system. Further, it can reduce the communication traffic because only the averaged information of the estimated distance is transferred to the location manager. The interval for averaging should be carefully adjusted according to the application requirement.

V. Related Work

In this section, we investigate location-aware host selection and positioning systems. First, we introduce the graphical user interfaces for sensor networks [5], [6]. These systems can display information from sensors onto a geographical map, such as Google Maps, Google Earth, or Virtual Earth. Geo-Coding [7] can reduce costs in terms of operations and management by simplifying the communication process from/to sensors through such interfaces. These approaches target pre-organized networks, i.e., locations of sensors are preliminarily configured manually and a dedicated server is used for managing sensors, which is different from the environment covered in this paper.

Second, several positioning systems have been proposed to autonomously estimate the locations of hosts in unorganized networks where the locations are not configured for the hosts.

Each host is equipped with a receiver device that is assigned an ID, and the location is estimated by using an appropriate positioning method. A host can be identified by its assigned ID and estimated location. These systems can be mainly classified into three types, i.e., ultrasound-based system, RF-based system, and their combinations [8] [9] [10]. Active Bat [8] and DOLPHIN [9] are ultrasound-based positioning systems. The main goal of these systems is to determine the precise locations of hosts in a specific environment wherein several ultrasound receivers are mounted in ceilings. This system was evaluated in an actual environment, which showed that the error of the estimated distance is from 3 cm to 20 cm. However, it is expensive to construct a specific environment.

As for RF-based systems, RADAR [11] and Bradio [12] have been proposed. These systems use wireless LAN devices to estimate the area (or room) where a target host is located. The estimation procedure consists of two phases: survey phase and estimation phase. In the survey phase, hosts measure the RSSI values to the access points in each target area and then the measured values associated with these areas are stored into a location database. In the estimation phase, the area is determined from the RSSI values measured by a host from the database. The survey phase needed by this approach is unsuitable for local and temporary networks.

Third, there are some studies for proximity sensing. NearMe wireless proximity server [13] provides a list of physically nearby hosts. The cost on estimating the proximity is lower than that on computing locations and in addition, several applications need proximity information rather than location information for nearby hosts. However, neighbor hosts existing in a similar distance cannot be distinguished by using only the proximity information, e.g., the hosts exists on the right side and left side. Location information is more intuitively for users to find the target host on the display as compared to the actual user's view. NearMe is unsuitable for local and temporary networks because the mechanism requires access points and central database.

VI. Conclusion

In this paper, we proposed middleware to realize efficient host selection for WVAC. The proposed middleware supports location-based host selection where a user can select a target host through its relative location in addition to its name. We described the middleware design and enhanced the method proposed by WiPS [3], [4] to estimate the relative locations for WVAC. We evaluated the middleware through several experiments using a prototype system in an actual environment. The result of the experiments suggests that this middleware can work even on resource-limited mobile devices under limited scenarios where the number of devices is less than 10. Moreover, we evaluated the effectiveness of the prototype system by developing a location-aware instant messenger, and examined the impact of errors for the distance measurement of wireless LAN devices.

In the future, we will endeavor to incorporate support for secure communications. For practical applications, we must consider secure communications in WVAC. However, it is difficult to safely authenticate public keys in such environments. Furthermore, we will address the determination problem of the user's view. The current system requires users to input the angle. It makes the automatic mechanism more useful in live environments. It is also our future work to deal with cases that the manager becomes down or multiple managers are established at the same time. The evaluation of this paper is conducted through simulation in an ideal environment. We will evaluate our middleware in more realistic simulation environment and also compare our results with other locating techniques rather than WiPS.

Acknowledgment

This work was partially supported by "The Kyushu University Research Superstar Program (SSP)", based on the budget of Kyushu University allocated under President's initiative and supported by Grant-in-Aid for Scientific Research.

References

[1] J. Hightower and G. Borriello, "A Survey and Taxonomy of Location Systems for Ubiquitous Computing," *Technical Report UW-CSE 01-08-03*, 2001.

[2] N. Patwari, J. N. Ash, S. Kyperountas, A. O. HeroIII, R. L. Moses and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, Vol.22, Issue.4, pp. 54–69, 2005.

[3] T. Kitasuka, K. Hisazumi, T. Nakanishi and A. Fukuda, "Positioning Technique of Wireless LAN Terminals Using RSSI between Terminals," *Proc. the 2005 Int. Conf. on Pervasive Systems and Computing (PSC-05)*, pp. 47–53, 2005.

[4] T. Kitasuka, K. Hisazumi, T. Nakanishi and A. Fukuda, "WiPS: Location and Motion Sensing Technique of IEEE 802.11 Devices," *Proc. Third Int. Conf. on Information Technology and Applications (ICITA'2005)*, pp. Vol.II, 346–349, 2005.

[5] G. Werner-Allen, P. Swieskowski and M. Welsh, "MoteLab: a wireless sensor network testbed," *Proc. the Fourth Int. Conf. on Information Processing in Sensor Networks*, pp. 483–488, 2005.

[6] S. Andre, N. Suman, L. Jie, P. Bodhi and Z. Feng, "SenseWeb: browsing the physical world in real time," Demo Abstract, *Proc. the Fifth Int. Conf. on Information. Processing in Sensor Networks*, 2006.

[7] C. Decker, T. Riedel, P. Scholl, A. Krohn and M. Beigl, "Graphically Geo-Coding of Sensor System Information," *Proc. Fourth Int. Conf. on Networked Sensing Systems (INSS'07)*, pp. 138–141, 2007.

[8] A. Harter, A. Hopper, P. Steggles, A. Ward and P. Webster, "The Anatomy of a Context-Aware Application," *IEEE Computer*, Vol. 34, No. 8, pp. 50–56, 2001.

[9] M. Minami, Y. Fukuju, K. Hirasawa, S. Yokoyama, M. Mizumachi, H. Morikawa and T. Aoyama, "DOLPHIN: A Practical Approach for Implementing a Fully Distributed Indoor Ultrasonic Positioning System," *Proc. Int. Conf. Ubiquitous Computing (UbiComp) 2004*, LNCS 3205, pp. 347–365, 2004.

[10] P. Bellavista, A. Corradi and C. Giannelli, "Coupling Transparency and Visibility: a Translucent Middleware Approach for Positioning System Integration and Management (PoSIM)," *Proc. 3rd Int. Symp. on Wireless Communication Systems (ISWCS'06)*, pp. 179–184, 2006.

[11] P. Bahl and V. N. Padmanabhan "RADAR: An In-Building RF-based User Location and Tracking System," *Proc. Nineteenth Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM) 2000*, Vol. 2, pp. 775–784, 2000.

[12] B. Yoshimi, G. B. Bolam, N. Sukaviriya, J. Elliott, B. Carmeli, J. Morgan and H. Derby, "Bradio: a wireless infrastructure for pervasive computing environments," *Proc. 21st IEEE Int. Conf. on Performance, Computing, and Communications*, pp. 309–316, 2002.

[13] J. Krumm, and K. Hinckley, "The NearMe Wireless Proximity Server," *Proc. Int. Conf. on Ubiquitous Computing 2004*, pp. 283–300, 2004.