Distributed Emulator for Developing and Optimizing a Pedestrian Tracking System Using Active Tags

Junya NAKATA*‡[1] Razvan Beuran*‡[2] Tetsuya Kawakami†[3] Takashi Okada‡*[4] Ken-ichi Chinen‡*[5] Yasuo Tan‡*[6] Yoichi Shinoda‡*[7]
* Hokuriku Research Center, National Institute of Information and Communications Technology
2-12 Asahidai, Nomi, Ishikawa Japan
[1]jnakata@nict.go.jp [2]razvan@nict.go.jp
† Panasonic System Solutions Company, Panasonic Corporation.
4-3-1 Tsunashima-higashi, Kohoku, Yokohama, Kanagawa Japan
[3]kawakami.tetsu@jp.panasonic.com
‡ Internet Research Center, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa Japan
[4]tk-okada@jaist.ac.jp [5]k-chinen@jaist.ac.jp [6]ytan@jaist.ac.jp [7]shinoda@jaist.ac.jp

***Abstract*—In this paper we introduce a distributed emulator for a pedestrian tracking system using active tags that is currently being developed by the authors. The emulator works on StarBED, which is a network testbed consisting of hundreds of PCs connected to each other by Ethernet. The three major components of the emulator (the processor emulator of the active tag micro-controller, RUNE, and QOMET) are all implemented on StarBED. We present the structure of the emulator, how it functions, and the results from the emulation of the pedestrian tracking system. The emulator accomplished quite accurate emulation of ubiquitous network systems with the technique of emulation. We found several issues originated from active tag's firmware or protocol by applying the emulator to the emulation of the tracking system. We confirmed the results obtained by running tests corresponding to a real-world experiment.**

***Keywords*—ubiquitous networks; distributed testbed; supporting software**

I. INTRODUCTION

As Panasonic Corporation. (hereafter referred to as Panasonic) is developing a pedestrian tracking system using active tags, one requirement is to carry out a large number of trials. Real-world experiments with wireless network systems, and active tags in particular, are difficult to perform when the number of nodes involved is larger than a few devices. Problems such as battery life or undesired interferences often influence experimental results. We are currently implementing a solution by developing an emulation system for active tag applications that runs the real active tag firmware within a virtual, emulated environment. Through emulation, much of the uncertainties and irregularities of large real-world experiments are placed under control. In the same time, using the real active tag firmware in experiments enables us to evaluate exactly the same program that will be deployed on the real active tags; this is a significant advantage compared to simulation. For performing the practical experiments we use StarBED, a network experiment testbed.

StarBED consists of 920 PCs connected by two separated networks, the management network and the experiment network, as shown in Figure 1. StarBED provides a simulation supporting software, SpringOS, to implement an easy-to-use simulation environment with which the users can write experiment scenarios in a specific scripting language that can later be executed automatically. In order to be able to use this testbed for active tag emulation we developed several additional subsystems, and integrated them with the existing testbed infrastructure [1]. These subsystems were developed on the basis of existing tools that are already used on StarBED, namely the wireless network emulator QOMET [2], and the experiment support software RUNE [3].

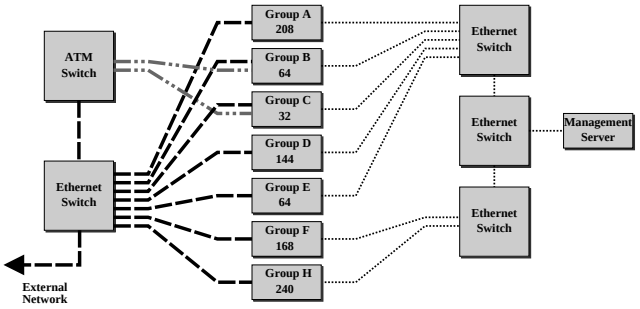

Fig. 1. Conceptual topology of StarBED.

Active tags were so far mainly studied through simulation, such as the work presented in [4]. Public domain wireless communication emulation research is currently mainly done in relation to Wireless LANs (WLANs). One can use real equipment, and hence be subject to potential undesired interferences. Two examples from this class that allow a controlled movement of wireless nodes are the dense-grid approach of ORBIT [5], or the more realistic robot-based

Mobile Emulab [6]. An alternative which avoids undesired interferences and side effects is to use computer models for real-time experiments. TWINE [7] is an example from this class. TWINE is a wireless emulator that combines wireless network emulation and simulation in one setup, but only supports 802.11b WLAN so far. Our development started from an existing wireless emulator, QOMET, which uses similar concepts.

There are already a number of implementations of experiment tools for ubiquitous systems that could be used in conjunction with active tag devices. Some of these tools focus on the operating system level, such as TOSSIM [8], which is a TinyOS simulator aiming to simulate TinyOS applications accurately in a virtual environment. ATEMU [9] is able to emulate TinyOS applications at processor level; its flexible architecture has support for other platforms too. ATEMU is thus closer to our purpose, since our low-cost active tags do not use any operating system. We aimed to run in emulation experiments the same firmware with the one used by the real devices. The manufacturer of the active tag processor , Microchip, only provides two alternatives for system development: real-time emulation in hardware using either the MPLAB REAL ICE In- Circuit Emulator, or the PICMASTER Emulator, or processor simulation using the MPLAB-SIM Simulator [10]. However none of these solutions are appropriate for our purpose; thus we developed our own real-time processor emulator running on PCs.

The pedestrian tracking system developed by Panasonic. makes use of active tags so as to provide to a central pedestrian localization engine the information needed to automatically calculate the trajectory to date and the current position of the active tag wearer. In the prototype system, three kinds of tags are used as shown in Figure 2. Mobile tags held by pedestrians transmit periodically ID packets which includes time information and sender's node ID. Fixed tags also transmit ID packets at certain intervals. Gateway tags have, in addition, a wired network connection to backend servers in which the data uploaded from mobile tags are sent. The trajectories of the pedestrians are calculated based on the data gathered by the gateway tags. Using the prototype of the pedestrian localization system, real-world experiments were carried out in March 2007. The experiment consisted in the orchestrated movement of 16 pedestrians both in indoor and outdoor environments. A system overview and experimental conditions will be presented later in this paper.

One of the important conclusions of the experiment was that it is very difficult to organize a real-world experiment for such applications of active tags. The number of people involved, and the accuracy of their movement following the predefined scenario, are only a few of the issues encountered. Nevertheless, the results of the above-mentioned experiment are currently being used as a basis for improving the prototype of the pedestrian localization system and extending it for use with very large groups of people, of the order of one thousand. The active tag emulation system that we designed and implemented plays an essential role at this point, since it makes it possible to continue the experiments in the development phase with ease and in a wide range of controllable conditions.

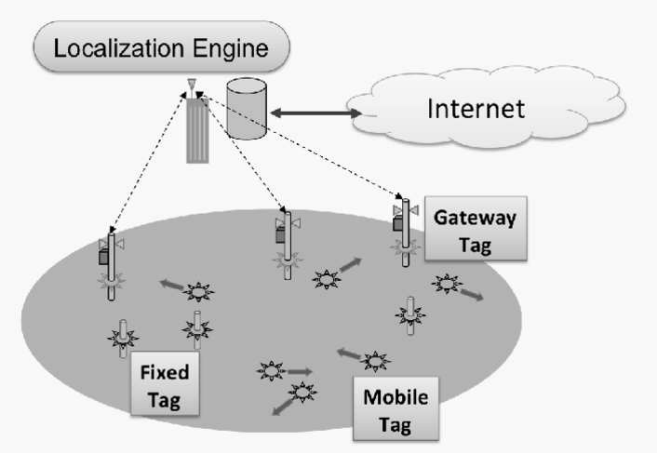

Fig. 2. Overview of the pedestrian tracking system.

The paper is organized as follows. Section II introduces our general approach to active tag emulation. This is followed by Sections III and IV, which describe the main components of the active tag emulation system: the wireless communication emulation, and the active tag processor emulation subsystems, respectively. Section V presents the preliminary real-world experiments carried out in order to validate the pedestrian localization system prototype. Section VI is dedicated to presenting experimental results obtained with the emulation system in the attempt to reproduce and extend the preliminary real-world experiments. The paper ends with sections on conclusions and references.

II. System Description

The technique of emulation implies creating a virtual environment in which the movement, the communication, and the behavior of active tags are all reproduced. Emulation has two main requirements in the case of our project: (i) Emulate in real time the wireless communication of the active tags; (ii) Emulate the active tag processor so that the same firmware used by the real devices can be tested in emulation experiments. The conclusions of the real-world experiment using the pedestrian localization prototype system were used as guidance during the design and implementation of the emulation testbed. In addition, we used our previous experience with emulation systems, such as those presented in [11] and [12], as we built the wireless communication emulation implementation on QOMET, as discussed in Section III.

The experiment-support software RUNE (Real-time Ubiquitous Network Emulation environment) is used to effectively run and manage the experiment in real time, as it can be seen in the overview given in Figure 3. RUNE Master and RUNE Manager are modules used in all RUNE-based experiments for controlling the experiment globally and locally, respectively. The active tag module was specifically designed and implemented for this application. This module includes: (i)Active

Tag Communication and chanel spaces, used to calculate and manage the communication conditions between active tags. These functions will be discussed in Section III; (ii)Active Tag Control space, which is powered by the active tag processor (PIC) emulator, and runs the active tag firmware in real time to reproduce the active tag behavior, as it will be discussed in Section IV. The experiment itself is performed using standard PCs (running the FreeBSD operating system) that are part of the StarBED testbed. They are labeled as Execution Units in Figure 3.

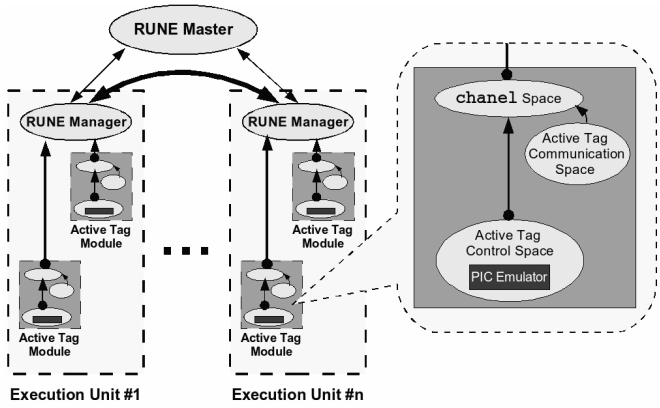

Fig. 3. Overview of the active tag emulation system.

The following RUNE configuration file is used for 16 pedestrian experiments:

```
#include "runebase.h"

BGNSPACELIST
  SPACE(mtag0,  xxx.yyy.zzz.2,   mtag.so)
  SPACE(mtag1,  xxx.yyy.zzz.3,   mtag.so)
                      ⋮
  SPACE(ftag0,  xxx.yyy.zzz.18,  ftag.so)
  SPACE(ftag1,  xxx.yyy.zzz.19,  ftag.so)
                      ⋮
  SPACE(gtag0,  xxx.yyy.zzz.22,  gtag.so)
  SPACE(gtag1,  xxx.yyy.zzz.23,  gtag.so)
                      ⋮
  SPACE(cspc0,  xxx.yyy.zzz.2,   cspc.so)
  SPACE(cspc1,  xxx.yyy.zzz.3,   cspc.so)
                      ⋮
ENDSPACELIST

BGNCONDUITLIST
  /* mtag0 */
  CONDUIT(mtag0,  cspc0)
  CONDUIT(cspc0,  mtag1)
  CONDUIT(cspc0,  mtag2)
                      ⋮
  /* mtag1 */
  CONDUIT(mtag1,  cspc1)
  CONDUIT(cspc1,  mtag2)
  CONDUIT(cspc1,  mtag3)
                      ⋮
ENDCONDUITLIST
```

III. QOMET

One of the most important elements when using emulation for studying systems that use wireless communication is to be able to recreate with sufficient realism the communication between them. For the active tags used in our pedestrian tracking system this was accomplished by extending the WLAN emulator QOMET to support the wireless transceiver used by active tags.

QOMET uses a scenario-driven architecture that has two stages. In the first stage, from a real-world scenario representation we create a network quality degradation (ΔQ) description which corresponds to the real-world events (see Figure 4).

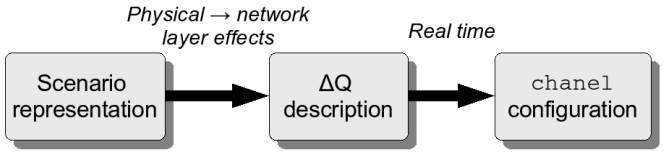

Fig. 4. Active tag communication emulation.

The ΔQ description represents the varying effects of the network on application traffic, and the wireless network emulator's function is to reproduce them.

The CHANel Emulation Library, chanel, is used to recreate scenario-specific communication conditions based on the ΔQ description (FER probabilities) computed by QOMET. Given that we emulate wireless networks, a second function of chanel is to make sure the data is communicated to all the systems that would receive it during the corresponding real-world scenario.

A. Active Tag Emulation

Our pedestrian tracking system uses the AYID32305 active tags from Ymatic Corporation., also known under the name S-NODE [13]. They were nicknamed communication tags or c-tags in the framework of the current pedestrian localization project. S-NODEs use as processing unit the PIC16LF627A microcontroller. The wireless transceiver of the active tag operates at 303.2MHz, and the data rate is 4800bps (Manchester encoding), which results in an effective data rate of 2400bps. The electric field emitted by active tags is 500μV/m; according to the specification, this produces an error-free communication range of 3-5m.

The active tag communication protocol was custom designed as a simple protocol based on time-division multiplexing. Each tag will select at random one of the available communication slots and advertise its identifier and the current time. Currently the number of available communication slots for advertisement messages is 9. There are additional communication slots that can be used on demand to transmit position tracking records from mobile tags to gateways.

The active tag communication model we currently use establishes the relationship between the distance between two nodes and the Frame Error Rate (FER, a data link layer parameter). This conversion is done based on measurements we made in an RF shielded room with the helicoidally shaped antenna,

also used in the practical experiment, and 4-byte frames. By fitting a second degree equation on the measurement results we obtained the following equation:

$$FER_4(d) = 0.1096d^2 - 0.1758d + 0.0371 \quad , \qquad (1)$$

where FER_4 is the frame error rate (the index shows it is based on 4-byte frame measurements) and d is the distance between the receiver and transmitter active tags. The above equation gives a goodness-of-fit coefficient, R^2, equal to 0.9588.

In order to extend the communication range we introduced the constant C, the scaling factor, in equation III-A. Note that equation one needs some small modifications in order to represent accurately active tag communication range. The extended equation is:

$$FER'_4(d) = 0.1096\left(\frac{d}{C}\right)^2 - 0.1758\frac{d}{C} + 0.0371$$
$$FER_4(d) = \begin{cases} 0, & if\ \frac{d}{C} < 0.5m \\ 1, & if\ FER'_4(d) > 1 \\ FER'_4(d), & otherwise \end{cases} \quad . \qquad (2)$$

Since the measurements were done using 4 byte data frames, the result of equation (III-A) must be scaled accordingly for other frame sizes, as given by:

$$FER(d) = 1 - \left(1 - FER_4(d)\right)^{\frac{H+x}{H+4}} \quad , \qquad (3)$$

where FER represents the frame error rate for a data frame of x bytes, and H is the frame header size in bytes. In our pedestrian tracking system, x and H are constant, 7 and 6 respectively.

Slot collisions arising during the time-multiplexed communication are an additional and independent source of errors. However they are handled in real time during the live experiment in the receiving procedure of the processor emulator.

The frame error rate induced by slot collision, FER_s, is expressed by the equation below:

$$FER_s = \sum_{m=1}^{n} {C_n^m}/{N_{slots}}^{m+1} \quad , \qquad (4)$$

where C_n^m is the notation for combinations of a set of n objects taken m at a time, N_slots represents the number of slots used for communication (currently 9), and n is the number of c-tags transmitters that are located in the reception range of the current tag. For a sufficiently large number of slots, equation (4) can be simplified by ignoring the terms with $m > 2$, which become very small. In this case we obtain the following simplified relation:

$$FER_s = \frac{n}{{N_{slots}}^2} \quad . \qquad (5)$$

Considering that FER_x is equal to 1 for out of range transmitters, the number n can be computed at each moment of time as the cardinal of the set of c-tags, E, for which the frame error probability due to distance when received by the current tag is inferior to 1:

$$n = |E|, E = \{e | FER_x < 1\} \quad . \qquad (6)$$

Finally, the overall frame error rate, FER, can be computed by taking into account the fact that the two error causes discussed above are independent, as follows:

$$FER = FER_x + FER_s - FER_x \cdot FER_s \quad . \qquad (7)$$

A more realistic approach is to take into account the slot collision in real time during the live experiment in the receiving procedure of the PIC emulator. This approach required more computational power, and was used only selectively. If live slot collision emulation is enabled, than the model above needs to consider FER_s equal to 0.

B. Communication channel emulation for non-IP applications

Given that the active tags we emulate do not generate IP traffic, we could not use a wired-network emulator such as dummynet for introducing network layer effects to traffic, as previously done when using QOMET. As a consequence we decided to implement our own communication channel emulation system, named CHANEL (communication CHANnel Emulation Library). This module is inserted between the space emulating the c-tag (Active Tag Control Space in Figure 3) and its connection to the other spaces using conduits. The advantage of this integration is that it becomes transparent from the point of view of emulation whether RUNE spaces are executed on the same PC or on different PCs, since communication itself is handled transparently by RUNE conduits.

The main role of CHANEL is to recreate scenario-specific communication conditions based on the ΔQ description (FER probabilities) computed by QOMET. This function is similar to that of any wired-network emulator, such as dummynet. Given that we emulate wireless networks, a second function of CHANEL is to make sure the data is communicated to all the systems that would receive it during the corresponding real-world scenario. This is done by using the ΔQ description to decide the conditions for the communication between the current active tag and the other active tags in its transmission range. Since unicast-like traffic coming from an active tag needs to be sent to multiple destinations, there are concerns regarding the performance of CHANEL when the number of destinations increases. Note however that give the small transmission range of active tags (4-5m), the number of receivers that can be in the transmission range at one moment of time is relatively small. We estimate that in general the number will be of a couple of active tags, and may reach about 10 active tags when emulating crowded areas.

The communication channel emulation library was optimized to increase performance. In addition we now started using a binary file (the output of QOMET) inside CHANEL instead of the text file used so far. A main advantage is that the reduced size of the file allows for faster reading, and therefore improves CHANEL performance. Most delays that we measure reach occasionally values around 250ms.

Although we do not know exactly the source of these errors, we believe they are related to kernel scheduling parameters in FreeBSD. Note however that since we run the experiment ten times slower than real time, a time slot of 500ms has a length of 5s, therefore a 250ms configuration error only represents a 5% error.

As we want to be able to run multiple instances of CHANEL on the same computer, as well as provide a thread-safe environment, several mutex structures were added, and now concurrent access to CHANEL data structures became possible in a safe manner.

IV. Processor Emulator

One advantage of network emulation is that already-existing network applications can be studied through this approach to evaluate their performance characteristics. Although this is relatively easy for typical network applications that run on PCs, the task is complex when the network application runs on a special processor. In order to execute the active tag application unmodified on our system, we emulate the active tag processor so that the active tag firmware can be run in our emulated environment without any modification or recompilation.

Processor emulation in our system had to take into account the following aspects that we implemented:

(i) Instruction execution emulation; all 35 PIC instructions are supported by our processor emulator.

(ii) Data I/O emulation; the only I/O access method used by the active tag application is USART (Universal Synchronous Asynchronous Receiver Transmitter). The application uses USART to interface with the active tag transceiver, and also with the back-end system in the case of gateway tags.

(iii) Interrupt emulation; all interrupts necessary for the active tag application, i.e., timer0, timer1, and timer2 are supported.

We used a pseudo-DMA data transfer technique which is not implemented by the real device instead of emulating the active tag transceiver. It makes easier to integrate the active tag application and the peripheral components of the experiment such as the chanel space etc. We also used random number generation functionality to compensate the original active tag software's weakness in random number generation.

When emulating active tag applications such as ours it is important to introduce cycle-accurate processor emulation. In our case active tags use the time information contained in messages to synchronize with each others autonomously. Incorrect time information may lead to artificial desynchronization problems and potentially communication errors, therefore it must be avoided.

One of the main concerns regarding a processor emulator is how well the execution speed is reproduced, especially in the case when running multiple instances of the emulator. In Figure 5 we show how emulation accuracy changes depending on the operating frequency and the number of instances of the PIC emulator that are run in parallel. We remind that frequency used in the active tag application is 4MHz. The figure shows that, good accuracy is obtained for up to about 40 instances running in parallel when the operating frequency is 4MHz. We tried some scheduling algorithms such as Round Robin, EDF (Earliest Deadline First) etc. in order to obtain better performance. But no significant difference could not be seen because the scheduling of the processor instances takes place always in synchronous manner unlike the process scheduling of operating system.

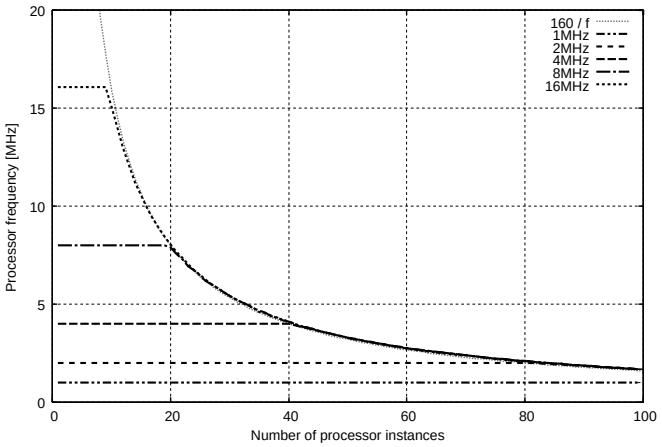

Fig. 5. Number of instances executed simultaneously at different frequencies on single PC

In our emulation, the PIC Emulator works as a part of Active Tag Control space as mentioned in Section 2. First of all, a PIC Emulator instance is allocated and initialized by invoking pic16f648Alloc(). The function allocates the data block for holding all processor internal states, registers, and memory and also launches the main emulation thread, which executes the fetch-decode-execute cycle repetitively. The main emulation thread controls the timing of progress of the emulation by using the RDTSC instruction of IA-32 architecture, which reads the Time Stamp Counter (TSC) register implemented in Intel IA-32 architecture processors. The advantage of this approach is: (i) The accuracy obtained in this way is theoretically the highest in a normal PC system, unless it has an external device which aids obtaining extremely accurate time such as GPS. (ii) It takes less time to execute the RDTSC instruction than typical C functions used to get system time, since the RDTSC instruction can be executed without the transition between kernel mode and user mode. There is also a thread created in the initialization process of the Active Tag Control space which takes care of the Pseudo-DMA data transfer. During emulation, both threads work together to accomplish real-time emulation of PIC processor.

V. Preliminary Trial

The real-world experiment was carried out in March 2007 by Panasonic. Each experiment participant was equipped with an active tag based pedestrian localization system prototype (c-tag).

A group of 16 participants were provided with instructions regarding the path they should follow in the 100 x 300m experiment area. An example of instructions, as received by participant #1 is shown in Figure 6.

The real-world experiment also included a number of tags with known position. These tags are divided into two classes: fixed and gateway c-tags, denoted in Figure 6 by F0 to F3, and GW0 to GW2, respectively. The role of fixed tags is to provide specific information to the mobile ctags that come in their vicinity to makes it possible to localize those tags. Gateway c-tags, in addition to c-tag communication, also allow information to be transferred between them and to the back end system. The gateways are placed at 3 known outdoor locations Gateways are also connected to the back-end servers; their data is used by the localization engine to determine the trajectories and positions of pedestrians.

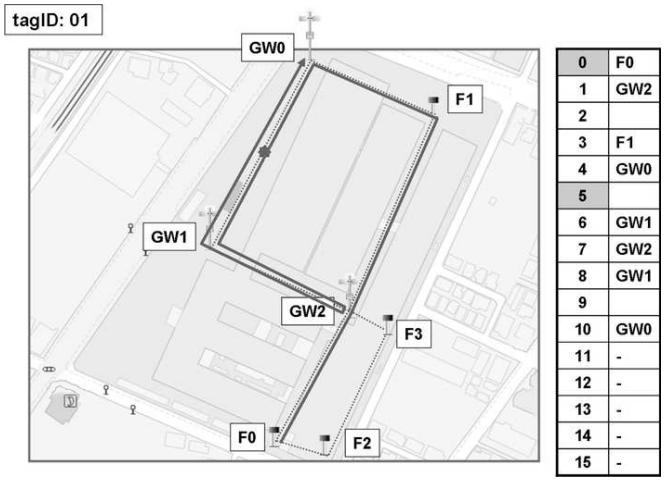

Fig. 6. Pedestrian movement instructions as received by participant #1.

The real-world experiment was successful in the sense that data collected from the active tags could be used to localize the pedestrians in most cases with sufficient accuracy. The active tag localization approach doesn't use any GPS-like or triangulation system. Instead the logs of each mobile tag, as collected by gateways, are used. The c-tag logs contain information regarding the time at which other mobile or known-position c-tags were encountered, and their identifiers. This information is used to predict the trajectory of c-tag wearers and track their position. The basic equation used to calculate the position Px of a pedestrian at moment of time tx is:

$$P_x = P_i + (P_j - P_i)\frac{t_x - t_i}{t_j - t_i}, \quad (8)$$

where P_i and P_j are the known positions of the pedestrian (from ctag logs) at moments of time t_i and t_j, with $t_i \leq t_x \leq t_j$. For more details about the experiment and the pedestrian localization engine one may consult [14] (in Japanese).

VI. Results

The emulation shown uses exactly the same conditions as the real-world experiment described in Section V, and was used to validate the emulation system. For simplicity each active tag and the associated chanel component are run on one PC. The emulational setup follows the overview presented in Figure 3.

A. Emulation results (16 virtual pedestrians)

The initial position of the 16 virtual pedestrians, the locations of the 4 fixed c-tags and 3 gateway c-tags, the building topology, and virtual pedestrian movement were all described by converting the real-world experiment instructions to the QOMET XML-based scenario description. Time granularity used when computing communication conditions, as well as during real-time execution was 0.5s. RUNE was used to configure the host PCs according to the emulation description and run the emulation.

We implemented a tool which converts the result of the emulation into KML format [15]. Visualizing the result with Google Earth™ [16] (Figure 7) [1] helps to figure out the motion of the virtual pedestrians in time and to easily identify localization problems.

In order to understand better the localization errors, it is possible to draw for each virtual pedestrian its emulated trajectory, and the trajectory localized by the system. By comparing them, and seeing where differences occur, one can determine where the algorithm needs to be improved.

Fig. 7. Visualization using Google Earth™.

We have performed several series of emulations trying to reproduce and extend the 16 virtual pedestrian experiment carried out by Panasonic. The communication range of active tags is one of the most important parameters, since it determines the area in which communication is possible. Communication range is given by transmitted power; therefore it is directly related to power consumption. In a series of emulations,

[1] Google Earth™ mapping service is a trademark of Google Inc.

we tried to see what is the performance of the localization algorithm for several communication ranges, as follows: 3m, 6m, 9m, 12m, 15m (see Figures 8 and 9). By looking at the mean localization error versus range in Figure 9, one can conclude that the range of 9 m seems to provide optimum performance in this case.

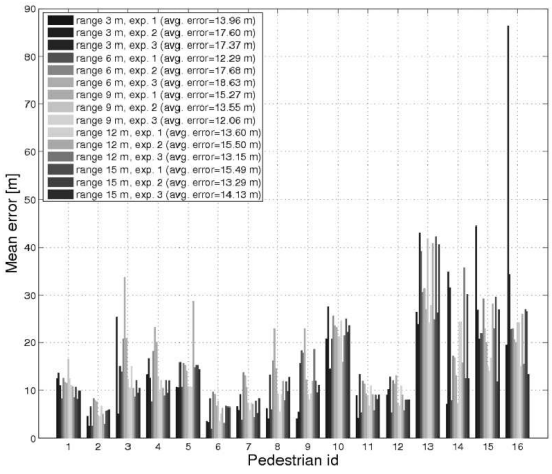

Fig. 8. Mean localization error per virtual pedestrian for several communication ranges (3 emulations per range).

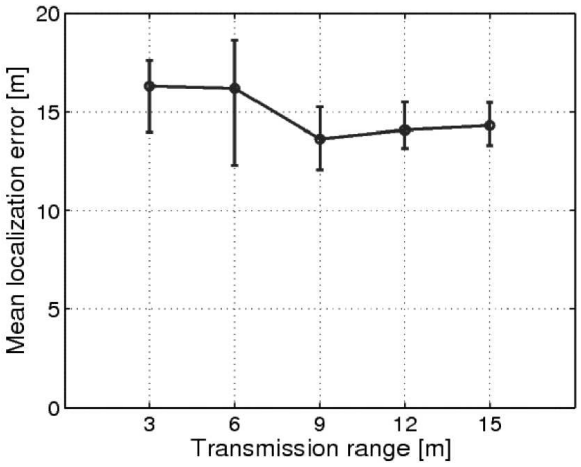

Fig. 9. Mean localization error versus communication range.

Emulation can be used to investigate a wide range of controllable conditions. We decided to use this approach to determine how localization performance changes when the number of slots allocated to communication between tags varies. For this purpose we performed several emulations, both with the tags configured for 3m range, and 9m range. In each series of emulation we varied the number of slots as follows: 3, 6, 9. Note that 9 slots is the value used by the real prototype. The results are shown in Figures 10 and 11. Analyzing the mean localization error versus the number of slots in Figure 11, we conclude that for 9 m range using 6 slots is enough, but the 3 m range does require the use of 9 slots to provide best performance.

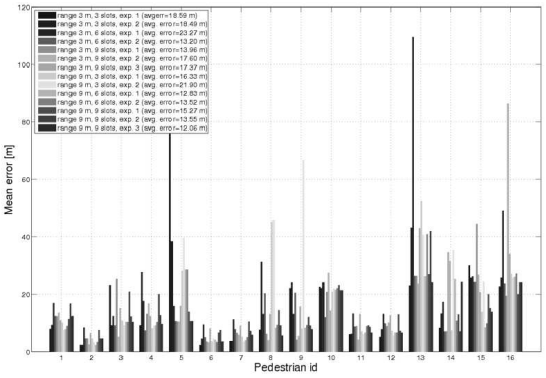

Fig. 10. Mean error per virtual pedestrian when varying the communication range and the number of communication slots used by each tag.

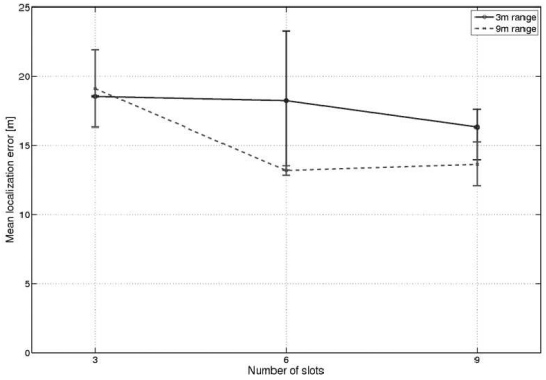

Fig. 11. The mean localization error per emulation versus the number of communication slots.

In Figures 12 and 13 we show the results for another emulation in which we configured the range of the active tags to 5m, 10m and 15m. The purpose was to demonstrate how our system could be used to determine the optimum range for the active tags.

Figure 13 indicates that for a range of 10m, the optimum performance is achieved. For 5m range, the variation of the error is quite high, since in some case the short communication range leads to the event that two tags may miss the chance to communicate, therefore their trajectory may not be correctly identified. On the other hand for communication range of 15m, the localization error increases slightly. This is explained by the fact that with a longer range the communication between tags starts and lasts until a longer distance, therefore the accuracy of localization decreases.

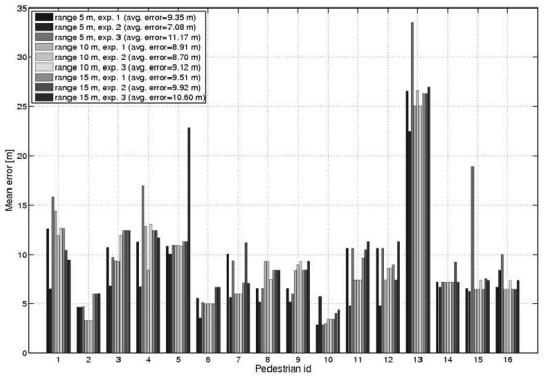

Fig. 12. Mean error for each virtual pedestrian for several communication ranges (3 emulations per range).

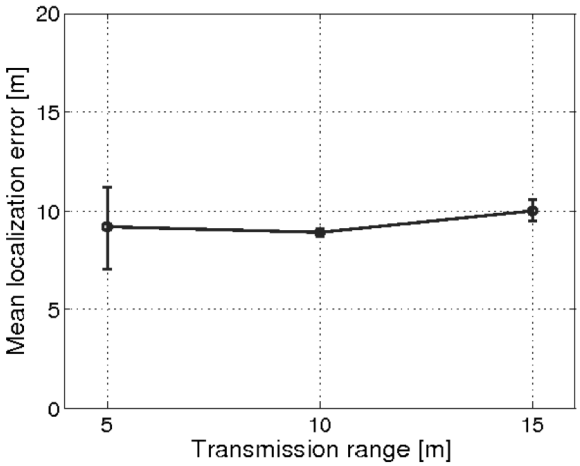

Fig. 13. Mean error for each transmission range.

In Figure 14, we show the visualization tool we use for the communication protocol of the active tags. Such a graphical representation gives an insight in the timing of the messages sent and received by active tags, as well as other elements of the communication protocol. This tool was successfully used to identify some potential firmware implementation problems. For instance, a weakness of the random number generator implementation led to the choice of the same time slot for communication in our emultaions. This fact produced an unusually large number of collision effects, for which the cause became obvious using the communication visualizer tool. As mentioned in Section IV, we implemented an alternative random number generator in the PIC emulator as a temporary solution. The implementation of the random number generation functionality is planned to be improved in the next prototype localization system.

Another issue we were able to identify by emultaions is related to time synchronization between active tags. At the moment a mobile active tag synchronizes its clock based

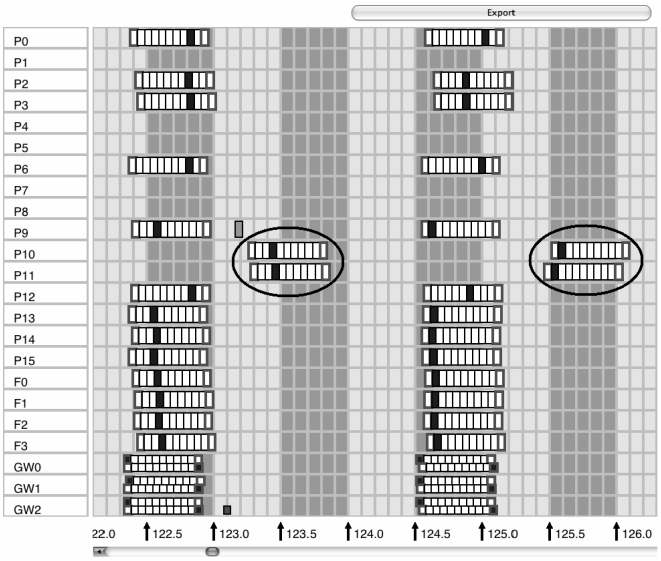

Fig. 14. Active tag communication visualization tool.

on the time received from neighboring tags. Gateways and fixed nodes do not synchronize their time. We observed in our emulation system that the time accuracy without time synchronization (e.g., for gateways) is better than with time synchronization (i.e., for mobile tags). The time drift of two or more mobile nodes that are not in the vicinity of a gateway or fixed tag becomes quickly significant using the current synchronization algorithm, while the gateways and fixed tags themselves seem to be relatively stable, although not using time synchronization. This issue had not been noticed in the real-world experiment, but it is very important. A significant time drift leads to localization inaccuracy and must be solved in the next prototype. We circled in Figure 14 an example of time drift for a pair of mobile tags (P10 and P11).

Figure 15 shows at where the #1 tag exchanged packets that used for localization to other tags. As the figure shows, enough number of packets necessary for localization were exchanged in our emulation. All the result presented in this section indicates the emulated tag software works properly even though we, unfortunately, have no way to confirm if the behavior of emulated tag software is correct by comparing with the result obtained from the real-world experiment since the real tags does not have any logging functions due to memory and processing ability restrictions.

B. Emulation results (100 virtual pedestrians)

One of the purposes of developing emulation was running large-scale emulation that cannot be executed very easily in the real world. In this content we ran several emulations with up to 100 virtual pedestrians. For the emulations, the motion of virtual pedestrians is generated by a motion generator using the real geographical information provided by GSI (Geographical Survey Institute, a Japanese governmental organization) as the constraint condition. Figure 16 shows an example of generated trajectory of virtual pedestrians.

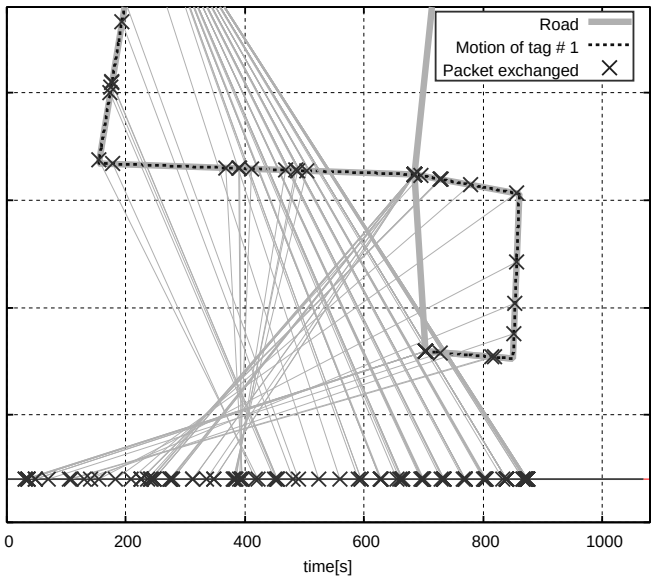

Fig. 15. Packet exchanged location and trajectory of the #1 tag.

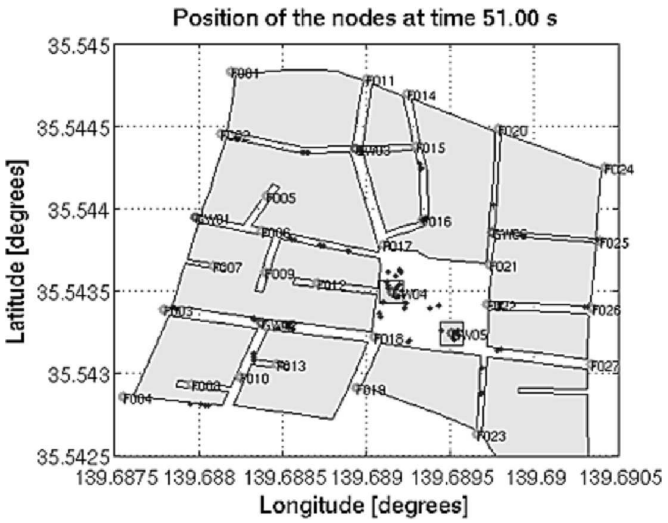

Fig. 16. Generated topology of 100 virtual pedestrian emulation.

Although we performed several series of 100 virtual pedestrian emulations, due to some problems in the localization engine we are not able to plot the localization results in the same way we did for the other emulation. The problem was the following: in the 100 virtual pedestrian emulation, the number of tags that reaches the destinations at GW4 and GW5 is high, therefore there are many collisions between the mobile tags as they try to upload their information. In addition, due to the relatively big size of the area, the number of encounters between mobile tags is rather small. These two reasons lead to the fact that not all the tags manage to upload information to gateways. Table I illustrates how many mobile tags never succeeded to upload information during the emulation. The table shows that almost half of the mobile tags never succeeded to upload any information although the rest of tags uploaded hundreds of packets as Figure 17 shows. The current version of the localization engine is not able to cope with the case when incomplete information is given, and was not able to produce any results. Panasonic is currently investigating this issue so that the robustness of the localization engine can be increased.

TABLE I
NUMBER OF TAGS SUCCEEDED TO UPLOAD INFORMATION.

Number of tags that uploaded at least one record	Number of tags that uploaded no record
55	45

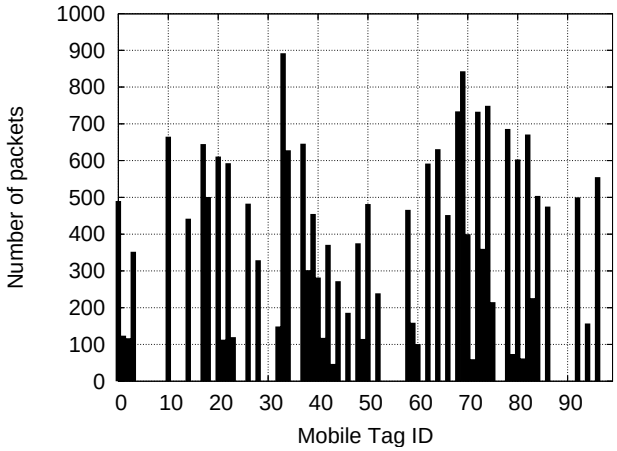

Fig. 17. Number of P records uploaded by each mobile tag.

Table II shows how many records are left in the memory of each mobile tag at the end of emulation. According to the table, over 90% of mobile tags had had information not uploaded at the end of emulation. This happens if a mobile tag lost the opportunity to upload information in the final part of its trajectory for some reasons, mainly collision in the emulation. So Panasonic is now designing a collision avoidance protocol, since our emulations have shown that without such an algorithm the active tag localization system cannot function for relatively crowded areas. This is one of the important findings of our emulations.

TABLE II
NUMBER OF TAGS HAVE INFORMATION LEFT IN MEMORY.

Number of tags have information left in memory	Number of tags have no information left in memory
91	9

VII. CONCLUSION AND FUTURE WORK

In this paper we presented an emulation system that we designed and developed for active tag applications. This emulation system is currently employed for the development phase emulations of a pedestrian localization system by Panasonic.

By using our system it was possible to simplify the development and testing procedures of the localization engine, and identify several firmware implementation issues.

In order to validate the emulation system we carried out tests that reproduced a real-world 16 pedestrian experiment that took place in March 2007 using the prototype of the active tag based pedestrian localization system. The emulation results show the good agreement that exists between the virtual motion patterns of pedestrians, reproduced according to the real-world scenario, and the actual conditions that were recreated in our emulation.

Through emulations we found several issues originated from active tag's firmware or protocol. Some of the issues such as those of time synchronization and random number generation are already fixed by modifying firmware in emulation first and then feeding it back to the real firmware. Some more fundamental issues such as unreliable behavior in crowded areas are under study by Panasonic and supposed to be fixed in the next version of the firmware. In both cases, our distributed emulation approach utilizing the real firmware made it easy to find out problems and fix them. This fact tells that the emulation environment implemented on distributed environment is useful to validate systems especially which consists of many small components such we targeted.

Our future work has several main directions: improve the scalability of the system so as to enable emulations of pedestrian groups as large as 1000; improve the realism of the wireless communication emulation by using more accurate 3D models for topology and electromagnetic wave propagation; combine the behavioral motion model with a GIS-based urban area description to create a realistic pedestrian trajectory generator for large-scale urban emulations.

REFERENCES

[1] Junya NAKATA, Razvan Beuran, Tetsuya Kawakami, Ken ichi Chinen, Yasuo Tan, and Yoichi Shinoda. Distributed emulator for a pedestrian tracking system using active tags. In *UBICOMM 2008: Proceedings of the 2008 The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 219–224, Washington, DC, USA, 2008. IEEE Computer Society.

[2] Razvan Beuran, Lan Tien Nguyen, Khin Thida Latt, Junya Nakata, and Yoichi Shinoda. Qomet: A versatile wlan emulator. In *AINA '07: Proceedings of the 21st International Conference on Advanced Networking and Applications*, pages 348–353, Washington, DC, USA, 2007. IEEE Computer Society.

[3] J. NAKATA, T. Miyachi, R. Beuran, K. Chinen, S. Uda, K. Masui, Y. Tan, and Y. Shinoda. Starbed2: Large-scale, realistic and real-time testbed for ubiquitous networks. In *The 3rd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(TridentCom 2007), Orlando, Florida, U.S.A.*, 2007.

[4] A. Janek, C. Trummer, C. Steger, R. Weiss, J. Preishuber-Pfluegl, and M. Pistauer. Simulation based verification of energy storage architectures for higher class tags supported by energy harvesting devices. volume 32, pages 330–339, Amsterdam, The Netherlands, The Netherlands, 2008. Elsevier Science Publishers B. V.

[5] Rutgers University and Wireless Information Network Laboratory. *ORBIT - Wireless Network Testbed*. http://www.orbit-lab.org/ 15.05.2009.

[6] David Johnson, Tim Stack, Russ Fish, Daniel Montrallo Flickinger, Leigh Stoller, Robert Ricci, and Jay Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *INFOCOM*. IEEE, 2006.

[7] Junlan Zhou, Zhengrong Ji, and Rajive Bagrodia. Twine: A hybrid emulation testbed for wireless networks and applications. In *INFOCOM*. IEEE, 2006.

[8] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM.

[9] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras. Atemu: A fine-grained sensor network simulator. In *Proc. of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara, California, U.S.A.*, 2004.

[10] Microchip Technology Inc. *MPLAB*. http://www.microchip.com/ 15.05.2009.

[11] R. Beuran, J. NAKATA, T. Okada, T. Miyachi, K. Chinen, Y. Tan, and Y. Shinoda. Performance assessment of ubiquitous networked systems. In *5th International Conference on Smart Homes and Health Telematics (ICOST2007), Nara, Japan*, pages 19–26, 2007.

[12] Takahashi Okada, Razvan Beuran, Junya Nakata, Yasuo Tan, and Yoichi Shinoda. Collaborative motion planning of autonomous robots. volume 0, pages 328–335, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[13] Ymatic Inc. *S-NODE specification*. http://www.ymatic.co.jp/ 15.05.2009.

[14] Y. Suzuki, T. Kawakami, M. Yokobori, and K. Miyamoto. A real-space network using bi-directional communication tags - pedestrian localization technique and prototype evaluation. In *IEICE Forum on Ubiquitous and Sensor Networks, techni cal report*, 2007.

[15] Open Geospatial Inc. *KML Standard*. http://www.opengeospatial.org/standards/kml/ 15.05.2009.

[16] Google Inc. *Google Earth*. http://earth.google.com/ 15.05.2009.