Distributed Emulator for a Pedestrian Tracking System Using Active Tags

Junya NAKATA
National Institute of Information
and Communications Technology
2-12 Asahidai, Nomi, Ishikawa Japan
jnakata@nict.go.jp

Razvan Beuran
National Institute of Information
and Communications Technology
2-12 Asahidai, Nomi, Ishikawa Japan
razvan@nict.go.jp

Tetsuya Kawakami
Matsushita Electric Industrial Co., Ltd.
4-3-1 Tsunashima-higashi, Kohoku,
Yokohama, Kanagawa Japan
kawakami.tetsu@jp.panasonic.com

Ken-ichi Chinen
Japan Advanced Institute of
Science and Technology
1-1 Asahidai, Nomi, Ishikawa Japan
k-chinen@jaist.ac.jp

Yasuo Tan
Japan Advanced Institute of
Science and Technology
1-1 Asahidai, Nomi, Ishikawa Japan
ytan@jaist.ac.jp

Yoichi Shinoda
Japan Advanced Institute of
Science and Technology
1-1 Asahidai, Nomi, Ishikawa Japan
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 system met the requirements to create a flexible experimental platform to support the development of the pedestrian tracking system. We confirmed the results obtained by running tests corresponding to a real-world experiment.

1. Introduction

As Matsushita Electric Industrial Co., Ltd. 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 organize and 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. In order to be able to use this testbed for active tag emulation we developed several subsystems, and integrated them with the existing testbed infrastructure. These subsystems were developed on the basis of existing tools that are already used on StarBED, namely the wireless network emulator QOMET [5], and the experiment support software RUNE [9].

Active tags were so far mainly studied through simulation, such as the work presented in [6]. 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

978-0-7695-3367-4/08 $25.00 © 2008 IEEE
DOI 10.1109/UBICOMM.2008.77

approach of ORBIT [1], or the more realistic robot-based Mobile Emulab [7]. An alternative which avoids undesired interferences and side effects is to use computer models for real-time experiments. TWINE [12] 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 [11] 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 [2]. 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 Matsushita Electric Industrial Co., Ltd. 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. Using the prototype of the pedestrian localization system, real-world experiments were carried out in March 2007, as reported in [**?**]. 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.

2. 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 put to use our previous experience with emulation systems, such as those presented in [4] and [10], as we built the wireless communication emulation implementation on QOMET [5], as discussed in Section 3.

The experiment-support software RUNE (Real-time Ubiquitous Network Emulation environment) [9]

is used to effectively run and manage the experiment in real time, as it can be seen in the overview given in Figure 1. 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 3; (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 4. 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 1.

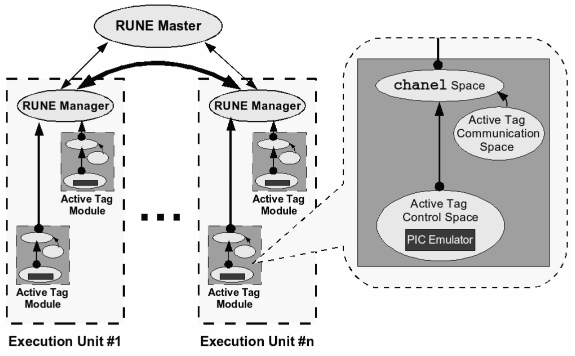


Figure 1. Overview of the active tag emulation system.

3. 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 [5] 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 2).

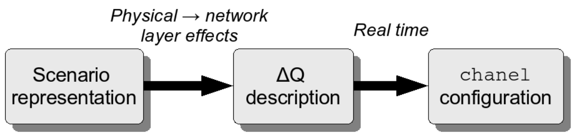


Figure 2. Active tag communication emulation.

The ΔQ description represents the varying effects of the network on application traffic, and the wireless network emulators 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.

3.1. Active Tag Emulation

Our pedestrian tracking system uses the AYID32305 active tags from Ymatic Corporation, also known under the name S-NODE [3]. 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 500uV/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:

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

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.

Since the measurements were done using 4 byte data frames, the result of equation (3.1) must be scaled accordingly for other frame sizes, as given by:

$$FER = 1 - (1 - FER_4)^{\frac{H+x}{H+4}}, \qquad (2)$$

where FER represents the frame error rate for a data frame of x bytes, and H is the frame header size in bytes.

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.

4. 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.

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.

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 3 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 couldn't be seen because the scheduling of the processor instances takes place always in synchronous manner unlike the process scheduling of operating system.

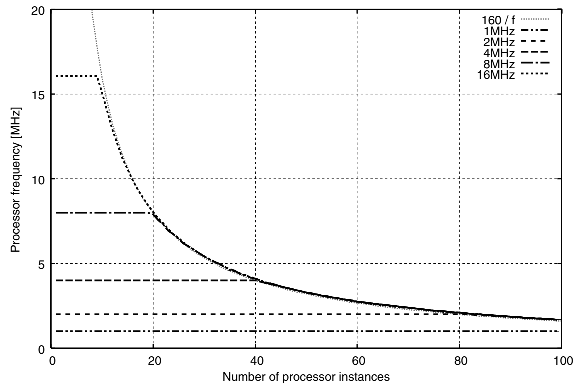


Figure 3. Number of instances executed simultaneously at different frequencies on single PC

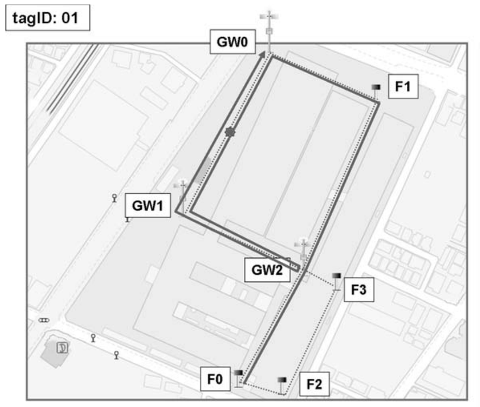
5. Preliminary Trial

The real-world experiment was carried out in March 2007 by Matsushita Electric Industrial Co., Ltd. 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 4.

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 4 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.

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



0	F0
1	GW2
2	
3	F1
4	GW0
5	
6	GW1
7	GW2
8	GW1
9	
10	GW0
11	-
12	-
13	-
14	-
15	-

Figure 4. Pedestrian movement instructions as received by participant #1.

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.

6. Results

The experimental results we present here were obtained by emulation on StarBED. The experiment shown uses exactly the same conditions as the real-world experiment described in Section 5, and was used to validate the emulation system. For simplicity each active tag and the associated chanel component are run on one PC. The experimental setup follows the overview presented in Figure 1.

The initial position of the 16 pedestrians and the locations of the 4 fixed c-tags and 3 gateway c-tags, the building topology, and 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 experiment description and run the experiment.

In Figure 5 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 emulation experiments. 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 4, we implemented an alternative random number generator in the PIC emulator as a temporary solution. random number generation will be improved in the next prototype localization system.

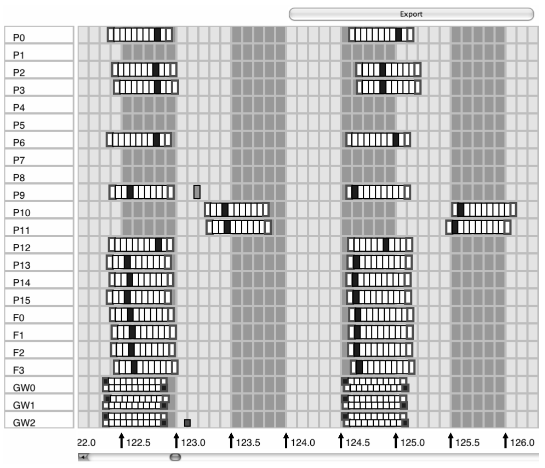


Figure 5. Active tag communication visualization tool.

Another issue we were able to identify by emulation experiments is related to time synchronization between active tags. At the moment a mobile active tag synchronizes its clock based 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 5 an example of time drift for a pair of mobile tags (P10 and P11).

Figure 6 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.

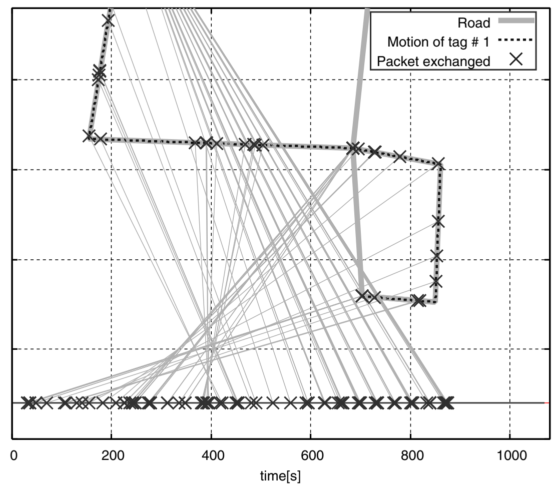


Figure 6. Packet exchanged location and trajectory of the #1 tag.

7. Conclusion

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 experiments of a pedestrian localization system by Matsushita Electric Industrial Co., Ltd. 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 results 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 experiment 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 experiment.

Our future work has several main directions: improve the scalability of the system so as to enable experiments 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 experiments.

8. Acknowledgment

This work was supported in part by the Japan Science and Technology Agency (JST) under a grant of 2007 3rd round of the Innovating Industry-University Collaboration Seeds project.

References

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

[2] Microchip Technology Inc., MPLAB, http://www.microchip.com/.

[3] Ymatic Inc., S-NODE specification, http://www.ymatic.co.jp.

[4] 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.

[5] R. Beuran, L. T. Nguyen, K. T. Latt, J. Nakata, and Y. Shinoda. Qomet: A versatile wlan emulator. In *IEEE International Conference on Advanced Information Networking and Applications (AINA-07) and Niagara Falls and Ontario and Canada*, pages 348–353, 2007.

[6] 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. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), Lubeck, Germany*, pages 463–462, 2007.

[7] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *IEEE INFOCOM 2006, Barcelona, Spain*, 2006.

[8] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proc. of the First ACM Conference on Embedded Networked Sensor Systems (SenSys03), Los Angeles, California, U.S.A.*, 2003.

[9] 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 3th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(TridentCom 2007), Orlando, Florida, U.S.A.*, 2007.

[10] T. Okada, R. Beuran, J. Nakata, Y. Tan, and Y. Shinoda. Collaborative motion planning of autonomous robots. In *3rd International Conference on Collaborative Computing (CollaborateCom 2007), White Plains, New York, U.S.A.*, 2007.

[11] 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.

[12] J. Zhou, Z. Ji, and R. Bagrodia. Twine: A hybrid emulation testbed for wireless networks and applications. In *IEEE INFOCOM 2006, Barcelona, Spain*, 2006.