Journal of Ambient Intelligence and Smart Environments 1 (2010) 1
IOS Press

Emulation framework for the design and development of active RFID tag systems

Razvan Beuran [a,b,*], Junya Nakata [a,b], Tetsuya Kawakami [c], Takashi Okada [b,a], Ken-ichi Chinen [b,a], Yasuo Tan [b,a] and Yoichi Shinoda [b,a]

[a] *Hokuriku Research Center, National Institute of Information and Communications Technology, 2-12 Asahidai, Nomi, Ishikawa, Japan*
E-mail: {razvan,jnakata}@nict.go.jp
[b] *Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa, Japan*
E-mail: {tk-okada,k-chinen,ytan,shinoda}@jaist.ac.jp
[c] *Panasonic System Solutions Company, Panasonic Corporation, 4-3-1 Tsunashima-higashi, Kohoku, Yokohama, Kanagawa, Japan*
E-mail: kawakami.tetsu@jp.panasonic.com

Abstract. The design and development of active RFID tag systems requires several iterations, and thorough testing in order to ensure that the prototype systems behave as intended. Facilitating these operations contributes significantly to faster product development and to a quicker time to market. The emulation framework that we present makes it easier to carry out the verification and testing related to the design and development of active RFID tag systems. We illustrate the use of this framework during the design and development of an active RFID tag based pedestrian localization system that is being developed by Panasonic Corporation. In this context emulation was successfully used: (i) to identify active RFID tag firmware issues; (ii) to assess the overall performance of the localization system; (iii) to explore the system parameter space so as to identify the values that provide optimum performance; (iv) to extend the range of tested scenarios beyond what could be done through real-world trials.

Keywords: emulation, active RFID tag, ubiquitous systems, testing environment, design and development

1. Introduction

Smart environments typically make use of a large number of small sensing and computing devices in order to achieve the Ambient Intelligence (AmI) functionality they are designed for, to "proactively, but sensibly, support people in their daily lives" [1]. Each of these devices has limited communication, computational and energy resources, and the resulting intelligence is usually a synergistic effect.

Whether embedded in homes, or deployed in outdoor environments, the AmI devices require algorithms and protocols for communication and data processing operations. It is precisely the embedded nature of AmI systems that makes controlled experiments difficult. As one cannot separate the algorithms and the real-world components, reproducibility of real-world trials may be rather low. Development is further complicated by the small form factor of such devices, and by the limited accessibility to their internal state. Moreover, the logistical difficulties of conducting experiments with a large number of physical devices makes it difficult, if not impossible, to exhaustively validate a system.

Before a product is ready for mass production, it has to go through several design and development phases. In the case of active RFID tag systems, which are the focus of this paper, one can distinguish the following three main development areas: hardware, firmware,

*Corresponding author. Address: Hokuriku Research Center, National Institute of Information and Communications Technology, 2-12 Asahidai, Nomi, Ishikawa. Tel: +81-761-51-8118. Fax: +81-761-51-8177. E-mail: razvan@nict.go.jp.

and software. If the system is being built from scratch, hardware design and implementation is clearly the first task to be taken into account. Once this phase ends, developers can proceed to the design and implementation of the system firmware. The firmware defines the behavior of the active RFID tag system, and works together with the hardware components to perform the functionality for which the system is being developed. Special software may be needed to combine the data of individual active tags so as to provide a higher level functionality by creating the synergistic effect mentioned before.

Although the hardware design process usually results in only one prototype system, many various algorithms and protocols can be implemented in firmware. It is desired to be able to test each of these algorithms and protocols so that their performance can be assessed, and so that they can be compared with each other. Usually the algorithms and protocols have several configurable parameters, and system performance depends significantly on the choice of these parameters. Finding the parameter values that ensure optimum performance under specific circumstances is also an important part of the system design and development phase. Performing all these tasks within a short time frame contributes significantly to a faster product development, and to a quicker time to market.

In this paper we present an emulation framework that can be used during the design and development of active RFID tag systems for testing and validating the behavior of the system being designed. Our emulation framework can be employed either as an intermediate step between numerical simulation and real-world trials, or even instead of numerical simulations. The emulation framework can be used in parallel with real-world trials as well, so as to explore a wider range of scenarios. An earlier version of this work appeared in [4]. We present here in more detail the components of the emulation framework, we show the validation of the emulation framework itself, and we illustrate by several examples the practical use of our system for evaluating the emulated active RFID tag system.

Let us discuss next the need for such an emulation framework. Traditionally, most of the investigations related to algorithms, protocols and parameter values are done using numerical simulations, so that a wide range of conditions is explored. Once enough confidence in the algorithms and protocols is built, developers proceed to the next phase, the real-world trials, in which the prototype of the system is being tested in a real-world environment. Unfortunately, active RFID tag simulators use logical models of the system to be tested, running in logical time, therefore it may be difficult to accurately assess how the system will behave in a real environment, when interacting in real time with other systems.

On the other hand, in real-world trials undesired factors such as wireless communication interference may influence the results in unexpected ways. Moreover, due to time and monetary constraints it is not possible to exhaustively explore through real-world trials all the conditions that could be met during the lifetime of a system. In conclusion, most developers currently use a limited number of real-world trials, preceded in many cases by more or less extensive evaluations through simulation; thus, system evaluation before deployment is generally poor.

Emulation is a technique intended to bridge the gap between simulation experiments and real-world trials. This is done by using a hybrid approach, in which some parts of the system are modeled (for example, the wireless communication between active RFID tags), whereas some other parts of the system are real (such as the active RFID tag firmware running on a processor emulator). Emulation allows the same level of control over experiment conditions as simulation does, therefore it can be used to study in an automated fashion a wide range of conditions, while avoiding undesired interferences. The use of real components increases the realism of the experiments compared to simulation, and makes the experimental results more useful in practice. For example, if one can test by emulation the same firmware that is deployed on the active RFID tags, then any result obtained will be directly pertinent for the real system as well. Table 1 summarizes some of the qualitative differences between the three experiment techniques.

Table 1
Experimentation technique comparison

	Simulation	Real world	Emulation
Done in real time	No	Yes	Usually
Easy to control	Yes	No	Yes
Range of conditions	Wide	Narrow	Wide
Cost to use	Low	High	Medium

Although the emulation framework that we designed and implemented is generic, the present work was done to enable experiments related to a pedestrian localization system that is being developed by Panasonic Corporation. Therefore some of the implementation details are specific to this particular appli-

cation. Nevertheless, the framework that we propose can be reused for the validation of other active RFID tag systems (and even sensor systems, or other ubiquitous computing devices) by simply altering/replacing the specific modules that are particular to the pedestrian localization system that we emulated: the wireless communication model, and the active RFID tag processor emulator (see Section 3).

The main contributions of this paper are:

- We present the active RFID tag system emulation framework that we designed and implemented, as well as the testbed used for running experiments;
- We show how this emulation framework was employed in the development phase of a specific active RFID tag based pedestrian localization system in order to validate its behavior;
- We illustrate the use of the emulation framework for selecting the optimum values of several configurable parameters of the pedestrian localization system, and for extending the range of tested scenarios beyond what was feasible through real-world trials.

The remainder of this paper is organized as follows. Section 2 discusses the active RFID tag based pedestrian localization system that we emulate. Section 3 introduces the emulation framework and its components. Section 4 presents the validation procedure of the emulation framework, as well as its practical usage for the verification of the active RFID tag system. Section 5 discusses related work. We conclude in Section 6, followed by a section of references.

2. Pedestrian localization system

The design and implementation of the emulation framework that we present was motivated by the experiments related to a pedestrian localization system using active RFID tags that is being developed by Panasonic Corporation. Location tracking is one of the important applications of active RFID tags that could play a vital role in situations such as disaster, public surveillance, etc. For instance, during the evacuation of a school following an earthquake, it is vital to be able to determine whether evacuation was completely successfully or not; if persons are still present in the disaster perimeter, their location should be identified. Some similar scenarios, albeit using GPS or omnidirectional cameras for localization, are discussed in [7].

Table 2 shows the main differences between position localization using cellular phones equipped with GPS systems that transmit their position via the cellular phone network, and the active RFID tag based technique developed by Panasonic Corporation. These differences emphasize the advantages of the active RFID tag approach for pedestrian localization. For a thorough discussion of location estimation techniques in smart environments see [11].

Table 2
Comparison of GPS-enabled cellular phone and active RFID tag based location tracking

GPS-enabled cellular phone	Active RFID tag system
Snapshot of position	Continuous movement
Each terminal is paged	Many tags simultaneously
Not available in disaster	Potentially more reliable
High-power consumption	Low-power consumption

2.1. System description

The location tracking system prototype uses active RFID tags manufactured by Ymatic corporation under the specification AYID32305 [19]; within the framework of pedestrian localization these tags were nicknamed *communication tags* or *c-tags*. The processing unit of c-tags is the PIC16LF627A micro-controller, with an operating frequency of 4 MHz. The c-tag wireless transceiver works on the 303.2 MHz frequency, and transmits at a data rate of 4800 bps with Manchester encoding, resulting in an effective data rate of 2400 bps. According to Ymatic Corporation, the error-free communication range of c-tags is between 3 and 5 meters, depending on the type of antenna being used. One can choose between an internal loop antenna (L-shaped), and an external helical antenna.

The pedestrian localization system developed by Panasonic Corporation uses these active RFID tags to provide to the localization engine software the information required for automatically computing the trajectory to date and the current location of the active tag wearer. Three types of c-tags are used to achieve this goal:

Mobile c-tags Worn by pedestrians, mobile c-tags transmit periodic ID packets that contain time information and the ID of the sender c-tag. Mobile c-tags also receive ID packets sent by other c-tags, and record the information contained in these packets. A picture of a mobile c-tag worn by a pedestrian is shown in Figure 1, and a detailed view is presented in Figure 2.

Fig. 1. Mobile c-tag prototype worn by a pedestrian.

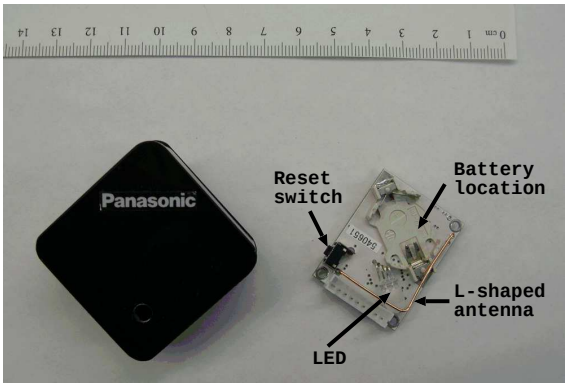

Fig. 2. Detailed view of the c-tag prototype using L-shaped antenna; centimeter marks in the top area of the figure are used to indicate prototype size.

Fixed c-tags Placed at known locations, fixed c-tags are identical in functionality with the mobile c-tags. The known location of the fixed c-tags is used by the localization algorithm to improve tracking accuracy.

Gateway c-tags Placed at known locations, gateway c-tags have some additional functions compared to other c-tags, such as sending beacons, coordinating the upload and receiving of record data from mobile c-tags, as well as making cumulative record data available to the localization engine by means of a wireless network connection using the IEEE 802.11j standard.

The communication protocol used by the active RFID tags was custom designed by Panasonic based on the time-division multiplexing paradigm. Before sending an ID packet, c-tags will randomly select one of the available communication slots, and use it to send the packet. The number of communication slots employed in the prototype system for such advertisement messages is 9. We examined through emulation the choice of the number of communication slots in Section 4.3.2. The received ID messages are also used by tags to synchronize their internal clocks. An issue with the synchronization algorithm that we discovered by emulation will be discussed in Section 4.2.

Given the memory limitations of the active RFID tags, the received ID packets are not saved individually, but merged in an unique ID record by using bit masks. The duration of the period during which merging occurs is controlled by a parameter that specifies the number of intervals that are used when merging. An interval in this case represents the time between two active periods of the tags, and is equal to approximately 2.23 s. This means that when using 15 intervals for merging (or 0x07 in hexadecimal notation), all the IDs of the tags met during a period of approximately 33.5 s will be merged into an ID record with a unique timestamp, representing the beginning of the corresponding period. One can notice that while merging saves memory, it also contributes to a decrease in the time accuracy of the information that is being recorded.

The number of memory locations available for ID records in the current prototype is 16. This means that, with the settings mentioned above, a c-tag can keep track of the tags met within a period of about 9 minutes. Several special communication slots can be used on demand by the mobile c-tags to upload the ID records to gateways when located in their vicinity. The ID records that are not uploaded by a mobile c-tag will be discarded as new tags are encountered, starting from the oldest record. We studied through emulation the influence of the ID record merging period on system performance in Section 4.3.3.

For pedestrian tracking, the localization engine retrieves ID record data from gateway c-tags, and then uses the contained time and ID information to determine the position and trajectory of the mobile c-tags. The basic idea of the localization algorithm is similar to that of interpolation: by knowing the position of a c-tag at two moments of time (for example, when a mobile c-tag meets a fixed c-tag, or another mobile c-tag with known trajectory), one can determine its location at intermediate moments of time by linear interpolation. The equation for calculating the position P_x of a mobile c-tag at an arbitrary moment of time t_x is:

$$P_x = P_i + (P_j - P_i)\frac{t_x - t_i}{t_j - t_i}, \quad (1)$$

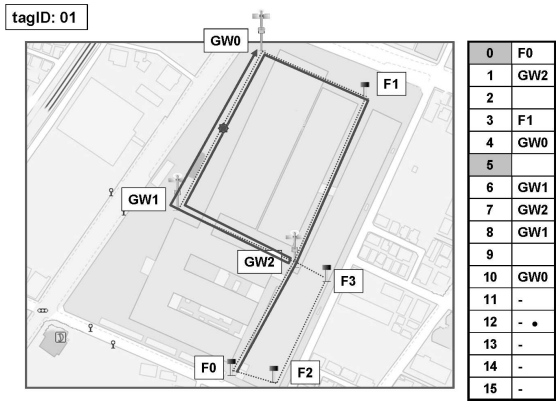

Fig. 3. Movement instructions for pedestrian #1 participating to the 16-pedestrian real-world trial.

where P_i and P_j are two known positions of the c-tag for the moments of time t_i and t_j, with $t_i \leq t_x \leq t_j$.

2.2. Real-world trials

Two real-world trials were done by Panasonic with the prototype system by orchestrating the movement of 16 pedestrians, each carrying a c-tag, in an area of approximately 100 x 300 m that included several buildings. In addition to the 16 mobile c-tags, 4 fixed c-tags, denoted by F0 to F3, and 3 gateways, denoted by GW0 to GW2, were used (see Figure 3). Out of these, the fixed c-tags F2 and F3 were placed inside buildings, and the other fixed and gateway c-tags were located outdoors.

Each pedestrian received movement instructions indicating the time in minutes, and the location at which the participant should be at that moment of time, including both indoors and outdoors trajectories (see Figure 3 for an example of movement instructions). Timing the motion was done by each participant on his/her own. Participants were also equipped with GPS devices, so that the robustness of the two tracking methods can be compared.

The experiment made it possible to validate the basic behavior of the prototype system. However, it also revealed several issues:

1. Organizing an experiment with even as few as 16 persons was time consuming: the 15 minutes experiment took several hours to prepare, and could only be repeated two times;
2. The accuracy of participants' movement in the two different trials was not good enough to allow reproducibility of the results; this was caused both by the variable movement speed of the participants, and by the fact that participants tended to use other information then the movement instructions to decide when to move, such as the fact whether their active tags were communicating with each other or not;
3. The wireless communication between the c-tags was not always reliable, which influenced both directly and indirectly experiment result repeatability (for example through movement variability, as explained above);
4. Battery depletion was relatively fast, and caused signal to weaken during and between trials;
5. The off-the-shelf GPS receivers had difficulties in providing a reliable location for small scale movements; moreover, they could not be used inside buildings.

These observations motivated the collaboration between NICT Hokuriku Research Center and Panasonic, that lead to the development of the emulation framework presented in this paper. The goal of our research was to make possible reproducible experiments with the pedestrian localization system, by emulating the communication between c-tags while running the same firmware that was used in the prototype system, so that its performance characteristics can be more thoroughly assessed.

3. Emulation framework

The emulation framework that we designed and implemented has three main components:

StarBED The large-scale network experiment testbed at NICT Hokuriku Research Center used as the infrastructure of the emulation framework. The associated experiment-support software tools, SpringOS and RUNE, are used to drive the emulation. In particular, RUNE plays an essential part in coordinating the execution of the emulated active RFID tags on StarBED hosts.

QOMET The wireless network emulator employed for reproducing on StarBED the wireless communication conditions between active RFID tags.

PIC Emulator The active RFID tag processor emulator in charge of emulating the active RFID tag micro-controller, so that the same firmware used in the real active tags can be evaluated in the emulation environment.

In the next subsections we describe in more detail each of these three components.

Fig. 4. StarBED: a large-scale network experiment testbed.

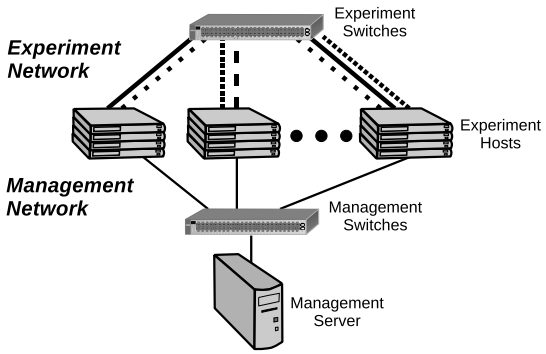

Fig. 5. StarBED topology for network experiment execution and management.

3.1. StarBED

StarBED represents the experiment execution infrastructure of our emulation framework [12]. StarBED is a testbed consisting of about 1000 commodity PCs, the *experiment hosts* (see Figure 4). The hosts have each between two and four network interfaces, operating either at 100 Mbps, or at 1 Gbps. This allows to have redundant full connectivity by means of two sets of switches, the *experiment switches* and the *management switches*. Thus there are two separate networks in StarBED, the *experiment network* and the *management network* (see Figure 5). Network separation ensures that the management traffic doesn't interfere with that of the experiments being run.

Experiment execution is controlled by a *management server*, which can be effectively any PC in the cluster that doesn't participate in the experiment. Specific switch configurations are used to produce logically separated experiment network topologies by using VLANs, so that several users can be active simultaneously. By means of the dedicated management network, experiment hosts can be loaded with the appropriate software, controlled and monitored, all without affecting the experiment.

The standard operating systems currently supported on StarBED are Windows, Linux and FreeBSD. One can also deploy specialized installations, such as software router systems, or use wired-network emulators, such as *dummynet* [15] running on FreeBSD. Users can also employ virtualization techniques, such as VMWare, to logically increase the number of experiment hosts.

The core network has several empty locations where users of the experiment environment can plug in their own devices, and thus integrate them in the experiment network. Examples of such devices are products under test, commercial routers, measurement equipment, etc. One can also connect StarBED to external networks and the Internet, so that remote locations can be included in experiments.

To assist StarBED users, two experiment-support software tools are available: SpringOS and RUNE. By using StarBED as an experimental platform assisted by these support tools, one benefits from the following features:

- Use of commodity PCs in a large-scale setup that makes it possible to emulate large network environments;
- Flexibility of the experiment environment that allows to easily switch between multiple different configurations, depending on the intended experiment;
- Powerful management and experiment-support software tools that enable easy control, quick reconfiguration, and concurrent use of the facility for independent experiments.

3.1.1. SpringOS

The main role of SpringOS is to manage experiment execution on StarBED. SpringOS also facilitates the usage of the testbed by multiple users simultaneously, both for hosts and switches. Access restrictions and mediation mechanisms for sharing such resources are built into SpringOS by design.

For using StarBED by means of SpringOS, users have to write a configuration file that describes the experiment. Based on the user-defined configuration file, SpringOS automatically performs the following tasks:

1. Assign experiment hosts from the pool of available cluster PCs;

2. Upload the appropriate operating system to the assigned experiment hosts;
3. Configure experiment switches to build the target network topology required by the experiment;
4. Drive the execution of the experiment scenario according to the configuration file, by starting and stopping the necessary applications, etc.

For more details about the SpringOS configuration file syntax, as well as regarding the effective operation of SpringOS, please see [12]. Note that while steps (1) to (3) above are essential for ensuring a correct execution of the experiment in a multi-user environment, step (4) is optional, and other tools can be employed for the same tasks, such as RUNE (see Section 3.1.2), or even shell scripts. In particular, for the experiments that we discuss in this paper, we use RUNE, the other StarBED support tool, to manage application execution.

3.1.2. RUNE

In our emulation framework, SpringOS is used to define basic experiment properties, such as the network topology used. However, due to the PC-oriented architecture of SpringOS, this tool cannot be used to run ubiquitous network experiments such as those related to active RFID tags. The main reason behind this is that SpringOS is intended for use with IP network applications running on commodity PCs. Therefore it lacks the fine-grain control needed when running custom software such as the active RFID tag firmware.

RUNE [13] is another experiment-support software that is being developed for StarBED. RUNE stands for Real-time Ubiquitous Network Emulation environment, and was designed on purpose to support the emulation of large ubiquitous networks. The most significant features of RUNE are:

– Support of real-time concurrent execution of numerous nodes;
– Provision of multi-level emulation layers;
– Support for the emulation of the surrounding environment, such as the thermal field.

The basic element of the logical structure in a RUNE-driven emulation experiment is the *space*. A space is an entity that behaves as any of the emulated devices, according to their function. Spaces can emulate any of the following:

– Nodes, that is physical devices such as active RFID tags;
– Environments, such as the thermal field in a room;

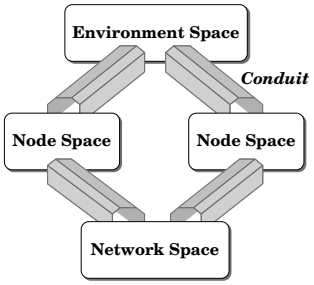

Fig. 6. Example of a RUNE-driven experiment topology.

– Networks, i.e., the communication media between nodes (for instance, the active RFID tag wireless communication).

Spaces are connected to each other by elements called *conduits*. The role of conduits is to behave as an abstract error-free communication pipe between two spaces, in a manner that is transparent to the user. Conduits play an essential role in the concurrent execution support of RUNE, since they make it possible to execute spaces on the same StarBED host, or on different hosts, without any modification. Note that, in order to emulate real communication channels, in which network degradation effects, such as frame errors and packet loss, delay and jitter, and bandwidth limitations occur, an appropriate network space needs to be used in-between the communicating spaces so as to reproduce these effects.

Figure 6 shows an example of a RUNE-driven experiment topology. In the figure we show two *node spaces* that play the role of any two ubiquitous computing devices. The *environment space* is in charge of emulating the physical environment in which the nodes are located in the virtual space. The communication conditions between the two devices are reproduced by the *network space*. In the emulation framework that we present in this paper we employ a very similar topology, with node spaces being the active RFID tags (effectively the PIC emulator running the active tag firmware), and QOMET components making up the network space that recreates the communication conditions between active RFID tags. For more details see Section 3.4.

In RUNE-driven experiments, there are two software modules that manage how the experiment takes place. On the PC that controls the progress of the experiment runs the module called *RUNE Master*. This module initiates the execution of all spaces deployed on multiple hosts. Another special module, *RUNE Manager*, is deployed on every emulation host, and

mediates the communication between them. Spaces implementing emulation targets exist on emulation hosts in the form of shared objects, loaded dynamically by RUNE Manager using the operating system dynamic loading mechanism.

RUNE Manager is in charge of operations such as loading the space objects, calling entry points (functions) in spaces, and relaying the communication between spaces via conduits. In RUNE architecture, each space is required to have five entry points, since any typical system can be broken into five operations: initialization, execution step, finalization, read, and write.

Once execution is initiated by RUNE Master, the emulation process performed by RUNE is as follows. First, RUNE Manager loads the objects corresponding to spaces, then notifies RUNE Master of completion. After that, RUNE Master starts the initialization process of all spaces to RUNE Managers on each host. A space will then allocate its heap memory area, which is permanently needed for emulation execution, and will return its pointer to RUNE Manager. Spaces do not use the stack memory area, but only the heap memory allocated by themselves for the execution of emulation. This approach contributes to an efficient usage of memory, since each space object can emulate multiple instances of the same node on one host, while sharing the binary code. This approach also ensures that spaces are thread-safe, since they do not have any static data.

Once initialization of all spaces is completed, RUNE Master starts the iterated invocation of the execution step symbol. Iterations last until one of the spaces in experiment returns an exit status. When RUNE Master receives the status, it starts finalization by notifying the end of experiment to all hosts. Spaces release then the work area allocated in the initialization process. At this point the experiment finishes, and the user will gather result data and logs. For more details about RUNE-driven emulation execution see [13].

3.2. QOMET

QOMET is a wireless network emulator that was initially dedicated to IEEE 802.11 networks [3], and was later extended to support other wireless technologies, such as ZigBee, and the wireless communication between active RFID tags that we present here.

QOMET employs a scenario-driven architecture with two stages, as it is depicted in Figure 7. In the first stage, from a real-world scenario representation, QOMET computes the network quality degradation (ΔQ) description that corresponds to the real-world

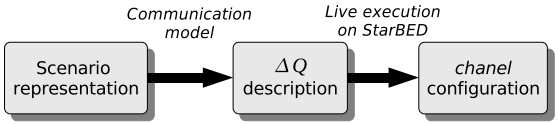

Fig. 7. Two-stage scenario-driven architecture for active RFID tag communication emulation.

events. This computation is done by a module called *deltaQ*. In the second stage, the ΔQ description is applied during the live experiment execution by the library called *chanel* (communication CHANnel Emulation Library). Thus, the wireless communication conditions between active RFID tags are recreated in the wired network during the effective emulation process.

A QOMET scenario consists of an XML-based description of the situation that must be emulated. The scenario representation will indicate, for example, the initial position of the active RFID tags, their motion pattern, the topology of the virtual environment which is being reproduced (such as buildings), and so on. This information is used to compute the communication conditions between any two given active RFID tags at each moment of time by using a specific communication model that will be presented next.

3.2.1. deltaQ *library*

The QOMET *deltaQ* library already provided support for IEEE 802.11 network emulation. However, the active RFID tags use a different and much simpler wireless communication technology and protocol, therefore *deltaQ* has been extended to support active RFID tag communication emulation as well.

For the purpose of testing by emulation the active RFID tag system developed by Panasonic, the communication model must only approximate the real communication conditions between active tags, since the main aspects to be tested are at protocol and firmware implementation level. Therefore, for simplicity purposes, we decided to use a model that establishes the relationship between the distance that separates two active RFID tags and the average frame error rate (FER) that occurs during communication. This model is based on measurements made by Panasonic in an RF shielded room while using a helical antenna, the same with the one used in the practical experiment. The measurements were done by using probe frames having a payload of 4 bytes. During measurements, the distance between sender and receiver was varied starting at 0.5 m, in intervals of 0.5 meters, until FER became 1. For each distance, 100 probe frames were sent and the number of successfully received frames was

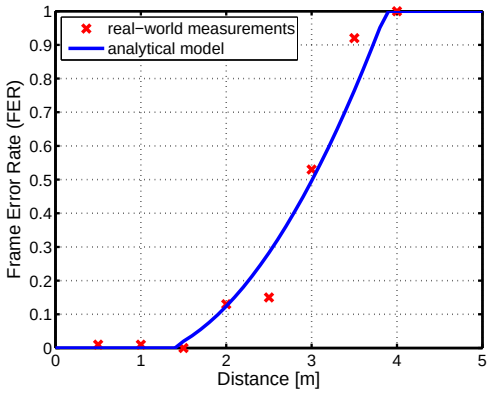

Fig. 8. Comparison of the active RFID tag communication analytical model with real-world measurements.

counted. This number was used to compute the empirical success and error rates for each distance (see Figure 8).

By fitting a second degree equation on the measurement results, we obtained the following formula for the frame error rate approximated by fitting, FIT:

$$FIT(d) = 0.1096d^2 - 0.1758d + 0.0371, \quad (2)$$

where d represents the distance between the communicating active RFID tags. This formula has a goodness-of-fit coefficient, R^2, equal to 0.9588.

Given the physical constraints on the frame error rate, $0 \leq FER \leq 1$, and to explicitly show the dependency of frame size, we write the following equation for the relationship of frame error rate versus distance when using 4-byte frames. The resulting model is plotted in Figure 8 for comparison against the real-world measurements it approximates.

$$FER(d,4) = \begin{cases} 0, & \text{if } d < 1.5m \\ 1, & \text{if } FIT(d) > 1 \\ FIT(d), & \text{otherwise} \end{cases}. \quad (3)$$

We can generalize Eq. (3) to consider the case of frames with payload size other than 4 bytes, as follows:

$$FER(d,S) = 1 - (1 - FER(d,4))^{\frac{H+S}{H+4}}, \quad (4)$$

where S is the frame payload size in bytes, and H is the frame header size, also expressed in bytes. The actual values for the active RFID tag system that we emulated are $S = 7$ and $H = 6$ bytes.

One of the purposes of designing and implementing the emulation framework was to make it possible to run experiments with the active RFID tag system in a wide range of emulated conditions. For this purpose we extended Eq. (3) by allowing users to specify a scaling factor for the communication range, denoted by C. Using this scaling factor it is possible to evaluate the performance of the system by emulation experiments in situations other than the default communication range of the active RFID tags. In principle range may also be changed in reality by varying the transmission power of the active tags. Since transmission power is related both to communication range, hence system performance, and to battery consumption, we investigated this relationship by emulation, as it will be discussed in Section 4.3.1.

A simple way to calculate the dependency between FER and distance for a certain scaling factor C is by replacing d with $\frac{d}{C}$ in Eq. (3). The resulting formula is given next:

$$FER_C(d,4) = \begin{cases} 0, & \text{if } \frac{d}{C} < 1.5m \\ 1, & \text{if } FIT(\frac{d}{C}) > 1 \\ FIT(\frac{d}{C}), & \text{otherwise} \end{cases}. \quad (5)$$

To compute FER for frames with arbitrary payload size, while enabling scaling of the communication range, we replace FER by FER_C in Eq. (4) to obtain:

$$FER_C(d,S) = 1 - (1 - FER_C(d,4))^{\frac{H+S}{H+4}}. \quad (6)$$

In Figure 9 we plot Eq. (6) for several values of the parameter C. The values used for the other parameters are the actual values of the active RFID tag system that we emulated, i.e., $S = 7$ and $H = 6$ bytes. Note that, to simplify presentation, we sometimes use the term communication range for the distance at which the frame error exceeds 0.6, since we consider this to be the effective range over which the RFID tags can successfully communicate with high-enough probability. However, this is just a convention when presenting the results, since during emulation experiments we always use the actual FER computed by Eq. (6) to recreate the communication conditions between the active RFID tags.

Note that the equations presented so far only account for communication in environments without obstructions. QOMET supports the use of real map data to define realistic virtual spaces, including buildings and streets. The streets are mainly used to constrain motion, so as to recreate pedestrian movement patterns similar to those occurring in reality. Buildings

Reprinted from Journal of Ambient Intelligence and Smart Environments, 2(2), Razvan Beuran, Junya Nakata, Tetsuya Kawakami, Takashi Okada, Ken-ichi Chinen, Yasuo Tan, Yoichi Shinoda, Emulation framework for the design and development of active RFID tag systems, 155-177.

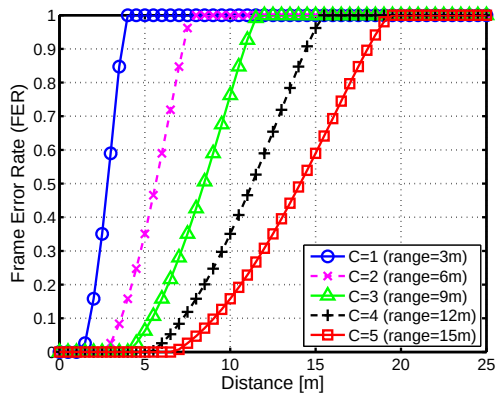

Fig. 9. Emulated active RFID tag frame error rate versus distance, depending on the scaling factor C.

constrain motion as well, but they also interfere with the active RFID tag communication. Therefore, large obstacles such as buildings must be taken into account by the wireless communication model. Given the low transmission power and short range of the emulated tags, we can ignore complex propagation aspects such as multi-path fading and absorption, and make a simplifying assumption. Thus, we assume in our model that only line-of-sight communication is possible. Hence the FER is equal to 1 (i.e., no communication) for tags between which no line-of-sight exists because of building obstructions, otherwise Eq. (6) is used, as discussed previously. Line-of-sight calculations are done by taking into account the position of the tags and the shape and position of the buildings present in the virtual environment that we emulated.

Some further remarks about the wireless communication model presented in this section are necessary:

- The basic component of the model, Eq. (2), is obtained by fitting a curve on measurement results done in an RF shielded room; as the goodness-of-fit coefficient indicates, it models sufficiently accurately the dependency between FER and distance under those conditions;
- Although we don't take into account advanced radio propagation issues such as noise, or fading, the model that we propose serves well enough the main goals of the project: test the system prototype, and extend the active RFID tags experiments done by Panasonic. This is demonstrated by our experimental results in Section 4;
- The scaling factor introduced in Eq. (5) allows testing conditions other than those given by the basic model. Moreover, the modular architecture of QOMET makes it possible to replace the current model with a more accurate one, should this be deemed necessary in the future.

3.2.2. *Collision probability*

A feature of the communication protocol used by the active RFID tags that we emulated is that the communication slots used for sending messages between two tags are selected at random, and independently, from the range of slots reserved for this purpose. As a consequence, two or more tags can select the same slot for communication, leading to collision between packets, hence to frame errors.

The probability of having such a collision, FER_{coll}, depends on the total number of slots used, denoted by N_S, and the total number of nodes in the interference range of a node (including itself), denoted by n. We consider a node to be in the interference range of another node if the FER for the communication between them is less than 1.

Let us calculate first the frame success probability in collision conditions, denoted by FSR_{coll}. In order for a frame to be successfully received, all the other senders in the interference range must select a different slot than the one selected by the current node. This gives the equation:

$$FSR_{coll} = \left(\frac{N_S - 1}{N_S}\right)^{n-1}. \quad (7)$$

The frame error rate due to collision, FER_{coll}, can easily be computed from FSR_{coll} as:

$$FER_{coll} = 1 - FSR_{coll}. \quad (8)$$

Figure 10 shows the variation (from the point of view of one node) of the frame error rate due to collisions versus the number of nodes, depending on the number of slots used in the communication protocol. One may observe that for 9 communication slots, the value used in the prototype localization system, collision-induced FER is expected to exceed 0.5 for as few as 6 nodes. This lead us to believe that the system may have a poor performance in situations when many tags are in the same region. Our intuition was confirmed by emulation experiments, as it will be shown in Section 4.4.

The total frame error rate caused both by the distance between active RFID tags and by collision, FER_{total}, can subsequently be computed as the total probability of two non-disjoint events as shown in Eq.

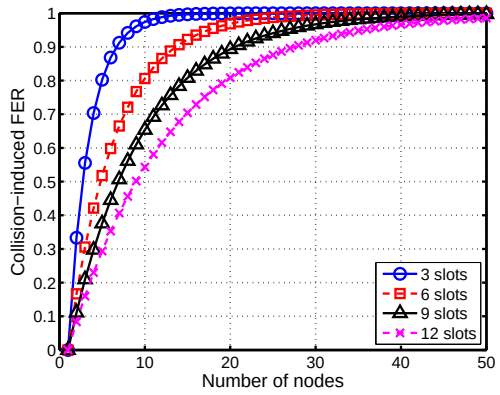

Fig. 10. Collision-induced FER versus the number of nodes, depending on the number of slots used in the communication protocol.

(9). Note that this assumption is not always true, as it does not account for the possibility that loss of a frame may lead to a successful transmission of another frame (i.e., if the collision event doesn't take place anymore). Nevertheless, Eq. (9) does provide an upper boundary on the total frame error rate, which was the most important aspect for our evaluation.

$$FER_{total} = FER_C(d,S) + FER_{coll} - FER_C(d,S) \cdot FER_{coll}. \quad (9)$$

The model given by Eqs. (7) and (8) is first of all important from a theoretical point of view, and as a design tool, for estimating the frame error rate due to collision that is to be expected during system operation. The model could also be used in experiments, as a simple way to account for this effect. However, at the request of Panasonic, we decided to determine in real time whether two or more frames collide as follows. Upon receiving a frame, the emulated active RFID tags will check whether another frame was received within the same time slot. If this is the case, then a collision event is decided, and the received frames are considered as errored; otherwise the reception is considered successful. Although this approach is more intensive from a computation point of view, it was favored due to its determinism. Thus, in the experiments we present in this paper, the collision probability FER_{coll} is not used in Eq. (9), which becomes:

$$FER_{total} = FER_C(d,S). \quad (10)$$

Nevertheless, modeling facilitated understanding the properties of the the pedestrian localization system, and helped designers make decisions for future development. The probability computed by Eq. (8) shows what is the collision probability from the point of view of a node. However, system designers were also interested in the overall probability that there is *at least* a collision for the tags located in a certain region, denoted by $OFER_{coll}$. This collision probability can be computed starting from the overall *complete* success probability (the probability that absolutely all transmissions are successful), denoted by $OFSR_{coll}$, as it will be shown next.

Firstly, in order to calculate $OFSR_{coll}$, one needs to consider the probability of the event that all nodes in an interference range select a different slot for communication (we make here the simplifying assumption that the interference ranges of all nodes in the considered region are identical, so as to obtain the worst-case scenario). The formula that results is:

$$OFSR_{coll} = \frac{(N_S-1)(N_S-2)\ldots(N_S-n+1)}{N_S^{n-1}} = \frac{(N_S-1)!}{(N_S-n)!} \cdot \frac{1}{N_S^{n-1}}, n \leq N_S. \quad (11)$$

As it is obvious that the complete success probability will be 0 if the number of nodes in the interference range exceeds the number of slots, we define:

$$OFSR_{coll} = 0, n > N_S. \quad (12)$$

$OFER_{coll}$ is calculated with reference to the value of $OFSR_{coll}$ that is given by Eqs. (11) and (12), as follows:

$$OFER_{coll} = 1 - OFSR_{coll}. \quad (13)$$

Figure 11 shows the variation of the overall frame error rate due to collisions versus the number of nodes, depending on the number of slots used in the communication protocol. It can be seen that for the case of 9 slots, as are used by the pedestrian localization prototype, error probability exceeds 0.5 starting from 4 nodes. Even if the number of slots is increased, the error probability increases quickly, therefore this is not a solution for improving system behavior in crowded areas. The experimental results in Section 4.4 demonstrate the poor performance in congestion conditions of the simple time-multiplexed protocol used by the active RFID tag system prototype. Panasonic Corpo-

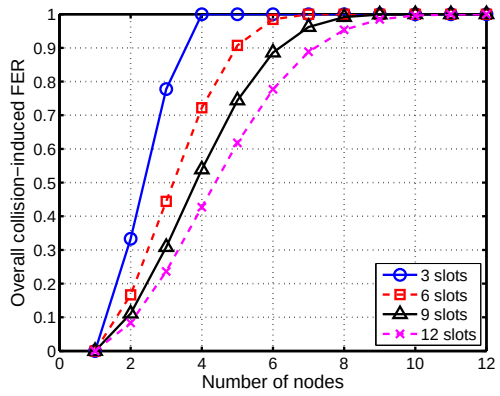

Fig. 11. Overall collision-induced FER versus the number of nodes, depending on the number of slots used in the communication protocol.

```
// reconfiguration thread
do forever
  retrieve next deltaQ configuration
  wait for next time step
  apply new deltaQ configuration
```

Fig. 12. Pseudo-code of the reconfiguration thread.

ration is now enhancing the protocol with collision avoidance features, so that large-scale situations can be better handled.

3.2.3. chanel *library*

During an emulation experiment, the main role of *chanel* is to recreate the scenario-specific communication conditions between active RFID tags using the ΔQ description computed by the *deltaQ* library. To achieve this goal, the *chanel* module is inserted between the space emulating the active RFID tag itself, and its connection to the other active RFID spaces by means of RUNE conduits. The algorithm of the reconfiguration thread is given in pseudo-code in Figure 12.

A second function of *chanel* is to make sure that the data sent by one active tag is communicated to all the active RFID tags 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 RFID tag and the other tags. By checking whether the current FER for a certain destination is less than 1 or not, *chanel* decides whether to forward the packet to that destination or not. We also use r, a pseudo-random number in the interval $[0, 1)$, to decide whether the frame which is being sent should be errored or not. The algorithm is executed as a parallel thread with that for reconfiguration; the corresponding pseudo-code is shown in Figure 13.

```
// transmission thread
do forever
  if frame must be sent
    for all destinations do
      if FER<1
        if r<FER
          set error flag
        send frame to destination
```

Fig. 13. Pseudo-code of the transmission thread.

In *chanel*, we use unicast transmissions to send the frames only to those destinations for which error probability is different than 1. Since the effective data rate is very low (2400 bps), and the number of nodes in a certain area is generally expected to be low, the total amount of traffic generated will not be very significant. If the case of many nodes in an interference range is considered, then Eq. (8) can be used to compute the collision rate. This will result in a high FER_{total} as given by Eq. (9), thus effectively bounding the number of nodes to which *chanel* needs to send the frames by unicast. If we use the value 0.9 as threshold on FER for deciding to whom packets are forwarded, and we consider 9 slots available for communication, then the maximum number of neighbors *chanel* has to forward a frame to is 20 (cf. Figure 10). The 2400 bps traffic from one node will result in around 50 kbps total traffic, which doesn't represent a large load in the 100 Mbps experiment network. We conclude that this approach is feasible even in large scenarios and when running multiple active RFID tag instances on the same StarBED host.

3.3. PIC 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 be able to execute the active RFID tag firmware on our system without any modification or recompilation, we emulated the active RFID tag micro-controller, which is a PIC processor.

Processor emulation in our framework had to take into account the following aspects that we implemented in the PIC emulator module:

Instruction execution All 35 PIC instructions of the active RFID tag micro-controller are supported.

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

Interrupts All interrupts necessary for the active RFID tag application (*timer0*, *timer1*, and *timer2*) are supported. At the moment, other interrupts which are not currently used by the active RFID tag firmware, such as comparator interrupt or USART TX/RC interrupt, are not supported.

When implementing the PIC emulator, we made two changes compared to the real system, one meant to improve execution performance on the StarBED hosts, and one to compensate a fault in the random number generation of the active RFID tag prototype system, as follows:

- Our PIC emulator supports not only access to the whole memory area of the tags, but also serial-to-parallel conversion. This makes it possible for *chanel* to send and receive sequences of bytes, although the data is sent and received in a bit-by-bit manner by the real active RFID tags. This feature was added for optimizing data transfer at PC level.
- We had to introduce support for software pseudo-random number generation so as to compensate the original active RFID tag firmware fault in random number generation that will be discussed in Section 4.2. Thus, the PIC emulator provides on request to the active tag firmware a pseudo-random number generated on the host PC, instead of using the corresponding active tag function call.

When emulating active tag applications such as ours it is important to introduce cycle-accurate processor emulation. In the pedestrian localization system, the active RFID tags use the time information contained in the exchanged messages to synchronize with each others autonomously. Incorrect time information may lead to artificial desynchronization problems, and potentially to communication errors, therefore it must be avoided. Timing is determined by how accurate the PIC emulator reproduces the execution speed of the active RFID tag micro-controller, especially in the case when running multiple instances of the emulator on the same StarBED host. In Figure 14 we show how emulation accuracy changes with the number of instances of the PIC emulator that are run in parallel. Experiments were performed on two operating systems from the FreeBSD family, and for 4 different processor frequencies. The PC labeled "FreeBSD 5.4, 1 GHz", is one of the StarBED hosts effectively used in the experiments presented in this paper.

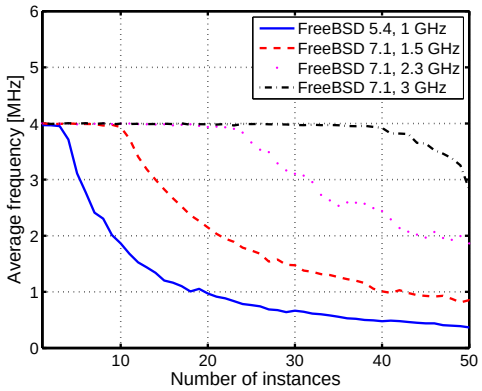

Fig. 14. PIC emulator accuracy for 4 combinations of operating systems and processor frequencies.

Figure 14 shows that even on a low-spec platform such as FreeBSD 5.4 1 GHz PC it is still possible to have 3 instances of the PIC emulator running in parallel at the correct frequency of 4 GHz. Performance improves quickly with processor frequency, as the operating system overhead becomes less and less important when more instances are run simultaneously. Thus, almost 40 instances of the PIC emulator could be run in parallel if experiments would be executed on FreeBSD 7.1 3 GHz PCs. Note that for the experiments presented in this paper we only used one active RFID tag instance per PC host.

3.4. Conceptual design

The conceptual architecture of the active RFID tag emulation framework that we designed and implemented is depicted in Figure 15, showing how the three components mentioned so far, RUNE-assisted StarBED, QOMET and PIC emulator are put together. As with all RUNE-driven experiments, RUNE Master is controlling the entire experiment execution. On each

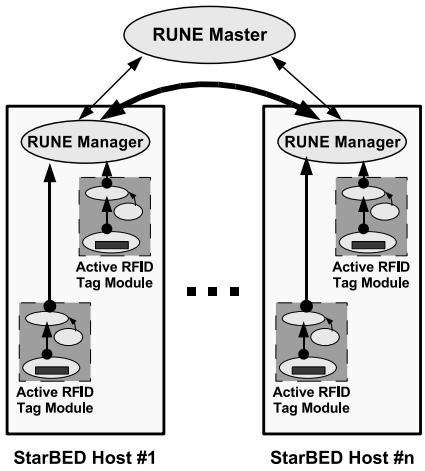

Fig. 15. Architecture of the emulation framework.

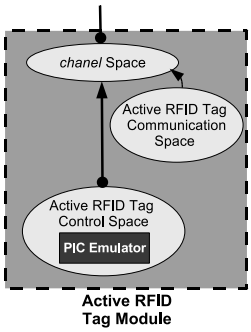

Fig. 16. Components of an active RFID tag emulation module.

PC participating to the experiment there is an instance of RUNE Manager, in charge of local execution.

Emulation is effectively run by one or more active RFID tag modules running on each StarBED host. Figure 16 details the components of each logical module in charge of the emulation of an active RFID tag. One can notice the active RFID tag *control space*, which executes an instance of the PIC emulator, and the active RFID tag *communication space* that is in charge of determining the communication conditions between the corresponding emulated active tag and all the other tags in the experiment (function powered by the QOMET *deltaQ* library). The communication conditions are provided to the other QOMET module, the *chanel* space, that will reproduce the communication conditions live on the testbed, as previously explained in Section 3.2.3. During an emulation experiment, the PIC emulator module will execute the active RFID tag firmware, thus reproducing the tag behavior in our framework.

Although we initially intended to run the experiments with the same speed as the real-world time (i.e., 1 emulated second lasts 1 real-time second), the jitter in the StarBED wired network prevented us from doing that. The reason is that each of the communication slots of the real active RFID tags has a duration of 53 ms. This value is only one order of magnitude higher than latency and jitter values in a wired network, which are in the order of milliseconds. As a consequence, there was a risk that network latency and jitter would perturb experimental results. For instance, two packets sent simultaneously from logical point of view may have physically arrived in different communication slots during emulation. To counter this problem we decided to execute the experiment 10 times slower than real-world time (i.e., 1 emulated second lasts 10 real-time seconds). In this way the actual real time duration of a communication slot in the emulation framework became 530 ms, two orders of magnitude greater than the undesired network effects. Note that even though most emulation experiments were not done in real time, they were still done live, with only a constant slow-down factor compared to real time. Therefore our framework differs in this respect from simulation, which uses purely logical time, and for which the duration of an experiment varies, depending on how many nodes are involved, the complexity of calculations, and so on.

4. Experimental results

Our emulation framework was used in connection with the pedestrian localization system developed by Panasonic Corporation for three main purposes:

- Emulate conditions similar to the real-world trial done by Panasonic so as to gain an insight into the dynamic behavior of the prototype localization system;
- Explore the parameter space of the active RFID tag system in order to make informed choices about parameter values that control the performance of the localization system;
- Extend the tested scenarios beyond what could be done by real-world trials so that the localization system performance can be assessed in large-scale situations.

Before discussing representative results for each of the three aforementioned research items, we will show a series of experiments that we did in order to validate

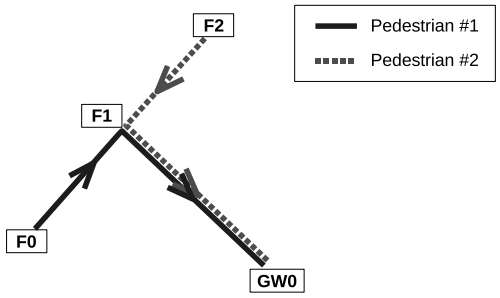

Fig. 17. Motion patterns for the participants in the 2-pedestrian experiment.

the most important aspects of the emulation framework. More precisely, for our current goals, the most critical issue was localization fidelity: how close are the localization results produced from emulation experiments to those produced from real-world trials.

4.1. Emulation framework validation

We present next a 2-pedestrian real-world trial that we carried out with the prototype localization system developed by Panasonic in an outdoor environment. The goal of the experiment was to validate our emulation platform by comparing the results obtained in the real-world trial with those of an emulation experiment that reproduced the real-world trial.

The scenario of the 2-pedestrian experiment is shown in Figure 17. In addition to the two pedestrians wearing mobile c-tags, the real-world trial also used three fixed c-tags, F0 to F2, and one gateway c-tag, GW0. The pedestrians #1 and #2 moved simultaneously from the fixed c-tag locations denoted by F0 and F2 towards the meeting point F1, then continued the movement to the gateway GW0.

The movement pattern of the participants in the 2-pedestrian experiment is summarized in Table 3. Note that the pedestrians spend around 30 s near each of the significant points on their trajectory. This time was not explicitly indicated in the initial 16-pedestrian experiment performed by Panasonic, but it was observed that pedestrians spend time at the "milestone" locations to ensure communication between c-tags takes place correctly, and to adjust the timing of their movement. Therefore, we included explicitly this pauses in the 2-pedestrian experiment movement description. As a consequence, motion effectively takes place only in the intervals 30 to 60 s, and 90 to 120 s. Distances between F0 and F1, and F2 and F1 were of about 30 m; the distance between F1 and GW0 was of about 60 m. This means that pedestrians moved with a speed of about 1 m/s.

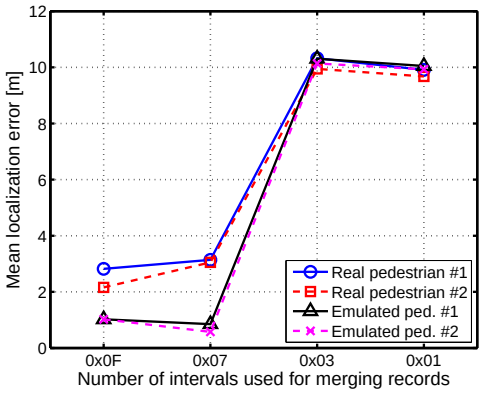

Fig. 18. Localization error per pedestrian in the 2-pedestrian experiment: real experiment versus emulation.

Table 3
Movement instructions for the participants in the 2-pedestrian experiment

Time [s]	Pedestrian #1	Pedestrian #2
0	F0	F2
30	F0	F2
60	F1	F1
90	F1	F1
120	GW0	GW0
150	GW0	GW0

In Figure 18 we plot the mean localization error computed for each pedestrian both in the real-world experiments (with 2 runs per setting), and in the emulation experiments (with 3 runs per setting). The parameter that we varied between experiments is the number of time intervals used by the active RFID tags when merging the identification information that is exchanged by tags, as it was explained in Section 2.1. The communication range used in emulation was 3 m (i.e., $C = 1$ in Eq. 5), the same with what we measured for the prototype system. For the real-world trials, the average localization error is computed as the mean of the distances at each moment of time between the position tracked by the pedestrian localization system, and the ideal position of the pedestrian in the real environment. For emulation experiments, the average localization error is computed as the mean of the distances at each moment of time between the position tracked by the pedestrian localization system, and the pedestrian position in the virtual emulation environment.

According to Figure 18, the agreement between the experimental results obtained by real-world trials and by emulation is good. The largest differences are less than 1 m, sufficiently small compared to the size of the experiment area, which was around 60 x 60 m. Moreover, when judging differences one has to also take into account the intrinsic motion inaccuracy of the real pedestrians, which we could not completely control. The average localization error differences between the real-world and emulation experiments are more significant for the values 0x0F and 0x07 of the number of intervals used for merging. This coincides with the situations in which the localization errors are smaller, as the correctly tracked trajectories are longer. We remind that a smaller numbers of intervals used for merging leads to the loss of records through erasure, since memory becomes full as the tags continue to exchange messages while walking towards their destination. Given that the correctly tracked trajectory is shorter, the error will be higher (see merging interval values 0x03 and 0x01 in Figure 18). As an indirect consequence, this produces a better match between real-world trial and emulation, since the incertitude of the real-world experiment becomes lower.

After we analyzed the results of such experiments, and built the required confidence in the correctness of the emulation framework, we proceeded to use it to investigate the properties of the localization system prototype by a series of experiments that will be presented in the next section.

4.2. Prototype system analysis

In order to validate the pedestrian localization prototype system, especially the firmware running on the active RFID tags, we performed several emulation experiments that reproduce the conditions of the 16-pedestrian experiment carried out by Panasonic.

We first analyzed how the localization error, the most important parameter characterizing the pedestrian localization system, differs between the real-world trials done by Panasonic and the results on our emulation testbed. In Figure 19 we compare the localization error per pedestrian from the real-world trial with the mean localization error per pedestrian from emulation experiments. This mean was computed for 3 emulation experiments done in conditions that mimic most closely the real system: 3 m communication range and 9 communication slots. We also plot the standard deviation of the emulation data as error bars. Although for most tags the deviation from average is small, a few of them, such as 3, 14, and particularly 16 have large values. On one hand this is related to the short communication range, which leads to few communication opportunities, as explained later in 4.3.1. In addition, there are the aggregated effects of various factors such as: pedestrian trajectory, the use of real time (compared to the perfectly deterministic logical time employed in simulations), and probabilistic components of the systems, such as the selection of the communication slot and the frame error probability. For the tag with id 16 we could confirm that, during one of the three emulation experiments, it was unable to communicate on multiple occasions, which made the localization error for that experiment be very high, although in the other experiments it had reasonable values. This explains the unusually high standard deviation.

Both for the real-world trial and emulation, localization error is calculated with respect to the motion scenario, which was followed as closely as possible by the pedestrians in the real-world trial, and was precisely reproduced in the virtual environment of the emulation experiments. It can be seen from Figure 19 that the localization error follows the same trend in both types of experiments, with the minor exception of pedestrian with id 16 that was discussed above. The trend is explained by the intrinsic properties of the motion scenario: depending on the trajectory of a tag relative to the other mobile and fixed tags, the precision with which it can be localized changes, and it will have a positive lower bound different than zero even in ideal conditions.

The fact that the results of emulation have the same trend with those in real-world trials shows that the emulation system works correctly, albeit closer to the ideal minimum localization error. The difference in absolute value between the two types of experiments comes from the many uncertainties referring to the real-world trial, mainly to the fact that motion cannot be fully controlled: it is impossible for all the real pedestrians to move with constant speed, and leave and reach the motion milestones at exactly the moment of time indicated in the scenario; thus reproducibility is low. In this sense, one can consider emulation as a best-case scenario, in which ideal motion takes place, hence the localization error will always be closer to the lower bound than that of a real-world trial. Still, the difference between the average localization errors for all pedestrians (28.71 m for the real world trial, and 16.31 m for the emulation experiments) is only of about 12 m, which represents around 5% of the largest distance

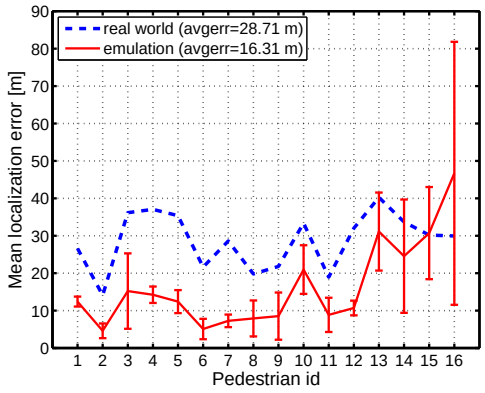

Fig. 19. Localization error in real-world trial versus emulation.

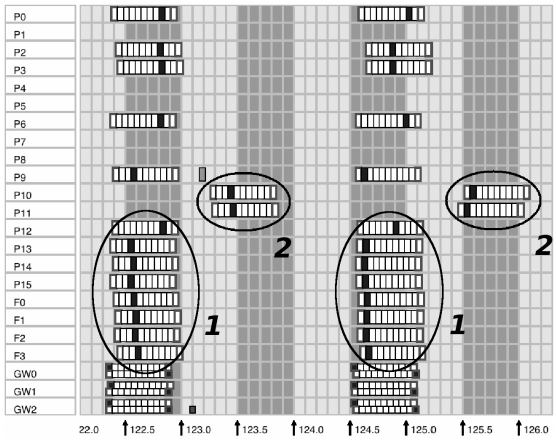

Fig. 20. Screenshot of our visualization tool for active RFID tag communication that emphasizes two issues we identified using the emulation framework.

in our virtual environment. This value was considered as sufficiently low, and validated the emulation testbed.

Through the analysis of the experimental data, we were also able to identify two issues with the active RFID tag based system, as follows:

1. A fault in the random number generation implementation;
2. A time synchronization issue.

These problems were both identified by using a visualization tool that shows the communication between the emulated c-tags. The graphical representation is constructed based on the log files that are produced by the PIC emulator module while the experiment is running. The visualizer shows for each c-tag the period in which it is available for communication as a sequence of white cases, and the slot effectively used for communication is marked with black.

In Figure 20 we show a screen snapshot emphasizing the two issues we identified (marked by the numbers “1” and “2”, respectively). P0 to P15 denote the mobile c-tags representing pedestrians, while F0 to F3 and GW0 to GW2 are the fixed c-tags and gateways, as explained in Section 2.2. One can note that pedestrians P13 to P15, as well as the fixed tags F0 to F3, always select the same slot for communication (problem 1). This is because the random number generation was not implemented correctly on the active RFID tags. The negative effect of this firmware issue was an artificially-induced high collision rate between c-tags. Following our observations, Panasonic has corrected the firmware and has fixed this problem.

Figure 20 also demonstrates problem 2, the time synchronization issue. It can be observed that tags P10 and P11, instead of being synchronized with all the other tags, have a different alignment, which means they use a different internal time (shown on the 0x axis). The cause of this issue was that these two particular tags had the same trajectory during the most part of the experiment. While moving, the tags continuously communicated with each other, exchanging ID messages. These messages are also used by c-tags to synchronize their internal clocks. The algorithm used in the prototype system lead to a continuous clock drift, as each of the tags would try to speed up to catch up with the other one. Panasonic has revised the algorithm so that this issue doesn’t occur in the next version of the system.

4.3. Parameter selection

One important use of the emulation framework that we designed and implemented is for system parameter selection. In what follows we shall illustrate how our system can be used to determine optimum values for the following parameters of the pedestrian localization prototype system:

– Transmission range;
– Number of communication slots;
– ID record merging period.

4.3.1. Transmission range

Transmission range is an important parameter of the localization system since it determines the average distance over which c-tags can communicate with each other. Transmission range mainly depends on the antenna and transmission power of the active RFID tags;

therefore it can be configured in reality as well. However, it may be difficult to perform many experiments while changing such hardware parameters. In such circumstances the approach of emulation simplifies considerably the task of experimentally evaluating different conditions.

One may imagine that the longer the transmission range the better. Although this may be true in some applications, it is not true in general. Considering our pedestrian localization application, we can make the following observations. A small transmission range will ensure that c-tags do not interfere with each other over large distances, and that the accuracy of localization is high (since the active tags will effectively be in almost the same location when they communicate with each other). On the negative side, a short transmission range may lead to the fact that tags fail to communicate with each other. Considering the approximately 2 second interval between the sending of ID records, and the average pedestrian speed of 1 m/s, it means that the tags will move by about 2 meters in the interval between sending two ID packets. This results in the fact that, for a communication range of about 3 m, there will be at most 3 chances to exchange ID records, but probably only 1 or 2 in real situations, especially if both tags are moving.

On the other hand, a long transmission range increases the active RFID tag communication probability. Nevertheless, in this case also there are negative effects: too long a communication range will lead to a large interference area around each tag; moreover, localization accuracy will decrease, since, even if the tags are not very close to each other, they may still be able to communicate.

These intuitions could be easily verified by emulation experiments. In Figure 21 we show the results of emulation tests we did for 5 different transmission ranges using the same scenario with the 16-pedestrian experiment. For each range 3 tests were carried out, and we used error bars equal to the standard deviation of the data to show the variation in the results. We remind that we compute the average localization error as the mean of the distances at each moment of time between the trajectory tracked by the pedestrian localization system, and the emulated pedestrian trajectory in the virtual environment.

The results in Figure 21 indicate that a range of 9 m is the most appropriate for the investigated scenario. As explained above, this is because shorter ranges lead to communication being unsuccessful on occasions, hence to localization errors. Meanwhile, larger ranges

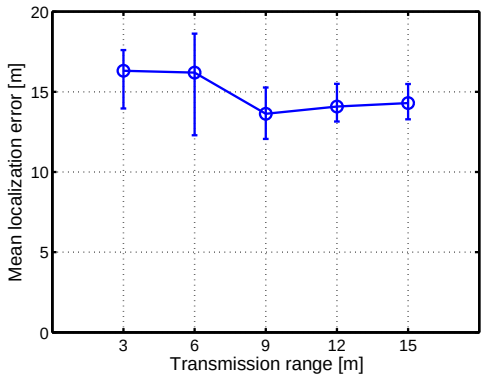

Fig. 21. Mean localization error versus transmission range.

decrease the accuracy in identifying the position of a tag, since the area in which tags can communicate with each other is large.

4.3.2. Number of communication slots

Another series of emulation experiments was made to study the effect of the number of slots used in the time-multiplexed communication on the overall localization performance in the 16-pedestrian experiment. The prototype localization system uses 9 slots for communication, preceded and followed by one guard slot. For two communication ranges, 3 and 9 m, respectively, we configured the active tag firmware to use 3, 6, and 9 communication slots. The advantage of using a smaller number of slots is that the active duration of the RFID tags is decreased, and therefore battery life potentially increases by 30% (when using 6 slots), or even by more than 50% (when using 3 slots). The disadvantage is that, when several tags want to communicate with each other, a smaller number of slots leads to a high collision rate, and impedes information exchange (cf. discussion in Section 3.2.2).

For each value of the number of slots we did 3 runs, and the results are plotted in Figure 22. One can notice that there is a performance gain when increasing the number of slots, but this gain is not so evident for the 3 m range, since the communication opportunities are reduced compared to the 9 m range. For the latter range, the advantage of using more than 3 slots is quite visible, and an optimum performance level seems to be reached for 6 slots already.

4.3.3. ID record merging period

One other parameter of the localization system is the number of intervals (active durations) used when merging ID record information, as it was already mentioned in 2.1. We shall use the emulation experiments

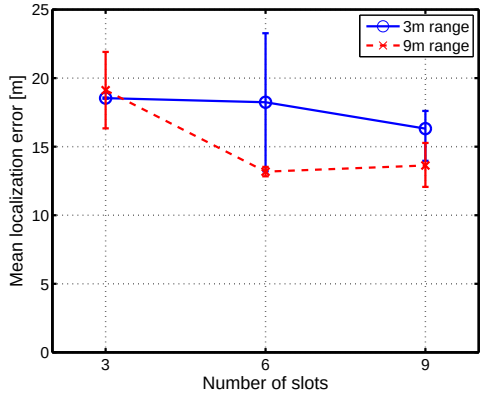

Fig. 22. Mean localization error versus the number of communication slots for two transmission ranges.

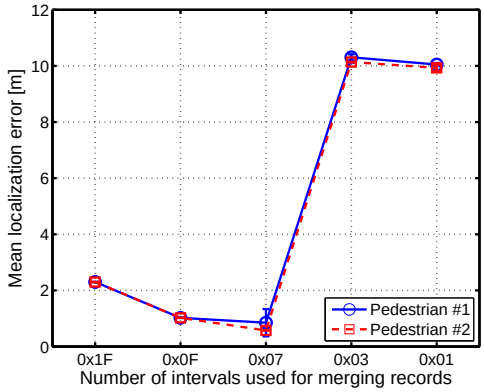

Fig. 23. Mean localization error versus number of intervals used for merging ID records.

done for the 2-pedestrian experiment to illustrate the use of our framework for analyzing this parameter.

We performed experiments with five different values of the number of intervals used for merging (three runs per case). In Figure 23 the mean localization error is shown for each pedestrian. Error bars equal to the standard deviation are used for each point, but given the high stability of the results the error bars only become visible in the case of the value 0x07. The best localization error is achieved precisely for this value, 0x07, i.e., when seven intervals are used for merging (totaling a period of around 15 s). However, given the fact that the results seem less stable at that point, one may choose the value 0x0F instead (15 intervals) as the optimum value, since the difference in localization error is not significant and stability is greatly improved.

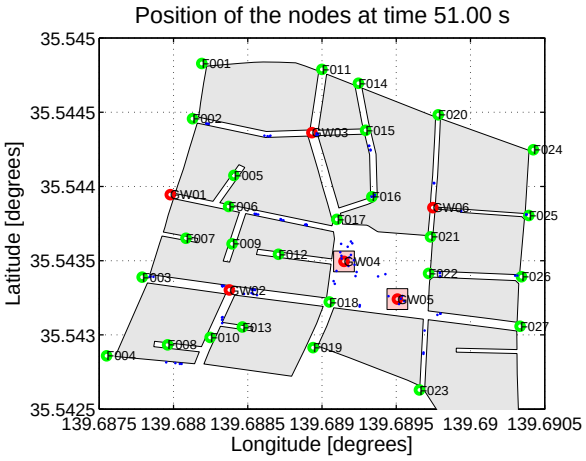

Fig. 24. Scenario with 100 pedestrians in a realistic topology based on geographic data.

4.4. *Large-scale experiments*

Perhaps one of the most important motivations for developing the emulation framework presented in this paper was running large-scale experiments that cannot be easily executed in the real world. In this context we ran several experiments with groups of 50 to 100 pedestrians. For these experiments, the movement of the pedestrians in the virtual space was automatically generated by using the behavioral motion model, one of the realistic mobility models supported by QOMET. This model creates a trajectory between a specified source and destination by taking into account constraints such as movement speed and the topology of the area (buildings, roads, etc.). In our experiments the topology of the virtual space is based on real geographical data for a region in Kawasaki, Japan, provided by the GSI (Geographical Survey Institute) of Japan. Figure 24 illustrates the topology of the virtual space and the location of active RFID tags for a 100-pedestrian experiment that also includes 6 gateways and 27 fixed c-tags. The pedestrians depart from the locations of the fixed c-tags, and head towards one of the two destinations located in the center of the area, nearby gateway c-tags GW04 and GW05.

Although we performed several series of 100 pedestrian experiments, due to an issue related to the active RFID tag communication protocol we were not able to obtain the localization results in the same way we did for the other smaller-scale experiments. The issue was the following: in the 100-pedestrian experiment, the number of tags that reach the destinations located around GW04 and GW05 is large compared to the pre-

vious experiments, therefore there are many collisions between the mobile tags as they try to upload their information (cf. discussion in Section 3.2.2). This reason leads to the fact that many tags don't manage to upload enough information to the gateways before the experiment ends.

Table 4 gives some statistics related to the status of the emulated active RFID tags for one of the 100-pedestrian experiments. The first two lines summarize the fact whether the mobile c-tags succeeded to upload any data to a gateway during the experiment or not. The table shows that almost half of the mobile c-tags never succeed in uploading any information, although we confirmed that the rest of tags uploaded hundreds of packets. Not having at least one piece of data regarding all the tags in the experiment revealed one other issue: the current version of the localization engine was not able to cope with the case when only incomplete information is available. This meant that although 55% of the tags did upload information, no localization results were produced. This robustness issue with the software implementation of the localization algorithm is now being considered by Panasonic.

Table 4
ID record statistics for one 100-pedestrian experiment

Tag status when experiment ends	Tag count
Uploaded at least one record	55
Uploaded no record	45
Still has data to upload	91
Has no more data to upload	9

Table 4 also shows how many mobile c-tags still have data to upload at the end of experiment. According to the table, over 90% of mobile c-tags still had not uploaded all their data at the end of experiment. This further demonstrates the low probability in uploading data in the final part of the experiment, which is due to collisions as said before. Panasonic Corporation is now designing an enhanced communication protocol that includes collision avoidance, since our experiments have clearly shown that without such an algorithm the active tag localization system cannot function for relatively crowded areas. This is one of the important findings that were made possible by using our emulation framework.

5. Related work

To the best of our knowledge, there exists currently no system which has an identical functionality with our emulation framework. There are however several approaches and tools that are partially related to our research.

5.1. Experiment techniques

For testing and validating active RFID tag systems, the two predominant experiment techniques that are used are simulation and the real-world trials. The work in [6] examines the use of cellular phones with active RFID tags in the context of emergency evacuation. The authors use a map of an underground shopping mall and agent-based simulation to simulate the evacuation procedure. Several statistic assumptions are made related to pedestrians in evacuation scenarios, therefore the technique presented is very specific.

The authors of [7] propose an interesting approach that doesn't focus on active RFID tags but deals with the general case of AmI systems. The authors propose two methods: *participatory simulation*, where agents and human-controlled avatars coexist in a virtual space to jointly perform simulations, and *augmented experiment*, where an experiment is performed in a real space by human subjects, scenario-controlled agents, and human extras. Since it combines simulated and real elements, this approach is very similar to ours, but the practical realization is different and the focus is on modeling human behavior not the system itself. The case study is also related to evacuation situations, but the methods used for pedestrian localization are GPS systems or omnidirectional cameras.

A similar project is SensorRAUM [2], a project whose goal is to transfer the physical environment into a quasi-realistic virtual representation within a computer. This representation allows for a better sensory perception, and makes possible the interaction between real sensors and this virtual world.

5.2. Experiment tools

Experiment tools for studying ubiquitous systems, again mainly based on simulation are, for example, TOSSIM and ATEMU. TOSSIM [9] is a TinyOS mote simulator, intended to ease the development of sensor network applications. TOSSIM authors claim that it scales to thousands of nodes, and compiles directly from TinyOS code. This makes it possible for developers to test not only their algorithms, but also their implementations. Since TOSSIM simulates the TinyOS network stack at the bit level, both low-level protocols

and top-level applications that are being developed for TinyOS can be studied.

ATEMU [14] is a software emulator for Atmel AVR processor based systems. Along with support for the AVR processor, it also includes support for other peripheral devices on the MICA2 sensor node platform [5], such as the radio. ATEMU can be used to perform sensor network emulation studies in a controlled environment. ATEMU is compatible at binary level with the MICA2 hardware. The ATEMU emulator core can simulate large numbers of nodes, and can model their execution and interactions between them, such as radio communication. The only difference between running an actual network of the MICA2 sensor nodes and emulating it in ATEMU is communication media.

Since both TOSSIM and ATEMU are essentially simulators focusing on particular processors, we deem the realism of the experiments themselves as lower than that obtained with our system. One difference is related to the communication modeling, which is very basic (for instance, ATEMU only support free-space propagation). The other difference is that there is no guarantee related to how far from real time the execution will be, since everything is done in logical time.

The PIC micro-controller used in the active RFID tags that we emulated is manufactured by Microchip. Microchip provides two alternatives for system development [10]: real-time emulation in hardware by means of the MPLAB REAL ICE In-Circuit Emulator, or the PICMASTER Emulator; processor simulation by means of MPLAB SIM. Obviously none of this solutions can be used for large-scale experiments such as the ones we did.

5.3. Experiment testbeds

Emulab [16] is perhaps on of the most widely known testbeds. In addition to the wired and wireless network facilities, Emulab also has a sensor network testbed that includes 25 MICA2 motes. All motes are equipped with a serial port, for maximum control and debugging capability. Another sensor testbed is MoteLab [17], which consists of a set of permanently deployed sensor network nodes connected to a central server which handles reprogramming and data logging, also providing a web interface for creating and scheduling jobs on the testbed. Albeit both Emulab and MoteLab are controlled environments, the devices used are still real, therefore subject to potential interferences, and no mobility is possible.

Real-world experiment testbeds in which the wireless nodes are more strictly controlled than it can be done if the participants are humans are those of ORBIT and Mobile Emulab. ORBIT [18] uses a novel approach involving a large two-dimensional grid of 400 IEEE 802.11 radio nodes which can be dynamically interconnected into specified topologies as the experiment progresses. Mobile Emulab [8] uses robots to achieve reproducibility of the motion of the wireless nodes. However, since both these systems use real wireless equipment, the experiments are subjected to potential undesired influences. Range and speed of motion are also limited.

Closer to our approach are the wireless network emulation testbeds, mainly related to IEEE 802.11 networks. A system such as TWINE [20] uses computer models to perform real-time experiments. This makes it possible to avoid undesired interference and side effects. QOMET on StarBED used a similar approach, therefore we extended it to support the wireless communication of active RFID tags as well.

6. Conclusion

We presented an emulation framework that can be used in the design and development phase of active RFID tag systems. This framework was used to emulate an active RFID tag based pedestrian localization system that is being developed by Panasonic Corporation. Our framework extends the wireless network emulator QOMET to support the emulation of active tag wireless communication. We also implemented an emulator for the micro-controller of the active tags, named PIC emulator. The experiments were performed on the large-scale network testbed StarBED, by taking advantage of its experiment-support tools, SpringOS and RUNE.

In this paper we showed several instances in which the emulation framework was used to identify issues with the active RFID tag firmware, issues that were already or will be corrected by Panasonic as a new prototype system is being developed. We also demonstrated how such an emulation framework can be used to explore the parameter space of the system under study, so as to determine their optimum values under certain conditions. The large-scale experiments that we carried out with 50 and 100 pedestrians helped emphasize an issue with the communication protocol of the active tags. The simple protocol used leads to high collision rates in congested areas where many active RFID tags

are simultaneously present. Panasonic is now working on refining the communication protocol, by introducing a collision avoidance algorithm.

Future work on the emulation framework will focus on two main directions:

- Improve the scalability of the system so as to enable experiments with groups of pedestrians as large as 1000;
- Improve the realism of the active RFID tag wireless communication emulation by employing more accurate 3D models for the indoor virtual environments, as well as electromagnetic wave propagation models.

References

[1] J. C. Augusto and P. McCullagh, *Ambient Intelligence: Concepts and Applications*, in: Intl. Journal on Computer Science and Information Systems, Vol. 4, No. 1, 2007, pp. 1-28.

[2] M. Beigl, *SensorRAUM*,http://www.duslab.de/sensorraum/.

[3] R. Beuran, L. T. Nguyen, K. T. Latt, J. Nakata and Y. Shinoda, *QOMET: A Versatile WLAN Emulator*, in: Proc. of IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA 2007), Niagara Falls, Ontario, Canada, May 21-23, 2007, pp. 348-353.

[4] R. Beuran, J. Nakata, T. Okada, T. Kawakami, K. Chinen, Y. Tan and Y. Shinoda, *Emulation of an Active Tag Location Tracking System*, in: Proc. of Ambient Intelligence Forum (AMIF 2008), Hradec Kralove, Czech Republic, October 15-16, 2008, pp. 53-60.

[5] Crossbow Technologies, *MICA2 Wireless Modules*, http://www.xbow.com.

[6] M. Daito and N. Tanida, *Effectiveness of Cellular Phone with Active RFID Tag for Evacuation - The Case of Evacuation from the Underground Shopping Mall of Tenjin*, in: Intl. Journal of Humanities and Social Sciences, Vol. 3, No. 1, 2009, pp. 72-83.

[7] T. Ishida and H. Hattori, *Participatory technologies for designing ambient intelligence systems*, in: Journal of Ambient Intelligence and Smart Environments (JAISE), Vol. 1, No. 1, 2009, pp. 43-49.

[8] 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: Proc. of IEEE INFOCOM 2006, Barcelona, Spain, April 23-29 2006.

[9] P. Levis, N. Lee, M. Welsh and D. Culler, *TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications*, in: Proc. of the ACM Conf. on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, California, U.S.A., November 5-7, 2003, pp. 126-137.

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

[11] A. Misra and S. Das, *Location Estimation (Determination and Prediction) Techniques in Smart Environments*, in: Smart Environments: Technology, Protocols and Applications, D. Cook, S. Das (eds.), Wiley Interscience, 2004.

[12] T. Miyachi, K. Chinen and Y. Shinoda, *StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software*, in: Proc. of Intl. Conf. on Performance Evaluation Methodologies and Tools (Valuetools 2006), ACM Press, ISBN 1-59593-504-5, Pisa, Italy, October 2006.

[13] 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: Proc. of Intl. Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2007), Orlando, Florida, U.S.A., May 21-23, 2007.

[14] J. Polley, D. Blazakis, J. McGee, D. Rusk and J. S. Baras, *ATEMU: A Fine-grained Sensor Network Simulator*, in: Proc. of the IEEE Conf. on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara, California, U.S.A., October 4-7, 2004, pp. 145-152.

[15] L. Rizzo, *Dummynet FreeBSD network emulator*, http://info.iet.unipi.it/ luigi/ip_dummynet/.

[16] University of Utah, School of Computing, *Emulab - Total network testbed*, http://www.emulab.net.

[17] G. Werner-Allen, P. Swieskowski and M. Welsh, *MoteLab: a wireless sensor network testbed*, in: Proc. of Intl. Symposium on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, California, U.S.A, April 25-27, 2005, pp. 483-488.

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

[19] Ymatic, Inc., http://www.ymatic.co.jp.

[20] J. Zhou, Z. Ji and R. Bagrodia, *Twine: A hybrid emulation testbed for wireless networks and applications*, in: Proc. of IEEE INFOCOM 2006, Barcelona, Spain, April 23-29 2006.