StarBED2: Testbed for Networked Sensing Systems

Junya NAKATA, Razvan Beuran, Toshiyuki Miyachi, Ken-ichi Chinen
Satoshi Uda, Kenji Masui, Yasuo Tan, Yoichi Shinoda

National Institute of Information and Communications Technology, Ishikawa, Japan
Japan Advanced Institute of Science and Technology, Ishikawa, Japan

***Abstract*— Nowadays many new technologies are being developed and introduced for Internet, home networks, and sensor networks. The new technologies must be evaluated in detail before deployment. However the above mentioned networks have a large number of nodes, and a complicated topology. We developed a large-scale, realistic and real-time network testbed, StarBED, using hundreds of PCs, and switched networks. We are now implementing StarBED2, which expands StarBED so as to be suitable for emulating ubiquitous networks. In this paper we describe StarBED2, its design policy, architecture, and additional components, including the custom experiment-support system, RUNE (Real-time Ubiquitous Network Emulation environment). We then show some experimental results obtained in this environment with emulated networked sensing systems.**

I. Introduction

Today various kind of ubiquitous networks, including sensor networks and home networks, are researched more and more actively, or are already in use.

The network consisting of the Internet and ubiquitous networks is large and its behavior cannot be easily foreseen. In order to introduce new technologies to these networks, we have to evaluate the same implementations for the real environment using large-scale topologies, since we have to know their behavior in the real environment, including potential bugs. An example of the complex environment that we intend to investigate through emulation is shown in Fig. 1.

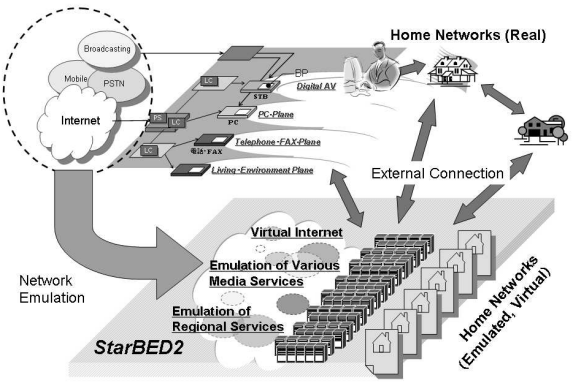

Fig. 1. Town network emulation

We implemented a large-scale network testbed, named StarBED. The current StarBED, however, cannot satisfy the requirements for evaluating implementations for ubiquitous networks, since in sensor networks and home networks the number of nodes can be huge and the nodes are naturally heterogeneous. For this purpose we designed StarBED2, by enhancing StarBED for realizing a large-scale ubiquitous network emulator. It enables emulation of ubiquitous networks with hundreds of thousands of heterogeneous nodes.

In this paper, we describe the requirements for emulating ubiquitous networks and existing methods for making experiments, then we explain the current StarBED, and the design of StarBED2 and its experiment-support software, RUNE. Next we show some illustrative experimental results.

II. Testbed Overview

A. Requirements for Emulating Ubiquitous Networks

Ubiquitous networks have different properties than computer networks in many aspects, such as: high node variability and network media variety, huge number of nodes and importance of the interaction with surrounding environment, etc. The required functionality for ubiquitous network testbeds is as follows:

1) Support the numerous nodes of ubiquitous networks. This can pose problems if emulation must be done under real-time constraints;
2) Emulate the surrounding environment and provide an interface between emulated nodes and environments;
3) Emulate various architectures of nodes and networks that form typical heterogeneous networks;
4) Provide an emulation support system that enables execution of experiments in a controlled manner through an automated execution mechanism.

B. Existing Methods for Experiments

There are already a number of implementations of emulators and testbeds for ubiquitous networks. TOSSIM [1] is a TinyOS simulator which aims to simulate TinyOS applications accurately in virtual environment. ATEMU [2] is also able to emulate TinyOS applications, and it has a more flexible architecture to support other platforms. MobiNet [3] is more a wireless network emulator than a testbed for ubiquitous networks. Such test environments satisfy only partially the requirements mentioned in the section II-A.

C. First Generation Testbed - StarBED

In StarBED many nodes are located on the same site. The environment is dividable and can be manipulated by several users simultaneously. The physical network of StarBED connects about 700 actual PCs, and makes it possible to build large-scale experiment topologies. All PCs have at least two interfaces connected to different network segments, the management network and the experiment network, so as to avoid interferences between management traffic and experiment traffic. To facilitate StarBED usage we designed SpringOS, a system that supports experiment execution. SpringOS executes experiments according to the configuration file. The details of SpringOS are described in [4] and [5].

D. Next Generation Testbed - StarBED2

In order to implement a testbed for ubiquitous networks that satisfies the requirements mentioned in section II-A, we are currently working on the implementation of StarBED2, a testbed for ubiquitous networks based on StarBED. The major aim of StarBED2 is to create an emulation environment in which various kind of nodes, networks, and environments can be emulated under real-time constraints.

StarBED2 is being constructed with StarBED architecture as basis. On top of StarBED, an instruction level emulation layer, a system call level emulation layer, and a middleware level emulation layer are being added. A layer is a sort of virtual machine within which various kinds of nodes with different architectures can work together during an emulation.

Obviously this physical architecture itself is insufficient for emulating ubiquitous networks. Therefore StarBED2 has a logical architecture as well. The main idea of the logical architecture of StarBED2 is a "*space*" and "*conduit*" structure. The elements of ubiquitous networks such as nodes, networks, and environments are described as *spaces*. Depending on what is implemented in a *space*, the *space* is classified into three categories: *node space*, *network space*, and *environment space*. The *spaces* interact with each other through *conduits*, executing their own process autonomously. The *conduit* is a communication channel which is set up between two *spaces*. Implementing *spaces* does not differ significantly from the development of usual programs, except that the StarBED2 API has to be used for *conduit* communication and real-time execution.

III. Implementation Details

In order to implement StarBED2, an experiment support software, RUNE, has been developed. The fundamental goal of RUNE is to implement an experiment environment in which a number of *spaces* that emulate each experiment target can work on either single or multiple nodes. RUNE provides a reasonably abstracted APIs for implementing *spaces* without much concern about the interaction between emulation nodes.

In RUNE architecture, "RUNE master" and "RUNE manager" play essential roles. RUNE master controls the progress of the experiment. The execution of all *spaces* deployed on multiple nodes is initiated by RUNE master. RUNE manager is deployed on every emulation node and mediates communication between them. *Spaces* implementing emulation targets exist on emulation nodes in the form of shared objects, loaded dynamically by RUNE manager using the operating system dynamic loading mechanism. Accordingly a *space* itself does not have context, but the RUNE manager instance on each node does. In RUNE architecture the shared objects that implement *spaces* are called *space* objects. Each function of *space* objects are invoked by RUNE manager when needed.

RUNE manager is in charge of operations such as loading *space* objects, calling entry points in *spaces*, relaying communication via *conduits*. In RUNE architecture, every *space* is required to have five entry points in it, since typical emulation can be broken into five parts: *initialization, execution step, finalization, read* and *write*.

The emulation process performed by RUNE is described next. First, RUNE manager loads the object. Then RUNE manager notifies RUNE master of completion of the attach process after that RUNE master indicates the initialize process of all spaces to RUNE managers on each node. A *space* allocates its work area which is permanently needed for the execution of emulation, and returns its pointer to RUNE manager. A *space* does not use stack area, but the work area allocated by itself for execution of emulation. This structure contributes to an efficient usage of memory because each space object can emulate multiple instances while sharing the binary code. This structure also ensures that *spaces* are thread-safe since they don't have any static data. RUNE master starts iterated invocation of the "step" symbol after the initialization of all spaces is finished. Iterations last until one of the spaces in experiment returns something else than normal status. When RUNE master receives the status, it starts finalization by notifying the end of experiment to all nodes. *Spaces* release then the work area allocated in the initialization process.

IV. Experimental Results

In this section we show some illustrative results that we obtained using the experiment support software RUNE and QOMET (Quality Of transforMing Environments Testbed) to emulate the WLAN communication environment [6].

The emulated topology is presented in Fig. 2. In this emulated home environment the air conditioner is wire-connected to a home controller, and works according to the information sent from the heat sensor. The temperature information is sent from the heat sensor to the home controller using WLAN, and then communicated to the air conditioner. An access point located in the house ensures WLAN access. The same access point is used by a human user and for simplicity reasons we assumed this user is performing a file transfer for the duration of the experiment. In a home environment there are many sources of interference with WLAN communication, such as microwave ovens or cordless phones. We assumed that during

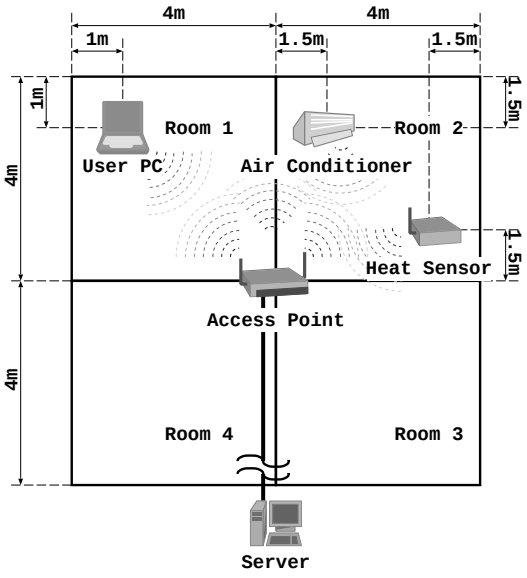

Fig. 2. Experiment topology

the experiment such interference will occur one minute, and then again three minutes after the experiment is started, and last each time for one minute. The interferences are with a noise level of -70 dBm and -60 dBm respectively.

In the experiment we used QOMET to emulate WLAN communication. QOMET emulates WLAN technology by calculating how network quality degradation depends first on the physical layer conditions. Then QOMET computes data link layer characteristics such as packet loss, effective bandwidth, delay and jitter by taking into account the 802.11 MAC layer behavior. QOMET also emulates the change of encoding method and operating rate that may be performed by the WLAN interface. Network quality degradation is derived in QOMET from models, such as those for radio wave propagation that use the parameters α (attenuation) and σ (standard deviation of the shadowing effect), as well as noise level. This derivation is the first stage of the WLAN emulation process. In the following step actual network quality degradation is enforced using *dummynet* [8] and applying the characteristics calculated in the previous stage. This takes place in real time on the testbed. The emulated WLAN environment used in our experiment had the properties shown in Table I.

The space allocation of the experiment is shown in Fig. 3.

The spaces corresponding to each equipment (air conditioner, heat sensor, user PC, and WLAN access point) work on separate nodes. In addition, a space that emulates heat transfer in the room is also deployed on a dedicated node. QOMET does not appear as one space but multiple spaces distributed on every node on which WLAN equipments are emulated. The spaces are called dnconf (DummyNet CONFiguration) space, and configure the *dummynet* pipe dynamically.

As mentioned before, each node in StarBED has at least

TABLE I
WLAN PROPERTIES

parameter		duration [s]	value
α		0 – 300	4.02
σ		0 – 300	7.36
normal noise level		0 – 60 120 – 180 240 – 300	-100 dBm
interference noise level		60 – 120	-70 dBm
interference noise level		180 – 240	-60 dBm
packet size	Heat Sensor → Air Conditioner	0 – 300	64 bytes
	Heat Sensor → Air Conditioner	0 – 300	64 bytes
	User PC → Server PC	0 – 300	1500 bytes
	Server PC → User PC	0 – 300	64 bytes

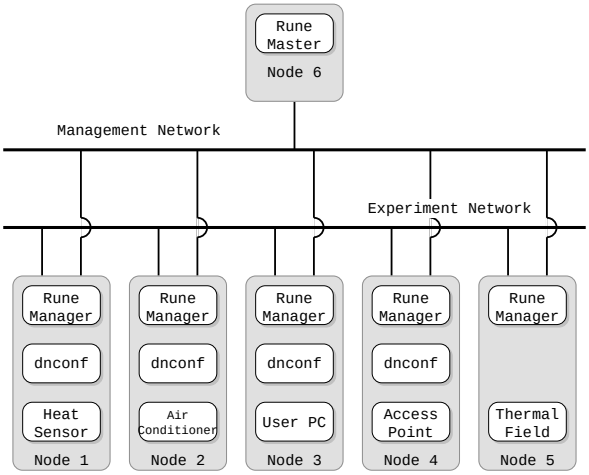

Fig. 3. Space allocation

two interfaces connected to different network segments. In the experiment, the management network is used for communication between RUNE master and RUNE managers, all of the experiment network for communication through the emulated WLAN.

In the experiment each space behaves as follows:

1) The Heat Sensor space obtains the temperature of a certain coordinate of the room from the thermal space, and sends the value to the air conditioner space every second.
2) The Air Conditioner space heats up the ambient air if the temperature sent from the heat sensor space is below the desired temperature (25 degree Celsius), otherwise stops heating. If the information is lost on the way from the heat sensor space for some reason, the air conditioner space keeps its current operation status. We assume that the heat source of the air conditioner is 120 degrees Celsius steam.
3) The User PC space generates bulk traffic which mimics a file transfer through the access point space. For this purpose *netperf* is used [7].
4) The Access Point space bridges the traffic between

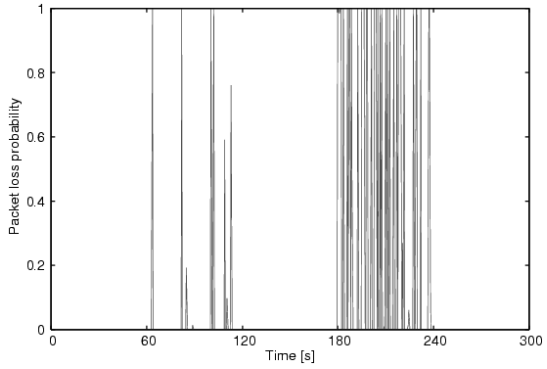

Fig. 4. Loss between heat sensor and access point

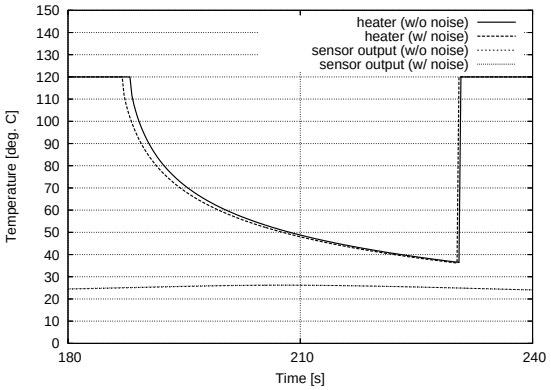

Fig. 5. Temperature transition for heater and sensor output

other spaces, and receives *netperf* traffic from the user PC space.

5) The Thermal Field space simulates the temperature transition in the room. For rapid heat transition, we assumed that room wall temperature is constant and equal to 10 degrees Celsius.
6) The Dummynet Configuration space applies repetitively the WLAN properties pre-calculated by QOMET.

Fig. 4 shows frame loss rate between the heat sensor and the access point. As it can be seen, network degradation occurs when the interference noise level raises. Losses have of course a significant effect on TCP/IP performance. Fig. 5 shows the variation of the temperature of the heat source and that observed by the heat sensor, both in the case when no interference noise is present, and in the case with noise. Since the differences are small, we show only the graphs for a time interval where effects are easily visible. The reason for the small scale of the effects is that packets are only rarely sent by the heat sensor (one per second). In addition, packet loss only affects system behavior if it occurs at the moments at which heat source state should change (from ”on” to ”off” or vice versa).

V. Conclusions & Future Work

Ubiquitous and sensor networks are large and complex, and it is difficult to predict their behavior. A lot of technologies for these networks are published and will be made public in the future. Since the expectations for these technologies' behavior are hard to assess, it is also difficult to make a testbed for that.

For this purpose we implemented an experiment support software, RUNE. RUNE assists executing emulation of ubiquitous network with a large number of PCs on multiple nodes. In a RUNE architecture, each emulation target is implemented as *space* and compiled into a shared object so that a single shared object can work as multiple instances.

To illustrate the use of the experimental platform we provided some test results that show the capabilities of StarBED2 and RUNE. We showed the different effects that interferences in WLAN environments can have on application performance, depending on each application characteristics.

Development of StarBED2 is still in progress. A high priority is given to adding support for processor and middleware emulation, which will increase the capabilities and functionality of StarBED2. In order to emulate more complex sensor networks other environment *spaces* will be added as well, such as acoustic environments and optical field for example. RUNE's basic functionality is already working, but more features will be added, such as, for example, more strict synchronization and mutual exclusion.

References

[1] Philip Levis, Nelson Lee, Matt Welsh, and David Culler: TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003) (2003)
[2] Jonathan Polley, Dionysys Blazakis, Jonathan McGee, Dan Rusk, John S. Baras: ATEMU: A Fine-grained Sensor Network Simulator Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004) (2004)
[3] Priya Mahadevan, Adolfo Rodriguez, David Becker, Amin Vahdat: MobiNet: A Scalable Emulation Infrastructure for Ad hoc and Wireless Networks. Proceedings of the International Workshop on Wireless Traffic Measurements and Modeling (WiTeMe 2005) (2005)
[4] Toshiyuki Miyachi, Ken-ichi Chinen and Yoichi Shinoda: Automatic Configuration and Execution of Internet Experiments On An Actual Node-based Testbed. 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(Tridentcom) (2005) 274–282
[5] The StarBED Project: http://www.starbed.org/
[6] R. Beuran, K. Chinen, K.T. Latt, T. Miyachi, J. Nakata, L.T. Nguyen, Y. Shinoda, Y. Tan, S. Uda, S. Zrelli, ”WLAN Emulation on StarBED”, IET International Conference on Wireless, Mobile & Multimedia Networks (ICWMMN) 2006, Hangzhou, China, November 6-9, 2006, pp. 856-859.
[7] Netperf: A Network Performance Benchmark. http://www.netperf.org/
[8] L. Rizzo: Dummynet: a simple approach to the evaluation of network protocols. ACM Computer Communication Review **27** (1997) 31–41