StarBED2: Large-scale, Realistic and Real-time Testbed for Ubiquitous Networks

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

Hokuriku Research Center, National Institute of Information and Communications Technology
Internet Research Center, Japan Advanced Institute of Science and Technology
School of Information Science, Japan Advanced Institute of Science and Technology

***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. Therefore it is difficult to analyze such networks using typical network simulators. Accordingly testbeds for these networks must be able to perform accurately emulation of large-scale networks with a complex topology. In order to implement a testbed that satisfies these requirements, 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 by introducing several new concepts. In this paper we describe first the present StarBED, its design concept, overall architecture, implemented functionalities, and some of the experiments we performed. Then we introduce StarBED2, its design policy, architecture, and additional components.**

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. As the ubiquitous networks have been connected to the Internet and introduced to our life, their influence on our life grows. This means that if some problems occur on the Internet and ubiquitous networks, they may affect our life.

Therefore, the importance of evaluating new algorithms and their implementation is increasing. The current Internet is already one of the important infrastructures for our life. Running and evaluating newly developed distributed software directly on the current Internet is no longer an option, because they may severely impact on the existing crucial services.

In order to evaluate new algorithms, it is popular to use software simulators and laboratory-level small-scale testbeds based on actual nodes. Software simulators can easily make experiments with large topologies. However, we cannot use the same implementations for the real environment and it takes a long time to execute experiments when using huge topologies and simulating node behavior in detail. Using a laboratory-level small-scale testbed that is based on actual nodes, we can use the same implementations for the real environment. It is, however, difficult to drive large-scale experiments because of costs for preparing physical nodes and controlling them, etc.

In order to evaluate new technologies for Internet connected to ubiquitous networks, a new testbed for ubiquitous networks is required. Since the behavior of elements on the Internet and ubiquitous networks is very complicated, we cannot predict the behavior before making experiments. Moreover the network consisting of the Internet and ubiquitous networks is quite large. In order to introduce new technologies for an infrastructure, we have to evaluate the same implementations for the real environment using large-scale topologies since we have to know their behavior on the real environment including potential bugs. Unless experiments on such testbed are made, these technologies may have bad influences on critical services running on the infrastructure.

We implemented a large-scale practical testbed based on actual nodes, named StarBED, that can emulate several thousand nodes. 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.

To 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 and discuss requirements for emulating ubiquitous networks and existing methods for making experiments, then we explain current StarBED and the design of StarBED2.

II. Requirements for Emulating Ubiquitous Networks

In order to determine what is required for a testbed for ubiquitous networks and how it can be implemented, we need to figure out the characteristics of ubiquitous networks, and clarify how they are different from computer networks. The differences between ubiquitous networks and computer networks are mentioned in the following subsection.

A. Characteristics of Ubiquitous Networks

Ubiquitous networks have different properties than computer networks in many aspects such as the following:

- variety of nodes
  In computer networks, similar computers are used as nodes which communicate with each other through a unified protocol, TCP/IP, on a unified network, IEEE802. On the other hand, nodes and networks to be used for ubiquitous networks vary in hardware and software architecture, amount of memory, and acceptable physical size, according to their purpose and requirements for them, such as cost efficiency, computational ability, thrifty power consumption, expected lifetime, etc. Cheaper cost and longer operational time are often required from the nodes and networks. This means the ability of each node of the ubiquitous networks is very limited in many aspects such as computational power, memory capacity, access networks coverage, etc. While most of nodes in ubiquitous network are relatively simple, rich nodes can participate in the same networks simultaneously. Thus ubiquitous networks must mediate the communication between multiple nodes with significantly different abilities.
- variety of network media
  For the same reasons mentioned above, the network media of ubiquitous networks are chosen in function of the requirements for the networks such as low power consuming, ease of installation, low running cost, and etc. The IEEE802 family, low power wireless networks, Power Line Communication (PLC), and telephone lines are used as access network for home networks. For sensor networks, low power wireless networks are used as access networks. The protocols used for communication are naturally different according to the access network media.
- huge number of nodes
  Ubiquitous networks consist of a huge number of small nodes existing close to each other, and access networks for which typically low power wireless networks are used. Because of the relatively poor ability of the nodes and limited coverage of the access networks, the nodes must stay in close connection , work cooperatively, and communicate frequently with each other. Accordingly, the number of nodes in a certain area naturally becomes greater than that in regular computer networks.
- importance of interaction with surrounding environment
  In most applications of ubiquitous networks, nodes of the network tend to interact actively with the surrounding environment in many ways. For example, the role of sensor network nodes is normally to obtain some kind of information from the nearby environment, and communicate it to other nodes. The role of nodes in home networks may also include some kind of measurement which may impact on their own behavior. For the above reasons, interaction between the nodes and environments is important in ubiquitous networks.
- importance of geographical information
  In computer network services, servers and clients can communicate with each other regardless of their location are. In other words, computer networks work without awareness of location. In contrast, location of nodes is significant for ubiquitous networks. For example, in sensor networks the location of nodes is one of the most important information which the nodes have to tell each other because the information is inseparably related to their location.
- persistency of network topology
  The topology of sensor networks is changing from hour to hour when mobile ad-hoc networks are used as access networks because of the condition of signal propagation and routing algorithm. Changing the topology may cause packet loss or increase of delay and latency.

The required functionality for testbeds will be presented in the following section.

B. Testbed Required Functionality

Some of the requirements for testbeds come from the characteristics mentioned above, so these characteristics affect the design and implementation of testbed for ubiquitous networks. To implement a testbed which is appropriate for emulation of ubiquitous networks, the testbed must have the following functionality:

- emulation of surrounding environment
  In ubiquitous networks, the information obtained from surrounding environment is more important in comparison to computer networks. Accordingly, a testbed for ubiquitous networks has to support emulation of the environment and provide the interface between the emulated nodes and environments by which the nodes can obtain necessary information.
- provision an interface between physical space and logical space
  In the past, evaluations of ubiquitous networks were done in either fully emulated test system using virtual nodes, networks, and surrounding environments, or fully equipped real environment. It is, however, sometimes useful that the evaluations of the networks are done in virtual and real mixed environment in which emulated nodes, networks, environment can work together with real ones. There are some abstraction levels in emulations of ubiquitous networks according to the proportion of virtual to real portion. To enable the evaluations in virtual and real mixed environment, a testbed for ubiquitous networks should have the interface between physical space, outside testbed, and logical space, inside testbed through which the virtual and real components communicate each other without distinction. Thinking about the interaction between the virtual and real space, it is also required that the testbed can work under real-time constraint.
- support numerous nodes
  The huge number of nodes is one remarkable property of ubiquitous networks. Testbed for ubiquitous networks must support emulation of such large-scale networks. Supporting emulation of a huge number of nodes is actually not so difficult if it is allowed that infinite time

can be consumed for the execution of the emulation, as software simulators do. But the issue here is that emulation must be done under real-time constraints if the emulation is run in virtual and real mixed environment. To support both the emulation with numerous nodes and emulation under real-time constraint a great amount of computational power is needed.

- emulation of various architectures of nodes and networks
  Nodes of ubiquitous network have a variety of architectures with respect to both hardware and software, as well as their access networks. To implement a testbed for such networks which supports emulation of heterogeneous networks, the testbed must have the flexibility which enables emulation of such heterogeneous networks.
- provision multilevel emulation layer
  In its development cycle, nodes of ubiquitous networks should be tested repetitively in various forms such as source code level emulation, binary level emulation, etc. Supporting these multilevel emulation in one testbed expands its capability of usage drastically. This functionality is of particular use if the purpose of emulation is to expand existing ubiquitous networks by introducing brand-new nodes into the network because user of testbed for ubiquitous networks need not implement the brand-new nodes in advance.
- provision emulation supporting system
  The emulation in which a great amount of nodes play the role of their own and various network media is used is hardly ever able to be carried out manually. Moreover, the timing of execution is important, and may affect the result of the emulation, if the emulation must be done under real-time constraint. For these reasons, a ubiquitous network emulator should have automated execution mechanism which enables execution of the emulation in a controlled manner.

In the following section, we will take a look at the existing methods for experiments.

III. Existing Methods for Experiments

Currently, software simulators and small-scale testbeds based on actual nodes are often used for experiments. In this section, we describe these existing methods.

A. Software Simulation

Software simulator makes experiments using abstracted network elements. It is the most popular approach to evaluate network technologies, and ns-2 by VINT Project [1] is the well known example of such simulators. It makes an experiment along configurations in which scenarios and topologies for experiment are described by the user. The cost for making experiments with software simulator is low because we can make experiments even with one computer and the experiments will be performed automatically with configurations written in advance.

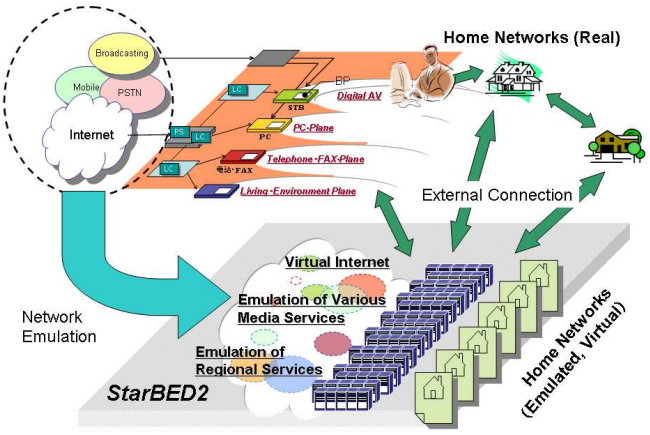

Fig. 1. Town emulation

However, we often cannot use implementations of target technologies directly on the Internet. Most software simulators require target systems to be described under their own modeling scheme, often using their own modeling language. These implementations may differ from what will actually be running on the Internet.

Software simulators run in logical time clocking. When we perform an experiment that requires detailed action for many nodes, it takes a long time with software simulators compared to the actual time that is described in the scenario. The duration required to run software simulation becomes problematic also as we try to simulate realistic target system under realistic environment where non-trivial aggregation of complex network services comes into play. On the other hand, the simulation will be finished within a short time when we make a simple experiment on simple topologies.

Using software simulators is a good way to validate algorithms or observing micro-behavior of communication protocols. However, it cannot be used to evaluate target technologies, including the bugs in the product implementations and behavior that is not described in the specifications before introducing new technologies to the Internet since what is simulated in those simulators is not the protocol itself but the model of the protocol. This is one big drawback of those simulators, even though they are really useful at some point in design and implementation phase of protocols.

B. Laboratory-level Testbed Based on Actual Nodes

A testbed based on actual nodes is a testbed built with network equipments and computers used in real environments. These testbeds consisting of a few dozen actual nodes are often used in research organizations concerning the Internet. We call these small-scale testbeds based on actual nodes ”laboratory-level testbed based on actual nodes.”

We can use software/hardware implementations of target technologies directly on the Internet with these testbeds. There-

fore, the behavior of target technologies and the result of experiment is realistic compared to the real Internet. However, the cost of making experiments with laboratory-level testbed is larger than that of software simulator. Because we have to prepare physical nodes needed by experiments, connect them with cables and configure nodes and switches if it is necessary. Especially controlling the experiments, how to execute commands in scenarios accurately or how to perform when an error occurs, etc., are important and difficult. When making experiments with large-scale topologies or complicated behaviors of experiment elements, the cost will become very big.

In this kind of testbeds, software for emulating network characteristics is often used to emulate bandwidth, delay, packet loss and etc. Dummynet [2] and NIST Net [3] are popular as such software.

Using these tools is not sufficient to simulate whole ubiquitous network system especially from scalability point of view, although they are more realistic than software simulators.

C. Tools for Ubiquitous Network Evaluation

There are already a number of implementations of emulators and testbeds for ubiquitous networks. ns/ns-2 is a discrete event-driven simulator which is widely-used for network simulations. TOSSIM [4] is a TinyOS simulator which aims to simulate TinyOS applications accurately in virtual environment. ATEMU [5] is also able to emulate TinyOS applications, and it has more flexible architecture to support other platforms. MobiNet [6] is rather a wireless network emulator than testbed for ubiquitous networks. MobiNet has the same concept as ours in some regards, such as utilizing PC cluster, real-time emulation, and high accuracy. Each test environment satisfies the requirements mentioned in the section II partially. None of them, however, provides the method to describe surrounding environments for user of testbed, and interface to interact with real nodes. Those functionalities makes simulations drastically realistic since simulation of ubiquitous networks needs interaction with surrounding environments. By implementing the functionality which enables simulation of surrounding environments, simulation of ubiquitous network system can be executed while obtaining necessary information from the simulated environments. In addition, the simulation can even interact with real environments if the interface between simulated and real environments. This is definitely of use for ubiquitous network simulation especially in the final phase of their development.

IV. First Generation Testbed - StarBED

The most suitable methods for making experiments depend on the phase of software development. Software simulation is suitable for the beginning phase, and laboratory-level testbed can be used for evaluating actual implementations just before introducing it to the Internet.

Our motivation is to evaluate actual implementations on the Internet-like environments to know the realistic behavior of these implementations on the real Internet and to avoid bad influences to critical services on the Internet. However, the gap between the Internet and laboratory-level testbed is very big, especially at points of scale and complexity. Technologies which are not sufficiently evaluated may impede critical services. We should avoid such extreme situations.

In order to solve this problem, we need a large-scale and realistic testbed for evaluation. We implemented StarBED, a large-scale practical network testbed based on actual nodes.

In this section, we describe StarBED and SpringOS which is a support system to drive experiments on StarBED.

A. Architecture of StarBED

In StarBED facility, many number of nodes are located on the same site, and all these resources are physically accessible. So we can get all information about the experiment traffic since all switches that connect experiment nodes are located in the same site. The environment is dividable and could be manipulated by several users. Figure 2 shows a space division of resource allocation considering three independent experiments, each set of nodes is completely independent from the others.

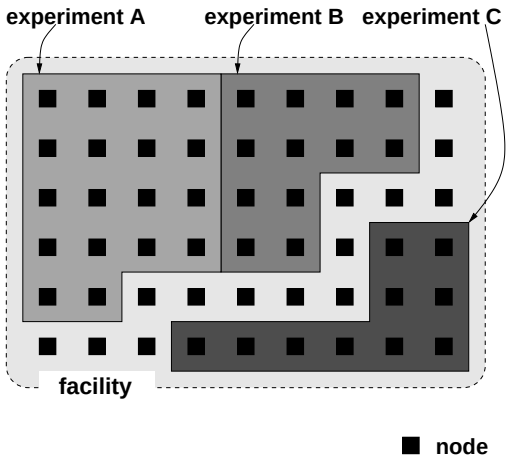

Fig. 2. Using space division of the facility

However, to build such environment, we need a variety of equipments and installations, which make such kind of projects quite difficult to achieve. To solve the difficultly of building experiment topologies, the network will be built beforehand based on fixed hardware connections and we build a customized topology using virtual networks.

These networks connect about 700 actual PCs equipped in StarBED, and enable to build large-scale experiment topologies. There are two networks, one is the management network and the other is the experiment network. Figure 3 shows this concept. By separating the management network from the network dedicated to the experiments, we can guarantee traffic separation and the precision of the experiments. Another purpose of the management network is to allow configuring the network interfaces in the experiment network. In this way, the configuration for experiments can be made based on user-

defined IP addresses during the building of the experiment topology.

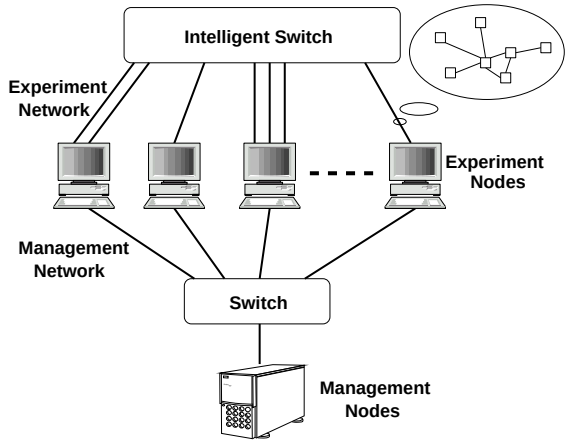

Fig. 3. Conceptual topology of StarBED

We designed the StarBED clients to run several ten virtual nodes using VMware [7]. When using virtual nodes, we should consider where virtual machines can be used, sometimes realism of experimental results becomes unacceptably low by using virtual machines.

B. SpringOS

We designed SpringOS, a system that supports the users to execute their experiments.

Using SpringOS, all what user has to perform is to pre-paid an experiment configuration file. SpringOS will execute experiments according to the configuration file; it will drive experiments according to the following steps:

1. Loading the user configuration file
2. Nodes assignment to an experiment
3. Software settings of the experiment nodes
4. Building topology for the experiment
5. Execution of an experiment according to the scenario

The details of the design and behavior of SpringOS are described in *Automatic Configuration and Execution of Internet Experiments On An Actual Node-based Testbed* [8], and StarBED Project web site [9].

V. Next Generation Testbed - StarBED2

In order to implement a testbed for ubiquitous networks that satisfies the requirements mentioned in section II, we are currently working on an implementation of StarBED2, a testbed for ubiquitous networks based on StarBED, our first generation testbed mentioned in the previous chapter. As the first step of the hardware implementation, the number of nodes on StarBED has been increased. As a result there are now about 700 nodes on StarBED.

Generally, to emulate ubiquitous networks is a highly computational-power-consuming task not only because of the huge number of nodes each network has, but also the complexity of their behavior. It is even more difficult to execute emulations under real-time constraints so that emulated nodes and real nodes can interact with each other. Thus we are about to utilize StarBED as a basis of StarBED2 since StarBED's ability to emulate thousands of network nodes accomplishes such a heavy task efficiently. 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 cooperatively with real nodes so that it can be widely used for prototyping, evaluation, testing of ubiquitous networks in any point of development.

A. Structure of StarBED2

StarBED2 is being constructed on StarBED architecture as its basis. On top of the StarBED portion, a multilevel emulation layer, 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 kind of nodes with different architectures can work together in a emulation. This framework of multilevel emulation layer gives the flexibility of use in any phase of the development of ubiquitous network.

From a physical point of view, StarBED2 is a large-scale PC cluster; in each node the emulation layer is implemented as can be seen from Figure 4.

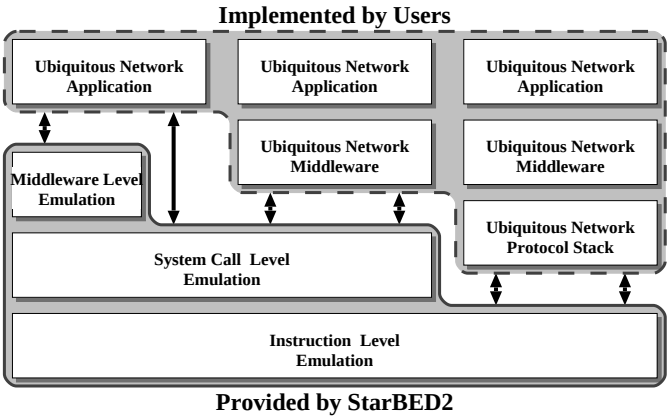

Fig. 4. Physical structure of StarBED2

Obviously this physical architecture itself is not sufficient for emulating ubiquitous networks. StarBED2 has a logical architecture as depicted in Figure 5, which makes StarBED2 able to executing emulate ubiquitous networks.

- *node space*
  The main idea of the logical architecture is a "*space*" and "*conduit*" structure. In the 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. In each *space* a user-defined coordinate system which is specific

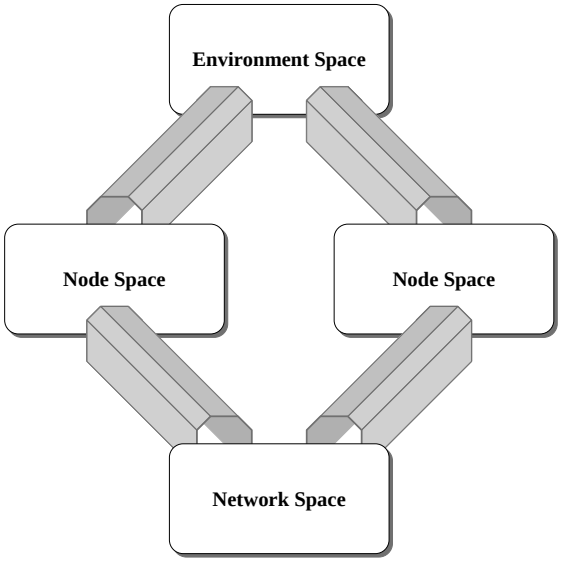

Fig. 5. Logical structure of StarBED2

to the *space* is used. For example, a pair of IP addresses and port numbers may be used in a *node space*, space coordinates in *network space* and *environment space* respectively. The *conduit* is a one-way communication channel which is set up between two *spaces*. All what a user of StarBED2 has to do for executing emulations is to implement each *space* and declare *conduits* between those *spaces*. This structure is roughly equivalent to the relationship between processes and Interprocess Communication (IPC), but they differ in some senses. One major difference is that a *conduit* works regardless of the location of both *spaces* which interact through the *conduit*. The *spaces* can be either on the same node or on different nodes since StarBED2 is a cluster system, and the emulation is done in distributed environment. Another difference is their address architecture. The end-points of IPC are bound to logical identifiers such as IP address and port number etc. In contrast, the end-points of *conduit* can be bound to either physical coordinate of objects etc., or logical identifiers according to implementation of the *spaces* the *conduit* is connecting. This is because the addressing using physical identifiers is necessary to realize the interaction between nodes and environments.

B. Space and Conduit Structure

In this section we explain how each component of a ubiquitous network can be implemented as *spaces* and *conduits*, taking a closer look at the *space* and *conduit* structure.

In *node space*, the behavior of nodes of ubiquitous network such as sensor nodes, appliance built-in nodes, home gateways etc. is emulated. The emulation is done in one of the multilevel emulation layer, instruction level emulation layer, system call level emulation layer, or middleware level emulation layer, under real-time constraints. The existence of this layer gives users the opportunity to evaluate nodes in the early phase of its development. For example, even if the hardware of the target node is newly designed and the operating system to be used has not yet ported on the target hardware, the node can still be emulated in the high layer, system call emulation layer or middleware emulation layer.

The system call level emulation layer converts calling conventions and emulates the system calls of other operating systems. In practice, the target of emulation has to be compiled and linked to the stub library of this layer in advance.

The middleware level emulation layer provides the APIs of ubiquitous networks to the emulation targets. By using this layer, the target of emulation can be executed without implementing any underlying protocol layers. In order to execute an emulation in this layer, the target has to be compiled and linked to the stub layer of this layer and the system call level emulation layer stub library.

The communication between a *node space* and other *spaces*, *network space* or *environment space*, is done through *conduits*.

- *network space*
  In *network space*, network media of ubiquitous networks such as IEEE802 family, Bluetooth [10], ZigBee [11] etc. are emulated. The network media can be emulated either in a statistical manner, in which the statistical model of the network is used, or in a realistic manner, in which signal propagation in the media is emulated. The ways of emulation can be switched according to the proceeding of the development. This means it is possible to use the statistical emulation for early prototyping in the early phase of the development, then switch to the realistic emulation for more precise analysis afterward.
- *environment space*
  In *environment space*, surrounding environments necessary for emulation such as thermal field, electromagnetic field, acoustic field etc. are emulated. The emulation of environment can be done either in a statistically manner, in which the statistical model of the environment is used, or in a realistic manner, in which any physical phenomenon in the environment is emulated.

The *spaces* mentioned above are implemented by the user of StarBED2, and executed as a process on each StarBED node autonomously, and interact with other *spaces* via *conduits* when needed. The *conduits* are bound to a certain coordinate in the source and destination *spaces*. When a read request is generated, the *conduit* makes a read request to the destination *space*, and sends the information obtained from the destination *space* to the source *space*. Write requests are processed similarly.

What is required for implementing the *spaces* does not differ significantly from the development of usual programs except that StarBED2 API has to be used for *conduit* com-

munication and real-time execution. The implementation of the *space* becomes a stand-alone binary after linking with StarBED2 library. The stand-alone binaries can be launched either manually or automatically by the emulation automating mechanism of StarBED2.

C. Functionality Considerations

By implementing the functionalities mentioned above, StarBED2 can be a testbed which satisfies the requirements for ubiquitous network emulation mentioned in section II.

Multilevel emulation layer enables emulation of a variety of ubiquitous network node platform and network media by its flexibility to emulate plenty of ubiquitous network components. The space and conduit structure provides a reasonably abstracted model of ubiquitous network system components which is valuable for setting up the emulation of ubiquitous network systems with a huge number of components and a complex topology. A combination of the about 700 PCs in StarBED and the virtual machines executed on the PCs can be the driving force of emulating a great number of nodes, networks and surrounding environments. SpringOS implemented as a part of StarBED supports dynamic changing of the topology of ubiquitous networks.

By using these functionalities, StarBED2 will be extremely useful for emulating whole ubiquitous network system, though it is still challenging to meet all the requirements simultaneously.

VI. Conclusions

Ubiquitous and sensor networks are currently hot research topics. They may be introduced in our daily life and connected to the Internet in the near future. These networks are large and complex, and it is difficult to predict their behavior. A lot of technologies for these networks are published now and will be made public in the future. Since the expectations for these technology's behavior are difficult to assess, it is also difficult to make a testbed for that. Using software simulators, one can only simulate well-known behaviors of technologies, and the implementations for real environments cannot be used. In order to evaluate implementations for these networks, testbeds based on actual nodes are strongly needed. However, building such testbeds is difficult because of the cost.

We have worked for years on StarBED, a large-scale network testbed, to reduce the cost for experiments based on large number of actual nodes. There are many nodes in StarBED just for experiments, users can use these nodes with time and space sharing. In StarBED, building topologies and driving scenarios for experiments are executed automatically by SpringOS. As a result of our effort, StarBED has been working and used for lots of practical experiments which opened new views on the design of network equipments. Our research goal is to implement a testbed for ubiquitous networks, StarBED2, based on StarBED. On top of StarBED, some components are added so that StarBED2 can be used widely for estimating ubiquitous networks.

In practice, emulation can be done with StarBED2 by implementing nodes, networks, and surrounding environment as *spaces*. They communicate with each other through *conduits* during the emulation process. This emulation is done with real-time constraints so that our system can work cooperatively with real components.

In this paper, we described the overall architecture of present StarBED from both software and hardware points of view, and how it is being expanded to StarBED2. Finally, we presented the architecture adopted for StarBED2 in order to implement a ubiquitous network emulator that meets specific requirements. In this paper, no evaluated results can be shown since StarBED2 is now in the design phase. But we have confidence that StarBED2 become a testbed which satisfies the requirements for ubiquitous network emulation mentioned in the section II by implementing specific functionalities such as PC clustering, experiment support system SpringOS, virtual machines, multilevel emulation layer, and space and conduit structure mentioned in the previous chapter.

References

[1] The VINT Project: http://www.isi.edu/nsnam/vint/index.html
[2] L. Rizzo: Dummynet: a simple approach to the evaluation of network protocols. ACM Computer Communication Review **27** (1997) 31–41
[3] NIST Internetworking Technology Group: NIST Net network emulation package, http://www-x.antd.nist.gov/nistnet/
[4] 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)
[5] 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)
[6] 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)
[7] VMware Inc.: http://www.vmware.com/
[8] Toshiyuki Miyachi and 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
[9] The StarBED Project: http://www.starbed.org/
[10] Specification of Bluetooth. https://www.bluetooth.org/
[11] Specification of ZigBee. https://www.zigbee.org/
[12] National Institute of Information and Communications Technology: http://www.nict.go.jp/
[13] Intel Corporation: Preboot Execution Environment (PXE) Specification Version 2.1, (1990)
[14] Takuji Iimura and Hiroaki Hazeyama and Youki Kadobayashi: Zoned Federation of Game Servers: a Peer-to-Peer Approach to Scalable Multi-player Online games, NetGames (2004)
[15] Eiichi Muramoto and Takahiro Yoneda and Atsushi Nakamura and Makoto Misumi and Toshiyuki Miyachi and Yoichi Shinoda: Report on a Method of Simulating Multicast Group Communication on the Internet, Towards Peta-Bit Ultra Networks (2003)
[16] Brian White and Jay Lepreau and Leigh Stoller and Robert Ricci and Shashi Guruprasad and Mac Newbold and Mike Hibler and Chad Barb and Abhijeet Joglekar: An Integrated Experimental Environment for Distributed Systems and Networks. OSDI02 (2002) 255–270
[17] IEEE standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. (1998)
[18] T. Ylonen: SSH – Secure login connections over the internet. Proceedings of the 6th Security Symposium (USENIX Association: Berkeley, CA) (1996) 37