

StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software

Toshiyuki Miyachi
School of Information Science
Japan Advanced Institute of
Science and Technology
1-1, Asahidai, Nomi
Ishikawa, Japan
toshi-m@jaist.ac.jp

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

Yoichi Shinoda
Center for Information Science
Japan Advanced Institute of
Science and Technology
1-1, Asahidai, Nomi
Ishikawa, Japan
shinoda@jaist.ac.jp

ABSTRACT

New technologies for the Internet should be evaluated on environments dedicated to experiments, in order to avoid influences to critical services on the Internet. Generally software simulation and small testbed using real nodes are used to satisfy these requirements. There are several stages in developing new technologies, however, and these technologies may not satisfy requirements for all stages.

We pointed the gap between the Internet and environment for experiment, especially in aspects of scale, complexity and reality. We proposed and implemented StarBED which is a testbed based on lots of actual nodes, in order to build large-scale, complex and realistic environments for experiments. StarBED consists of 512 PCs and switches which connect these PCs. The PCs on StarBED are designed to run 10 virtual PCs on a physical PC. It enables to build a topology for experiments using up to 5120 nodes. It is difficult to manage and control such a lot of nodes. We also designed and implemented SpringOS, which is a supporting software for making experiments. SpringOS makes the topology and drives the scenario for experiment according to the user's configuration automatically.

Many experiments were performed on StarBED, and this shows StarBED's effectiveness.

Categories and Subject Descriptors

C.2.3 [Network Operations]: [Network management, Network monitoring, Public networks]

General Terms

Experimentation, Verification

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VALUETOOLS '06, October 11-13, 2006, Pisa, Italy
Copyright 2006 ACM 1-59593-504-5 ...\$5.00.

Keywords

Network testbed, supporting software for experiments

1. INTRODUCTION

The past Internet was an experimental network, so new technologies for the Internet were evaluated on the Internet itself. The current Internet, however, has become the most versatile and the most wide spread communication infrastructure, with so many critical services running on it. We cannot make experiments for evaluating new technologies on the current Internet in order to avoid bad influences to these services.

As the growth of the Internet has continued, there are many demands to make experiments on the Internet for many purposes. To satisfy these demands, software simulators and laboratory-level testbeds based on the actual nodes are used as Internet-like environments.

Software simulation is the most popular approach to evaluate them, and ns-2[19] is a well known example of such simulators. The environments of software simulation are very useful for evaluating algorithms of technologies, however, we often cannot use implementations of target technologies for the Internet directly. This is because they often require simulator specific implementations which are different from what will actually be running on the Internet. The implementations for simulators may differ from implementations for the Internet.

For knowing the abstract behavior and evaluating the algorithms of these technologies, software simulators are a good method. However, we should evaluate them including bugs and behavior that is not described in specifications, by using real implementations on a realistic testbed, before introducing the new technologies to the Internet.

We propose StarBED, a large-scale network testbed based on actual nodes, and SpringOS, a supporting software for making experiment on StarBED. In this paper, we describe StarBED and SpringOS.

2. EXISTING METHODS

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

2.1 Software Simulation

A software simulator makes experiments using abstracted network elements. It is the most popular approach to evaluate network technologies, and ns-2[19] by VINT Project[18] is a well known example of such simulators. It makes an experiment according to configurations in which scenarios and topologies for experiment are described by the user. The cost for making experiments with software simulators is low because we can make experiments even with one computer and the experiments will be performed automatically with configurations written in advance of starting the experiments.

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.

Moreover, software simulators run in logical time. 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. Because we can make an experiment environment which is completely the same as user's assumption, without any kind of disturbance and bugs of implementation. 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.

2.2 Laboratory-level Testbed Based on Actual Nodes

Testbed based on actual nodes means testbed built with network equipment 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 testbed 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. Therefore, the behavior of target technologies and the result of experiment is realistic compared to the real Internet. This approach also enables performance tests in realistic environments.

However, the cost of making experiments with laboratory-level testbeds is larger than that of software simulation. Because we have to prepare physical nodes needed by experiments, connect these nodes 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 act when an error occurs, etc., are important and difficult in this case. 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

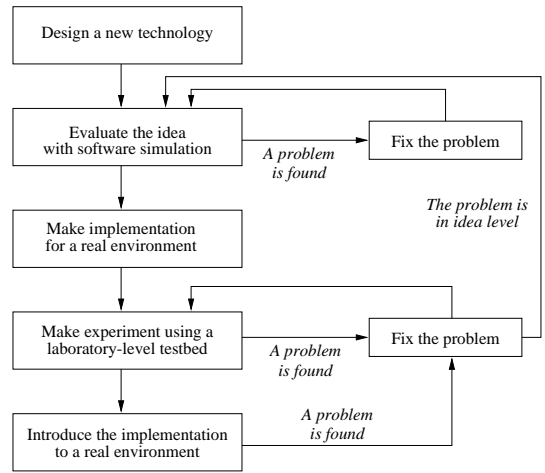


Figure 1: An Example Steps of Development Process

characteristics is often used to emulate bandwidth, delay, packet loss, etc. Dummynet [15] and NISTnet [3] are popular examples of such software. Using these tools, the testbed should be more realistic.

2.3 The Gap between the Existing Methods and the Internet

The suitable methods for making experiments are different in each phase of software development.

Figure 1 shows an example of general development process. Software simulator is suitable for 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 evaluated sufficiently may destroy critical services, sometimes it means that one of the social infrastructure may be destroyed. We should avoid these critical situations. In order to solve the problem, we need a large-scale and realistic testbed just for evaluation. Figure 2 shows an example development cycle which we propose using a large-scale testbed.

3. RELATED WORK

Netbed[22] is an integrated experimental environment. It can access simulated, emulated and wide-area testbed transparently. There are many functions to manage experiments easily including GUI interface. Simulated nodes using nse [17] can be used for experiment to build large topologies. As a user requests to perform experiment to the master server of Netbed, the system will perform it when enough resources for this are available.

Netbed has no mechanism for node rendezvous, so it is difficult to make an experiment which needs strict and complex synchronizations. The experiment tends to be the cycle of finding bugs of the implementations or scenarios for the experiment, and fixing it. The user's experiments in Netbed

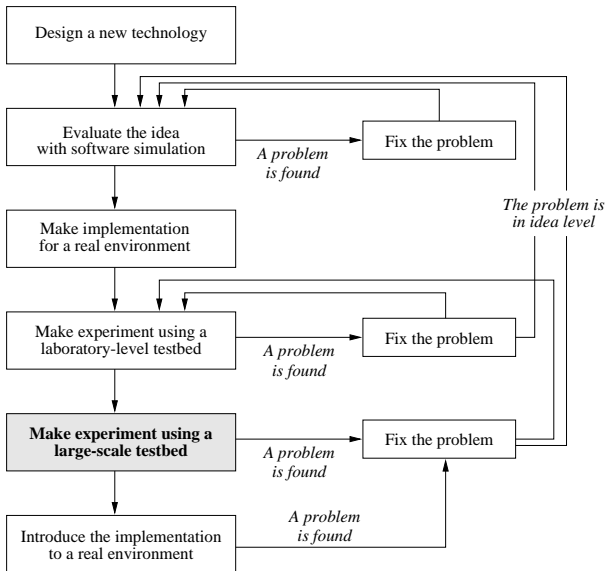


Figure 2: An Example Steps of Proposed Development Process

are preformed as batch processing, therefore, sometime it may be not suit users experimental cycles.

PlanetLab [1] is a testbed using an overlay environment. The nodes are distributed on the globe, these nodes are connected via the Internet. A user can build a large-scale topology virtually on PlanetLab. It enables to introduce real links and it's characteristics in the experiment.

The topologies for experiments are built as overlay networks, so they may be affected by the Internet traffic. Therefore, the results of the experiment is often more realistic. However, it may be different if the same experiment is performed several times. A user cannot access the network equipments on the remote sites, so sometimes troubleshooting is difficult.

4. STARBED

We designed and implemented StarBED, a large-scale general purpose network testbed based on actual nodes. StarBED was founded by National Institute of Information and Communications Technology(NICT)[10] in 2002 as Hokuriku IT Open Laboratory¹.

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

4.1 Concept of StarBED

We assume that many nodes are located on the same site, and all these resources are physically accessible. So we can get all of information about the experiment traffic since all switches that connect experiment nodes are located on the same site.

In order to build an Internet-like environment, we need a variety of equipments and installations, which make such kinds of projects quite difficult to achieve. To solve the difficulty of building experiment topologies, the network will be built beforehand based on fixed hardware connections

¹It is renewed as NICT Hokuriku Research Center with updating the facility in April 2006.

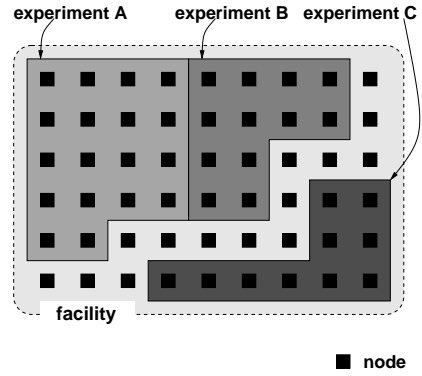


Figure 3: Using StarBED Facility by Space Division

and we build customized topology using virtual networks. We can set the desired topology for the experiments using VLAN[4] and VP/VC. The experiment can be driven on the virtual environment, thereby we don't have to change the physical topology.

We also decided to make our environment a multi-user system. Nodes are assigned by the administrator to the users. Figure 3 shows a space division of resource allocation considering three independent experiments, each set of nodes is completely independent from the others.

Users should make a reservation and hold a group of nodes long enough for their experiment and they can use reserved nodes freely including install new OS during the reserved period. Users set up these nodes by installing the needed software for their experiments. With this reservation method, if they meet a problem during an experiment they can fix the problem and retry the experiment.

We designed a supporting software, SpringOS, which builds the desired experiment topology and drives the experiment following the user defined steps, automatically. Moreover, this software makes the synchronization of experiment nodes. We don't force users to use SpringOS, other supporting software or simple programs are also accepted to use on StarBED.

4.2 Construction of StarBED

There are 512 actual PCs in StarBED, in order to build large-scale experiment topologies. Nodes in StarBED are classified into 5 groups by their functionalities, such as the number of network interfaces. The descriptions of nodes are showed in Table 1. Users can choose suitable clients for their experiment. As support to manage nodes for experiments, we can access all consoles of nodes using Raritan [13] products.

The actual management mechanisms are deployed using a separate network, and each node in our environment has a network interface dedicated to this purpose. By separating the management network from the network dedicated to the experiments, we guarantee the traffic separation and the precision of the experiments. On the experiment network, a network interface is not configured when building the experiment topology. In order to configure the network interfaces of nodes remotely, a configured network interface is needed. Figure 4 shows this concept. There are 4 intelligent switches to connect all nodes and make topologies for experiment using VLANs. A switch is connected to other switches with

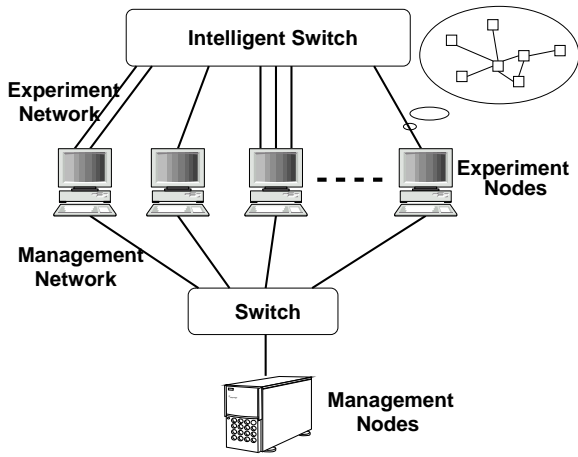


Figure 4: Concept Topology of StarBED

Table 1: Clients in StarBED

client group	number of nodes	Network Interface for experiment			disk type
		Gigabit Ethernet	Fast Ethernet	ATM	
A	208	1	0	0	ATA
B	64	0	1	1	ATA
C	32	0	4	1	SCSI
D	144	0	1	0	ATA
E	64	0	1	0	ATA

10 Gbps links. The management network topology is quite simple and flat, because basically a user doesn't change the topology of the network.

There are two external links to outside of StarBED, one is to JGN2[11] network via 10Gbps and the other is to WIDE[23] network also via 10Gbps. We can introduce real traffic to an experiment via these links.

Racks for introducing user's own equipments are available in StarBED and a user can connect these equipments to existing networks. With the racks and network connections, a user can evaluate their targets in the same situation on laboratory-level testbed. When making hardware performance tests, this is an essential function.

We design the StarBED clients for running up to 10 virtual nodes using VMware[21]. VMware is one of the tools to realize virtual nodes. VMware realizes virtual machines, so generic OSes can be run on virtual nodes by VMware. There are many tools to realize virtual nodes for which the abstraction level of nodes is various. Generally as the abstraction level of emulating computers becomes high, the number of virtual nodes on one computer increases and virtual nodes are less reality. We think the reality of nodes are important, so we adopt VMware as PC level emulator. When using virtual nodes, we should consider where virtual machines can be used, as sometimes reality of experiment results become low by using virtual machines.

5. SPRINGOS

In the previous section, we presented the connectivity and the characteristics of our experimental environment, as

well as the needed preparations and management procedures that we assumed from here on. During the experiment, however, we need to define various other procedures and that will be the topic of the following paragraph. We designed SpringOS, a software that supports the users to execute their experiments. SpringOS can manage and control many nodes including virtual nodes by VMware.

The assumed environment for SpringOS is like StarBED, dividable and can be manipulated by several users, concurrent resource usage can happen, such as nodes assignment or switch configurations. For that reason we need to have an access restriction and mediation mechanisms to share and manage those resources.

More details on the design and behavior of SpringOS are described in [20] and on the StarBED Project website [16].

5.1 Resource Management

Nodes are assigned to users following the criteria described in the configuration files. To compensate the possible failure of some nodes, users can acquire more nodes than described in the configuration file as spare nodes. The number of spare nodes is configurable.

VLAN IDs are needed to build topology for an experiment. We also manage VLAN IDs and assign it to users according to the configuration file.

When a user requests nodes to the Resource Manager, it will assign nodes based on the requirements for each node. The main requirement is the number of network interfaces. The Resource Manager will search nodes which satisfy the user's requirements. The algorithm to search a node is NP-hard problem[14], so we employ the first match algorithm now.

5.2 Booting nodes

The Wake on LAN mechanism is used to boot up nodes. Experiment nodes load OS using the bootloader provided by PXE[6]. The bootloader specifies whether the node starts as diskless system or boots from local partition. The bootloader configurations are specified by the DHCP[2] server. Before booting up leased nodes, the user may setup the PXE so that the node boots from a specified partition and using Wake on LAN to boot up nodes.

5.3 Software Installation on Nodes

Almost all of the time we have the same OS and applications installed on nodes having the same role in the experiment. Therefore, disk images could be created for every role and installed in the node collection having the same role at the same time. In many cases we only need a few disk images to setup all the experiment nodes. Moreover, the setup of IP addresses of each network interface is done according to the configuration file.

5.4 Building Topology for Experiment

The topology for an experiment is built automatically by describing the desired topology in the configuration file using VLANs. Virtual networks are identified by VLAN IDs, which will be used to build the network topology for an experiment. These virtual networks are created using configurable switches according to the needs of each experiment.

5.5 Driving Scenario

To post the scenarios on each nodes, one way is to man-

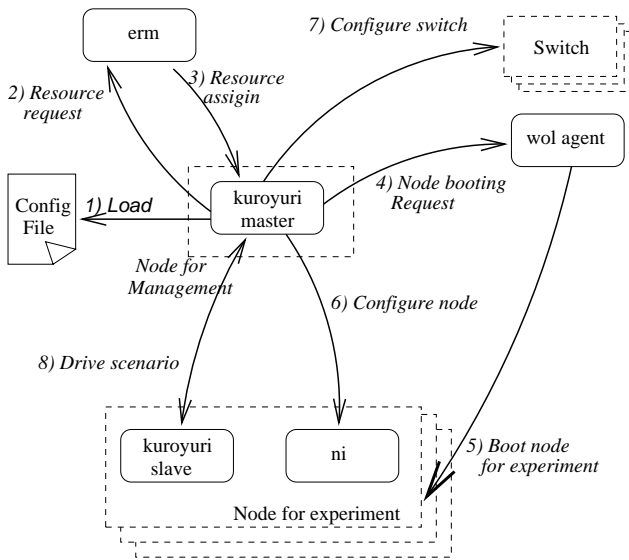


Figure 5: Overview of SpringOS's Behavior

ually perform a remote login followed by the scenario execution, another way could be remote execution using rsh[7] or ssh[24] etc. These manual procedures have many inconveniences. One of them is a synchronization problem: we cannot control precisely the execution time of scenarios on each node. It becomes difficult to interpret the experiment results and outputs. Moreover, we cannot reproduce the same experiment because of the imprecision of the human factor.

There are several methods for driving experiments automatically following the user's scenario. One method is to copy the scenario before the experiment starts, and each node will be programmed to execute the scenario independently at a specified time. Another method is to use a scenario master: the scenario master transmits commands to the experiment nodes at a specified time. Using the first method, experiment nodes cannot synchronize with each other. By using the second method, the scenario master has to manage sessions for sending scenarios to all experiment nodes. When there are many events at the same time, the management cost will be large and event timings might be delayed.

Therefore, our method uses scenario master only when some experiment nodes have to be synchronized. The usual method is to copy the scenario before the experiment starts, and generally each nodes execute the scenario independently. When an experiment node has to synchronize, it connects to the scenario master to mediate with other experiment nodes. When a node should synchronize with other nodes, the scenario master controls the nodes with message passing. The details of the scenario driving of SpringOS, especially concerning synchronization of nodes, are described in [8].

5.6 Modules of SpringOS

SpringOS is a collection of modules. Following is the list of the major modules constituting SpringOS.

Kuroyuri master

The *kuroyuri master* is the core function for driving experiments. It loads configuration files and then it conducts the other modules to make experiments.

Kuroyuri slave

The *kuroyuri slave* executes scenarios on experiment nodes. After introducing software into the nodes, the *kuroyuri slave* communicates with *kuroyuri master*, gets the scenario for the node, and then executes the scenario.

Experiment resource manager (erm)

The *erm* manages the actual nodes and VLAN IDs. When the *kuroyuri master* requests resources to *erm*, it decides which are the suitable actual nodes according to the number of network interfaces or the type of network interface media described in the configuration file, and then assigns the nodes to the *kuroyuri master*. After node assignment, the *erm* locks the nodes to prevent assigning the same node to another experiment. Moreover, it is able to manage defective nodes by avoiding their assignment to any experiment.

Node initiator (ni)

The *ni* is used to setup the node software for the experiment. It works on nodes booted from the network and assimilated diskless systems. It manages disk images installed on the node's harddisk partition by downloading these images from an FTP server whose address is provided by the *kuroyuri master*, and then saves the downloaded image onto the local harddisk.

Wake on LAN Agent (wol agent)

In the environment we assumed, actual nodes boot-up using WoL. WoL uses broadcast to send its magic packet. Sometimes there are multiple management segments. Since WoL magic packets don't go beyond local segment, a function to relay magic packets to remote management segments was implemented.

5.7 Experiment steps with SpringOS

Using SpringOS, all the users have to perform is preparing an experiment configuration file. Figure 5 shows the overview of SpringOS behavior. SpringOS will execute experiments according to the configuration file, which drives experiments with the following steps:

1. Loading of user configuration file

Users have to write the network topology of the experiment, and information to needed set up the nodes as well as the execution scenario in a configuration file. This file must be loaded by *kuroyuri master*, and processed to set up the environment and execute the experience scenario.

2,3. Nodes request and assignment to an experiment

Kuroyuri master requests nodes following the criteria described in the configuration files. Then *erm* assign nodes which have enough interfaces for the experiment to *kuroyuri master*. In compensation of nodes which are in failure, users can acquire more nodes than described in the configuration file as spare nodes. The number of spare nodes is configurable.

4.5. Booting nodes for the experiment

Kuroyuri master requests booting assigned nodes to *wol agent*. Then *wol agent* send WoL magic packets to specified nodes.

6. Software settings of the experiment nodes

Ni on each node for experiment communicates with *Kuroyuri master* to know the disk image for the node, which is described in user's configuration. Then *ni* downloads the disk image and writes the it to the harddisk.

7. Building topology for the experiment

The topology for an experiment is built by describing the desired topology in the configuration file. *Kuroyuri master* configure the switches of the experiment network to build the topology virtually using VLANs.

8. Execution of an experiment according to the scenario

Kuroyuri master sends the scenario for each node; when a node receives the scenario, *kuroyuri slave* begins to execute the scenario. When it needs rendezvous with other nodes, it waits for control messages from *kuroyuri master*.

6. EXPERIMENTS WHICH HAVE BEEN EXECUTED ON STARBED

StarBED was used by many researchers, and these researchers obtained results using the StarBED.

In [9], the authors investigate impact of group communications with multicast to ISP backbones. To build realistic ISP backbones and end nodes, they used over 1000 nodes including virtual machines. Virtual machines are used to generate user traffic by emulating user behavior, and physical nodes are used as parts of core networks.

In [5], the authors propose introducing peer-to-peer communication to Multi-player Online Games (MOGs). The authors evaluated their software on StarBED after evaluating it on a laboratory-level testbed. In this research, a topology with 300 nodes was used, which is difficult to build on a laboratory-level testbed.

The authors of [12] propose the hierarchical IP traceback architecture. They plan to evaluate their implementation using StarBED. The topology for experiment should have many Autonomous Systems(AS), so many actual nodes are needed.

And many other experiments were executed on StarBED as follows:

- L2 Switch benchmarks with multicast traffic
- Observation of TCP behavior
- Comparison of behavior between hardware routers manufactured by popular vendors and open source software routers running on PCs
- Emulating wireless networks on wired networks
- Emulating G4 cellular phone networks
- Overlay networks

These experiments required an environment such as StarBED. Users chose StarBED because of the following facilities offered:

- Running real implementations of applications on a large number of nodes
- Accessing the console of all nodes from one screen
- Using their equipments in experiments
- Introducing real traffic via wide-band external networks

The fact that many experiments have been made on StarBED shows the need for StarBED and its importance.

7. CONCLUSION AND FUTURE WORK

Software simulators and laboratory-level testbeds are a good way to evaluate new ideas and implementations on the development process of new technologies. The suitable methods differ according to the development stage. For proving new ideas, software simulation is suitable, because it can simulate the algorithms and surrounding environment which is assumed by the user to function correctly without disturbances. Laboratory-level testbeds are suitable to evaluate the real implementations after evaluation of the idea using software simulation. In this stage, we can evaluate the implementation including bugs and disturbances on the small testbed using actual nodes. However, the difference between the laboratory-level testbed and the real environment is often large. Therefore, we consider that a testbed located in between a laboratory-level testbed and the Internet is needed.

We proposed and implemented StarBED which is a cluster composed of 512 PCs dedicated to experiments, and SpringOS which is a supporting software for making experiments on StarBED. Using StarBED and SpringOS, we can make larger topology than general methods, and the result of the experiments is more realistic. Each node on StarBED can run 10 virtual machines with VMware, which enables to construct a 5120-node topology.

As the Internet continues to grow, a testbed which enables to build large-scale topology will be more important. We continue the development of our tools to suit next generation networks. As physically approach, we add actual nodes and develop network facilities. Furthermore we'll discuss using virtual node technologies to experiments.

Using virtual nodes, it is possible to build larger topologies for experiments, however the realism tends to be reduced. A method to use virtual nodes without reducing the reality is quite important.

Sensor networks and ubiquitous networks become hot topics, and practical experiments for technologies on these networks are also needed. There are many thin nodes which connect using wireless networks in these networks. It is difficult to build an environment for experiments using many wireless nodes because it requires wide space to move the nodes and we should have to move many nodes at the same time. We are considering a method to emulate a wireless network on wired network for performing wireless experiments easily.

When we designed SpringOS, we discussed the general architecture of supporting software for experiment. Now we have more knowledge about experiments and testbeds, we'll redesign the general architecture of the supporting software based on this knowledge.

8. REFERENCES

- [1] Planetlab website. <http://www.planet-lab.org/>.
- [2] R. Droms. Dynamic Host Configuration Protocol, RFC2131. March 1997.
- [3] N. I. T. Group. *NIST Net network emulation package*. <http://www-x.antd.nist.gov/nistnet/>.
- [4] IEEE standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. Dec. 1998.
- [5] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *NetGames*, 2004.
- [6] Intel Corporation. *Preboot Execution Environment (PXE) Specification Version 2.1*, sep 1990.
- [7] B. Kantor. BSD Rlogin, RFC1282. December 1991.
- [8] Ken-ichi Chinen, Toshiyuki Miyachi and Yoichi Shinoda. A rendezvous in network experiment — case study of kuroyuri. In *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(Tridentcom)*, Mar. 2006.
- [9] E. Muramoto, T. Yoneda, A. Nakamura, M. Misumi, T. Miyachi, and Y. Shinoda. Report on a method of simulating multicast group communication on the internet. Towards peta-bit ultra networks, Sept. 2003.
- [10] National Institute of Information and Communications Technology. <http://www.nict.go.jp/index.html>.
- [11] National Institute of Information and Communications Technology. *JGN2 HomePage*. <http://www.jgn.nict.go.jp/>.
- [12] M. Oe, Y. Kadobayashi, and S. Yamaguchi. An implementation of a hierarchical IP traceback architecture. In *Proceedings of IPv6 Workshop, SAINT 2003, Orland, USA*, Jan. 2003.
- [13] Raritan, Inc. <http://www.raritan.com/>.
- [14] R. Ricci, C. Alfeld, and J. Lepreau. A Solver for the Network Testbed Mapping Problem. In *SIGCOMM Computer Communications Review*, volume 33, No.2, pages 65–81, Apr. 2003.
- [15] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, 1997.
- [16] The StarBED Project. <http://www.starbed.org/>.
- [17] The VINT Project. *Network Emulation with the NS Simulator*. <http://www.isi.edu/nsnam/ns/ns-emulation.html>.
- [18] The VINT Project. *The VINT Project*. <http://www.isi.edu/nsnam/vint/index.html>.
- [19] The VINT Project. *The ns Manual*. Apr. 2002. <http://www.isi.edu/nsnam/ns/ns-documentation.html>.
- [20] Toshiyuki Miyachi and Ken-ichi Chinen and Yoichi Shinoda. Automatic configuration and execution of internet experiments on an actual node-based testbed. In *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(Tridentcom)*, Feb. 2005.
- [21] VMware. <http://www.vmware.com/>.
- [22] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. pages 255–270, Boston, MA, Dec. 2002. USENIXASSOC.
- [23] WIDE Project. <http://www.wide.ad.jp/>.
- [24] T. Ylonen. SSH - secure login connections over the internet. In *Proceedings of the Sixth USENIX Security Symposium*, pages 37–42, Jul 1996.