International Journal of Informatics Society

10/22 Vol. 14 No.2 ISSN 1883-4566

Informatics Society

Editor-in-Chief: Hiroshi Inamura, Future University Hakodate

Associate Editors: Katsuhiko Kaji, Aichi Institute of Technology
Yoshia Saito, Iwate Prefectural University
Takuya Yoshihiro, Wakayama University
Tomoki Yoshihisa, Osaka University

Editorial Board
Hitoshi Aida, The University of Tokyo (Japan)
Huifang Chen, Zhejiang University (P.R.China)
Christian Damsgaard Jensen, Technical University of Denmark (Denmark)
Teruo Higashino, Kyoto Tachibana University (Japan)
Tadanori Mizuno, Aichi Institute of Technology (Japan)
Jun Munemori, The Open University of Japan (Japan)
Yuko Murayama, Tsuda University (Japan)
Ken-ichi Okada, Keio University (Japan)
Norio Shiratori, Chuo University / Tohoku University (Japan)
Ian Wakeman, University of Sussex (UK)
Ismail Guvenc, North Carolina State University (USA)
Qing-An Zeng, North Carolina A&T State University (USA)
Tim Ziemer, University of Bremen (Germany)
Justin Zhan, University of Cincinnati Computer Science Faculty (USA)
Xuyun Zhang, Macquarie University (Australia)

Aims and Scope

The purpose of this journal is to provide an open forum to publish high quality research papers in the areas of informatics and related fields to promote the exchange of research ideas, experiences and results.

Informatics is the systematic study of Information and the application of research methods to study Information systems and services. It deals primarily with human aspects of information, such as its qu ality and value as a resource. Informatics also referred to as Information science, studies t he structure, algorithms, behavior, and interactions of natural and a rtificial systems that store, process, access and communicate information. It also develops its own conceptual and theoretical foundations and utilizes foundations developed in other fields. The advent of computers, its ubiquity and ease to use has led to th e study of info rmatics that has computational, cognitive and social aspects, including study of the social impact of information technologies.

The characteristic of informatics' context is amalgamation of technologies. For creating an informatics product, it is necessary to integrate many technologies, such as mathematics, linguistics, engineering and other emerging new fields.

Guest Editor's Message

Akihiro HAYASHI

Guest Editor of the Forty-first Issue of the International Journal of Informatics Society

We are delighted to have the Forty-first issue of the International Journal of Informatics Society (IJIS) published. This issue includes selected papers from the Fifteenth International Workshop on Informatics (IWIN2021), which was held online, Sept. 12-13, 2021. The workshop was the fifteenth event for the Informatics Society, and was intended to bring together researchers and practitioners to share and exchange their experiences, discuss challenges and present original ideas in all aspects of informatics and computer networks. In the workshop, 29 papers were presented in seven technical sessions. The workshop was successfully finished with precious experiences provided to the participants. It highlighted the latest research results in the area of informatics and its applications that include networking, mobile ubiquitous systems, data analytics, business systems, education systems, design methodology, intelligent systems, groupware, and social systems.

Each paper submitted to IWIN2021 was reviewed in terms of technical content, scientific rigor, novelty, originality, and quality of presentation by at least two reviewers. Through those reviews, 19 papers were selected for publication candidates of IJIS Journal, and they were further reviewed as a Journal paper. We have three categories of IJIS papers, Regular papers, Industrial papers, and Invited papers, each of which was reviewed from different points of view. This volume includes papers among those accepted papers, which have been improved through the workshop discussion and the reviewers' comments.

We publish the journal in print as well as in an electronic form over the Internet. We hope that the issue would be of interest to many researchers as well as engineers and practitioners over the world.

Akihiro HAYASHI received his MBA and Ph.D. from the University of Tsukuba, Tokyo, Japan in 1997 and 2010, and his Ph.D. in Software Engineering from Nanzan University, Japan in 2019. His professional career includes global companies such as Motorola USA, NTT, and IBM Japan. In 2018 he has become a Professor of Department of Information Design, Faculty of Informatics, Shizuoka Institute of Science and Technology. He received the Annual Best Paper Award from Software Interprise Modeling (SWIM) of the Institute of Electronics, Information and Communication Engineers (IEICE) in 2005 and 2008, 39th Quality and Technology Award from the Japanese Society for Quality Control (JSQC) in 2010, Best Industrial Award from IWIN2018, and Industrial Paper Award from IWIN2021. His current research interests include process improvement, quantitative project management, operating research and management strategy. He is a member of IEICE and INFSOC.

Industry Paper

Verification of Shell Script Behavior by Comparing Execution Log

Hitoshi Kiryu[†], Satoshi Suda[‡], Shinpei Ogata*, and Kozo Okano*

[†]Graduate School of Engineering, Shinshu University, Japan
21w2025g@shinshu-u.ac.jp
[‡]Advanced Technology R&D Center of Mitsubishi Electric
Suda.Satoshi@ay.mitsubishielectric.co.jp
*Faculty of Engineering, Shinshu University, Japan
{ogata, okano}@cs.shinshu-u.ac.jp

Abstract - Scripts written in shell languages including Bash are widely used to automate tasks on UNIX families such as Linux. These scripts, especially those used for automating tasks, are often used continuously even after the operating environment such as OS is updated. The behavior of the shell scripts may change due to OS updates which also usually include commands' upgrades. For this reason, developers must in advance know the changes of the commands used in the script and also check each script in the OS, for every time of the releases. It also produces the cost of debugging scripts and the associated commands. This paper proposes a method to verify if the behavior of a script does not change between two different OS versions. It also detects the cause of the difference. An automated tool for the proposed method is also presented. The proposed method embeds commands which generate execution logs into the scripts and executes those scripts on two different OS versions. The tool compares the generated log files from each OS, and if the behavior is changed, it presents the commands that is cause of differences to the developers. As a result of evaluation of the proposed method, we confirmed that the proposed method can verify the different behavior and detect the commands that cause the difference for a simple example. In addition, the result shows that it was not possible to detect the cause of commands which behavior changes in the area that does not appear in the standard output. In order to detect these commands as the cause, it is necessary to collect logs from a different approach than the standard output.

Keywords: Behavior verification, Debug, Execution log, Shell script, Bash

1 INTRODUCTION

1.1 Background

Bash [1] is a typical shell language that runs on UNIX OSes such as Linux. It is a POSIX-compliant Bourne Shell language with extended history and aliasing features, and it is mainly adopted as default login shell in many UNIX OSes. These scripts written in shells are widely used in corporate business systems for automating tasks. The behavior of shell scripts written in the Bash may change due to OS updates and associated command upgrades. For Instance, because of commands upgrades, it is possible to lose access to environment variables, which are a set of variables used to configure the shell and various commands. It is explained in Section 3. Therefore, developers must understand the commands which are executed in the scripts and the changes in the OS specifications when the OS is upgraded. It produces a lot of costs for debugging scripts and the associated commands.

In this paper, we propose a method to support developers in solving problems that occur when updating the OS by verifying whether the behavior of Bash scripts is equivalent between two different version OSes and detecting the cause of the inequality. Although there are many distributions of Linux, in this paper, we focus on CentOS, which has been used for business systems in companies.

1.2 Related Work

Various studies [2]–[5] have been conducted to localize bugs related to these problems. These studies proposed methods to identify statements that cause bugs by using bug reports, trace information, visualization. These studies aim to fix and identify bugs, while our purpose is to identify the causes of the differences in behavior.

A similar role to Bash scripts is played by Dockerfile that describes a series of procedures to build containers. In the field of micro-services architecture, services based on Docker are promising. The services are defined by Doker files, and the files sometimes contain complex scripts. The analysis of Dockerfile has been studied particularly from the viewpoint of developer support. For instance, Kaisei Hanayama and his co-authors [6] have suggested a method to create a code-completion tool for writing Dockerfiles is proposed by using machine learning of Dockerfiles on Github. In another study [7], a survey of Self-Admitted Technical Debt(SATD) in Dockerfile was conducted for an actual DockerHub project. From the survey, patterns of SATD in Dockerfile and their proportion are revealed.

In addition, CBMC [8] and JBMC [9] which is based on CBMC are software model checking tools for verifying the be-

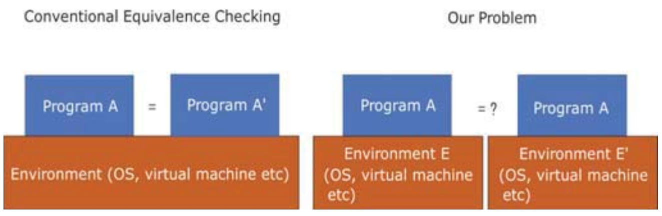

Conventional Equivalence Checking
Program A
=
Program A'
Environment (OS, virtual machine etc)
Our Problem
Program A
=?
Program A
Environment E (OS, virtual machine etc)
Environment E' (OS, virtual machine etc)

Figure 1: Differences from our previous study

havior of programs. These tools take a program and assertions as input and use bounded model checking [10] to verify that the assertions are valid. Although it is possible in theory to verify the behavior using these tools, they are not tools for directly checking changes of behavior, which is the focus of this paper, since they are for verifying whether the behavior is as specified.

The author's group has been researching behavioral equivalence verification of programs [11]–[13]. In these studies, we verified the differences in the behavior of modified programs as shown on the left side of Fig. 1. The problem that we will address in this paper is the difference in the behavior in changes in the environment, as shown on the right side of Fig. 1. Therefore, the methods used in these studies cannot be applied to this problem.

The authors did not find any research on the behavioral equivalence of Bash when the environment changes. Therefore, it would be useful for developers to create a tool to verify whether a script behaves equally across operating systems. Bash has a debugging tool [14], but these tools do not work with inter-pipeline output and scripts called within a script.

1.3 The Approach

In the proposed method, we first embed commands which generate execution log in script files to be verified. The log generating commands(Loggers) are programs that log standard output, output between pipelines, and variable assignments. Scripts with Loggers embedded are executed to generate and retrieve the logs. This process is performed on each of the two target operating systems built on VirtualBox [14], and the obtained logs are compared to detect differences in behavior. If the logs match, it is assumed that there is no change in the behavior. If the logs do not match, we present where the logs differ as the commands which is cause of the difference(hereinafter called "cause command") to the developer. The information to be presented is executed command, execution log, and stack trace information of the function or script.

We conducted three evaluations of the proposed method. In the first evaluation, we intentionally created commands with different behaviors and experimented to detect the difference in behaviors and to identify the cause. In the second evaluation, we experimented to see if the proposed method can detect the difference in the behavior of a script containing the command sudo, whose behavior was changed by updating the command in the past and to identify the cause of the change. In the third evaluation, we applied the proposed method to a script, which access given URL via a proxy server imitatively downloading files in secured network.

As a result of our evaluation experiments, we confirmed that our proposed method is helpful for developers in verifying whether behaviors changed. We also found that it is possible to detect differences in the behavior of commands with small side-effects, e.g. commands that only calculate and return arguments. However, it is difficult to directly identify the cause of the differences in commands with side-effects, such as commands that affect the environment variables.

In the following sections, Section 2 describes the techniques related to this research, such as bug localization and lexical and syntactic analysis. In Section 3 we describe the motivating example, and in Section 4 we describe the proposed method. In Section 5 we present the results in the evaluation experiments and in Section 6 we discuss the results in the experiments. Finally, we conclude in Section 8.

2 PRELIMINARIES

2.1 Bug Localization

Various methods are being researched to identify software bugs. In the paper [15], a method to identify statements with bugs from the program spectrum which is execution records of success or failure of software test cases is studied. It identifies the statement as the most suspicious statement that includes bugs. The suspicion of statement is calculated from program spectrum information using newly proposed metrics. Another research [16] proposes a method to support identifying defects by visualizing data transitions and execution flow with dynamic analysis from Java source code. This research targets to detect functional defects in logic and visualizes their details the detailed processing flow and dependencies of variables to assist users in understanding. In evaluation experiments on the tool, it was confirmed that the use of the tool can reduce the time required to locate defects.

2.2 Model Checking

Model checking is a method to verify that the behavior of a system satisfies the specification. A model checking tool judges whether or not a specification is valid for a state transition model that represents the behavior of a system. In many model checking tools, specifications are described by temporal logic such as CLT and LTL. There is a study [17] that applies model checking to real system development. In this study, a model checking tool UPPAAL [18] was used for the development of medical infusion pumps to formalize the models and specifications, verify the safety of the models and generate code from the verified models. Also, since the description of the state transition model and the specification is essential for model checking, the research [19] has been conducted to support the description of the model and

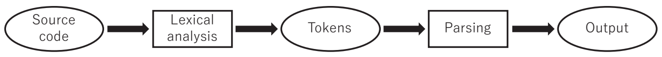


Figure 2: Outline of lexer and parser

the specification for engineers who do not have such knowledge. Automatic generation of checking models from the state transition table of the system and the table which summarizes the actions which occur in each state.

2.3 Lexical and Syntactic Analyzer

Lexical and syntactic analysis is a series of parsing processes in programming languages, as shown in Fig. 2. A series of pipelined syntactic analyzers check that the token sequence given by the lexical analysis satisfies the defined grammar. We use PLY (Python Lex-Yacc) [20] as a lexical and syntactic analyzer to embed Loggers into Bash scripts given as input. PLY is an implementation of Lex, a lexical analyzer for Python, and Yacc which is a syntactic analyzer. In the lexical analysis part, regular expressions are used to define the string-to-token conversion rules, and in the syntactic analysis part, the LALR(1) grammar [21] is used to define the syntax rules. A syntax tree is constructed from the given rules and parsed for a given sentence.

3 THE MOTIVATION EXAMPLE

Listing 1: foo.bash

```
VAR='baz'
export VAR
sudo bash bar.bash
```

Listing 2: bar.bash

```
echo ${VAR}
```

The two Bash scripts in the Listings 1 and 2, foo.bash and bar.bash, are example scripts whose behavior changes depending on OS versions. In CentOS5 and later versions, the behavior of the sudo command in foo.bash has changed due to the upgrade of the command. Versions that are older than CentOS5 output the string "baz," while in CentOS5 and later versions not output a blank line. This is because, in 1.6 and earlier versions of the sudo, the command was able to preserve environment variables when executed by sudo, however, in 1.7 and later versions, it is necessary to specify the '-E' option to preserve environment variables. The environment variable is not referred due to the change of the command specification of the newer version. This change causes the difference in behavior between versions older than CentOS5 using version 1.6 bash and later versions of CentOS.

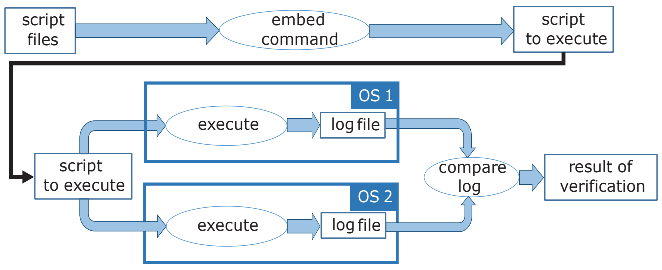


Figure 3: Outline of the proposed method

4 THE PROPOSED MOTHOD

The proposed method generates execution logs of Bash scripts and judges the difference in behavior by comparing the logs. The execution log contains the standard output, the output between pipelines, and the variable assignments along with the executed commands. The stack trace information of the command is also stored to make it easier to identify the cause.

4.1 Outline of the Proposed Method

The schematic diagram of the proposed method is shown in Fig. 3. The input and output of the proposed method are shown below.

- Input : Script file to be verified
- Output: verification results (the behavior is different and which command behaves differently)

The procedure of the proposed method is as follows.

(1) It embeds the Logger into the Bash script given as input. The Logger refers to a program that logs the standard output, output between pipelines, and variable assignment. The details of each program are described below.

(2) It executes script embedded commands on each of the two operating systems.

(3) It compares the obtained logs and judges whether the behavior is different. If the behavior is different, the command that causes the difference and its log information is presented.

We have created a tool that automatically executes the above procedure.

4.2 Execution Log

The format of the execution log is shown in Listing 3.

Listing 3: Abstraction of the log

```
<log identifier>:<commands>, line: <
    lineno>, stack: <stack trace>
<output log>
:<log identifier>
```

1. <log identifier> : "assignment" if the log is an assignment, "command" if the log is a command execution

2. <commands> : Executed commands
3. <line> : line number of executed command
4. <stack trace> : Stack trace information of the script and function when the command is executed.
5. <output log> : Assignment log and standard output log.

4.3 Log Generating Command(Logger)

The following four Loggers are embedded in the script. The behavior of each is shown below.

- Standard output log command : Generate standard output log
- Assignment Log command : Generate Variable Assignment Log
- Stack push command : Record Stack trace information
- Stack pop command : Record Stack trace information

4.3.1 Standard Output Log Command

This command logs the standard output and the output between pipelines. It also takes two arguments, a command to be executed and its line and logs the string to be executed. The standard output log command is "stdout_logger" in Listing 5. This logging command can generate the standard output log by pipelining this command to a line that does the normal standard output as shown in the examples in Listings 4 and 5.

Listing 4: Before embedding example for standard output

```
echo hello
```

Listing 5: After embedding example for standard output

```
echo hello | stdout_logger 'echo
    hello' 1
```

The log shown in Listing 6 is generated by executing the script shown in Listing 5.

Listing 6: Log example for standard output

```
command:echo hello, line: 1, stack:
hello
:command
```

Also, the output between pipelines can be logged like the standard output by embedding it as Listings 7 and 8.

Listing 7: Before embedding commands for pipeline

```
commandA | commandB
```

Listing 8: After embedding commands for pipeline

```
commandA | stdout_logger 'commandA' |
    commandB
```

4.3.2 Assignment Log Command

In this step, we generate log of assignments to variables. The assignment log command is "assign_logger" in Listing 10. It takes the assignment command to be executed, the variable name, and the value of the variable as arguments and records them in the execution log. When the assignment command is taken as an argument, single quotes are escaped to avoid the expansion of variables. The Listings 9 and 10 show an example of before and after embedding the log.

Listing 9: Before embedding commands

```
VAR='baz'
```

Listing 10: After embedding commands

```
VAR='baz'
assign_logger 'var='\''baz'\''' VAR "
    $VAR" 1
```

Running the script ex.bash that executes Listing 10 will generate the log shown in Listing 11.

Listing 11: Log example for assignment

```
assignment:VAR='baz', line: 1, stack:
VAR=baz
:assignment
```

4.3.3 Stack Push, Stack Pop Command

In order to identify the execution path of commands, command to stack push and pop records stack trace information into a text file. At the start of the script or function, their name is pushed. At the end of that, the pushed name is popped. In addition, the call command string is pushed just before the script or function call, and the pushed string is popped after the calling. Listings 14 and 15 shows the result of embedding for ex.bash and test.bash shown in Listings 12 and 13.

Listing 12: ex.bash before embedding commands

```
test.bash 'test'
```

Listing 13: test.bash before embedding commands

```
echo ${1}
```

Listing 14: ex.bash after embedding commands

```
push_stack ex.bash
push_stack 'test.bash '\''test'\'''
test.bash 'test'
pop_stack
pop_stack
```

Listing 15: test.bash after embedding commands

```
push_stack test.bash
echo ${1} | stdout_logger 'echo ${1}'
    1
pop_stack
```

Listing 14 generate the log shown in Listing 16.

Listing 16: Log example for stack and push command

```
command:echo ${1}, line: 1, stack: ex
    .bash->test.bash 'test'->test.bash
test
:command
```

The log "ex.bash->test.bash 'test'->test.bash" which is the stack trace information following "stack:" in the log, indicates that the log was generated in test.bash called by test.bash 'test' from within ex.bash.

4.4 Embedding Commands

Embed Loggers into scripts given as input. The tool embeds each command in the following cases.

- Start of the script or function : Stack Push Command
- End of the script or function : Stack Pop Command
- The line just before the call of script or function : Stack Push Command
- The line just after the call of script or function : Stack Pop Command
- Pipeline : Standard Output Log Command
- Variable Assignment : Assignment Log command

4.5 Executing Scripts

Scripts which is embedded Loggers are executed. In running the script, copy it to OS built using VirtualBox and run. Verification can be done within a single machine.

4.6 Comparing Logs

The tool compare 2 execution logs generated by running. Execution log contains following information.

- Executed command
- Stack trace information
- Log identifier of standard output or assignment
- Log of standard output or assignment

In case that no difference between the logs is detected, the tool judge that behavior is consistent. If not so, the tool judge that behavior is changed and display different logs as causes.

4.7 Implementation

We create the tool based on the proposed method. The tool gets scripts as input, embed commands, run embedded scripts, compare logs, and display results automatically.

The Loggers were embedded using a lexical and syntactic analyzer written in Python Lex-Yacc. It embeds Loggers if the script matches defined grammars. For example, if assignment operator "=" appears in a single command, it is recognized as an assignment and the assignment log command is embedded. The four Loggers and the log comparison program were implemented using C++.

Target OSes of verification are built on VirtualBox on Windows. Scripts given as input are embedded Loggers on Windows and copied to target OSes, then the execution logs are provided by running the scripts on each OS. The logs are copied to the Host OS and verified. By executing the script on target OSes using SSH from the Host OS, this method is executed automatically.

5 EVALUATION EXPERIMENT

In order to evaluate the proposed method, we conducted evaluation experiments for the following three scripts to see if it is possible to detect and identify the cause of different behaviors.

1. A script that executes commands created to behave differently between operating systems.
2. A script containing the sudo command described in Section 3
3. A script that imitates a situation of building environments via proxy

The environment and OS used for these evaluation experiments are as follows.

- Host OS : Windows10 Pro
- VirtualBox ver 5.2.18 r124319
- Guest OS 1: CentOS 4.6
- Guest OS 2: CentOS 8.2.2004

Both experiments followed steps below.

1. Embed commands into scripts given as input on the Host OS.
2. Copy the Scripts which is embedded commands to the target OSes.
3. Execute the scripts on each target OSes and copy generated logs to the Host OS.
4. Compare the obtained logs and get verification results.

5.1 Experiment 1

We prepared a command “sample” and a simple script to execute the command for each OS. The command “sample” takes two integer arguments and outputs the result of addition on CentOS4.6 and multiplication on CentOS8.2. The proposed method is applied to the scripts we created and conducted experiments to evaluate whether the tool can detect a difference in behavior between scripts with a different function, and whether “sample” can be identified as the causative command.

The script used for the experiment, “ex.bash,” is shown in Listing 17.

Listing 17: ex.bash

```
result=$(sample 2 3)
echo ${result}
```

Script after embedding the commands into ex.bash is shown in Listing 18.

Listing 18: ex.bash after embedding command

```
push_stack ex.bash
result=$( sample 2 3 | stdout_logger
    ’sample 2 3’ 1 )
assign_logger ’result=$( sample 2 3
    )’ result "$result" 1
echo ${result} | stdout_logger ’echo
    ${result}’ 2
pop_stack
```

Obtained logs by executing the above script on each target OSes are shown in Listings 19 and 20.

Listing 19: Log on CentOS4.6 in Experiment 1

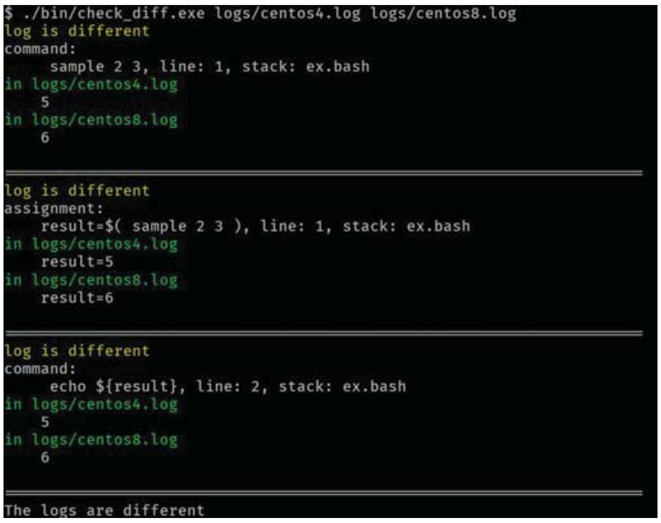
```
command: sample 2 3, line: 1, stack:
    ex.bash
5
:command

assignment:result=$( sample 2 3 ),
    line: 1, stack: ex.bash
result=5
:assignment

command: echo ${result}, line: 2,
    stack: ex.bash
5
:command
```

Listing 20: Log on CentOS8.2 in Experiment 1

```
command: sample 2 3, line: 1, stack:
    ex.bash
6
:command
```



```
$ ./bin/check_diff.exe logs/centos4.log logs/centos8.log
log is different
command:
     sample 2 3, line: 1, stack: ex.bash
in logs/centos4.log
    5
in logs/centos8.log
    6

================================================================
log is different
assignment:
    result=$( sample 2 3 ), line: 1, stack: ex.bash
in logs/centos4.log
    result=5
in logs/centos8.log
    result=6

================================================================
log is different
command:
     echo ${result}, line: 2, stack: ex.bash
in logs/centos4.log
    5
in logs/centos8.log
    6

================================================================
The logs are different
```

Figure 4: Result of Experiment 1

```

assignment:result=$( sample 2 3 ),
    line: 1, stack: ex.bash
result=6
:assignment

command: echo ${result}, line: 2,
    stack: ex.bash
6
:command
```

The results of the comparison of the two logs are shown in Fig. 4.

Different logs are suggested. According to the results in Fig. 4, differences in the script behavior were detected. The cause of the different behavior of the script “ex.bash” is the command “sample,” and the first presented log shows the command “sample 2 3.” Therefore, the difference in behavior was detected and the command that caused the difference was identified.

5.2 Experiment 2

The proposed method is applied to scripts shown in Listings 1 and 2, which are indicated in Section 3 and conducted experiments to evaluate whether the tool can detect differences in the behavior of the scripts and identify the command “sudo” as the cause of the difference.

The Bash scripts after embedding the commands into the scripts are shown in Listings 21 and 22.

Listing 21: foo.bash after embedding command

```
push_stack foo.bash
VAR=’baz’
assign_logger ’VAR=’\’’’baz’\’’’ VAR "
    $VAR" 1
```

```
export VAR | stdout_logger 'export
    VAR' 2
push_stack 'sudo bash ./bar.bash'
sudo bash ./bar.bash
pop_stack
pop_stack
```

Listing 22: bar.bash after embedding command

```
push_stack bar.bash
echo ${VAR} | stdout_logger 'echo ${
    VAR}' 1
pop_stack
```

logs generated by executing the scripts shown in Listings 21 and 22 on each target OSes are shown in Listings 23 and 24.

Listing 23: Log on CentOS4.6 in Experiment 2

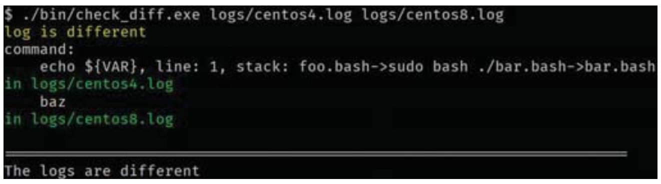
```
1  assignment:VAR='baz', line: 1, stack:
      foo.bash
2  VAR=baz
3  :assignment
4
5  command:export VAR, line: 2, stack:
     foo.bash
6  :command
7
8  command:echo ${VAR}, line: 1, stack:
     foo.bash->sudo bash ./bar.bash->
     bar.bash
9  baz
10 :command
```

Listing 24: Log on CentOS8.2 in Experiment 2

```
1  assignment:VAR='baz', line: 1, stack:
      foo.bash
2  VAR=baz
3  :assignment
4
5  command:export VAR, line: 2, stack:
     foo.bash
6  :command
7
8  command:echo ${VAR}, line: 1, stack:
     foo.bash->sudo bash ./bar.bash->
     bar.bash
9
10 :command
```

There is a difference in the 9th line of each log: CentOS4.6 outputs "baz," but CentOS8.2 outputs an empty string. This is the same as the result mentioned in Section 3.

The results of the comparison of the logs are shown in Fig. 6. From the results in Fig. 6, the difference in behavior is detected by comparison of the execution logs. However, the command "echo $VAR" suggested as a cause is not true cause as described in Section 3. The true cause command "sudo" wasn't identified as a cause of the difference in the behavior of the script.



```
$ ./bin/check_diff.exe logs/centos4.log logs/centos8.log
log is different
command:
    echo ${VAR}, line: 1, stack: foo.bash->sudo bash ./bar.bash->bar.bash
in logs/centos4.log
    baz
in logs/centos8.log

The logs are different
```

Figure 5: Result of Experiment 2

5.3 Experiment 3

The following script shown in Listing 25 updates packages with package managing command "yum" via a proxy server, which is described by environment variables including "http_proxy." Some enterprises and institutions often create proxy servers to protect internal networks from cyberattacks, and users in organizations access external networks via proxy servers. A tool to manage packages like "yum" in the script is necessary command for building server environment. This script imitates such a situation.

Listing 25: Example script of updating command via a proxy server

```
http_proxy="http
    ://192.168.56.1:3128/"
https_proxy="https
    ://192.168.56.1:3128/"
ftp_proxy="ftp://192.168.56.1:3128/"
export http_proxy https_proxy
    ftp_proxy
sudo yum update -y
```

In this Experiment, We applied the method to a script that is the essence of the script in Listing 25. The script extracts the title of the given URL "http://example.com" with the curl command via a given proxy server. In order to evaluate that the tool can detect differences in behaviors and identify cause command. This script just extracts the title of a given URL via proxy server.

Listing 26: test.bash

```
export http_proxy="http
    ://192.168.56.1:3128/"
echo "title is" $(sudo curl -sS "
    example.com" 2>&1 | grep -Po "(?<=
    title>)(.+)(?=</title>)")
```

The script embedded command is shown in Listing 27.

Listing 27: test.bash after embedding command

```
push_stack test.bash
export http_proxy="http
    ://192.168.56.1:3128/"
assign_logger 'http_proxy="http
    ://192.168.56.1:3128/"' http_proxy
     "$http_proxy" 1
echo "title is" $( sudo curl -sS "
    http://example.com" 2>&1 |
    stdout_logger 'sudo curl -sS "http
    ://example.com"' 2 | grep -Po
    "(?<=title>)(.+)(?=</title>)" |
    stdout_logger 'sudo curl -sS "http
    ://example.com" 2>&1 | grep -Po
    "(?<=title>)(.+)(?=</title>)"' 2 )
     | stdout_logger 'echo "title is "
     $(sudo curl -sS "example.com"
    2>&1 | grep -Po "(?<=title
    >)(.+)(?=</title>)")' 2
pop_stack
```

Listings 28 and 29 show generated logs on the each OSes.

Listing 28: Log on CentOS4.6 in Experiment 3

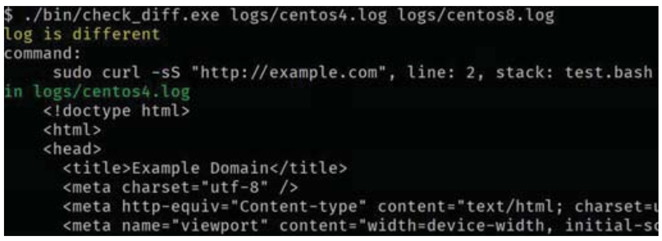
```
assignment:http_proxy="http
    ://192.168.56.1:3128/", line: 1,
    stack: test.bash
http_proxy=http://192.168.56.1:3128/
:assignment

command: sudo curl -sS "http://
    example.com", line: 2, stack: test
    .bash
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
    (Omission)
</html>
:command

command: sudo curl -sS "http://
    example.com" 2>&1 | grep -Po "(?<=
    title>)(.+)(?=</title>)", line: 2,
     stack: test.bash
Example Domain
:command

command: echo "title is" $(sudo curl
    -sS "example.com" 2>&1 | grep -Po
    "(?<=title>)(.+)(?=</title>)"),
    line: 2, stack: test.bash
title is Example Domain
```

Listing 29: Log on CentOS8.2 in Experiment 3



```
$ ./bin/check_diff.exe logs/centos4.log logs/centos8.log
log is different
command:
    sudo curl -sS "http://example.com", line: 2, stack: test.bash
in logs/centos4.log
    <!doctype html>
    <html>
    <head>
      <title>Example Domain</title>
      <meta charset="utf-8" />
      <meta http-equiv="Content-type" content="text/html; charset=u
      <meta name="viewport" content="width=device-width, initial-sc
```

Figure 6: Result of Experiment 3

```
assignment:http_proxy="http
    ://192.168.56.1:3128/", line: 1,
    stack: test.bash
http_proxy=http://192.168.56.1:3128/
:assignment

command: sudo curl -sS "http://
    example.com", line: 2, stack: test
    .bash
curl : (6) Could not resolve host:
    example.com
:command

command: sudo curl -sS "http://
    example.com" 2>&1 | grep -Po "(?<=
    title>)(.+)(?=</title>)", line: 2,
     stack: test.bash
:command

command: echo "title is" $(sudo curl
    -sS "example.com" 2>&1 | grep -Po
    "(?<=title>)(.+)(?=</title>)"),
    line: 2, stack: test.bash
title is
:command
```

The result is shown in Fig. 6.

The result shows that the tool detect changes in logs and the commands “sudo curl -sS "http://example"” are suggested as cause commands.

6 DISCUSSION

In the all Experiments, the difference in behavior between the script which executes commands with different behaviors is detected.

The cause command of the difference is identified in evaluation Experiment 1. In the experiment, differences in execution logs were detected in command substitution and assignment before they appear on standard output. Shell scripts that have complicated processing will almost certainly use such operators. Therefore, It is helpful for developers to find the difference before it comes out as standard output. This is also true for the pipeline which our method supports.

While, in the Experiment 2, the wrong command was suggested as the cause command. The reason is that the environment variables are not referred in the script called by the “sudo” command in CentOS8.2, and the difference in behavior is surfaced at the stage of “echo $VAR” which performs standard output. In the case of change in the behavior of such a command which has no standard output, differences in behavior are detected in standard output or assignment. This is because the proposed method generates execution logs which focus on standard output and assignment. Therefore, it will be difficult to directly identify such commands as the cause command. Similarly, it will be difficult to precisely identify the cause commands in case of commands like “sudo,” which has a function to affect shared resources such as environment variables changed, i.e. a command with a strong side effect.

In Experiment 3, The tool identified cause command “sudo”. The command “sudo curl -sS "http://example.com"” accesses the URL via proxy server, which is described by “http_proxy”. However, in CentOS8.2, the access to URL fails due to the command “sudo” doesn’t preserve environment variables. There are commands that refer to environment variables when performs, such as curl and tools to manage packages including yum and apt. These commands are generally used in building environment on server.

From result of the evaluations, Our proposed method can support developers in checking behavior changes in scripts.

7 FUTURE WORK

Defined grammars on lexical and syntax analyzers are simple. The analyzers cannot support complex grammars. Therefore, the tool needs to expand the grammars of analyzers.

Since this method only collects logs that appear in the standard output or assignment. If the behavior differs in areas that don’t appear in the standard output or assignment, e.g. signal trapping, it is not possible to verify whether the behavior changed.

The proposed method needs to evaluate the execution time and used memory for large and complex scripts such as recursive.

Thus, addressing these issues will be the main task in the future.

8 CONCLUSION

In this paper, we proposed a method to verify whether the behavior of shell scripts written in Bash is changed before and after OS upgrade and created a tool based on the method. The tool based on the proposed method can verify the behavior and identify the causes of the differences by comparing the execution logs of the shell scripts generated by log generating commands(Loggers). From evaluation experiments, we confirmed that our proposed method can verify differences in the behavior in standard output and assignment, and the method can support developers in verifying whether behaviors in scripts is changed before and after OS update efficiently. Furthermore, we conclude that it is difficult to precisely identify the cause commands in case of commands which have no standard output or strong side effects.

ACKNOWLEDGEMENT

Part of this work is supported by fund from Mitsubish Electric Corp.

The research is also being partially conducted as Grant-in-Aid for Scientific Research A (19H01102) and C (21K11826).

REFERENCES

[1] “GNU Bash,” https://www.gnu.org/software/bash/ (referred May 13, 2022).

[2] J.Nam, S.Wang, Y.Xi, and L. Tan: “A bug finder refined by a large set of open-source projects,” Information and Software Technology, Vol.112, pp.164–175 (2019).

[3] S.Kim, T.Zimmermann, K.Pan, and E.J.Whitehead Jr.: “Automatic Identification of Bug-Introducing Changes,” 21st IEEE/ACM International Conference on Automated Software Engineering (ASE’06), pp.81-90 (2006).

[4] S.Tsakiltsidis, A.Miranskyy, and E.Mazzawi: “Towards Automated Performance Bug Identification in Python,” 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp.132-139 (2016).

[5] K.Matsushita, M.Matsumoto, K.Ohno, T.Sasaki, T.Kondo, and H.Nakashima: “A Debugging Method Based on Comparison of Execution Trace,” Symposium on Advanced Computing Systems and Infrastructures (SACSIS), Vol.2011, pp.152-159 (2011) (in Japanese).

[6] K.Hanayama, S.Matsumoto, and S.Kusumoto: “Humpback: Code Completion System for Dockerfiles Based on Language Models,” In 1st Workshop on Natural Language Processing Advancements for Software Engineering(NLPaSE 2020), pp. 1-4 (2020).

[7] H.Azuma, S.Matsumoto, Y.Kamei, and S.Kusumoto: “Survey of Self-Admitted Technical Debt in Container Virtualization Technology,” IEICE technical report, Vol.120, No.193, pp.25-30 (2020) (in Japanese)

[8] E.Clarke, D.Kroening, and F.Lerda: “A tool for checking ANSI-c programs,” International Conference on Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2004, Lecture Notes in Computer Science, Vol.2988, pp.168-176 (2004).

[9] L.Cordeiro, D.Kroening, and P.Schrammel: “JBMC: Bounded Model Checking for Java Bytecode,” International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS 2019), Lecture Notes in Computer Science, Vol.11429, pp.219-223 (2019).

[10] A.Biere, A.Cimatti, E.Clarke, and Y.Zhu: “Symbolic Model Checking without BDDs,” In Proceedings of the Workshop on Tools and Algorithms for the Construction

and Analysis of Systems(TACAS 1999), Lecture Notes in Computer Science, Vol.1579, pp.193-207 (1999).

[11] K.Okano, R.Karashima, S.Harauchi, and S.Ogata: "Regression Verification for C Functions with Recursive Data Structure," International Journal of Informatics Society, Vol.11, No.2, pp.107-115 (2019).

[12] K.Okano, S.Harauchi, T.Sekizawa, S.Ogata, and S.Nakajima: "Consistency Checking between Java Equals and hashCode Methods Using Software Analysis Workbench," IEICE Transactions on Information and Systems, Vol.E102, No.8, pp.1419-1422 (2019).

[13] R.Karashima, S.Harauchi, S.Ogata, and K.Okano: "Proposal and evaluation for property verification for Java functions with recursive data structures by SAW," Proceedings of International Workshop on Informatics 2019 (IWIN2019), pp.155-162 (2019).

[14] "BASH Debugger," http://bashdb.sourceforge.net/

[15] "Oracle VM VirtualBox," https://www.virtualbox.org/ (referred May 13, 2022).

[16] C.Oo and H.Min Oo: "Spectrum-Based Bug Localization of Real-World Java Bugs," International Conference on Software Engineering Research, Management and Applications, pp.75-89 (2019)

[17] T.Sato, T.Katayama, Y.Kita, H.Yamaba, K.Aburada, and Naonobu Okazaki: "Development of TFVIS (Transitions and Flow VISalization) for Java Programs," Journal of Information Processing (JIP), Vol.59, No.4, pp.1137-1149 (2018) (in Japanese).

[18] B.Kim, A.Ayoub, O.Sokolsky, I.Lee, P.Jones, Y.Zhang, and R.Jetley: "Safety-assured development of the GPCA infusion pump software," 2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT), pp. 155-164, (2011).

[19] G.Behrmann, A.David, and K.G.Larsen: "A Tutorial on Uppaal," In Formal Methods for the Design of Real-Time Systems, Vol.3185, pp.200-237 (2004).

[20] T.Koike: "Model Checking Support Environment based on State Transition Matrix," SIGEMB, Vol.2008, No.116 pp.91-96 (2008) (in Japanese).

[21] "PLY (Python Lex-Yacc) — ply 4.0 documentation," https://ply.readthedocs.io/en/latest/ (referred May 13, 2022).

[22] F.DeRemer and T.Pennello: "Efficient Computation of LALR(1) Look-Ahead Sets," ACM Transactions on Programming Languages and Systems, Vol.4, No.4, pp.615–649 (1982).

(Received: October 30, 2021)
(Accepted: February 23, 2022)

Hitoshi Kiryu is a graduate student of Shinshu University. His areas of interest include formal verification.

Satoshi Suda Suda Satoshi received his M.E. degree in mathematical from Osaka University, Osaka, Japan, in 2016. He joined Mitsubishi Electric Corp. Currently he is a researcher of Solution Engineering Dept. at Advanced Technology R&D Center and mainly engaging in research on software development efficiency.

Shinpei Ogata is an Associate Professor at Shinshu University, Japan. He received his BE, ME, and PhD from Shibaura Institute of Technology in 2007, 2009, and 2012 respectively. From 2012 to 2020, he was an Assistant Professor, and since 2020, he has been an Associate Professor, in Shinshu University. He is a member of IEEE, ACM, IEICE, IPSJ, and JSSST. His current research interests include model-driven engineering for information system development.

Kozo Okano received his BE, ME, and PhD degrees in Information and Computer Sciences from Osaka University in 1990, 1992, and 1995, respectively. From 2002 to 2015, he was an Associate Professor at the Graduate School of Information Science and Technology of Osaka University. In 2002 and 2003, he was a visiting researcher at the Department of Computer Science of the University of Kent in Canterbury, and a visiting lecturer at the School of Computer Science of the University of Birmingham, respectively. Since 2020, he has been a Professor at the Department of Electrical and Computer Engineering, Shinshu University. His current research interests include formal methods for software and information system design. He is a member of IEEE, IEICE, and IPSJ.

Industry Paper

On the Effectiveness and Stability of Multi-channel Time Division Multiple Access Using LoRa

Ryotaro Sakauchi*, Shuto Ishikawa*, Yuki Eto*, Yota Nomoto*, Shunsuke Segawa*, Hikaru Yabe*, and Mikiko Sode Tanaka**

*Kanazawa Institute of Technology, Japan
**International College of Technology, Japan
*{b1905429, b1816880, b1918986, b1901724, c1016620, b1800744}@planet.kanazawa-it.ac.jp
**sode@neptune.kanazawa-it.ac.jp

Abstract -We are developing a bus operation management system using LoRa. The main part of the system is a bus location information system called "BusDoko System" which presents the current location of the bus. This system uses GPS and IoT SIM to collect real-time location information from running buses. We are currently in the process of changing to LoRa communication to reduce operational costs. In recent years, the number of users of the 920 MHz band has been increasing, making stable operation of the system more difficult. Therefore, in this paper, we studied how to enable stable communication even in such an environment. In a bus location system, the accuracy and stability of the system are important. Therefore, we constructed a multi-channel LoRa communication system using Time Division Multiple Access (TDMA) and a channel hopping method. This enables us to cope with changes in the communication environment over time. First, we confirmed the effectiveness of these methods through the results of indoor experiments. Next, we confirmed whether the system actually works in the outside, and we will introduce the results. From these results, we will report the normal operation of the system and its effectiveness against communication interference in the communication system.

Keywords: LPWA, LoRa, TDMA, Channel Hopping, Bus location

1 INTRODUCTION

Community buses play an important role in the transportation of local residents. However, the operation of it is depend on weather and traffic conditions. Therefore, it may give the bus users anxiety about whether the bus will arrive. In recent years, efforts have been made to develop and operate the bus location systems that present the current location of buses in order to remove unstable elements [1] - [3]. These use LoRa, which does not require communication costs, in order to reduce operating costs. Fig. 1 shows the overall bus location system, the "BusDoko System" which we are developed [4]. This system presents the current location of community buses Notty circling Nonoichi City, Ishikawa Prefecture, on a website.

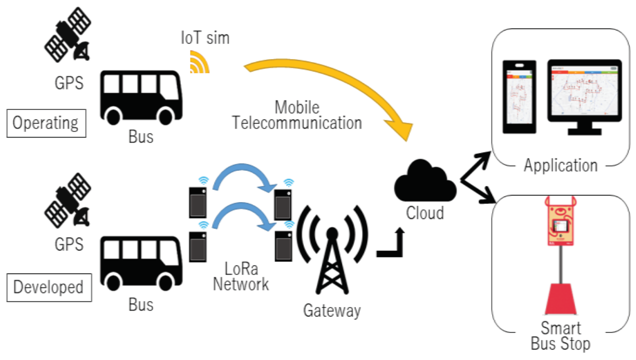
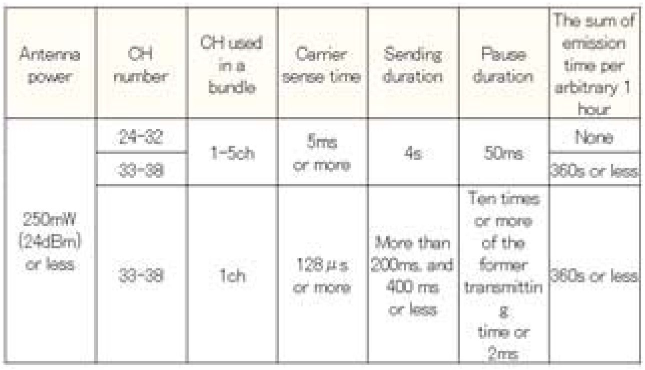


Figure 1: Bus location system which is called BusDoko system

In the system, GPS, IoT sim and LoRa module has been used to collect the real-time bus location information from running buses. In recent years, there are many users in the 920MHz band which is unlicensed bands, and stable system operation has become difficult. In the system interference is a problem that needs to be resolved to improve system stability, and to enable frequency sharing. Many wireless standards in recent years have capable of long-distance transmission, and since the timing and transmission channel can be freely set, they are susceptible to interference. Therefore, in order to ensure the communication quality of the service(QoS), the time occupancy rate of other system signals arriving at the Gateway (hereinafter referred to as the out-of-system interference time ratio) is estimated each channel with high accuracy, and the channel of LoRa of the our system is decided based on the estimation result. We need to assign the communication between bus and gateway to the appropriate channel to maintain the system stability.

There are existing studies [5] - [7] as a highly accurate radio wave environment prediction method. The method of acquiring, accumulating, and utilizing the radio wave condition is an important method from the viewpoint of radio wave utilization efficiency. In the existing studies [6] - [7], the actual observation type database is discussed, and its usefulness is reported. This database is a collection of environmental information such as received signal power and reception position information acquired by the receiver in the actual environment. Using this, the communication



Antenna power	CH number	CH used in a bundle	Carrier sense time	Sending duration	Pause duration	The sum of emission time per arbitrary 1 hour
250mW (24dBm) or less	24-32	1-5ch	5ms or more	4s	50ms	None
	33-38					360s or less
	33-38	1ch	128μs or more	More than 200ms, and 400 ms or less	Ten times or more of the former transmitting time or 2ms	360s or less

Figure 2: 920MHz band channel allocation specified by ARIB [12]

parameters of wireless channels, and channels can be adaptively assigned. In the existing research [5], a method of allocating to a channel with a low interference ratio has been proposed. However, this method assigns to only one channel. In recent years, the use of this database has been discussed in the field of frequency sharing. On the other hand, research is being conducted on LoRa to suppress interference [8] – [9]. However, these do not allow the system to consider the effects of interference.

In order to improve the accuracy and reliability of the system, we have created a method for multi-channel LoRa communication method using Time Division Multiple Access (TDMA) for the bus location system. We also create the channel hopping method which use the observation type wireless database. This makes it possible to handle changes in the communication environment over time. Multi-channel TDMA have been used to improve the QoS [10] – [11]. The conventional methods require submillisecond (sub-ms) accuracy synchronization. Therefore, expensive equipment is required. In addition, dynamic management of the TDMA slot needs to be done promptly. However, in the bus location system using LoRa it was difficult. Because LoRa have a transmission distance of several km with a large transmission delay. According to the report of [1], it is a dozen seconds. Therefore dynamic management of the TDMA slot is difficult in short TAT. The method we propose does not require accurate time synchronization and is not require fast dynamic TDMA slot management.

We have confirmed the effect of TDMA as a result of experiments in the room and the outdoor. We will explain the details of the experiments. And we report on the successful operation of the communication system and its effectiveness against communication interference.

2 BUSDOKO SYSTEM

In this section, we will explain the hardware and software configuration of the BusDoko system. The BusDoko system is a system that allows bus users to check the location information of buses in the real time on a Web page, a mobile terminal, or a terminal installed at bus stops.

By using the bus location system, it is possible to visualize the arrival, the departure, and the location of buses, and it is possible to eliminate the anxiety and dissatisfaction

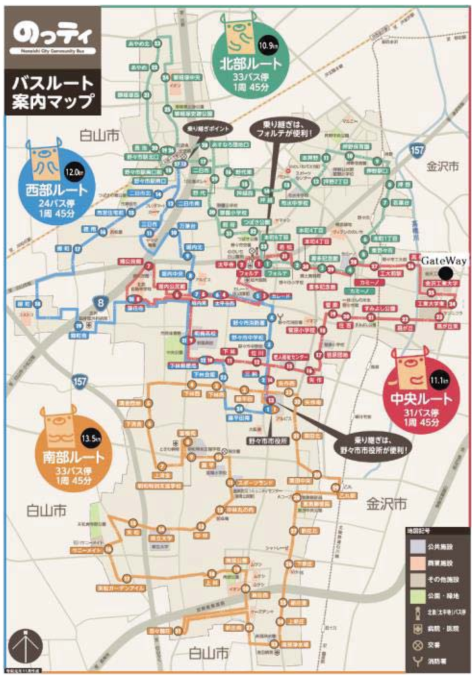


Figure 3: Bus route of the community bus Notty

that bus users have. Figure 1 shows the configuration of the bus location system we have developed. The buses are equipped with an in-vehicle device equipped with Private LoRa, IoT sim and GPS, acquires the current position information from the GPS at intervals of several seconds, sends it to the gateway using Private LoRa. And then gateway send data to cloud via the Internet and store the data in the upper server. By displaying the bus position information on a Web page using the stored the bus position information data, the bus position can be confirmed on the user's terminal. In addition, some of the bus stops we have developed have a built-in tablet, and some of the bus stop are equipped with a panel that allows you to check the bus position using LEDs. The LoRa gateway send the bus locations to the bus stops. Therefore the bus stops can display the current position of the buses.

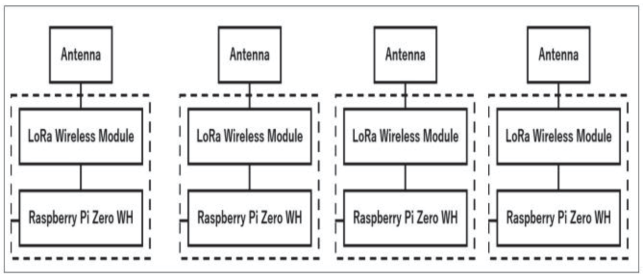
The domestic standard for the use of the 920MHz band is specified in ARIB STD-T 106-108. Figure 2 shows an excerpt of the ARIB STD-T108 standard. LoRa is a low-power wireless system that uses the 920MHz band, and it is possible to perform communication according to the purpose by changing the output power to 20mW or less and 250mW or less. The standard defines a combination of usable channel bandwidth, carrier sense time, maximum transmission time, pause time, and total transmission time per hour for each of 20 mW or less and 250 mW or less. We are using 250mW LoRa in the BusDoko system.

Figure 3 shows the bus route map of the community bus Notty. The gateway was installed on the roof of the 12-story Kanazawa Institute of Technology Library Center. There are four bus routes for the community bus. As a system requirement to realize QoS, location information of bus must be displayed at least once between bus stops.

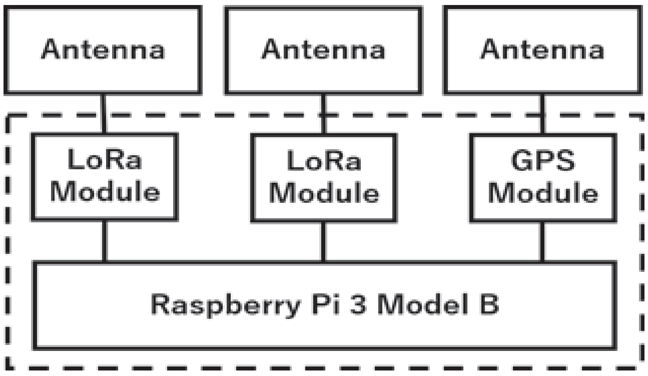
2.1 Hardware Design

Figure 4(a) shows the hardware configuration of the LoRa gateway and Fig. 4(b) shows the hardware configuration of the end devices on the bus. The LoRa gateway has two boxes with four Raspberry Pi Zero WH and four LoRa modules each. The end device is a Raspberry Pi 3 Model b with a built-in GPS module and a LoRa module. The LoRa module is the RM-92C module of RF-LINK Co., Ltd. The RM-92C module operates in the 920Mhz band and achieves a maximum receive sensitivity of -137dBm by changing the transmit power from 13dBm to 24dBm and setting the SF (Spreading Factor). The SF parameters of LoRa can be set from 6 to 12.

Figure 5 shows the gateway installed on the rooftop of the library center, which contains two boxes with four LoRa modules. In order to determine the exact location of the bus, the gateways were installed in two locations: the library center and a facility at the same height as the library center. The experiment was conducted within the area covered by the gateway installed at the library center



(a) LoRa Gateway



(b) The end device mounted in bus

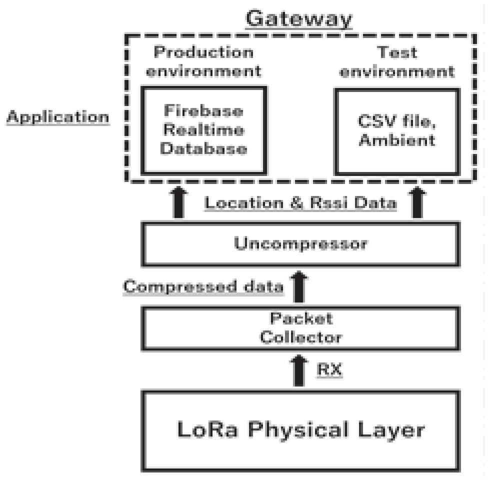
Figure 4. Hardware configuration of the system

Figure 5. Gateway on the Library center

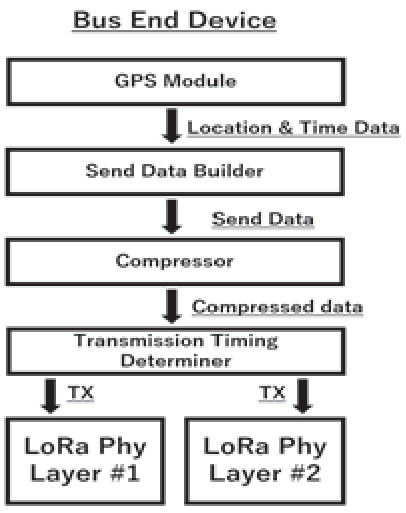
2.2 Software Configuration

Figure 6(a) shows the software configuration diagram of the LoRa gateway system of the BusDoko system. Figure 6(b) shows the software configuration diagram of the bus end device system of the BusDoko system. In the LoRa gateway, the data acquired by the LoRa Physical Layer is imported into the host computer by the Packet Collector. After that, the uncompressor decompresses the compressed data. The data is then uploaded to the Firebase realtime database in the production environment.

In the LoRa end device, the GPS module acquires the current position data and current time data. Of the data acquired by Send Data Builder, the current position data and bus ID are combined to create the send data. The Compressor compresses the transmission data. Transmission Timing Determiner determines which of the two LoRa Modules is used for transmission based on the current time data acquired by GPS.



(a) Gateway system



(b) End device system

Figure 6. Software configuration diagram of the BusDoko system

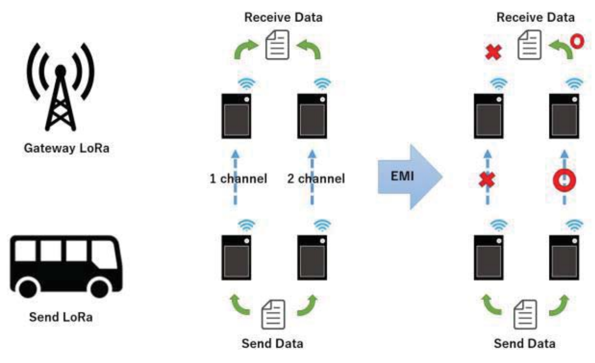


Figure 7. Effect of using multiple channels

3 MULTI-CHANNEL COMMUNICATION

If an end device on a bus sends data using one channel, it cannot be sent if that channel is used by another system. To avoid this danger, we decided to use two channels (see Fig. 7). This allows the location of the bus to be sent to the gateway, even if either channel is used by the other system. On the other hand, due to the limited number of channels, it is not good for one end device to occupy two channels. Therefore, we decided to use two Channels in common by two end devices.

3.1 Multi-channel Communication

In the case of communication using a single channel, regular communication may become difficult due to interference by others, which may impair the quality and real-time performance of the system. Therefore, we decided the multiple devices will communicate using different channels to solve these problems. Figure 8 shows the system diagram of LoRa communication using multi-channel in Nonoichi Notty bus routes. In this communication, two LoRa modules are installed in the end device of each bus, and the location information acquired by GPS is transmitted alternately. Each LoRa parameter sets a different channel. At the LoRa gateway, a module adapted to the channel receives the data and uses the bus route ID assigned to the data to determine the route.

3.2 Time-division Multiple Access of LoRa Link

When multiple communications are performed in the same channel, data loss due to communication collision is considered. We decided to synchronize the time between buses and gateway by using the the time that is gotton at the same time by acquiring the position of the bus using the GPS. By synchronizing the time in all LoRa of the end device, setting the time frame, and controlling the transmission module and time, the communication within the same channel is possible between two devices. Figure 9 shows the time frame diagram of the TDMA-based multi-channel communication. The LoRa gateway is always active

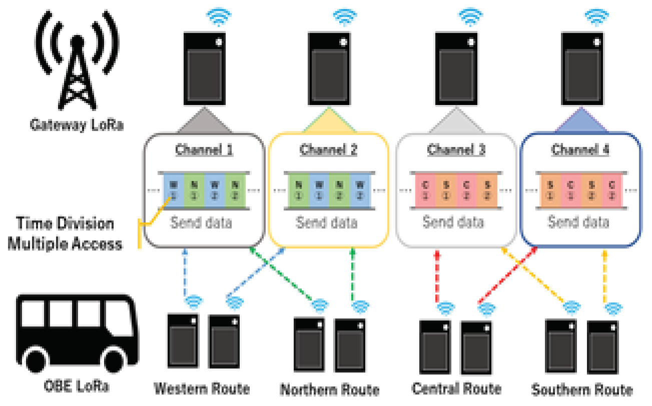


Figure 8. Channel assignment of Nonoichi nottey bus route

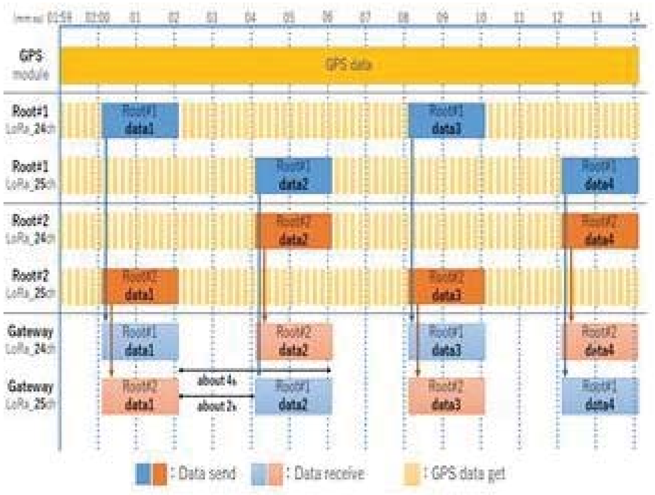
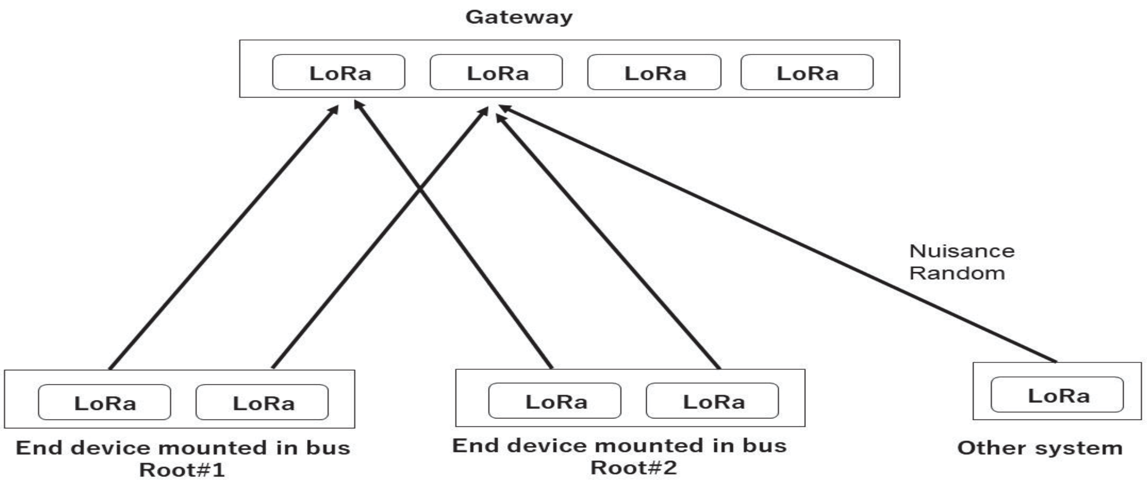


Figure 9. Multi-channel communication image

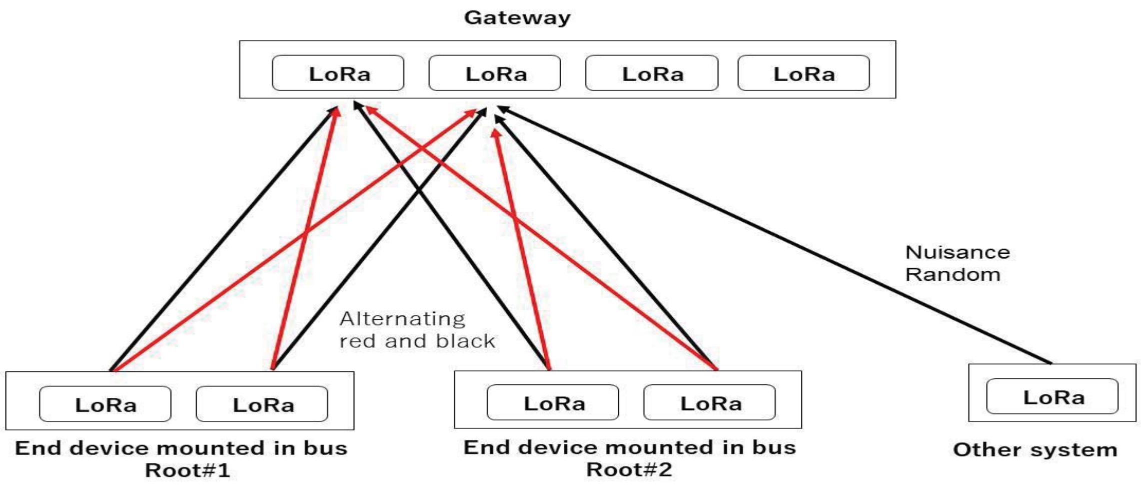
and waits for transmission from the LoRa end device. The end device synchronizes the time based on the time data obtained from GPS. The time frame starts at exactly even minutes, and the end device communicates every 4 seconds on the selected channel. By limiting the number of transmitting modules in the same time frame, one connection is established for each channel to realize TDMA.

3.3 Channel Hopping

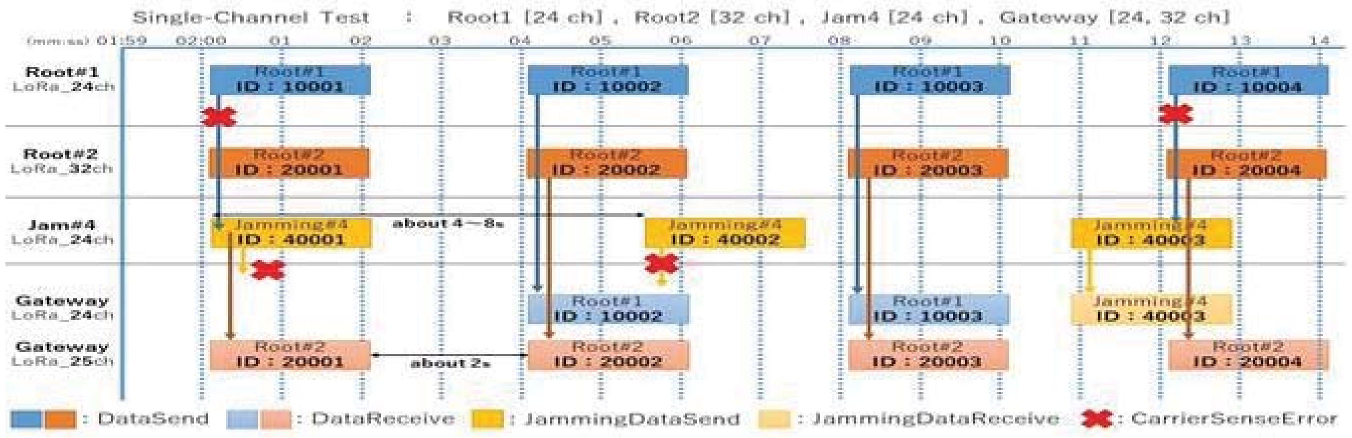
The channel used by the end device is determined using an actual observation database. The actual observation database is collected and created using a gateway data. The channel number will be updated using the time when all buses gather at Nonoichi City Hall for transfer on each bus route and stop for about 5 minutes. That is, the channel number to be transmitted is changed once an hour. This makes it possible to handle temporal channel congestion.



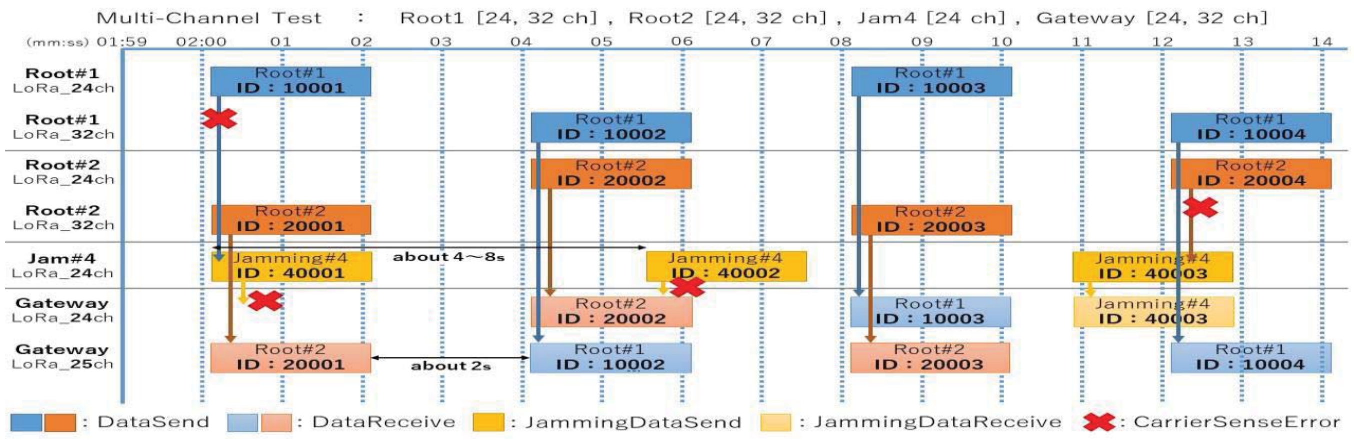
(a)State of the experiment: pattern 1



(b) State of the experiment: pattern 2



(c) Test sequence of the experiment: pattern 1



(d) Test sequence of the experiment: pattern 2

Figure 10. Test overview of with jamming and without jamming

4 EXPERIMENTAL RESULTS

Experiments were conducted both indoors and outside to verify the effectiveness of multichannel communication in the presence of jamming. The experimental configuration diagram is shown in Fig. 10. Figure 10(a) shows the configuration diagram of the experiment in which the end-bus equipment transmits using one channel. Figure 10(b) shows the block diagram of the experiment where the end-bus device transmits using two channels. Figure 10(c) shows the test sequence for one channel. Figure 10(d) is the test sequence for two channels. The end device sent test data for two routes using 24 and 32 channels, and the 24 channels of the jamming device sent different data at intervals of 4 to 8 seconds. The experiment was conducted indoors on one and two channels, and outside on two channels for actual use.

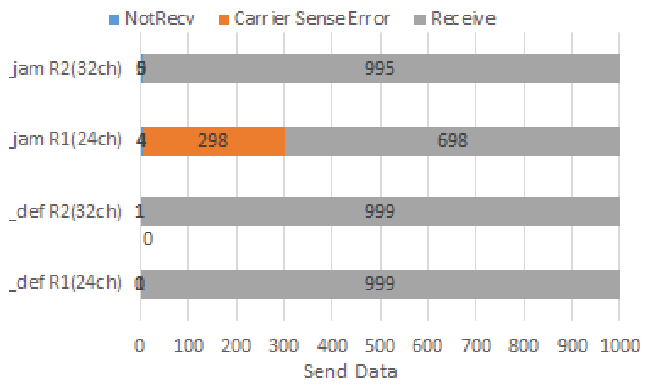
The national standard for the use of 920 MHz band is defined in ARIB STD-T 108 [12]. In BusDoko system, the parameters of LoRa are set to 24 dBm output power and no limit on the sum of transmission time in order to achieve real-time communication in the path. In this experiments, the parameters of LoRa are limited to 24 dBm output power, SF=12, and 24,32 channels used in the room. We also assume that the TDMA time frame is 4 seconds, the device of payload tendency to transmit is 5 bytes, the transmission time is 2 seconds, and the carrier sense and pause time is about 2 seconds. In the outside, LoRa parameters are limited to 24 dBm transmit power, SF=10, and 24,32 channels used. The TDMA time frame is set to 4 seconds, the device of payload tendency to transmit is set to 7 bytes, the transmission time is set to 4 seconds, and the carrier sense and pause time is set to about 4 seconds.

4.1 Indoor Experiments

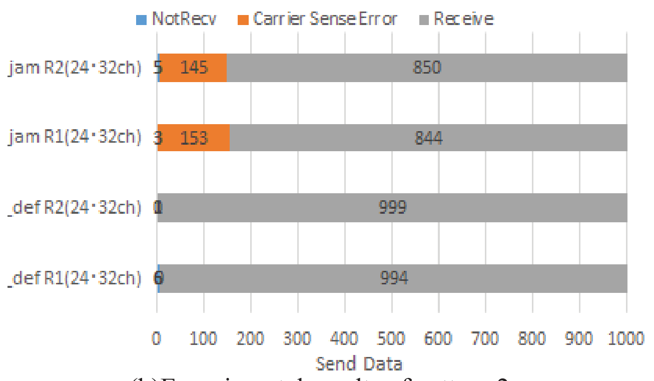
We will explain the experiment that was conducted indoors. Figure 11 shows the number of successful and unsuccessful transmissions when interference is given. Figure 11 (a) shows the result of an example using only one channel in the end device. Figure 11 (b) shows the results of an example using two channels in the end device. In the 1-channel communication, the number of carrier sense errors is biased toward route 1. In the 2-channel communication, the number of errors is successfully distributed.

Figure 12 shows the transmission time when interference is given in 1-channel communication. Figure 12 (a) shows the result without the disturbing wave, and Fig. 12 (b) shows the result with the jamming wave. Figure 13 shows the transmission time when interference is given in 2-channel communication. Figure 13 (a) shows the result without the disturbing wave, and Fig. 13 (b) shows the result with the jamming wave. From these results, it can be seen that the multi-channel system eliminates the bias of transmission errors and suppresses the variation of transmission time.

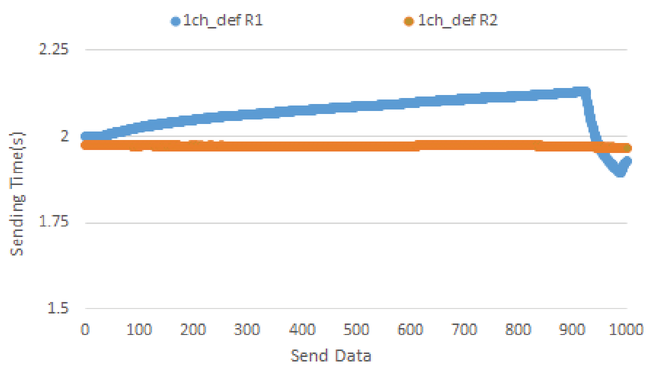
From the experiments, we checked whether data could be transmitted at an acceptable time, even in situations where other systems used the same channel as the bus-end device and had collisions. From the results, it was found that the data could be transmitted within a reasonable time and the system worked.



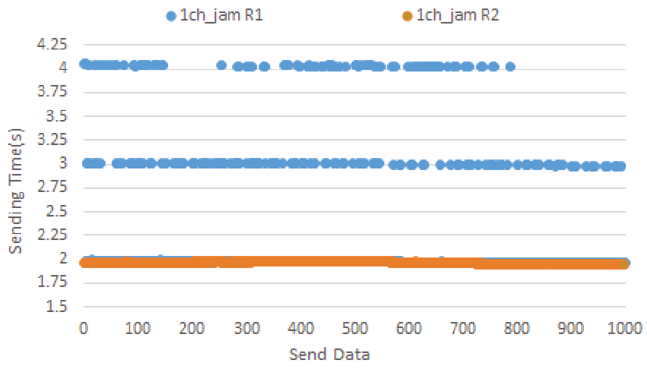
(a) Experimental results of pattern 1
(With / Without interference)



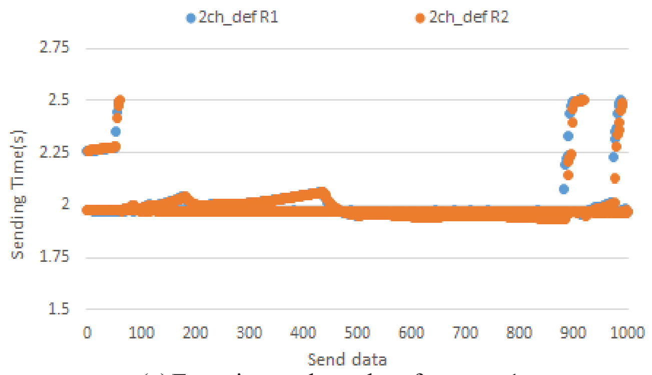
(b)Experimental results of pattern 2
(With / Without interference)
Figure 11. Transmission success/failure results



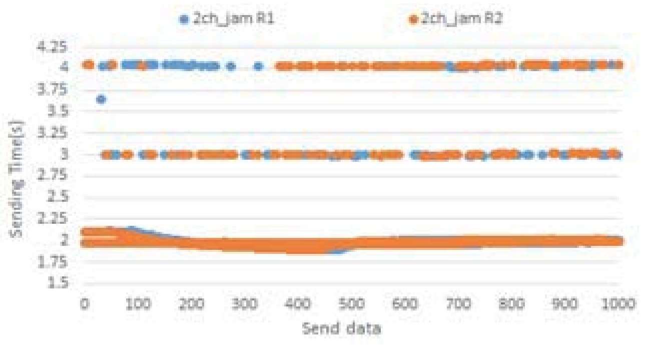
(a)Experimental results of pattern 1



(b)Experimental results of pattern 2
Figure 12. Transmission time of with jamming and without jamming



(a) Experimental results of pattern 1



(b)Experimental results of pattern 2
Figure 13. Verification experiment contents

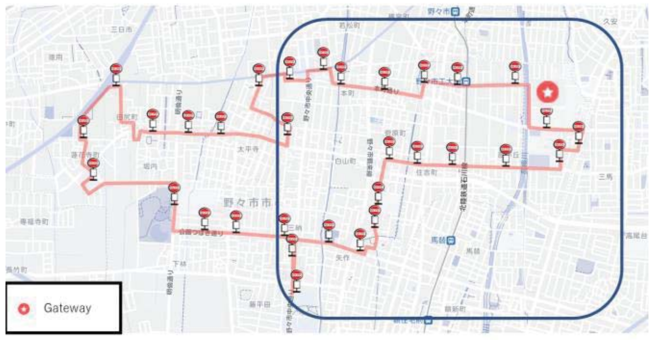


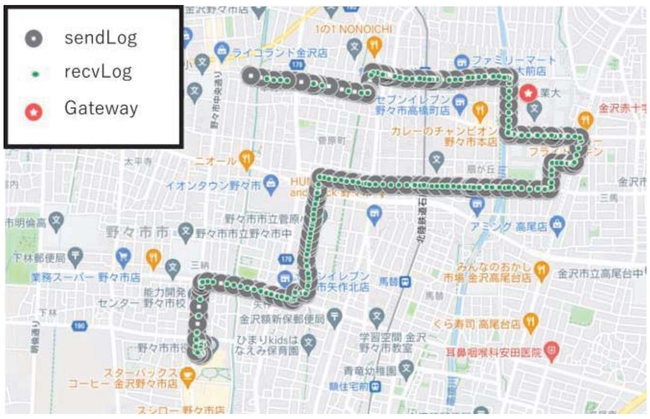
Figure 14. Experimental route map

4.2 Outdoor Experiments

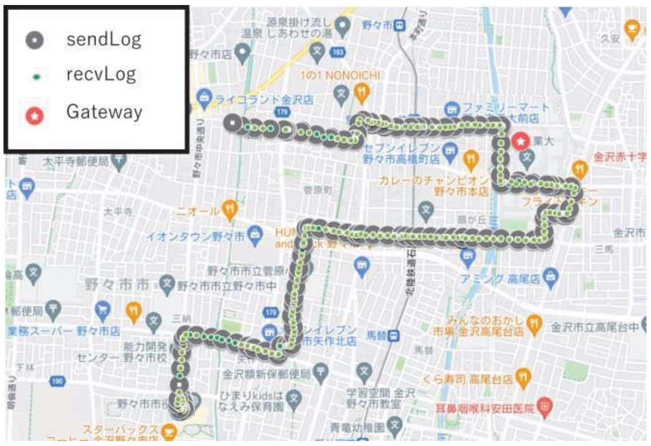
We will introduce the experiments conducted on the actual bus route. The experiment was conducted on a bus route within the range covered by the gateway installed on the roof of Kanazawa Institute of Technology, which is one of the bus routes, the north route. Figure 14 shows the entire route of the north route, the range used in this experiment, and the location of the gateway. The experiment was conducted by installing two end devices in one car. One end device is assumed to be route 1 in Fig. 10, and the other end device is assumed to be route 2. Since we assumed a different route, we installed the two devices diagonally and conducted the experiment.

LoRa parameters are limited to transmit power 24dBm, SF = 10, channels 24 and 32 used. The TDMA time frame is set to 4 seconds, the payload to send is set to 7 bytes, the send time is set to 4 seconds, and the carrier sense and pause time is set to about 4 seconds. Jamming is randomly transmitted to channel 24 every 4 to 8 seconds.

Figure 15 shows the results of the experiment with Pattern 2. Figure 15 (a) shows the result when there is jamming in pattern 2, and Fig. 15 (b) shows the result when there is no jamming in pattern 2. The gray circle in the figure is the transmission location, the green dot. Indicates where the transmitted data could be retrieved at the gateway. And the red circle is the gateway. You can see that the transmitted point and the received data are in the same place in both Fig. 15 (a) and Fig. 15 (b). Therefore, you can see that the data has been received properly. Also, comparing Fig. 15 (a) and



(a)Experimental results with jamming of pattern 2



(b)Experimental results without jamming of pattern 2

Figure 15. Results of communication experiments in the outside

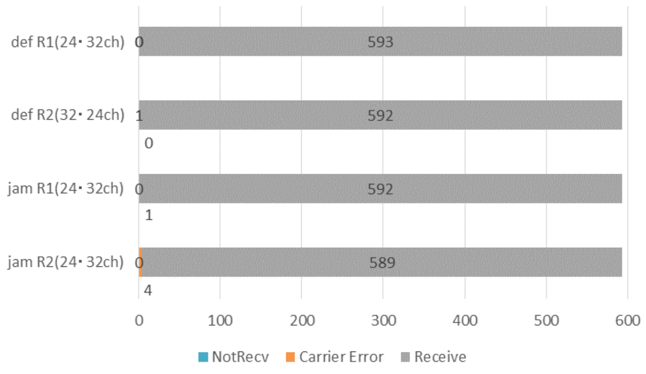
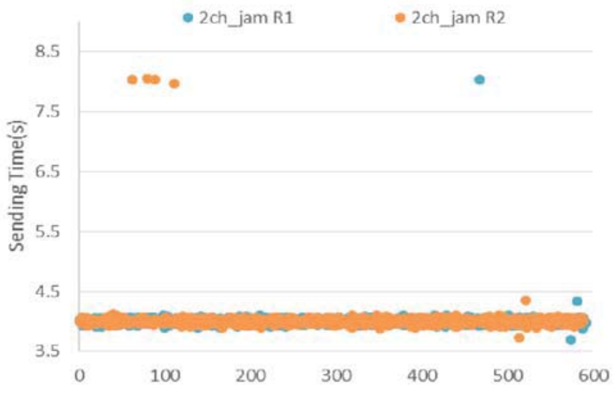
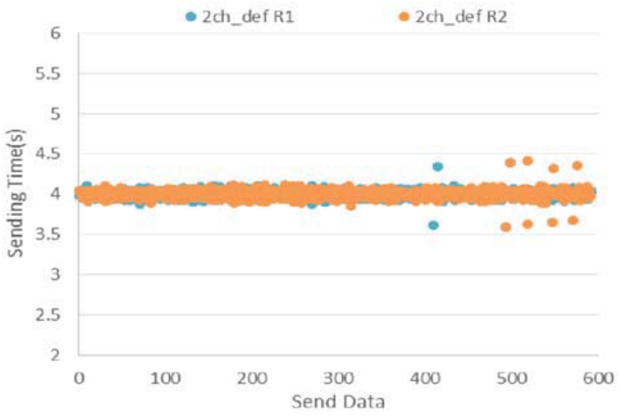


Figure 16. Transmission success/failure results of pattern 2



(a) Experimental results with jamming of pattern 2

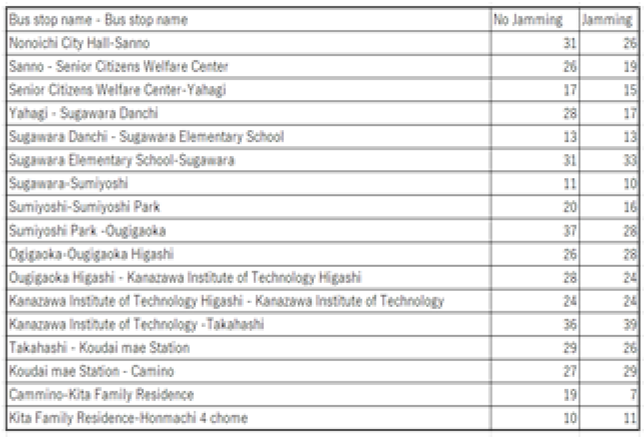


(b) Experimental results without jamming of pattern 2

Figure 17. Transmission time when jamming in the outside

Fig. 15 (b), the results are almost the same. In other words, it can be said that our system is robust in terms of communication performance. It meets the system requirements for achieving QoS even under the condition that noise is inserted at intervals of 4 to 8 seconds.

Figure 16 shows the results of the transmission success rate conducted in Pattern 2. It shows the number of successful and unsuccessful transmissions with and without interference. In Fig. 16, the top two are the experimental results without jamming radio waves, and the bottom two are the experimental results with jamming radio waves. R1 indicates route 1 and R2 indicates route 2. In this experiment, the jamming signal was generated on channel 24. Therefore, you can see that a carrier sense error has occurred on 24 channels. In the example with jamming waves, one carrier sense error occurred in R1 and four carrier sense errors occurred in R2. However, it can be seen that it operates without major errors even in a harsh environment where disturbing waves are randomly emitted at intervals of 4 to 8 seconds. From these results, the effectiveness of this system was shown.



Bus stop name - Bus stop name	No Jamming	Jamming
Nonoichi City Hall-Sanno	31	26
Sanno - Senior Citizens Welfare Center	26	19
Senior Citizens Welfare Center-Yahagi	17	15
Yahagi - Sugawara Danchi	28	17
Sugawara Danchi - Sugawara Elementary School	13	13
Sugawara Elementary School-Sugawara	31	33
Sugawara-Sumiyoshi	11	10
Sumiyoshi-Sumiyoshi Park	20	16
Sumiyoshi Park -Ougigaoka	37	28
Ogigaoka-Ougigaoka Higashi	26	28
Ougigaoka Higashi - Kanazawa Institute of Technology Higashi	28	24
Kanazawa Institute of Technology Higashi - Kanazawa Institute of Technology	24	24
Kanazawa Institute of Technology -Takahashi	36	39
Takahashi - Koudai mae Station	29	26
Koudai mae Station - Camino	27	29
Cammino-Kita Family Residence	19	7
Kita Family Residence-Honmachi 4 chome	10	11

Figure 18. Number of times received between bus stops

Figure 17 shows the result of the transmission time in pattern 2. Figure 17 (a) shows the result with the disturbing wave, and Fig. 17 (b) shows the result without the jamming wave. If there is a carrier sense error, the transmission time will be longer because the carrier sense is repeated. It can be seen that the transmission time may increase in the presence of the interference wave shown in Fig. 17 (a). However, this result shows that the ratio is small and the increase in transmission time is within the allowable time. From this result, it was confirmed that even if another system is using the same channel as the bus end device and a collision occurs, data can be transmitted within an acceptable time. It took less than 1 second from the time when the bus location information was obtained at the gateway until it was displayed on the WEB page. Figure 18 shows the number of times bus location information was successfully sent to the gateway between bus stops. The probability of successful location data transmission at least once between bus stops was 100% with jamming.

5 CONCLUSION

We are working on the development of a bus operation management system using LoRa. In recent years, there are many users in the 920MHz band, and stable system operation has become difficult. Therefore, in this paper, we have examined a method that enables stable communication even under such circumstances. In order to improve the accuracy of the system, we have created a method for multi-channel LoRa communication using TDMA for the bus location system. We also created the channel hopping method. This makes it possible to handle changes in the communication environment over time. We have confirmed the effect as a result of experiments in the room and the outdoor, so we explained the details. And we reported on the successful operation of the communication system and its effectiveness against communication interference. By adopting the proposed method, we were able to improve QoS. It is predicted that the number and density of wireless ad hoc networks in the 920MHz band will continue to increase in the future. We plan to study a stronger communication method.

ACKNOWLEDGMENTS

The research is supported by Nonoichi City.

REFERENCES

[1] H. Yabe, S. Ishikawa, S. Tomioka, S. Tsukahara, R. Sakauchi, M. Sode Tanaka, “Bus Location System with LoRa to Cover the Entire Nonoichi City”, IEEE 3rd Global Conference on Life Sciences and Technologies(LifeTech), pp.419-420, pp. 419-420, doi:10.1109/LifeTech52111.2021.9391821, (2021).

[2] T. Boshita, H. Suzuki and Y. Matsumoto, "IoT-based Bus Location System Using LoRaWAN," 2018 21st International Conference on Intelligent Transportation Systems(ITSC), pp.933-938, doi:10.1109/ITSC.2018.8569920, (2018).

[3] P. Guan, Z. Zhang, L. Wei and Y. Zhao, "A Real-Time Bus Positioning System Based on LoRa Technology," 2018 2nd International Conference on Smart Grid and Smart Cities (ICSGSC), pp.45-48, doi:10.1109/ICSGSC.2018.8541282, (2018).

[4] https://nottydoko-demo.sodeproject.com/

[5] S. Kakuda, Y. Yamazaki, K. katagiri, T. Fujii, O. Takyu, M. ohta, and K. Adachi, “Channel Allocation for LoRaWAN Considering Intra-System and Inter-System Interferences”, IEICE Tech. Rep., Vol.121, No.30, IEICE-SR2021-12, pp.79-85 (2021).

[6] Y. Ye and B. Wang, “RMapCS: Radio map construction from crowdsourced samples for indoor localization,” IEEE Access, Vol.6, pp.24224-24238, doi:10.1109/ACCESS.2018.2830415 (2018).

[7] S. Bi, J. Lyu, Z. Ding, and R. Zhang, “Engineering radio maps for wireless resource management,” IEEE Wireless Commun., Vol.26, No.2, pp.133–141 (2019).

[8] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, “Low overhead scheduling of LoRa transmissions for improved scalability,” IEEE Internet of Things J., Vol.6, No.2, pp.3097-3109 (2019).

[9] Z. Qin and J.A. McCann, “Resource efficiency in low-power wide-area networks for IoT applications,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Singapore, pp.1-7, doi:10.1109/GLOCOM.2017.8254800 (2017).

[10] Jabandžić, S. Giannoulis, R. Mennes, F. A. P. De Figueiredo, M. Claeys and I. Moerman, "A Dynamic Distributed Multi-Channel TDMA Slot Management Protocol for Ad Hoc Networks," in IEEE Access, Vol.9, pp.61864-61886, doi:10.1109/ACCESS (2021).

[11] A. Aijaz and A. Stanoev, "Closing the Loop: A High-Performance Connectivity Solution for Realizing Wireless Closed-Loop Control in Industrial IoT Applications," in IEEE Internet of Things Journal, Vol.8, No.15, pp.11860-11876, (2021).

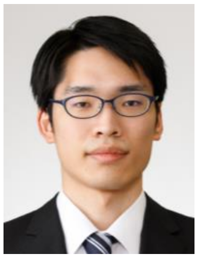
[12] 920MHz-BAND TELEMETER TELECONTROL AND DATA TRANSMISSION RADIO EQUIPMENT, ARIB STD-T108 Version 1. 3 (2021).

(Received: October 31, 2021)
(Accepted:July 20, 2022)

Ryotaro Sakauchi is 4th year undergraduate student in the Department of Information Technology, Faculty of Engineering, Kanazawa Institute of Technology (2022 Current). He interest in Wireless Communication Technology, Mobile Communication and Low Power Wide Area.

Shuto Ishikawa received bachelor of Engineering from Kanazawa Institute of Technology, Japan in 2022. He joined Sony Network Communications Inc. in April 2022. He engaged in LPWA-related research.

Yuki Eto is 4th year undergraduate student in the Department of Informatics, Kanazawa Institute of Technology (2022 Current). He interest in Wireless Communication Technology, Smart city and Regional activation.



Yota Nomoto is 4th year undergraduate student in the Department of Information Technology, Faculty of Engineering, Kanazawa Institute of Technology (2022 Current). He interest in Wireless Communication Technology, Mobile Communication and Low Power Wide Area.

Shunsuke Segawa is 3th year undergraduate student in the Department of Electrical and Electronic Engineering, Kanazawa Institute of Technology (2022 Current). He interest in Wireless Communication Technology, AI image processing and Low power consumption technology.

Hikaru Yabe graduated in March 2022 from the Department of Information Technology, Faculty of Engineering, Kanazawa Institute of Technology, and joined Softbank Corp in April 2022.

Mikiko Sode Tanaka received Dr. Eng. degrees from Waseda University in Fundamental Science and Engineering. She joined NEC Corporation, NEC Electronics Corporation, and Renesas Electronics Corporation. She is Associate Professor of International College of Technology, Kanazawa. Her research interests include wireless communications, AI chip, and personal authentication. She is a member of IEICE (Institute of Electronics, Information and Communication Engineers). Also, she is senior member of IPSJ (Information Processing Society of Japan) and IEEE (Institute of Electrical and Electronics Engineers).

Regular Paper

Performance Evaluation of a CyReal Sensor System

Kei Hiroi[†], Akihito Kohiga[‡], and Yoichi Shinoda[‡]

[†]Disaster Prevention Research Institute, Kyoto University, Japan
[‡]Japan Advanced Institute of Science and Technology, Japan
hiroi@dimsis.dpri.kyoto-u.ac.jp

Abstract - IoT devices are expected to enable inexpensive and easy measurement and collection of a wide range of environmental information, especially in the field of disaster prevention. Whereas, preliminary verification is difficult, because of a functional design that assumes their distributed deployment, the cost of development itself. Therefore, we develop a sensor system emulator with CyReal concept, which integrates the virtual device with the actual device, and federated with other simulators. This paper presents a prototype of our sensor system, its feasible performance, and discusses a design for flexible integration, in order to figure out that our sensor system supports large-scale virtual sensor devices for verification of the development, update, debugging, and operation of sensor systems as a distribution network for disaster prevention information.

Keywords: sensor network, emulation system, sensor testbed

1 INTRODUCTION

IoT (Internet of Things) devices equipped with sensors and related technologies are expected to enable inexpensive and easy measurement and collection of a wide range of environmental information. In the field of disaster prevention, they are being utilized for various fields such as river observation and slope measurement, and are expected to be useful for collecting data at points where it has been difficult to figure out the condition of the environment.

Conventional environmental monitoring for disaster response is usually installed at a few vulnerable sites that require monitoring. The environmental observation had been carried out through a robust monitoring system using a leased network lines. However, due to the frequent and large-scale river floods in recent years, we face on pressing importance requiring a larger-scale monitoring network, even for small rivers and waterways that have not been monitored well in the past. An observation network using IoT has the potential to minimize the cost of implementation and operation. This implies that a large-scale observation system can be applied.

However, installation of a large number of sensor devices creates a new problem. IoT devices are expected to be used in urban areas and mountainous regions. Devices can be installed in large numbers to provide valuable measurements that have not been possible with conventional observation systems. Whereas, installing a large number of devices over a wide area makes it complicated to handle them and difficult to improve them by relocating them after installation. Especially in data measurement where the relationship between devices is meaningful, it is important to design functions and consider how to utilize them with distributed deployment. Preliminary verification in this paper indicates to test the operation of the entire sensor network before installation, to find bottlenecks, and to consider solutions based on this functional design. By conducting preliminary verification before installing sensor devices, we can clarify data from the unique characteristics of sensor devices and the relationship between data from multiple sensor devices, as well as the transmission characteristics of data due to terrain and communication infrastructure, and use these results to design measurements that satisfy the purpose of use. Nevertheless, since a large number of devices are required, it is difficult to verify the functions in advance by preparing the necessary number of actual devices, which increases the cost of development itself and improving such as relocation. Although various sensor emulators have been developed to enable such preliminary verification, they are limited to verification of some functions such as network performance and device performance after using virtual devices.

In this study, we develop a sensor emulator system to support research, development, and operation of disaster prevention information collection and operation, and to enable preliminary verification assuming a specific installation environment and operational configuration. A sensor emulator is a virtualization technology that emulates the computers performance involved in sensor devices and their functions. In this research, we develop an emulator system that can exchange the virtual device with the actual device and can be federated with other simulators in order to support the development, update, debugging, and operation of sensor systems as a distribution network for disaster prevention information. By separating the Node and Sensor, and incorporating a connection mechanism between the virtual and physical devices for each of them, we can verify the functions of a large-scale sensor device. By enabling connection and verification not only with the virtual device but also with the physical device, it is possible to assume functions and communication environments that cannot be demonstrated with the virtual device, and it also facilitates connection to the cloud, thereby reducing operating costs and enabling multiple use of resources based on the assumption of an actual operating environment. Federation with other simulators can be possible through data exchange using the IoT linkage infrastructure. Through collaboration, data that assumes actual operation can be incorporated into the sensor, and preliminary verification, including operational forms such as disaster response based on data collection, becomes possible.

2 RELATED WORKS

Sensor emulator is a powerful tool that helps researchers and users to consider the design of sensor networks. An enormous amount of effort has been invested in developing emulators for various technologies related to sensor networks; communication protocols, computer processes, application software. As the number of sensors increases with the spread of IoT, the need to handle a large number of sensors has led to the development of a number of sensor virtualization technologies. SenaaS [1] is IoT virtualization framework to support connected objects sensor event processing and reasoning. This framework provide an ontology design by a semantic overlay of underlying IoT cloud and a policy-based service access mechanism in terms of semantic rules. Bose et al. [2] have presented resource abstraction at the sensor level on Sensor-Cloud infrastructure with virtualization of sensors for developing applications in various fields. This is a design for virtual sensor on cloud station. SenseWrap [3] is a middleware architecture providing virtual sensors as representatives for any type of physical sensors. This midlleware supports sensor-hosted services and a standardized communication interface that applications can use without having to deal with sensor-specific details.

Wireless networks, an indispensable technology for sensor networks, are also subject to emulation. TOSSIM [4] focuses on simulating a wide range of network interactions. TOSSIM, which features a high fidelity and scalability, can capture network behavior while scaling to thousands of nodes. COOJA [5] is a sensor network emulator aimed at cross-level simulation, allowing simultaneous simulation at many levels of the system; sensor node platforms, operating system software, radio transceivers, and radio transmission models. Many other emulators for sensor network verification have been researched and developed, such as EmStar [6], Avrora [7], and J-Sim [8]. These sensor emulators have been developed on the premise of wireless sensor networks. Recently, based on recent developments in low power wide area networks, including the rise of Long Range (LoRa) technology, a LoRa Coverage Emulator [9] has also been developed. This LoRa emulator consists of a transmitter and receiver and provides a reliable network coverage estimation based on the LoRa network design framework.

There are many researches on emulators from Hardware / Software point of view as well as network. The Freemote Emulator [10] is an emulator for developing software for nodes. It provides developers with a system architecture in several layers: Physical, Data Link (MAC), Routing and Application. Similarly, ATEMU [11] is a well-known sensor network emulator with a lot of contributions. A unique feature of ATEMU, which can operate on different application/hardware platforms, is its ability to simulate a heterogeneous sensor network. ATEMU emulates the processor, radio interface, timers, LEDs and other devices. Kasprowicz et al. [12] focused on CCD sensors as devices and developed a hardware emulator to speed up and streamline post-assembling tests and debugging. Furthermore, research on emulators has also focused on commoditization, with an emphasis on lightweight and small IoT systems. Brady et al. [13] developed an emulator for an IoT environment using the popular QEMU system emulator to build a testbed of inter-connected, emulated Raspberry Pi devices. The effort to emulate functionality extends to applications that anticipate not only specific devices, but also power supply and utilization. Deda et al. [16] have designed a battery emulator/tester system to reduce development time for developing and testing of high-voltage power supply systems. Abrishambaf et al. [14] have developed a laboratory emulation model for energy scheduling in an agriculture system using real nodes.

The conventional emulators so far can be said to be emulators that specialize in a certain function of the computer. By using these technologies, hardware, software, and network can be emulated in an integrated manner. However, with the evolution of IoT, we have to treat power supply, heterogeneous sensor devices, and network devices. We need to provide an operator-friendly verification environment. The benefits of the IoT have increased the opportunities for sensing technologies to be more readily available to a wider user. And this advantage means that sensor networks can be built more widely and in more places than ever before. Verifying the required functionality with a few emulators may provide the required results. Althogh, for actual operators whose work is on the use of application services, it is very difficult to verify the functions in isolation. Nonetheless, emulators that require special technology are considered to have little affinity with this kind of operators. Also, for operators, when considering many things that could not be verified with previous emulators, such as geospatial and distributed installation for business efficiency, these requirements are not considered with previous emulators.

3 SENSOR SYSTEMS BY CYREAL EMULATOR

3.1 Overview

The sensor emulator system developed in this paper enables preliminary verification of IoT devices based on the assumption of their specific installation environment and operational configuration. Based on the concept of CyReal, this sensor emulator system is designed to support the development, update, debugging, and operation of sensor systems, so that the virtual machine and the actual machine can be exchanged and can be federated with other simulators.

3.2 System Requirement

We aim to develop a verification environment for sensor systems for actual operators whose work is on the use of application services. What is the most user-friendly verification environment for operators? Our definition is an emulator that does not require any special skills and produces output that is directly relevant to thier work. With the development of IoT, there is a need to master the power supply, different types of sensor devices, and network devices. What is important for operators who work with IoT-based applications and services is that the data from the IoT is stable and their applications and services can work smoothly.

In case of a disaster, there is a risk of failure of the sensor system, power supply to operate the sensor system and obtain data, and communication problems. Thus, it is necessary to verify that the operation of the system as a whole is stable, not only the operation of the IoT sensor system, but also the power supply and communication status. Further, in considering how to improve the efficiency of their work, it is essential for operators to verify this in light of utilization, such as device locations in geographical environments and distributed installations that address disaster vulnerabilities. Our primary focus in this verification environment is to capitalize on the IoT sensor devices currently being operated by operators. This is because using real installed and operating devices as part of the emulator not only enables preliminary verification including actual performance, but also aims to operate as a real-time emulator in the future. For these reasons, we develop a verification environment that is compatible with real systems and can handle the large scale of IoT devices.

We defined the requirements for a sensor emulator system to meet these requirements as follows.

- The sensor system emulator should be compatible with a real system.

- The sensor system emulator should be able to handle a large number of sensors.

To meet the above requirements, we develop an emulator that incorporates the concept of CyReal.

3.3 CyReal Approach

CyReal is intended to be an entity that plays an intermediate role in the concept of digital twin [15]. CyReal enables flexible replacement and integration of real and virtual entities, such as computer systems, people, and environments. The configuration based on CyReal strongly promotes the digital twin of disaster prevention IT systems. Digital twin refers to the creation of digital objects to be handled in the real world and computer systems. This is the concept of debugging various properties on the created twin, in the case of computers, and is an important concept in the AI world.

Namely, the configuration on the left in Fig. 1 is one in which all subsystems are configured by simulators, and the subsystems are connected via a federation platform. On the other hand, the various simulations and systems that configure this platform can all be replaced with real things, real systems. The configuration on the right side of Fig. 1 shows a situation in which the subsystems are all real and in actual operation, such as people, natural phenomena, and a real disaster prevention information system. These two are in a digital twin relationship. Then, we have further extended the digital twin concept to allow real systems and simulators to be integrateded in a subsystem (The center of Fig. 1). This concept is beginning to be called CyReal, as an integration of Cyber and Real.

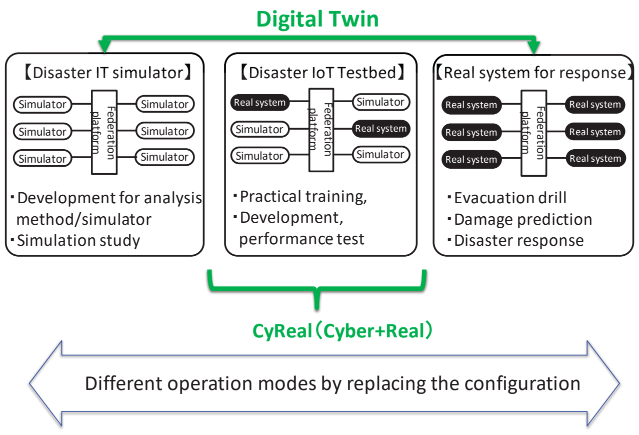


Figure 1: CyReal Approach for Sensor System

3.4 CyReal Sensor System

The sensor system we are developing is configured in accordance with the CyReal concept. The sensors are connected as subsystems in Fig. 1. The sensors can be replaced by real or virtual ones. We expect that this CyRealization allows the subsystems to work in various ways and the system to be used for various purposes. That is, depending on the system we replace, this configuration can be transformed into a system with various purposes.

If the system is entirely composed of simulators, it will work as a disaster management IT simulator. For example, new analysis technologies and simulators can be connected to the disaster prevention IT simulator, and all simulators can be run based on data from past disasters. Since data is difficult to collect in disaster research, evaluating the performance of analysis technologies and simulators is a difficult and costly task. By creating this disaster prevention IT simulator, we have the prospect of providing an evaluation environment and facilitating development. Alternatively, if the subsystems are replaced with real ones (i.e., if the sensor system is replaced with a real one), it becomes a test bed that can be used for research and development and performance evaluation of the sensor system.

This configuration eliminates the limitation of devices, simulators, and systems that can be verified, which is a requirement of this paper. In other words, we can connect not only sensor devices, but also geospatial and operational simulations to verify the functionality and effectiveness of the system in actual operations. In order to develop a system that can flexibly switch between these three modes, we have embarked on a sensor system as one of the proofs of concept for the digital twin and CyReal that bridges the gap between Real and Virtual.

3.5 Significance of CyReal Sensor System

This sensor system is not a simple sensor virtualization. We have designed the system to anticipate the actual data utilization of the sensors. Conventional sensor virtualization has only simulated the data and functions involved in sensing. On the other hand, we not only simulate the data, but also develop the sensor device and its environment as an instance.

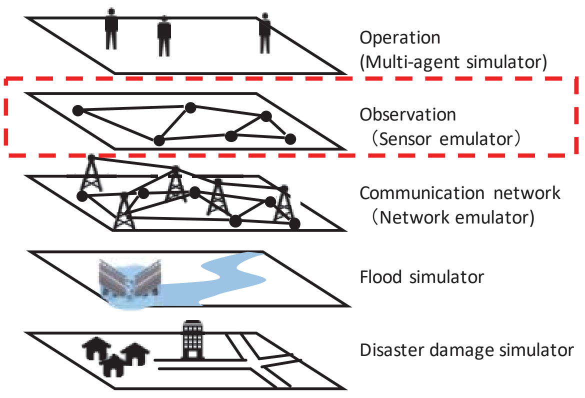


Figure 2: Federation with External Simulators

Our expected applications are as follows. Data missing is an unavoidable problem in large-scale sensor deployments. It is difficult for a system that only simulates data to represent the handling of such missing data. Our system design is able to produce data by incorporating not only the sensor device itself, but also the external environment, such as the wireless environment and the conditions of the location where the sensor is installed. This makes it possible to represent fluctuations in the data due to the influence of the external environment(Fig. 2). In addition, data diversity can be expressed by integrating with external simulators, such as simulators for operations and simulators for natural phenomena. This is the advantage of an emulator that can replace and integrate actual and virtual nodes and sensors. Such a sensor system has many uses in verifying operations, but it also has many challenges. This paper shows proof-of-concept for a sensor system according to this concept, investigates its performance when running virtual sensors on a large scale, and consider the challenges of implementation.

3.6 Structure of CyReal Sensor System

Based on the CyReal concept, we have developed a sensor emulator system that can connect and verify not only virtual devices but also real devices. A sensor emulator is a kind of virtualization technology that enables preliminaly verification of sensor-related systems, such as network performance, device performance, and computer processing performance. The sensor emulator is used for functional verification using virtual sensor devices. Conventional technologies and prior research to date have only supported virtual sensor devices, so that it is difficult to conduct verification based on actual operational environments. Recent IoT devices and related technologies are said to make it possible to distribute sensors over a wide area and to measure and collect data easily and inexpensively. However, in reality, it is difficult to design functions assuming distributed deployment and to prepare the necessary number of actual devices for functional verification in advance, and this has the disadvantage of leaving the decision to the user. Therefore, we have attempted to develop a CyReal sensor system to support research and development and operations related to the collection and distribution of disaster prevention information. In Fig. 3, Sensor refers to a measurement device and Node refers to a computer that processes and communicates measurement data. Here, both sensor and node are designed to be interchangeable between actual and virtual machines. Furthermore, the system can be communicated with other simulators. This is based on data from past disasters, and is intended to be extended to verification based on actual situations such as power outages, device failures, and network disconnection.

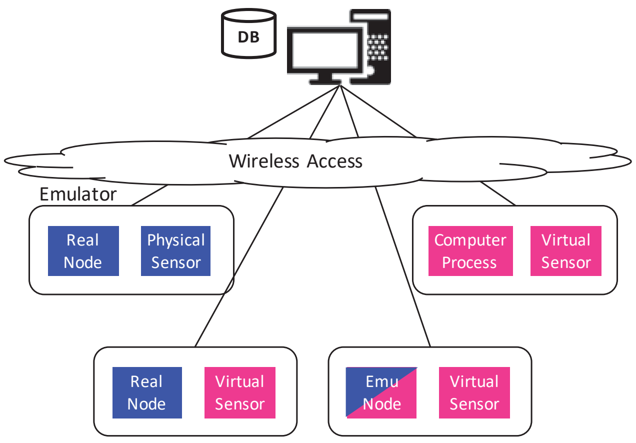


Figure 3: Structure of CyReal Sensor System

4 IMPLEMENTATION AND PERFORMANCE TEST

4.1 Overview

The implementation of this paper is based on two points. The first point is the use of existing IT technologies. The first point is that we use existing IT technology, using AWS, a commercial IoT cloud, to aggregate and process data without preparing any onpremise servers. This allows us to consolidate and process data without any servers. Since there are no servers, there is no need for maintenance. This is an attempt to minimize the cost of implementation and operation.

The second point is API unification. By using SigFox and AWS, whose specifications are open to the public, we can take advantage of the functions that already exist. Also, as an open system, others can take advantage of this mechanism. By using the existing functions, we can simplify the development of the system, and finally, the internal structure is quite simple.

The data from the CyRealized sensor emulator system is currently only recorded in the slack as a logger. If this data is put into a DB, it can be displayed in various UIs and used in applications. In addition, we are planning and implementing data exchange with other simulations.

4.2 Implementation

Figure 4 shows a prototype configuration of the sensor emulator system we have implemented. From Real Node, we used Sigfox, an IoT network and cloud to aggregate the measurement data and put it into the AWS cloud. We used an Arduino Uno as the hardware for Real Node. The Arduino

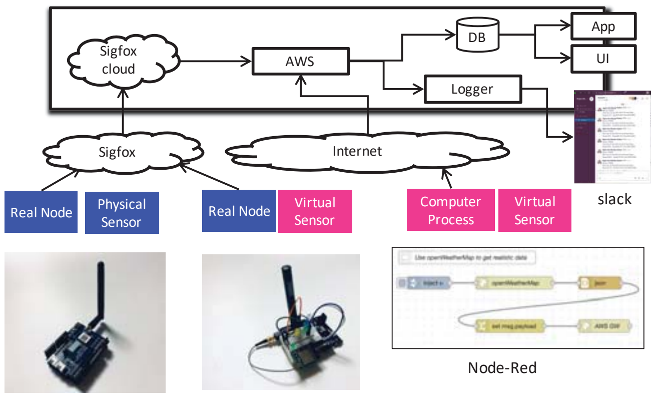


Figure 4: Prototype Configuration

Uno is equipped with a Physical Sensor that measures temperature, humidity, and air pressure. The Arduino Uno sends these data to the AWS API gateway via the Sigfox una shield. Although the hardware used in this paper is an Arduino Uno, it also has the commutativity to connect data obtained from other devices.

To show the commutativity, we use a Raspberry pi as another Real Node. The Raspberry pi collect the data from the Virtual Sensor. In this proof of concept, data is retrieved from Open Weather Map[1], which is a weather information API. We use Node-Red to retrieve the data and send it to the AWS API gateway using the flow shown in Fig. 4. We are using AWS as a Serverless service. The physical/virtual sensor data sent to the AWS API gateway is sent to the Slack webhook API via AWS Lambda. Figure 4 shows the data sent to Slack and recorded.

In addition, we created a combination of a computer process and a Virtual Sensor as a Virtual Node. The sensor data generated by the Virtual Sensor is sent to AWS via the computer process. Using this function, we can of course incorporate open weather information published on the Internet as sensor data. In addition, sensor data generated by rainfall simulation and inundation prediction simulation can be used as data for the entire system. In addition, sensor data generated by rainfall simulation and inundation prediction simulation can be used as data for the entire system. With this configuration, we are demonstrating the CyReal environment that can operate simultaneously heterogeneous computers, as a Real/Virtual Node and a Physical/Virtual Sensor.

4.3 Performance Test and Discussion

We have proposed and implemented a design using SigFox and AWS for a CyReal sensor system. In 4.3, we conduct a test to measure the performance of this sensor system. The performance test focuses on the sensor system connected with only virtual devices. Considering that preliminary verification of these devices can be conducted on this sensor system, it is necessary to investigate the performance of scalability to show how large an emulation can be performed using this sensor system. Therefore, this paper sets up virtual devices on our sensor systems and tests the following evaluation points using the number of virtual sensors as a parameter.

- Performance of a large scale installation of virtual sensors
- Limitations of devices, simulators, and systems, and discussion based on actual operations

[1]Open Weather Map: https://openweathermap.org/

Table 1: Performance Test Scenario

Scenario	Duration(min)	Time interval(s)
A	100	1.0
B	30,60,120,180,240	0.5, 1.0
C	30	60.0

4.3.1 Setup

For the performance test, we use a computer process and virtual sensors implemented in Python. As the Ministry of Land, Infrastructure, Transport and Tourism is working to install 3,000 water level sensor devices in small and medium-sized rivers, and in addition to the Japan Meteorological Agency, many companies have installed rainfall observation sensor devices, with each company operating about 100-400 sensor devices. As a result, nearly 4,000 sensor devices are expected to be in constant use in the field of disaster prevention. Thus we configure the virtual sensors on a scale of 2^n $(n = 0 - 16)$. Data is transmitted via MQTT.

For performance testing, we experimented and discussed two evaluation points. In the first test, we calculate CPU usage and processing time when emulating the CyReal sensor system on a PC. This test is to investigate that this sensor system can handle large scale virtual sensors. Secondly, we discuss whether the requirements in this paper have been implemented and further for future practical use. Both the emulated and virtualized environments are run on a PC with a 2.4 GHz Intel Core i9 (4 cores / 8 threads), 64 GB RAM, under MacOS 10.15.7 64-bit.

4.3.2 Performance Test Scenario

In the future, we aim to federate the sensor system with other simulators in order to obtain useful results for disaster response. The purpose of federating with simulators is to verify the operation of the entire disaster response system, including the operation of the IoT sensor system, power supply, and communication network status. Moreover, the sensor system is intended to be a verification environment for operators to consider how to improve the efficiency of their disaster response work. For the performance test, we prepared three scenarios, Scenario A)-C).

Scenario A) Scenario A) assumes 100 minutes of sensing time duration at 1 minute intervals, and all virtual sensors transmitted this data every second for emulation. This is a test to estimate the performance and processing time when federating an emulator that uses sensor system data with other simulaters for efficiency improvement of operation and damage prediction. Simulations for efficiency improvement and prediction can be repeated many times by changing the predicted values of sensor data and simulator parameters to calculate results for various situations. Obviously, the faster the processing time is, the greater the number of possible trials in a limited time period. For this reason, it is desirable to have a fast processing time, although it is necessary to achieve the utmost performance possible with a commodity computer in light of its utilization. The purpose of Scenario A) is to investigate the performance of a large-scale operation using a commodity computer with ordinary performance.

Scenario B) In Scenario B), we prepare several different sensing time durations as shown in Table 1, in order to investigate the performance for each of them. When the sensing time duration is 30 minutes, we assume a short simulation. For example, it can be used to analyze the prediction accuracy of flood simulations immediately before a disaster, or to simulate effective emergency measures in the operations department. With a sensing time duration of 120 minutes, it is assumed that the simulation will be used for predictions based on a flood simulation and emergency measures based on the results of the predictions, or for the simulation of emergency measures at the disaster site in case of sensor device or network failure. In the case of a relatively long sensing time duration, such as 300 minutes, we plan to simulate operations when long-term warning is required due to typhoons or long rains, and scenarios of sensor device and network failures and recovery. We will examine the performance of such short and long time simulations to find out how much processing time will be available for calcuration. For each sensing time duration, all virtual sensors transmit their data every 0.5 seconds or every 1 second. We assume 100 minutes of sensing scenario at 1 minute intervals, and all virtual sensors sent this data every second for emulation.

Scenario C) Then, in Scenario C, we estimate the performance of the sensor system in real time. In Scenarios A and B, data generated every minute was transmitted at one-second time interval. Our objective is to verify the system in a preliminary manner by federating with other simulators such as disaster response and damage prediction. Scenario C measures performance by connecting to real sensor devices and operating in a integrated environment with virtual sensors. That is, we verify whether our sensor system is capable of handling a large number of sensor devices in a integrated environment with real sensors. This is the scenario in which the CyReal sensor system shows its full potential. Here, data is transmitted every minute with a time duration of 30 minutes.

4.3.3 Result and Discussion

Scenario A) The Scenario A result obtained for the performance of the large scale installation of virtual sensors are shown in Fig. 5. This figure shows the maximum CPU usage for $n = 0 - 16$ with the number of virtual sensors as 2^n sensor devices. When the number of sensors was set to 2^0, the CPU usage was 6.8%, and the transmission time was about 90 seconds. Then, the number of sensors was set to 2^9, the CPU usage reached 97.64%. The processing time for transmission

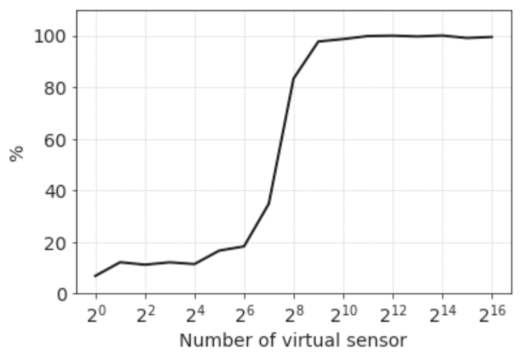


Figure 5: Scenario A) Performance (CPU usage)

in this case was 119 seconds. For $n = 10 - 16$, the CPU usage was 98.58-99.95%, almost 100%. The processing time was about 120 seconds up to the 2^{11} unit, gradually increased as the number of sensor devices increased. It reached 212 seconds with 2^{12} sensor devices, which is close to the number of 4,000 sensor devices that will be operated for disaster prevention in the near future. 1,071 seconds with 2^{14} devices, and 2,528 seconds with 2^{16} devices.

Considering the use of such a sensor system emulator during a disaster situation, two patterns can be considered: preliminaly verification, and scenarios that require real-time data verification. In the case of preliminaly verification, there is no time limitation, thus it is possible to use a processing speed of $n = 14$ or more. In contrast, in the case of real-time use, considering the sensor data missing during a disaster and related actions, it is appropriate to use about $n = 12$. It is certainly that this performance vary depending on the computer performance. However, in the case of disaster response, where it is difficult to use high performance computers, this result is useful to consider usecase scenarios.

Scenario B) Figure 6a-6d show the results of the performance test for five different time duration patterns. Figure 6a shows the processing time for transmitting sensor data with a time interval of 60 seconds, and Fig. 6b shows the CPU usage. Given that the sensing time duration is 30 minutes, the data that is sensing every minute is being transmitted at 60x speed. Therefore, if data is being transmitted and received successfully, all data transmission and receiving would be completed in about 30 seconds. Up to $n = 11$, transmission and receiving were performed in about 30 seconds, whereas as the number of sensors increased to $n = 12$ or more, the processing time required for transmission and receiving increased to 66 seconds. Similarly, in the case of 60 minutes of time duration, if processing proceeds smoothly, all data transmission and reception can be completed in about 60 seconds, while processing time is about doubled for the number of sensors above $n = 13$.

The probable reasons for this are saturation due to failure of virtual sensor process generation due to insufficient memory of the computer used for performance testing, CPU shortage, and overhead of process switching. For the number of sensors above $n = 12$ or $n = 13$, the CPU shortage and the overhead of process switching caused the increase in processing time.

Also, in the case of $n = 16$ with 120 minutes time duration, processing time decreased in spite of the increase in the number of sensors. It is considered that the process that actually did the work was less than the designated process because the transmission process of the virtual sensor was not generated due to insufficient memory. Then, with the number of sensors above $n = 12$ or $n = 13$, correct sensor transmission and receiving were not performed. This means that the commodity computer used for the performance test can be operated with this number of sensor devices or less.

Figure 6c shows the processing time for transmitting sensor data at a shortened to 120x speed, and Fig. 6d shows the CPU usage. In other words, in the case of 30 minutes of time duration, sensor data is transmitted and received every 0.5 seconds, and all data transmission is completed in 15 seconds. In the case of this transmission time interval, data was transmitted and received successfully when $n = 13$, while the processing time increased from $n = 14$. This is a typical method of estimating system performance, and depending on the time interval and time duration, as well as the computer that processes the virtual sensors and the computer of the sensor system, we can determine the number of sensors that are capable of transmission. It is essential to determine the number of sensors according to these verification environments when operating the sensor system in federation with simulation. In the commodity computer environment used for this performance test, the number of sensors that can be operated is about $n = 12$ or $n = 13$. Considering the number of sensors that are actually used in Japan, we found that this number is large enough for preliminary verification.

Scenario C) We measured the processing time of a sensor system that transmits data in real time, namely every minute, when the data is transmitted at 1x speed. Up to $n = 16$, no saturation occurred, and the data was successfully transmitted and received in 30 minutes, the same value as the time duration. This means that it is possible to operate a number of virtual sensor devices with $n = 16$ in this performance test environment. Scenario B), with 30 minutes time duration, was saturated with 2000 processes. Assuming that the overhead of process switching in particular dominates and leads to this result. Since the process switching time is 2000 processes per second, we can assume that it is about 500 [μs] per process. Considering the switching time of 500[μs/process] in a time interval of 60 seconds, the process can be estimated to saturate in $12{\times}10^4$ processes. Figure 7 shows the result of increasing the number of devices to $n = 17$ and transmitting/receiving data. At $n = 16$, the number of virtual sensor devices is 65,536, and even if all sensor devices transmit data simultaneously, the number of processes is less than $12{\times}10^4$ processes. In this case, saturation is not considered to occur. For $n = 17$, the number of virtual sensor devices is 131,072, and the number of processes exceeds $12{\times}10^4$ process. Thus, the process of the sensor system would saturate, and the processing time is expected to be more than 30 minutes.

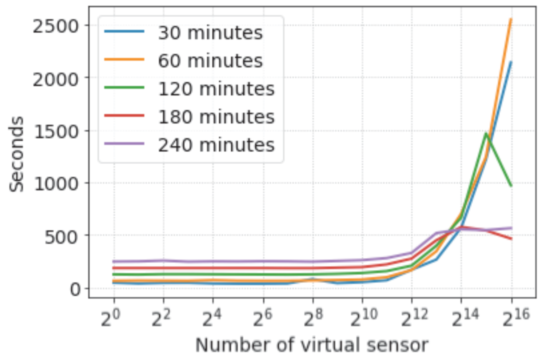
At $n = 16$, the processing time was 1,798 seconds, which is within 30 minutes. Meanwhile, at $n = 17$, the processing time was 2,043 seconds, which is considered to be saturated. This means that if the sensor system is operated in real time with a time dutation of 30 minutes and data transmission every minute, our sensor system supports 65,536 processes. On the other hand, 131,072 processes could not be handled, which is consistent with our estimate of $12{\times}10^4$ processes. This result shows that when the sensor system operates in real time under these environments, our sensor system can operate 2^{16} virtual sensor devices, considering the actual number of devices installed, this number is large enough.

Discussion As for limitations of devices, simulators, and systems, and discussion based on actual operations, we address two functional requirements for the sensor system emulator; The emulator should be compatible with a real system and be able to handle a large number of sensors. Performance tests have shown that it is possible to emulate a large number of sensors in a typical usage environment and the expected number of sensors. The compatibility with real devices already in operation is achieved by implementing the prototype in the configuration shown in Fig. 4. This configuration integrated three types of real/virtual systems: Real Node / Physical Sensor, Real Node / Virtual Sensor, and Computer Process / Virtual Sensor. We have achieved the integration requirement, then we plan to expand the types of sensor devices and communication ports. As a sensor device, we are also planning to incorporate an Emulator Node/Virtual Sensor. By emulating the Node, we will be able to test a wider variety of devices.

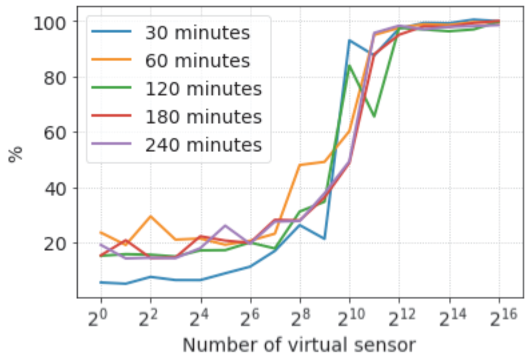
The key issues are CyReal-ness and software transparency. For the CyReal-ness, we are incorporating virtual sensors.Then we need to consider the degree of virtuality, and CyReal integration. We will provide more specialized virtualization and integration of hardware, algorithms, data, and its utilization (missing data and heterogeneous sensor data integration), rather than per node or sensor. We believe that this will enable more realistic emulation in large-scale IoT environments. The other is software transparency. The number of nodes and sensors that can be connected to this sensor system is planned to increase. By considering software that can be used in common regardless of the nodes, devices, and systems connected, the sensor system emulator will be able to easily support a large number of sensors. In addition, we will develop other simulators for specific use cases; for example, functions for optimize sensor installation and development support through fedelating a flood simulator, an evacuation simulator, a sensor system emulator for river level sensors and precipitation sensors, or a multi-agent simulator for disaster response, and a network emulator and sensor system emulator.

5 Conclusion

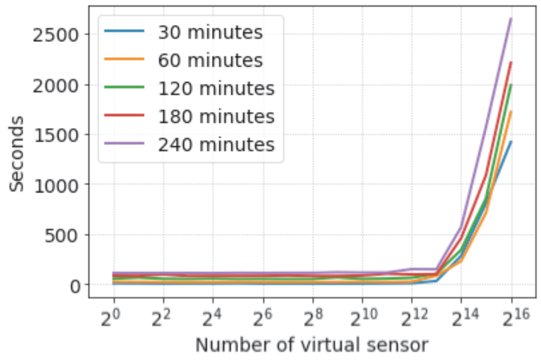
It is very difficult to prepare properly preliminaly verification for a large-scale IoT environment, in consequence, we have not benefits from IoT technologies. To address this problem, this paper proposes a sensor system emulation environment based on the CyReal concept that integrates real and virtual systems. We aim to support research, development, and operation of disaster prevention information collection



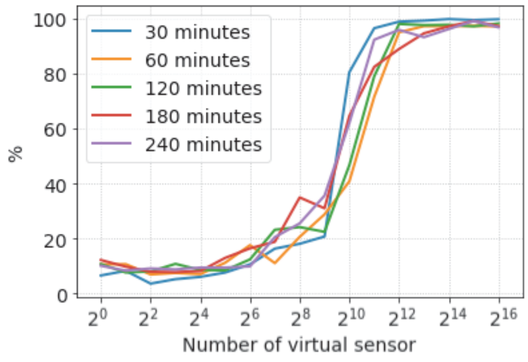
(a) Processing time (60 seconds time interval)



(b) CPU usage (60 seconds time interval)



(c) Processing time (30 seconds time interval)



(d) CPU usage (30 seconds time interval)

Figure 6: Scenario B) Performance

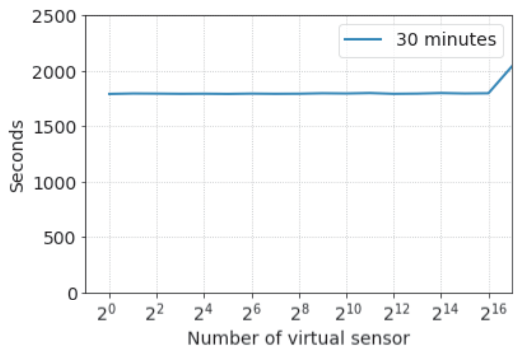


Figure 7: Performance of Sensor System (CPU usage)

and operation, and to enable preliminary verification assuming a specific installation environment and operational configuration. We expect to further develop the sensor system emulator in this paper and for verification by federating with other simulators in order to improve the efficiency of disaster response.

This paper developed a prototype of a sensor system that integrates real and virtual systems. The result of the experiments on the prototype showed the number of devices that can be used as a guide for larger scale in a general use environment, which is sufficient to operate the number of sensors currently used for disaster response. We plan to define CyReal-ness and improve the system to allow selective use of Real / Virtual at several layers: hardware, algorithms, and data. We expect preliminaly verifivcation for IoT systems from both functional and utilization aspects, including more realistic verification such as data missing.

Our objective is to develop a sensor emulator that enables preliminary verification of the behavior of the entire sensor network before installation to identify bottlenecks and determine how to resolve them, based on the premise of functional design. It is intended to be used to design measurements that satisfy the purpose of use by clarifying data due to the unique characteristics of sensor devices, relationships between data from multiple sensor devices, and data transmission characteristics due to terrain and communication infrastructure. In this paper, we have shown that Physical/Virtual Sensor/Nodes can be integrated in a CyReal Sensor System, and the performance of virtual sensors in a typical computational environment. In order to use the CyReal Sensor System for functional design in the future, it is necessary to develop an evaluation environment with large-scale physical devices, and to add functions for various conditions such as network conditions, power consumption, and environmental exposure due to the installation location, and then integrate these functions with the physical and virtual conditions. Even if we consider only the network, the variety of infrastructures available makes it necessary to consider a complex configuration to build these functions on the sensor device. Therefore, we are trying to solve this problem by developing a simulation layer that enables CyReal of network and power on a separate layer from the sensor device, and federating these simulations. Moreover, we plan to develop a verification environment for distributed installation from the perspective of geographical characteristics of sensor installation and utilization through MQTT-based federation with other simulators for external environment.

Acknowledgement

This work was supported by JST, PRESTO Grant Number JPMJPR2036, Japan.

REFERENCES

[1] A. Sarfraz, M. MR Chowdhury, J. Noll, "Senaas: An Event-driven Sensor Virtualization Approach for Internet of Things Cloud", In 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications, pp. 1–6 (2010).

[2] S. Bose, A. Gupta, S. Adhikary, N. Mukherjee, "Towards a Sensor-cloud Infrastructure with Sensor Virtualization", In the Second Workshop on Mobile Sensing, Computing and Communication, pp. 25–30 (2015).

[3] P. Evensen, M. Hein, "SenseWrap: A Service Oriented Middleware with Sensor Virtualization and Self-configuration", In 2009 IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 261–266 (2009).

[4] P. Levis, N. Lee ,M. Welsh, D. Culler, "Tossim: Accurate and Scalable Simulation of Entire Tinyos Applications, In Computer Communications and Networks", International Conference on Embedded networked sensor systems, pp. 126–137 (2003).

[5] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, "Cross-level Sensor Network Simulation with Cooja", In 31st IEEE conference on local computer networks, pp. 641–648 (2006).

[6] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, D. Estrin, "Emstar: A Software Environment for Developing and Deploying Heterogeneous Sensor-actuator Networks", In ACM Transactions on Sensor Networks (TOSN), Vol.3, No.3, pp. 1–34 (2007).

[7] B. Titzer, D. Lee, J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing", In IEEE Fourth International Conference on Information Processing in Sensor Networks(IPSN' 05), pp. 477–482 (2005).

[8] A. Sobeih, J.C. Hou, L. Kung, N. Li, H. Zhang, W. Chen, H. Tyan, H. Lim, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks", IEEE Wireless Communications, Vol.13, No.4, pp. 104–119 (2006).

[9] B. Al Homssi, k.Dakic, S. Maselli, H. Wolf, S. Kandeepan, A. Al-Hourani, "IoT Network Design Using Open-Source LoRa Coverage Emulator", IEEE Access, No.9, pp. 53636–53646 (2021).

[10] T. Maret, R. Kummer, P. Kropf, J. F. Wagen, "Freemote Emulator: A Lightweight and Visual Java Emulator for WSN", In International Conference on Wired/Wireless Internet Communications, pp. 92–103 (2008).

[11] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. Baras, "Atemu: a Fine-grained Sensor Network Simulator, In Sensor and Ad Hoc Communications and Networks", pp. 145–152 (2004).

[12] G. Kasprowicz, L. Mankiewicz, K. T. Pozniak, R. S. Romaniuk, S. Stankiewicz, G. Wrochna, "Hardware Emulator of the High-resolution CCD Sensor for the Pi of the Sky Experiment", In Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2007, Vol.6937, pp. 693–708 (2007).

[13] S. Brady, A. Hava, P. Perry, J. Murphy, D. Magoni, A. O. Portillo-Dominguez, "Towards an Emulated IoT Test Environment for Anomaly Detection using NEMU", In 2017 Global Internet of Things Summit (GIoTS), pp. 1–6 (2017).

[14] O. Abrishambaf, P. Faria, Z. Vale, "Laboratory Emulation of Energy Scheduling in an Agriculture System. In 2020 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)", pp. 1–5 (2020).

[15] S. Boschert, R. Roland, "Digital Twin—The Simulation Aspect, Mechatronic futures, Springer", pp. 59–74 (2016).

[16] S. Deda, A. Eder, V. Mhetre, A. Kuchling, R. Greul, O. Koenig, "Designing a Battery Emulator/Tester from Scratch to Prototyping to Automated Testing within a HIL Digital Twin Environment", In International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management, pp. 1–8 (2020).

[17] Z. Ye, F. Hu, L. Zhang, Z. Chu, Z. O'Neill, "A Low-Cost Experimental Testbed for Energy-Saving HVAC Control Based on Human Behavior Monitoring", International Journal of Cyber-Physical Systems (IJCPS), Vol.2, No.1, pp. 33–55 (2020).

[18] P. Evensen, M. Hein, Sensor Virtualization with Self-configuration and Flexible Interactions, In the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems", pp. 31–38 (2009).

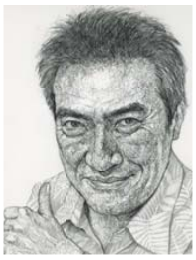
[19] H. Debnath, N. Gehani, X. Ding, R. Curtmola, C. Borcea, "Sentio: Distributed Sensor Virtualization for Mobile Apps", In 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 1–9 (2018).

[20] Z. Wang, M. Liu, S. Zhang, M. Qiu, "Sensor Virtualization for Underwater Event Detection", Journal of Systems Architecture Vol.60, No.8, pp. 619–629 (2014).

(Received: October 30, 2021)
(Accepted: July 19, 2022)

Kei Hiroi received her Master of Media Design and Ph.D. in Media Design in 2011 and 2014, respectively from Keio University. She has been an assistant professor in the department of Information and Communication Engineering, Graduate School of Engineering, Nagoya University. She is currently an associate professor in Disaster Prevention Research Institute, Kyoto University. Her research interests include disaster simulation, and crisis computing.

Akihito Kohiga is a Project Researcher of Information Sciences at Japan Advanced Institute of Science and Technology. He received Ph.D (Information Science) in 2020. His research interests lie in cloud computing, massive distributed simulation, modeling and architecture, especially in flood and evacuation fields.



Yoichi Shinoda is currently a professor of Japan Advanced Institute of Science and Technology. His research interests include Impact of digital technologies on human activities, Parallel and Distributed Systems, Networking Protocols and Systems, and Information Handlinf Systems.

Regular Paper

Adaptive NDN Content Delivery Mechanism on Mobile Networks

Taichi Iwamoto[†]and Tetsuya Shigeyasu[‡]

[†]Graduate School of Comprehensive Scientific Research, Prefectural University of Hiroshima, Japan
[‡]Department of Management Information Systems, Prefectural University of Hiroshima, Japan
sigeyasu@pu-hiroshima.ac.jp

Abstract - Recently, named data networking (NDN) has attracted considerable attention from network researchers as a network architecture based on a content-centric manner. NDN delivers user requests according to the shortest path recorded in the forward information base (FIB) of relay content routers (CRs). Incidentally, the development of portable ICT devices enables the publishing of content even in the mobile environment. The current version of NDN, however, does not consider publisher migration. If the location of the content publisher changes, NDN cannot deliver the *Interest*/content to the appropriate destination accurately. In this study, we discuss how to address the publisher migration problem on NDN and propose a new adaptive content delivery method and priority cache holding method. The proposed methods are evaluated under the scenario that the mobile publisher changes its location. The results of the evaluations confirm that our proposal improves the performance of content delivery even in a mobile environment.

Keywords: NDN, publisher migration, cache management, adaptive content delivery

1 INTRODUCTION

Recently, many new network architectures based on information centric data delivery have been proposed [1]-[4]. Named data networking (NDN) is a popular architecture that has garnered considerable attention from network researchers as a network architecture based on a content-centric manner. NDN delivers users' requests and their corresponding contents according to the shortest path recorded in the forward information base (FIB) of relay content routers (CRs). Incidentally, the development of portable ICT devices enables the publishing of content even in the mobile environment.

The current version of NDN, however, does not consider publisher migration. For example, NDN does not update relay information in the FIB in accordance with the dynamic topology change. When the topology change occurs by node migration, relay information in the FIB can no longer be used. Wrong FIB information misleads *Interest* the content requested by users and increases network traffic needlessly. Therefore, a new mechanism for updating FIB information quickly in response to publisher migration is required.

Based on the situation of a publisher's migration, contents generated on the migrating publisher cannot be reached from the network. Network reachability for those contents will recover once the migrating publisher completes its location change and reconnects to the network. Thus, during the period from the beginning and end of migration, the users will not be able to obtain the contents of the migrating publisher if the CRs do not hold the cache. Hence, a new cache management method for maintaining the CR cache of a mobile publisher during its migration is required.

To address this publisher migration problem on NDN, this study proposes a new adaptive content delivery mechanism for mobile networks, comprising two parts: pre-forwarding with update method of old/wrong FIB information, and preferentially cache-keep method for migrating publishers' contents.

In the first method, the old/incorrect FIB information of relay CRs is canceled by control information on pre-forwarded contents transmitted from the migrating publisher before its migration. Subsequently, *Interest* destined for contents of migrating publishers will not be forwarded to the previous location of the publisher. In addition, after the migrating publisher arrives at a new location, another control information is transmitted to form new FIB information for relay CRs. Thereafter, *Interest* destined for the contents will be forwarded to an appropriate new location.

In the second method, during the publisher migration, the cache of the contents generated by the publisher will be preferentially stored on CRs. Therefore, content can be obtained more easily, even if there is no network access to the publisher.

The proposed methods will be evaluated considering scenarios where the mobile publisher changes its location. The results of the evaluations confirm that our proposal improves the performance of content acquisition ratios.

2 PROCEDURE OF NDN

NDN is an architecture that performs both content discovery and content delivery in a content-centric manner. NDN performs the procedure using two packet types: *Interest* and *Data*, as shown as Fig.1.

The *Interest* is used for requesting the desired contents. If a content request has newly arrived at a user, the user transmits this *Interest* toward a publisher/CR with original/cache content of the corresponding content. Meanwhile, *Data* is used for returning the content corresponding to *Interest*. The publisher/CR returns the content as *Data* if and only if it has the contents/cache corresponding to the received *Interest*. The *Data* will be forwarded along the reverse path of the previously received *Interest*.

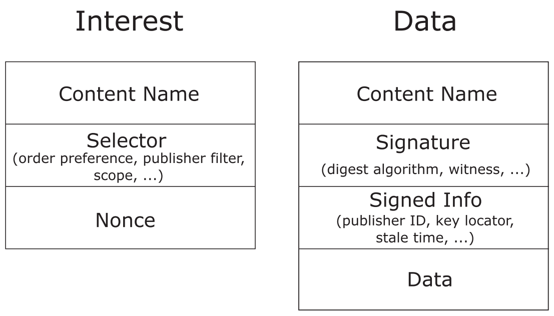

Interest
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

Figure 1: Frame format of *Interest* and Content on NDN.

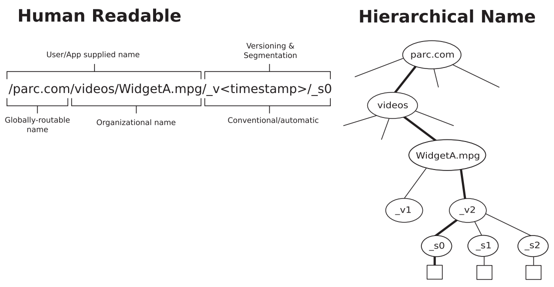


Figure 2: Naming structure of NDN.

In NDN, the name of the content is used as an identifier. All contents will be requested by specifying the names of each content. As shown in Fig. 2, the contents are named using a hierarchical structure divided by ”/”.

2.1 Construction of CR on NDN

As shown in Fig. 3, CR performs packet forwarding according to the following three tables.

- FIB
  FIB consists of information used for forwarding *Interest* toward a content publisher. When a CR receives a new *Interest*, the CR forwards it according to the information registered in its FIB. In the FIB, each record consists of two fields: prefix of the contents (i.e., the name of contents) and an interface number that must be forwarded the *Interest*.

- PIT (Pending *Interest* Table)
  PIT maintains a history of received content requests corresponding to the forwarded *Interest*. Each PIT record consists of two fields: the prefix of the forwarded *Interest* and several interfaces to which the *Interest* was forwarded. By referring to a PIT entry, the CR can return *Data* to the appropriate direction if the CR receives it in the future.

- CS (Content Store)
  CS is a buffer to store caches of the content temporarily. An entry of CS consists of the prefix and cache of the content. CR returns the cache of the content instead of the publisher generating the original content if and only if the CR has the corresponding cache of the requested contents. When the content is returned to a user from the CR in shorter hops than those required by the original publisher, the amount of network traffic and response time can be reduced effectively.

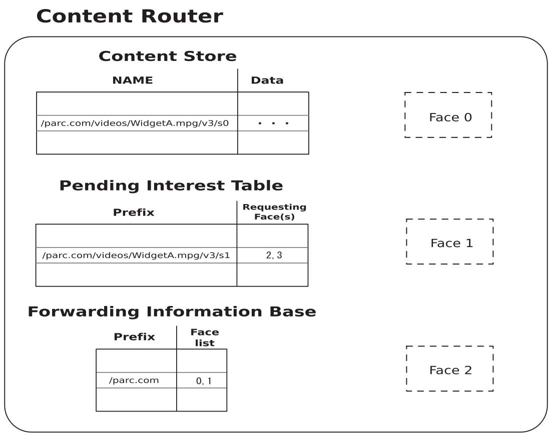


Figure 3: Contents Router in NDN.

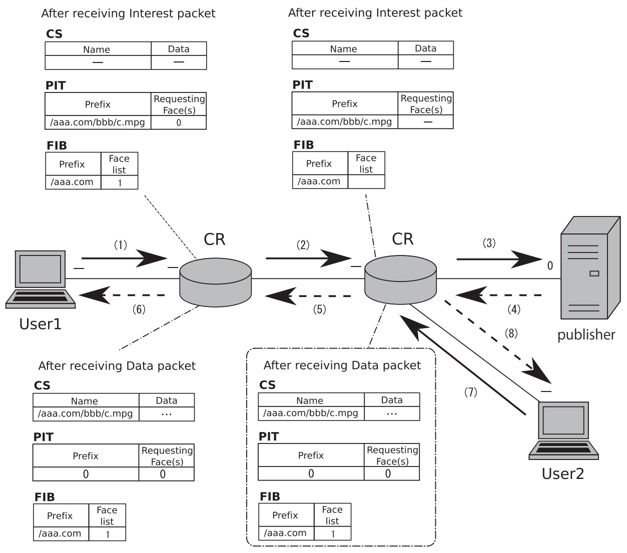


Figure 4: Basic procedure of NDN

2.2 Basic Procedure of NDN

Figure 4 illustrates the basic procedure of NDN. As shown in the figure, (1) the user transmits the *Interest* to the neighboring CR. The CR receiving the *Interest* checks whether the corresponding cache to the *Interest* is stored in its buffer. If a cache exists, the CR returns it as *Data* through the interface that the *Interest* came from; otherwise, the CR registers the information of the *Interest* into its PIT; (2) the *Interest* is forwarded to the next upstream CR. The next upstream CR is selected according to the entry of FIB. If there is no cache corresponding to the *Interest*, (3) the *Interest* is forwarded to the original publisher, who then returns the contents to the user.

During *Data* returning, the CR receiving the returning *Data* stores the *Data* to its CS as a cache, and the CR checks if it

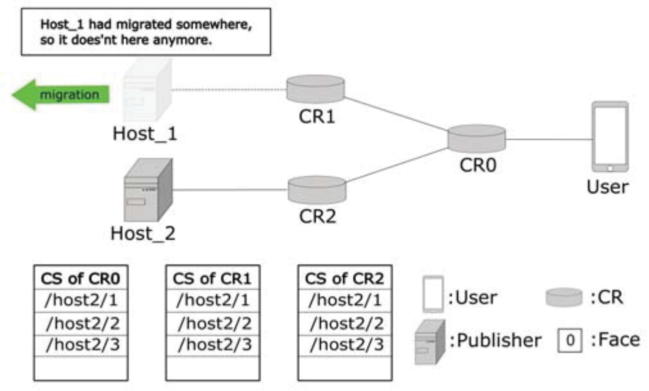


Figure 5: Missing access problem to contents during publisher migration

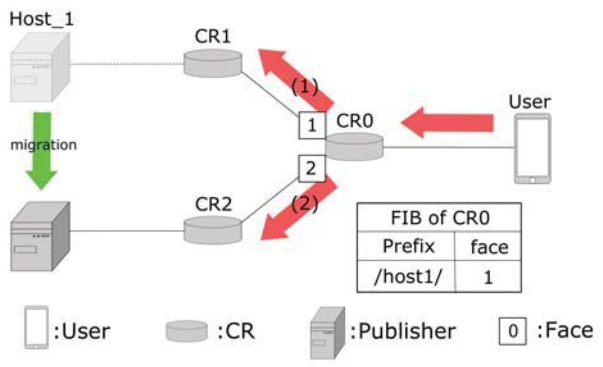


Figure 6: Incorrect *Interest* forwarding problem induced by old FIB information

has an entry corresponding to *Data* on its PIT. If the entry exists, the CR returns the *Data* through the interface(s) in accordance with the entry and removes the entry from the PIT. By repeating the process, CR can deliver the *Data* to the user requesting the contents, even if the *Interest* excludes the location information of the user.

In addition, when other user(s) request the same content, (7)(8), the CR responds by returning the cache of content stored in CS, with a shorter delay compared with traditional host-centric network. Moreover, if the CRs estimate the popularities and keep caches of the higher popular content, the cache hit ratio at the the CR can be effectively improved, and the amount of network traffic can be effectively reduced in NDN.

3 PROBLEMS ON CONVENTIONAL NDN

This section describes the three problems induced by publisher migration on conventional NDN.

3.1 Missing Access Problem to Contents during Publisher Migration

Once the mobile publisher starts its migration to a new location, users requesting the contents may lose access to the original content generated by the publisher. Figure 5 illustrates this problem. As shown, after the beginning of the migration of Host_1, *Interest*s destined for Host_1 cannot reach the desired contents if the requested contents are not stored in CSs on intermediate CRs (CR0, CR1) between the user and Host_1.

3.2 Incorrect Interest Forwarding Problem Induced by Old FIB Information

On conventional NDN, old FIB information will be kept holding on CRs, even if a publisher changes its location by its migration. Thus, the *Interest*s destined to a content generated by the publisher will be delivered to the old publisher location according to the old FIB information. As shown in Fig. 6, the *Interest* destined to the Host_1 content will be forwarded to direction (1), despite the right path to the current Host_1's new location being (2).

4 RELATED WORKS

This section describes the developments of network architectures related to the NDN. In order to reduce both of the network traffics and delays for content delivery, NDN is expected to be replaced instead of traditional location centric network, namely, IP network architecture. However, because current Internet infrastructure is established based on IP, most applications used worldwide are also guaranteed to work only over the IP architecture. Hence, for a smooth architecture migration from TCP/IP to NDN, some methods have been discussed.

The literature [5] discussed methods for coexisting IP and NDN architectures. In this literature, authors classify the existing methods into three types: stack modification, encapsulation and translation. In the literature [6], a method, named, IP/NDN, for coexisting IP and NDN by introducing translation mechanism has been. proposed. In the IP/NDN, translation is implemented to realize coexisting two architectures in same network. IP datagrams from senders are captured at TUN device, and translated into NDN Interest or Data. Those architecture coexistence methods can push architecture migration, IP to NDN.

Incidentally, NDN belongs to the category of Information Centric Networking (ICN) which is most focusing on getting/delivering contents efficiently instead of knowing node location. Currently, the most practical ICN system is CDN (Content Delivery Network). The literature [7] has proposed that content migration method to realize efficient content delivery under the client node mobilities on vehicular network. In the literature, authors proposed a strategy to place contents on network node as edge router cache, based on deep reinforcement learning approach. However, in this method, only client mobilities are considered. In other words, how to deal with the mobilities of publisher is not discussed.

Previously, a method for a proactive selective neighbor caching strategy has been proposed [8]. In this strategy, the contents going to be requested by mobile users will be cached proactively on selected neighbors. Neighbor candidates for proactive caching are selected based on the cache cost delay and mobility behavior. This strategy is for mobility support on the CCN, focusing on user mobility and not on publisher mobility.

DONA also handles events related to the user (consumer mobility) [9]. In DONA, content request/delivery is handled by introducing a resource handler (RH) mechanism, which acts as a domain name system (DNS) server. According to user requests, RH provides the information for rendezvous for content delivery from publishers. In DONA, the user's mobility event is handled by changing RH information.

Although the above mechanism focuses on mobility support on information-centric network architectures, they are focused on user (consumer) mobility rather than on publisher mobility.

4.1 PMC: Publisher Mobility Support Protocol in CCN

To solve the publisher migration problem on NDN, the publisher mobility support protocol (PMC) was proposed [10]. In the PMC, to cope with publisher migration, a publisher selects a *HomeNode*. The publisher registers its new location to the *HomeNode* when it migrates to another place. Using the *HomeNode*, *Interest*s that arrive at the old publisher's location owing to the old/incorrect FIB entry can be correctly forwarded to the new publisher's location. Figure 7 demonstrates the procedure of *Interest* forwarding on PMC. As shown, after a publisher migration, the publisher sends an MR request, including its new location information, to the *HomeNode*. The *HomeNode* returns an MR response to the mobile publisher. At this time, CRs belonging to the forwarding path of the MR response update their FIB entry according to the received MR response.

After the transmission of the MR response, CRs can forward the future *Interest* destined for the contents generated by the mobile publisher.

In the PMC, publisher mobility can be solved by introducing location support of the mobile publisher using *HomeNode*, holding information of both the old and new locations of the migrating publisher. However, without support from the *HomeNode*, PMC cannot address the mobility problem of the publisher, autonomously, Hence, in the following sections, we propose a new autonomous solution for the mobility problem by the publisher.

5 PROPOSAL

We propose a new solution for coping with the problem induced by publisher migration on conventional NDN. Our solution consists of two parts: 1) content pre-forwarding and updating old FIB information, and 2) prioritizing holding for caches generated by mobile publishers until it recovers network connections.

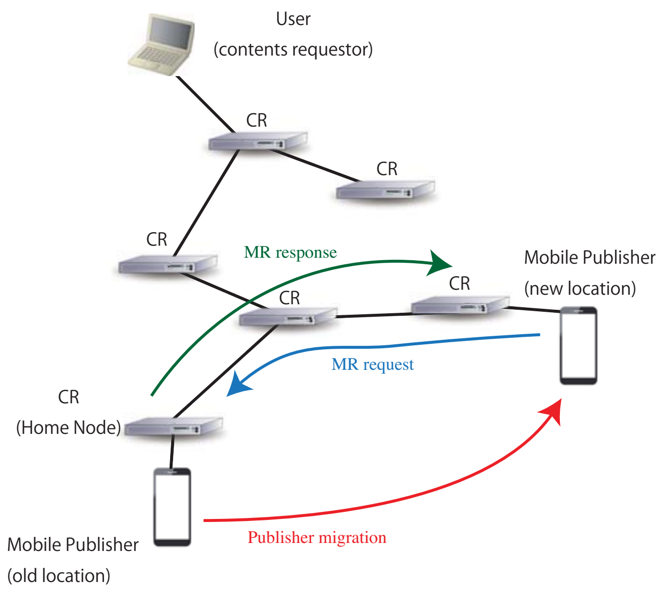


Figure 7: Procedure of PMC

5.1 Content Pre-forwarding and Updating Old FIB Information

This section describes the first part of the proposal. The first proposal, namely content pre-forwarding and updating FIB information, is further divided into two methods: before leaving method and after arrival method.

Hence, the followings describe these two methods in detail.

5.1.1 Before Leaving Method: A Method for Contents Pre-forwarding and Updating FIB Information

As we mentioned previously, during the publisher migration, all users willing to acquire content lose access to them if the desired contents are not stored in the CS of CRs belonging to the *Interest* forwarding path. Therefore, this section proposes a method for content pre-forwarding from the mobile publisher to the neighboring CR before publisher migration and for updating FIB information according to the header information of the pre-forwarding contents. By pre-forwarding, *Interest* can be delivered to the CS holding the corresponding contents even if the mobile publisher changes its location. This phenomenon increases content acquisition ratio. By using the pre-forwarded contents, this method also deletes the old (wrong) FIB information on CRs belonging to the shortest path from users to the previous publisher location. For the deletion of FIB information, caches of pre-forwarded content are marked with a *M-flag* (Migration flag), which indicates that the publisher that generated the content already left the network from the old location. Similar to the conventional approach, the cache of the pre-forwarded content is returned when the CR receives the corresponding *Interest*. However, the CR deletes the FIB information corresponding to the content name when it receives the content with the *M-flag*. Figure 8 demonstrates the procedure of this mechanism.

The detailed procedure of NDN employing both content pre-forwarding and FIB information updating by *M-flag* is described as follows:

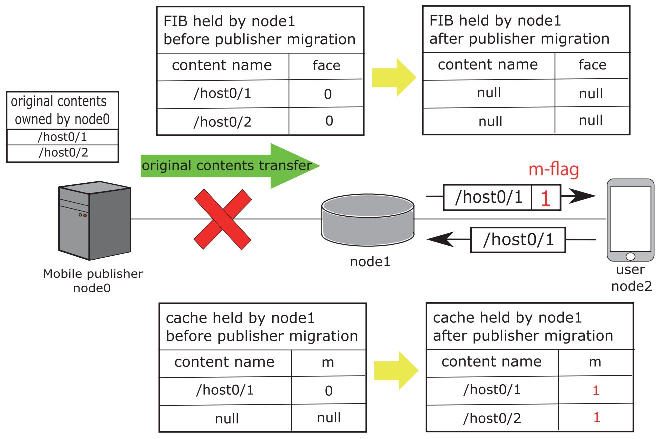

FIB held by node1 before publisher migration
content name
face
/host0/1
0
/host0/2
0
FIB held by node1 after publisher migration
content name
face
null
null
null
null
original contents owned by node0
/host0/1
/host0/2
original contents transfer
m-flag
/host0/1
1
/host0/1
Mobile publisher node0
node1
user node2
cache held by node1 before publisher migration
content name
m
/host0/1
0
null
null
cache held by node1 after publisher migration
content name
m
/host0/1
1
/host0/2
1

Figure 8: Procedure of content pre-forwarding and updating FIB information by *M-flag*

1. Procedure on CR when receiving contents with *M-flag*
    - (a) Store received content into its CS with *M-flag*.
    - (b) Forward received content with *M-flag* to its downstream CRs registered in its PIT.
    - (c) Delete its FIB information corresponding to the same content name with the received content.
2. Procedure on CR when receiving contents without *M-flag*.
    - (a) Search FIB information having the same content name as the newly received content. Register the new FIB with the information of the received face if the corresponding FIB information is not recorded yet.
    - (b) Search its CS whether content with the same content name is cached or not. If not, store the received content. Otherwise, when the content having the same content name is already cached, CR further checks whether the *M-flag* is set. If so, it unsets the *M-flag*.
    - (c) Forward the received content without M-flag if the corresponding entry is recorded into its PIT.
3. Procedure on CR when it receives the *Interest* (This is similar to the conventional NDN procedure.)

    CR returns corresponding content when it contains the desired cache. Otherwise, it records the requested information regarding the received *Interest* on its PIT and forwards the *Interest* to its upstream CR.

As described in the above pre-forwarding process, a CR that receives an Interest for a content for which M-flag is set returns the content only to the CRs listed in the face registered in its own PIT.

M-flag has the role of deleting FIB information that is no longer needed in order to prevent unnecessary forwarding of Interest to the mobile publisher that has moved to new location.

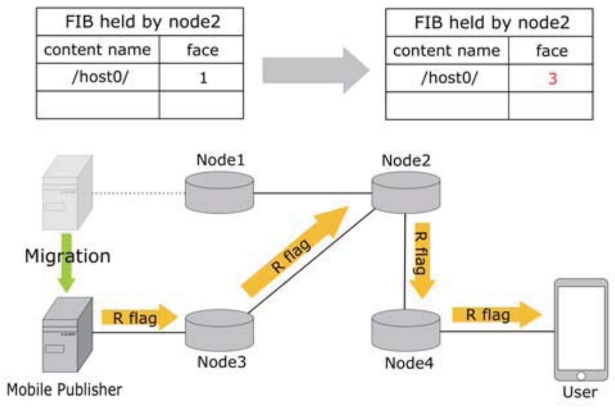


Figure 9: Procedure of dissemination of new FIB information by *R-flag*

However, as described in the following section, the mobile publisher will send another R-flag to set the new location information to all potential users after the move.

Therefore, the pre-forwarding process reduces unnecessary Interest forwarding traffic by removing FIB information from users who have sent Interest to the relevant content and CRs on their Interest forwarding path.

5.1.2 After Arriving Method: A Method for Dissemination of New FIB Information

This section describes the after arriving method that disseminates new FIB information to CRs. In this method, the migrated publisher disseminates the control packet from the new location toward users when its migration is finished. The control packet with *R-flag* (Rebuild flag) on its packet header is transmitted from the migrated publisher. By receiving the control packet with *R-flag*, the CRs that belong to the shortest path among the mobile publisher and users renew their FIB information immediately.

Figure 9 illustrates the procedure for dissemination of new FIB information using *R-flag*. In this method, mobile publishers transmit new control packets with *R-flag* after the completion of their migration. The control packet is then forwarded to the user. At this time, the control packet with *R-flag* is forwarded in all directions that received at least one *Interest*, regardless of its content name.

CRs receiving the content with *R-flag* set new FIB information according to the arrival face of the content with *R-flag*. Accordingly, CRs can obtain the desired content without turning on multicast *Interest* forwarding, which increases needless traffic. The content with the R-flag is forwarded to the downstream CRs until end users.

The detailed procedure for introducing the *R-flag* is described as follows:

1. Procedure for CRs receiving content with *R-flag*
    - (a) Check the FIB information relating to the same content name with the content possessing an *R-*

flag. If it does not exist, the CR adds the FIB information.

(b) Store the received content into it CS.

(c) Forward the content with R-flag to the downstream CRs. To reduce the forwarding traffic, the content is only forwarded to the downstream CRs from all faces that have received any previous Interest.

2. Procedure for users receiving content with *R-flag*

A user receiving content with *R-flag* checks its FIB information. If there is no corresponding FIB entry, the user adds the FIB entry according to the received content with the *R-flag*.

The proposed method uses R-flag to inform all potential users of the mobile publisher's new location. The CR that receives the R-flag forwards the R-flag to downstream CRs from all faces that have received any previous Interest.

In other words, the R-flag to update FIB information is forwarded to all CRs that have users under them. Therefore, as the number of mobile publishers in the network increases, the network load due to R-flag forwarding also increases.

5.2 Priority Cache Holding for Mobile Publisher Contents

In the previous section, we proposed a method that employs both pre-forwarding and updating old FIB information. By implementing this method, owing to cache pre-forwarding, the cache hit ratio can likely be improved even when the mobile publisher is disconnected from the network.

The pre-forwarded caches, however, will be removed from the CR buffer when the buffer overflows by the new arrival cache. If the pre-forwarded cache is removed from all CRs' buffers, although the mobile publisher has not yet completed its migration, the cache cannot be obtained by any user.

Therefore, to improve cache acquisition performance, it is desirable to keep holding the pre-forwarded cache preferentially, while the mobile publisher is disconnecting from the network.

Figure 10 presents an overview of the proposed priority cache holding method. Before starting its migration, the mobile publisher records the current neighboring CR (in Fig. reffig:priori, the current neighbor is Node1).

The mobile publisher transfers its original content according to the pre-forwarding method, as described in 5.1.1. On priority cache holding, pre-forwarded caches in CRs are recognized by referring to their *M-flag*; such caches are given preference over those without *M-flag*. In case of the necessity to remove any cache from CS on CR due to the arrival of new content, the cache without *M-flag* will be removed first. If and only if there is no cache without *M-flag* in the CS, the pre-forwarded cache, which is the cache with *M-flag*, will be removed from CS (Please note that the cache without *M-flag* can be obtained from the original publisher because the publisher is connected to the network).

After the mobile publisher finishes its migration, the publisher sends a control packet, named *Address* packet, to the CR (Node1, in the case of Fig. 10) recorded before the migration. The CRs receiving the *Address* forward it to the designated CR according to the location information without referring to the PIT, FIB, and CS, indicating that *Address* will be transferred by location base.

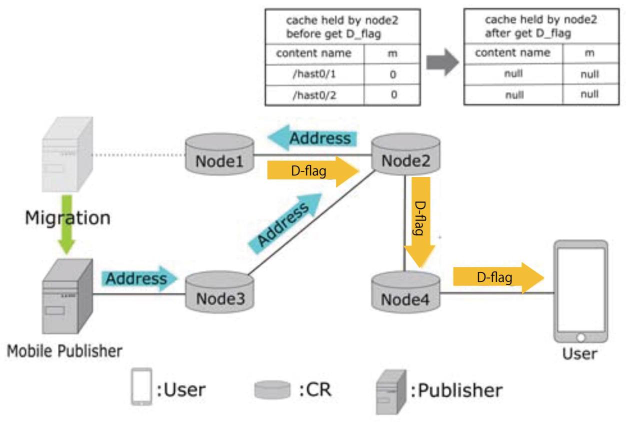


Figure 10: Method for priority cache holding for mobile publisher contents

The destination CR transfers the control packet with *D-flag* (Delete flag) to the downstream from all IFs that transferred the pre-forwarded cache, namely content with *M-flag*. The forwarding is repeated until the packet with the *D-flag* arrives at the user end. The CRs that receive the packet with *D-flag* unset *M-flag* from all caches corresponding to the received *D-flag*. Subsequently, the caches are treated the same as common caches on NDN.

5.3 Advantage and Disadvantage of Proposal

5.3.1 Effects on Content Acquisition Delay

As described in the section4, PMC method has been proposed to cope with the publisher migration. However, on the PMC, all first requests for migration publisher's content must be forwarded to the *HomeNode* for reaching the new publisher location. This leads to increase unneeded delay. In addition, consumers could not get the mobile publisher's content before the publisher finish its migration, on the PMC.

On the other hand, our proposal method can deliver content without reaching old location (*HomeNode* in case of PMC). In addition, consumers can get the mobile publisher's content if the desired content is pre-forwarded to the neighbor CR of the mobile publisher, even before the mobile publisher finish its migration.

5.3.2 Overhead Induced by Proposal

As described in section 5.1.1, the *M-flag* process in the proposed method is not a method to immediately rewrite old FIB information on all CRs. Therefore, it is not a method that significantly increases the network load.

In addition, the effect of pre-forwarding the *M-flagged* content depends on the cache capacity of the neighboring CRs

Table 1: Simulation paprameters

Parameter	Value
Number of nodes	24
Interest generation rate	100 [request/sec]
Number of contents	1,000
Interest Packet size	1,024 [bytes]
Content Packet size	1,024 [bytes]
Cache capacity	infinity
Simulation length	50[sec]
Time to start publisher migration	10 [sec]
Time to finish publisher migration	40 [sec]

before migration of the mobile publisher. When the cache capacity is small, the effect is not high, but it does not degrade the performance of content delivery compared to conventional NDN that does not consider publisher migration.

On the other hand, the proposed method delivers *R-flags* to all CRs having users under them when a publisher migration is completed.

Of course, R-flags are delivered together in a single packet where routes for users overlap. Therefore, in a network where many users are connected to a small number of edge CRs, the load generated by the R-flag forwarding process is not a significant problem. However, in a network where many users are equally connected to many edge CRs, the load caused by *R-flag* processing becomes significant.

6 PERFORMANCE EVALUATION

To clarify the applicability of the proposed methods, this section evaluates the methods using computer simulation.

6.1 Effects of Ratio of Pre-forwarding Contents on Proposed Method

This section evaluates the effects of the ratio of pre-forwarding contents on the proposed method for content pre-forwarding and updating old FIB information.

6.1.1 Evaluation Environment

The evaluation environment is described in this section. Table. 1 lists the parameters used in the simulation.

In this paper, we assumed to use the NDN architecture in which mobile publisher publishes SNS messages consisting text-based information mainly (not the multimedia content). Hence, the size of the content will be delivered to the consumers are short in the evaluations. For the simplification of the evaluations, we used same value, 1024 byte as the length of the both Interest and Content. Although it would be better to use more accurate length of the both packets (at least, Content length is larger than Interest in terms of SNS message), length of the both packets does not influence the characteristics of the evaluation results because of we have mainly focused on content acquisition ratio.

Figure 11 shows the simulation topology. The number of publishers, CRs, and users are 1, 18, and 5, respectively. The mobile publisher starts to migrate from the neighbor of CR1

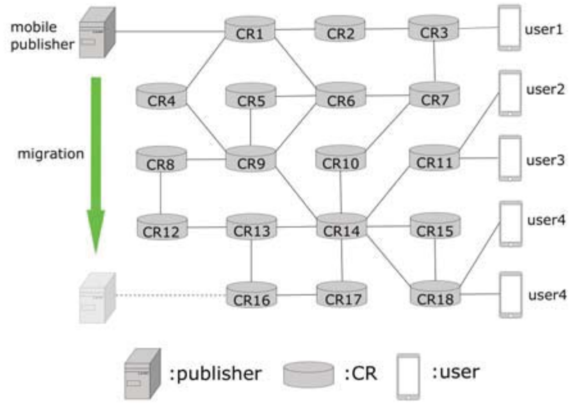


Figure 11: Evaluation topology

to the neighbor of CR16 during the simulation. Content requests arrive at all users at a rate of 100 [request/s].

The mobile publisher published 1,000 original contents. The content name recorded in *Interest* is randomly selected within the range. Pre-forwarding contents are also randomly selected at the beginning of publisher migration.

In the simulation, the ratio of pre-forwarding content, time to live (TTL) of *Interest*, and the capacity of the PIT are varied to clarify the characteristics of the proposed method. For the evaluation, the content acquisition ratio is derived as a value of the number of contents obtained divided by the number of contents requested.

6.1.2 Relationship between Content Acquisition Ratio and Ratio of Pre-forwarding Content

This section reports the characteristics of the content acquisition ratio under varying amounts of forwarding content. Values of TTL and PIT are 0.1 [s] and infinity, respectively.

Figure 12 presents the simulation results of three methods, namely pre-forwarding with *M-flag* control, pre-forwarding with both *M-flag* and *R-flag* control, and conventional as pre-forward(M), pre-forward(M/R), and Conventional, respectively.

In addition, Conventional implies the performance of original NDN. The figure shows that both pre-forwarding with *M-flag* and pre-forwarding with *M-flag* and *R-flag* increase the content acquisition ratio according to the amount of pre-forwarding content. In addition, pre-forwarding with both *M-flag* and *R-flag* always achieves a higher performance than that with only *M-flag*. The difference between the two methods is due to the fast rebuild of the *Interest* forwarding path by the *R-flag*.

6.1.3 Relationship between Content Acquisition Ratio and Length of TTL

This section reports characteristics of content acquisition ratio under varying lengths of TTL. For this evaluation, all contents are forwarded to the neighbor of the publisher at the beginning of the publisher?s migration.

Figure 13 presents the evaluation results, wherein the horizontal and vertical axes represent the length of TTL and con-

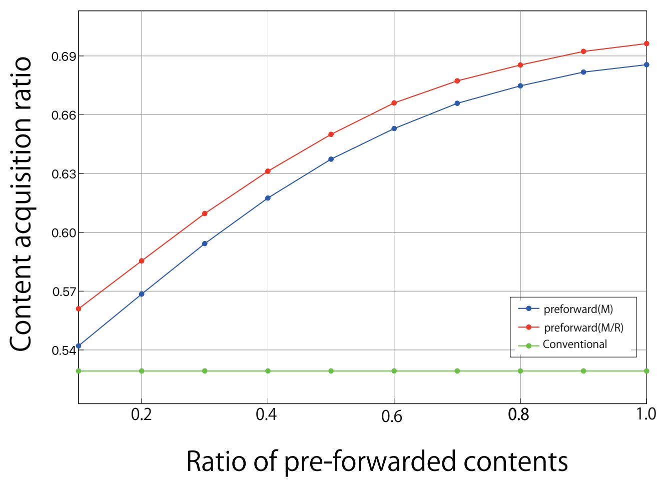


Figure 12: Characteristics of content acquisition ratio - amount of pre-forwarding content

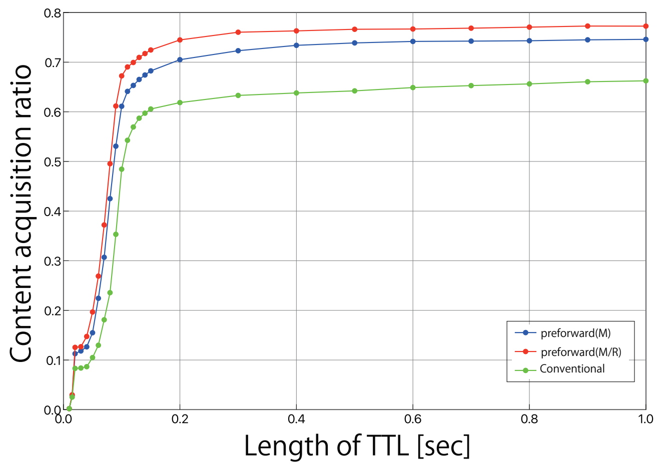


Figure 13: Characteristics of content acquisition ratio - Length of TTL.

tent acquisition ratio, respectively. The colored lines show the same mean as the previous figure.

All methods increase the content acquisition ratio with increasing TTL length. In addition, the increase in the content acquisition ratio becomes small when the TTL exceeds 0.1 [s]. This is because the average round trip time (RTT) for content acquisition is 0.2 [s]. Furthermore, if the TTL is more than the RTT, the content acquisition ratio did not increase.

As the figure shows, pre-forward(M/R) always maintains the highest performance compared to the other two methods.

6.1.4 Relationship between Content Acquisition Rate and PIT Capacity

This section evaluates the characteristics of content acquisition rate and PIT capacity. We use 0.1 [s] as the TTL on the evaluation.

Figure 14 presents the results of the evaluation. As shown, the content acquisition rate of all methods increases in accordance with the PIT capacity when the PIT capacity is smaller than 5. However, the increase in the content acquisition rate

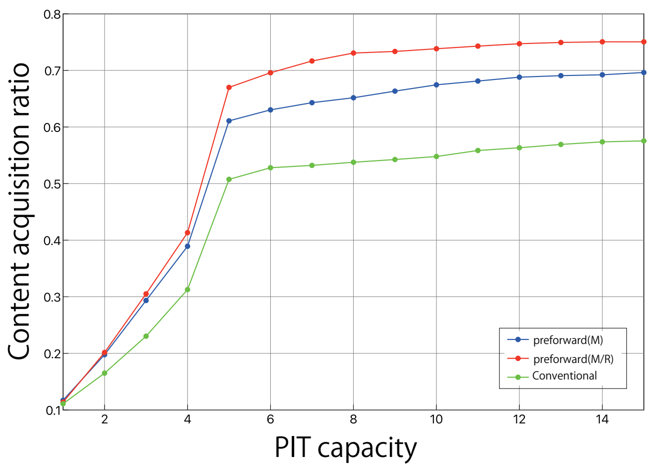


Figure 14: Characteristics of content acquisition ratio - PIT capacity.

Table 2: Simulation paprameters

Parameter	Value
Number of nodes	24
Interest generation rate	100 [request/sec]
Number of each publisher's content	100
Interest Packet size	1,024 [bytes]
Content Packet size	1,024 [bytes]
Cache capacity	100 [caches]
Simulation length	100 [sec]
Time to start publisher migration	20 [sec]
Time to finish publisher migration	80 [sec]

decreases when the PIT capacity exceeds 5.

As shown in this figure, pre-forward(M/R) always has an advantage over the other two methods.

6.2 Effects of Priority Cache Holding for Mobile Publisher's Contents

This section evaluates the effects of the second part of the proposal, namely the priority cache holding for mobile publisher's contents.

Simulation parameters used in this section are shown in Fig. 2. In this evaluations, we assume to use our proposal at the suburban area. At the mountain district, many points are outside the communication range due to the geographical features. Hence, network inaccessibility time by publisher migration will be longer. As shown in the table, the mobile publisher starts its migration at 20 [s] after the beginning of the simulation and finishes its migration at 80 [s].

Interest is generated at each user end every 0.01 [s]. In each *Interest*, one content name is selected randomly from the contents of two publishers regardless of the publisher's mobility (mobile or stationary). Before the migration of the mobile publisher, it pre-forwards its content to the neighbor CR (CR1 in this case). In addition, pre-forwarded contents are treated using the method of pre-forwarded cache holding.

Figure 16 present the results of content acquisition ratio under the varying amounts of pre-forwarding contents. As

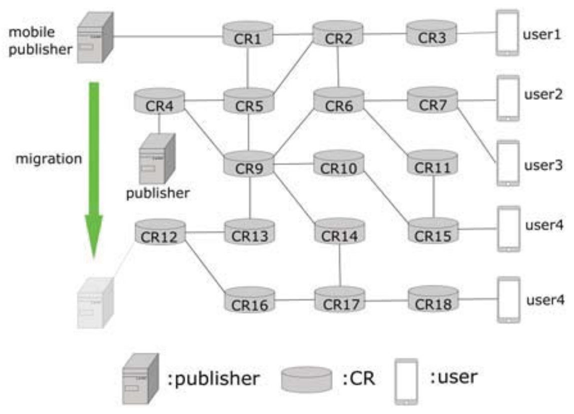


Figure 15: Topology for evaluation of the effects of priority cache holding

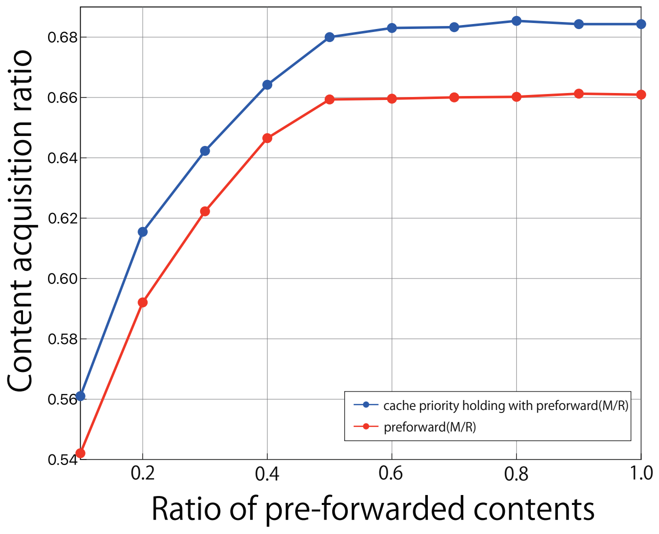


Figure 16: Relationship between content acquisition ratio and ratio of pre-forwarding content on priority cache holding method

shown, by adding the method of priority cache holding to the first part of our proposal (pre-forward(M/R)), higher content acquisition ratio is achieved, regardless of the ratio of pre-forwarded contents. By holding the cache of the mobile publisher's contents while migrating, users can obtain the mobile publisher's content. Moreover, the contents generated at the stationary publisher can be obtained directly from the publisher, even without a CR holding the cache of a stationary publisher. Hence, the priority cache holding of the mobile publisher has no negative effects in terms of contents acquisition ratio.

7 CONCLUSION

The development of ICT enables the publishing of content for delivering many network users, even by mobile devices. This makes it more flexible and adds more richness to the content generation activities. Moreover, ensuring content provisioning is gaining importance for such mobile content publishing activities.

This paper proposed solutions to deal with such issues by implementing new features on NDN that has received much attention as future network architecture. We proposed a method for content pre-forwarding and updating old FIB information effectively, which avoids degrading the content acquisition ratio. In addition, the priority cache holding of mobile publishers? contents is effective for further improvement of the content acquisition ratio. The applicability of the proposed methods is clarified by computer simulations.

In this study, we evaluated the performance of the proposed system in a scenario where the network contained one mobile publisher. However, in the real situation, multiple mobile publishers must be considered. In addition, it is obviously that our proposed content pre-forwarding method is only useful when the root CR, which means the next CR to the mobile publisher can store the additional cache during the publisher migration process. In this paper, in order to clarify the constitutive performance of the proposal, we have evaluated the methods under the situation that CR holding unlimited cache buffer. However, in the real situations, all CRs hold a finite cache buffer. Hence, we will continue to discuss detailed parameters for implementing our method, particularly for the priority cache-holding policy.

REFERENCES

[1] G. Tyson, A. Mauthe, S. Kaune, P. Grace, and T. Plagemann, “Juno: An adaptive delivery-centric middleware,” Proc. of 2012 IEEE Consumer Communications and Networking Conference (CCNC), pp. 587-591 (2012).

[2] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, “Developing information networking further: From PSIRP to PURSUIT,” in Proc. of Broadnets, pp. 1–13 (2010).

[3] B. Ahlgren et al., “Design considerations for a network of information,” Proc. of CoNEXT, pp. 66 (2008).

[4] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs and R. Braynard, “Networking named content,” Proc. of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT ’09, pp. 1–12 (2009).

[5] F. Fahrianto and N. Kamiyama, “Comparison of Migration Approaches of ICN/NDN on IP Networks,” Proc. of 2020 Fifth International Conference on Informatics and Computing (ICIC), pp. 1–7, (2020).

[6] S. Luo, S. Zhong and K. Lei, “IP/NDN: A multi-level translation and migration mechanism,” Proc. of NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–5 (2018).

[7] S. Malektaji, A. Ebrahimzadeh, H. Elbiaze, R. H. Glitho and S. Kianpisheh, “Deep Reinforcement Learning-Based Content Migration for Edge Content Delivery Networks With Vehicular Nodes,” Trans. on IEEE Network and Service Management, Vol. 18, No. 3, pp. 3415-3431 (2021).

[8] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis, “Proactive selective neighbor caching for enhanc-

ing mobility support in information-centric networks," Proc. of the second edition of the ICN workshop on Information-centric networking (ICN '12). pp.61–66 (2012).

[9] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture, " SIGCOMM Comput. Commun. Rev., Vol. 37, No. 4, pp. 181–192, (2007).

[10] D. Han, M. Lee, K .Cho, T. Kwon and Y. Choi, "Publisher mobility support in content centric networks," Proc. of The International Conference on Information Networking 2014 (ICOIN2014), pp. 214-219 (2014).

(Received: October 30, 2021)
(Accepted: September 5, 2022)

Taichi Iwamoto received his Bachelor and Master degrees from Prefectural University of Hiroshima, 2020 and 2021, respectively. He has worked in the field of Content Centric Networking architecture.

Tetsuya Shigeyasu received his B.S. and M.S. degrees from Yamaguchi University, 2000, and 2002, and his Ph.D. degree from Osaka University in 2010, respectively. In 2022, he joined Hiroshima International University as a research associate. In 2011, he moved to Prefectural University of Hiroshima. He is currently a professor of the Department of Management Information Systems. He is a dean of Faculty of Management and Information Systems, and a dean of Faculty of Regional Development. His current research interests include computer networks, mobile networking, He is a member of IEEE, ACM, IEICE, and a senior member of IPSJ.

Regular Paper

A Proposal on New Control Mechanisms Based on ICN for Low Latency IoT Services

Atsuko Yokotani[*], Hiroshi Mineno[*], Satoshi Ohzahata[**] and Tetsuya Yokotani[***]

[*]Graduate School of Science and Technology, Shizuoka University, Japan

{yokotani.atsuko20@, mineno@inf.}shizuoka.ac.jp

[**]Graduate School of Informatics and Engineering, University of Electro-Communications, Japan
ohzahata@is.uec.ac.jp

[***]College of Engineering, Kanazawa Institute of Technology, Japan
yokotani@neptune.kanazawa-it.ac.jp

Abstract – Information Centric Network (ICN) is a promising candidate to mitigate protocol overheads on the Internet to transfer information. Currently, the Internet invokes Internet protocol (IP) address base routing and translation between the IP address and the indicator by the domain name system (DNS) to obtain information. In contrast, ICN obtains information directly and provides the networked cache function to reduce duplicate information transfer. In particular, these features provide advantages in Internet of Things (IoT) communication, including low latency services. Prioritized information should be transferred with low latency to users and should be shared with multiple users. For this purpose, it is proposed that networked cache provides dedicated space to store prioritized information. This paper describes detailed mechanisms on bandwidth reservation and networked cache functions and performance evaluation by network traffic emulation. This paper proposes an architecture referred to as "C-NAT" and mechanisms of ICN with traffic control functions (e.g., bandwidth reservation and cache control) and their performance evaluation. C-NAT is an abbreviation of "Content-centric network (CCN) with Network initiative And Traffic control" with some modifications of CCN, which is a typical mechanism in ICN technologies.

Keywords: IoT, ICN, Traffic control, Cache control, Low latency service

1 INTRODUCTION

The Internet of Things (IoT) is a worldwide topic of interest. As various services utilizing IoT are deployed, the communication network plays an important role. Most IoT services expect wide-area network services, including Internet services [1]. However, in the mature stage of IoT services, if these services are deployed over the Internet as it currently exists, some serious problems will be highlighted, e.g., large overheads of the legacy protocols, processing resources of their overhead in communication equipment, and processing power of Internet protocol (IP) address translation by the domain name system (DNS).

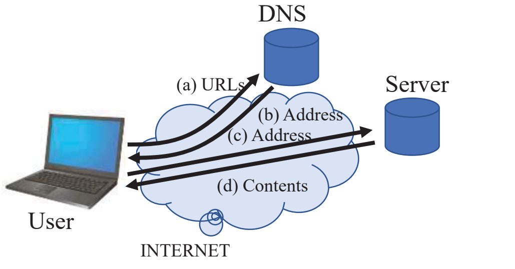
To mitigate these problems, Information-Centric Network (ICN) technologies have been discussed to facilitate IoT services. ICN technologies invoke independent communication of IP. They also provide networked cache to reduce duplicate traffic transfer.

This paper describes the possibilities of ICN technologies for IoT services and proposes architecture and operations of ICN base networks for various IoT services, referred to as "C-NAT" which is an abbreviation of "CCN with Network initiative And Traffic control." CCN is an abbreviation of "Content-Centric Network" and is summarized in the next section. Most significantly, this paper proposes operations with traffic control mechanisms with prioritized traffic flows on ICN base networks for low latency IoT services.

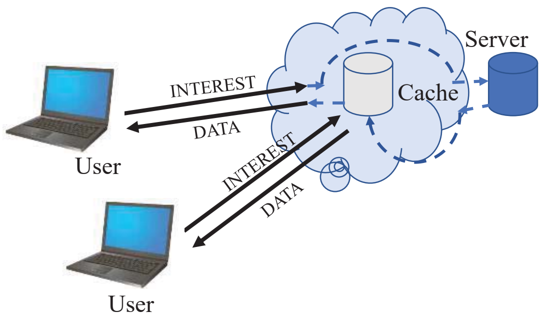
2 SURVEY ON ICN TECHNOLOGIES

ICN technologies include various mechanisms [2]. One typical mechanism is CCN proposed by [3]. CCN can provide simplified communication sequences to obtain information from servers. This paper focuses on CCN in ICN technologies.

Figure 1 compares sequences in the Internet and in CCN in the case of information transfer from a server to users.



Communication sequences in the Internet

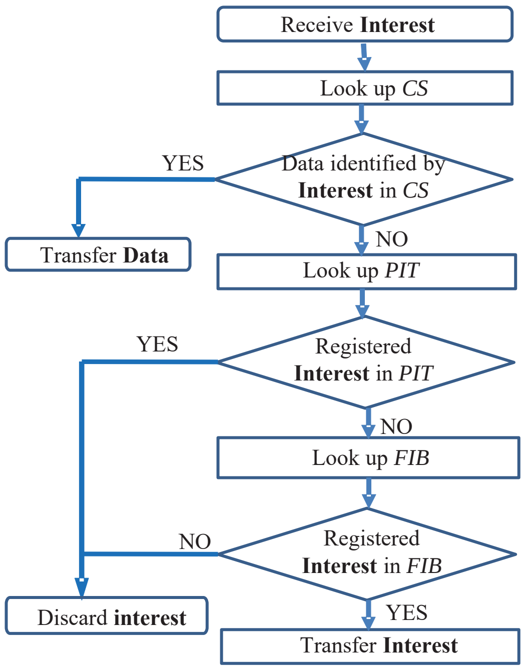


Communication sequences in CCN

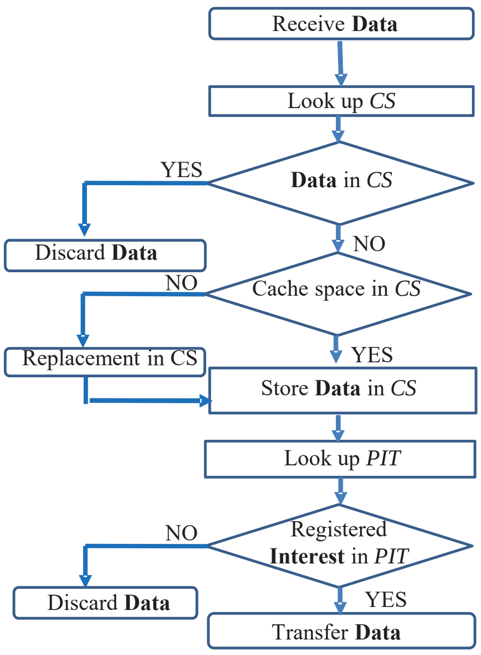
Figure 1 Comparison of communication sequences

With the Internet, before a user accesses a server to obtain particular content, the destination IP address must be derived from translation of a uniform resource locator (URL) by the DNS; see (a) and (b) in Fig. 1. Then, this user accesses the server using the derived IP address; see (c) in Fig. 1. As a result, this user obtains the requested content; see (d) in Fig. 1. In this case, because the user is connected to the server using these sequences, each user must invoke the same sequence whenever they access the server.

By implementing CCN, a user accesses the server using content names directly, without translation between an IP address and URL indicating the location of the content. Moreover, transferred content can be temporarily stored at some intervening, or interworking, points in networks. When another user accesses the server to obtain that content, the interworking point provides the requested content from its cache instead of the server. Therefore, duplicate transfer of contents by the server and the limited processing power of content transfer in the server are mitigated. In CCN, a request for content and the response, including the content, are referred to as **Interest** and **Data** messages, respectively. In this paper, these terms will reflect the same meaning. The detailed operations of CCN are described in Fig. 2. In CCN, there are three components to an Interworking point (IWP) (i.e., Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB)), which handle **Interest** and **Data**. IWPs are connecting points between CCN links and can be positioned as routers in the current Internet. Figure 2 (a) and (b) show processing sequences for **Interest** and **Data**, respectively.



(a) Processing sequences of **Interest**



(b) Processing sequences of **Data**

Figure 2 Processing sequences in CCN

3 COMMUNICATION SEQUENCES IN IOT SERVICES

To deploy IoT services, communication sequences are classified into three types, as shown in Fig. 3 [4]. Generally, an end device consists of various sensors, actuators and a communication device to connect a broadband network, e.g., high speed LAN, fiber to the X (FTTX), or radio access networks (RAN). Components in an end device are connected by wireless networks, e.g., LoRa and W-SUN, as a proximity network. The proximity network is relatively small and is configured for a dedicated purpose, so it can be optimized for specific services.

Of these, Type 1 seems to be in the majority because most IoT services require information from a large number of end devices, including sensors as end points of communication. In these sequences, some interworking points relay information of IoT services.

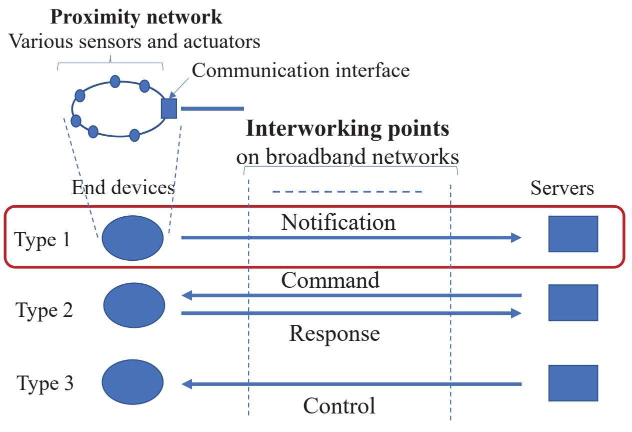


Figure 3 Types for IoT services in communication sequences

The studies on IoT services based on ICN technologies, especially CCN, have been published in previous research, e.g., [5] and [6]. Moreover, these studies have been discussed by the ICN Research Group of the Internet Research Task Force (IRTF) [7] as one of the next standardized subjects.

In common agreement in these studies, when ICN technologies are applied to IoT services, these technologies provide simpler communication for IoT services than conventional Internet technologies. For example, in IoT services, the huge number of end devices create tiny information blocks and transfer these blocks across networks, e.g., Type 1 in Fig. 3. In this situation, large protocol headers, i.e., the hypertext transfer protocol (HTTP), and three-way handshake procedures in transmission control protocol and IP (TCP/IP) cause an increase in traffic volume. Moreover, processing in DNS generates a heavy load for communication equipment.

ICN technologies are designed to mitigate these problems in deployment of IoT service. However, ICN may cause a security issue. Especially in the case of Type 1 of Fig. 3, when suspicious devices are connected to networks, distributed denial of service (DDoS) attacks may be initiated. This problem has been indicated in [8]. In that paper, authors proposed that an interworking point in networks could provide a screen of transfer traffic prior to endpoints as one of the regulation mechanisms on incoming IoT traffic, as shown in Fig. 3.

Generally, in the case of Type 1, end devices transfer information periodically to networks. In particular, real-time services in IoT will require periodic transfer sequences [9]. Requirements of these services are surveyed in [9] as described in Table 1. These services are typical examples. In addition to these, services categorized as Ultra-Reliable and Low Latency Communications (URLLC), a type of services in the 5G mobile system, have the same characteristics [10].

Generally, these services invoke memory-to-memory communications [11]. Each server includes the Shard memory to receive information from end points as shown in Fig. 4. The update of each field is cyclically invoked according to service requirements. The update cycle includes the transfer and processing delay. For instance, when this cycle is small, the transfer delay will be small. Therefore, information is transferred using multiple priorities in the network. This approach has been applied to low-latency communication systems in the industrial field, e.g., [12] and [13].

Table 1 Summary of QoS requirements of IoT services with low latency

	Latency (ms)	Packet loss ratio	Cycle (ms)	Size (B)	Device density
Factory	0.25～10	10^{-9}	0.5～50	10～500	0.33～3/m^3
Industrial plant	50～100	10^{-4}～10^{-3}	100～5000	40～100	10000/Plant
Smart grid	3～20	10^{-6}	10～100	80～1000	10～2000/km^2
Transportation (Safety drive)	10～100	10^{-5}～10^{-3}	100～1000	～1000	500～3000/km^2

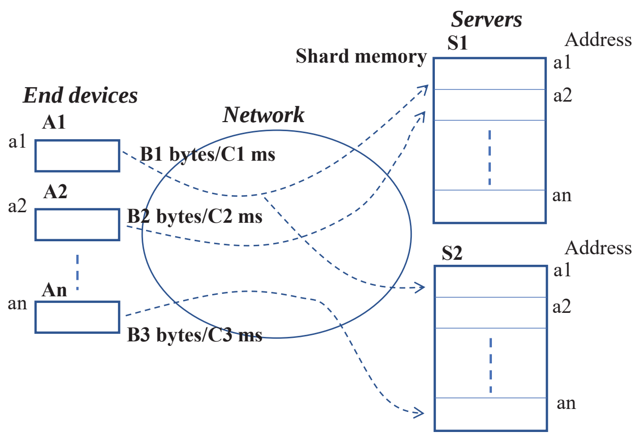


Figure 4 Examples of information transfer between end devices and servers

In Fig. 4, End devices A1, A2, … and An cyclically transfer information to Servers S1 and S2 across a network. Information is saved in the Shard memory in each Servers. In this case, the area in the Shard memory and the update cycle are allocated according to the service requirements. In this figure, B2 byes of information are shared from End device A2 to Sever S1 and updated every C2 ms. The network must comply with these conditions using traffic control functions with multiple priority levels.

4 PROPOSED MECHANISMS OF ICN WITH TRAFFIC CONTROL

In Section 3, the possibilities and new issues of IoT services based on ICN technologies were described. However, ICN technologies for IoT services do not provide traffic control functions. Generally, IoT services are overlaid across ICN base networks. In this situation, some traffic functionalities should be provided to improve the quality of service (QoS). In this section, the authors propose an architecture for new data transfer based on CCN with traffic control functions, referred to as “C-NAT”, and detailed mechanisms according to this architecture. In this paper, mechanisms with traffic control functions focusing on reservation of bandwidth and priority control in cache are proposed.

4.1 Architecture of C-NAT

In C-NAT, triggers for information transfer are provided by the IWP accommodating end devices, although end devices initiate information transfer by **Interest** in CCN. Because low latency IoT services invoke periodic information transfer as described in the previous section, the IWP can provide these triggers. The conceptual operations are shown in Fig. 5. In this figure, if information transfer is performed in less than half the cycle period identified in Table 1, the cycle time in Table 1 is guaranteed.

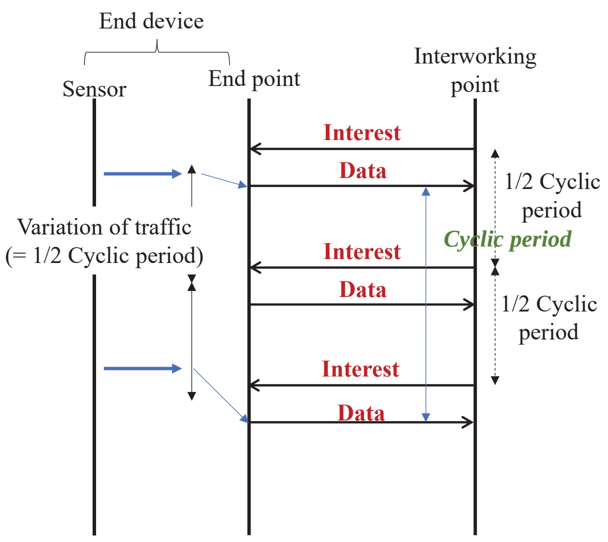


Figure 5 Conceptual operations in C-NAT

These operations are useful for the Type 1 communication sequence in Fig. 3. They also protect the network from DDoS attacks. However, information relay among multiple IWPs and traffic control on relayed routes should be specified. These points are specified in the subsequent sub-sections.

4.2 Information Relay Mechanisms for IoT Services

Information relay mechanisms based on CCN are shown in Fig. 6.

In Fig. 6, end devices are deployed according to offered services, e.g., monitoring of industry plants. At first, Server #1 submits an **Interest** to obtain information through some Interworking points (IWPs). Then, end devices reply to Server #1 with **Data** containing target information. After that, IWP #1 transfers that **Interest** periodically according to provisioning timing, e.g., required cycles. As the PIT in each IWP is set after the first **Interest**, **Data** is transferred to each IWP and can be stored in cache (Content Store) for every periodic **Interest**.

If other servers, e.g., Server #2, intend to obtain information regarding the **Interest**, IWP #3 updates the PIT to indicate requested information by a new server and then provides the **Data** stored in cache.

The processing sequences of **Interest** at IWP, which accommodates End devices, i.e., IWP #1 shown in Fig. 5, should be modified as in Fig. 7(a). The processing sequences of **Data** at the IWP, which relays **Data**, i.e., IWP #2 shown in Fig. 7(b), should be added to the original sequence. Other sequences are compiled in the original sequence shown in Fig. 2.

Sequences shown in these figures are clarified as follows. In Fig. 7(a), the first access to obtain information is the same as the original sequence. In short, the server generates **Interest** to end device. At this time, PIT was already set in IWP #1. In the second access and after, **Interest** to end devices is generated periodically according to provisioning timing, which is preset based on the required cycle in each service. In Fig. 7(b), IWP #2 monitors receiving **Data** periodically. If **Data** is not received, PIT is reset in IWP #2 automatically.

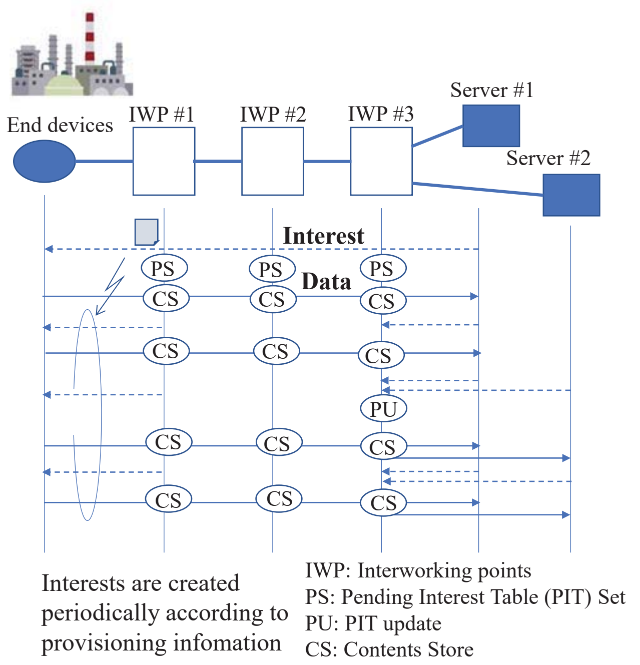
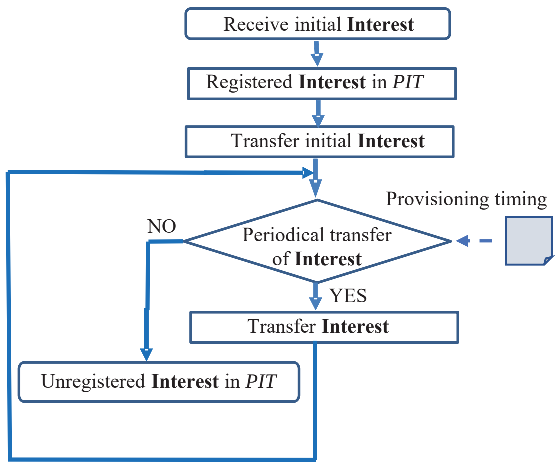
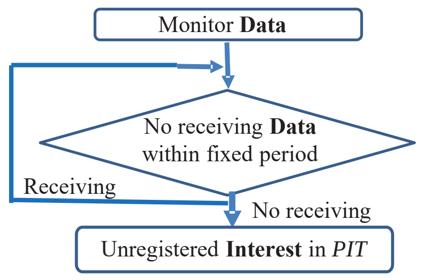


Figure 6 Operations in basic transfer mechanisms


Receive initial Interest
Registered Interest in PIT
Transfer initial Interest
Provisioning timing
NO
Periodical transfer of Interest
YES
Transfer Interest
Unregistered Interest in PIT

(a) Processing sequences of **Interest** in IWP #1


Monitor Data
No receiving Data within fixed period
Receiving
No receiving
Unregistered Interest in PIT

(b) Additional processing sequences of **Data** in IWP #2

Figure 7 Detailed operations in proposed mechanisms

In these mechanisms, triggers to transfer information are provided by the IWPs. Therefore, DDoS attacks initiated from end devices cannot be successful. Moreover, these mechanisms do not increase traffic volume by taking advantage of ICN technologies.

4.3 Mechanisms with Traffic Control

In Section 4.1, basic transfer mechanisms were proposed. However, when services are aggregated on networks, traffic control functions should be provided to prioritize IoT services requiring low latency. For this purpose, bandwidth reservation and dedicated space in the cache of IWPs are proposed.

A summary of these proposed mechanisms is shown in Fig. 8. Multiple QoS requirements are specified in this system as described in Section 3. Transferred information can be classified into multiple priority levels according to the delay and/or loss requirements. It is important to note that priority control is operational issues and does not majorly impact cost of these IoT services.

In this section, mechanisms on two-priority level are described, e.g., prioritized and non-prioritized information.

Before information is transferred, priority levels should be provisioned. Then, priority levels can be recognized by attributes in **Interest** and **Data**. Each Interest and Data is transferred according to their indicated priorities across networks.

(1) Bandwidth reservation and priority control

In Fig. 8, each link has guaranteed bandwidth for prioritized information, which consists of an **Interest** and **Data** pair. In this system, prioritized information can utilize the full capacity if non-prioritized information is not transferred. In each IWP, dedicated cache space is assigned for prioritized **Data**. With these traffic control functions, guaranteed bandwidth is reserved by the dual leaky bucket mechanism [14]. The bandwidth less than the commitment information rate (CIR) should be reserved for prioritized information. The bandwidth of more than the CIR and less than the sustainable information rate (SIR), e.g., link rate, can be reserved for prioritized information according to availability of non-prioritized information. Moreover, at each IWP, prioritized information is transferred prior to non-prioritized information according to head of the line (HoL) scheduling [15].

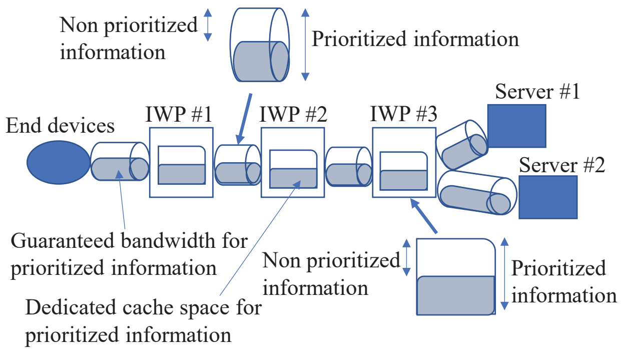

Non prioritized information
Prioritized information
End devices
IWP #1
IWP #2
IWP #3
Server #1
Server #2
Guaranteed bandwidth for prioritized information
Non prioritized information
Prioritized information
Dedicated cache space for prioritized information

Figure 8 Traffic control functions in networks

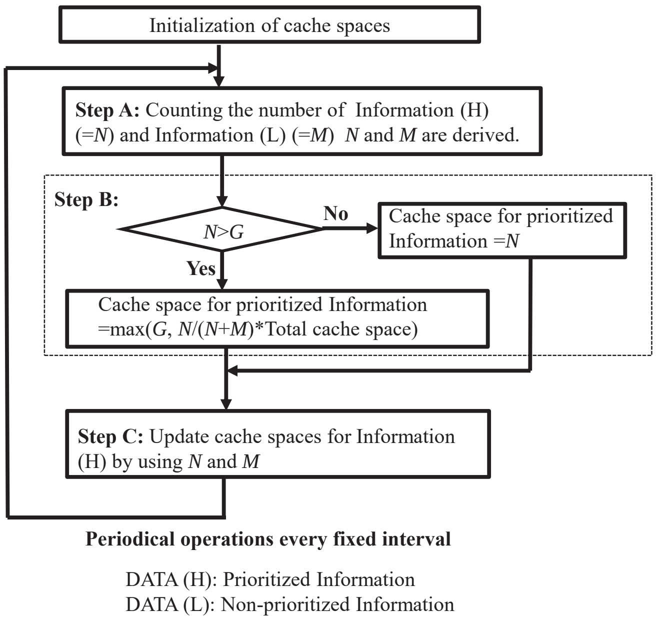

Initialization of cache spaces
Step A: Counting the number of Information (H) (=N) and Information (L) (=M) N and M are derived.
Step B:
N>G
No
Yes
Cache space for prioritized Information =N
Cache space for prioritized Information =max(G, N/(N+M)*Total cache space)
Step C: Update cache spaces for Information (H) by using N and M
Periodical operations every fixed interval
DATA (H): Prioritized Information
DATA (L): Non-prioritized Information

Figure 9 Cache control mechanisms for dedicated space

(2) Cache control

With cache control function, cache control mechanisms have been discussed in many articles. Control of dedicated cache space for prioritized information is provided as follows. Typical control mechanisms are the Least Recent Used (LRU), Least Frequency Used (LFU), and their combined mechanism [16].

As a legacy cache control, LRU is a policy under which stored information with the longest elapsed time after the last access is replaced when there is a shortage of cache space. LFU is a policy under which stored information with the smallest access frequency is removed from the cache space first. Prioritized information is always stored in cache spaces under this policy. Therefore, latency of prioritized information can be reduced [17]. However, these methods just relatively assign priority of information [18] and do not always guarantee communication quality of information transfer.

Authors have promoted new cache control mechanisms to improve these conventional mechanisms by utilizing real-time incoming information [19]. In this paper, these mechanisms are enhanced to adopt IoT communications. Essentially, full cache space can be assigned for prioritized information if non-prioritized information is not stored in cache as shown in Fig. 8. The proposed mechanism on control of dedicated cache space specifies the following operations.

In Fig. 9, each IWP counts incoming information within a fixed interval (Step A in Fig. 7). The numbers of information, prioritized and non-prioritized, are indicated by N and M, respectively. The volume of dedicated space for prioritized information is indicated by G. If N is larger than G, cache space for prioritized information is updated according to Equation (1). Otherwise, N is assigned to its space (Step B in Fig. 9).

$$\text{Cache space for prioritized information} = \max\left\{G, \frac{N}{N+M} \times Total\ cache\ space\right\} \quad (1)$$

Then, dedicated cache space is updated periodically (Step C in Fig. 9).

5 PERFORMANCE EVALUATION

In this section, the proposed mechanisms are evaluated using the CCN software platform referred to as CCNx [20].

5.1 Network Configuration

Two network configurations for performance evaluation are shown in Figs. 10 and 11. In Fig. 10, an IWP connects end devices and eight servers. The two servers processed prioritized information. Other servers processed non-prioritized information. Therefore, in the link between end devices and the IWP, bandwidth was reserved for prioritized information. This configuration corresponds to the small-scale deployment, e.g., IoT services across LAN. For example, various information generated from sensors in a site is monitored by several services specified by every service. This case referred to as the Local deployment case.

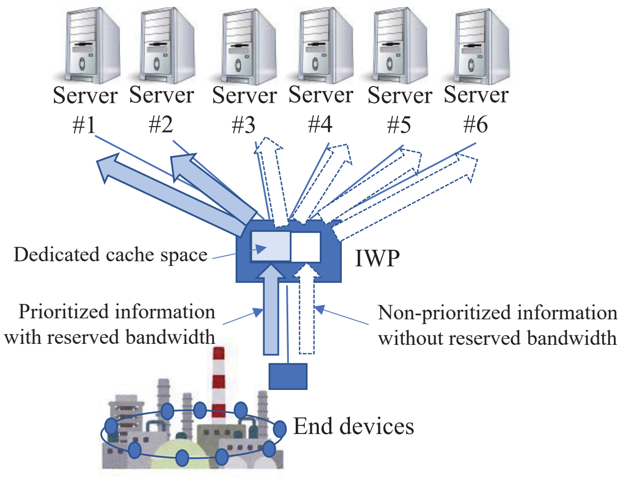

Server #1
Server #2
Server #3
Server #4
Server #5
Server #6
Dedicated cache space
IWP
Prioritized information with reserved bandwidth
Non-prioritized information without reserved bandwidth
End devices

Figure 10 Network configuration with one IWP

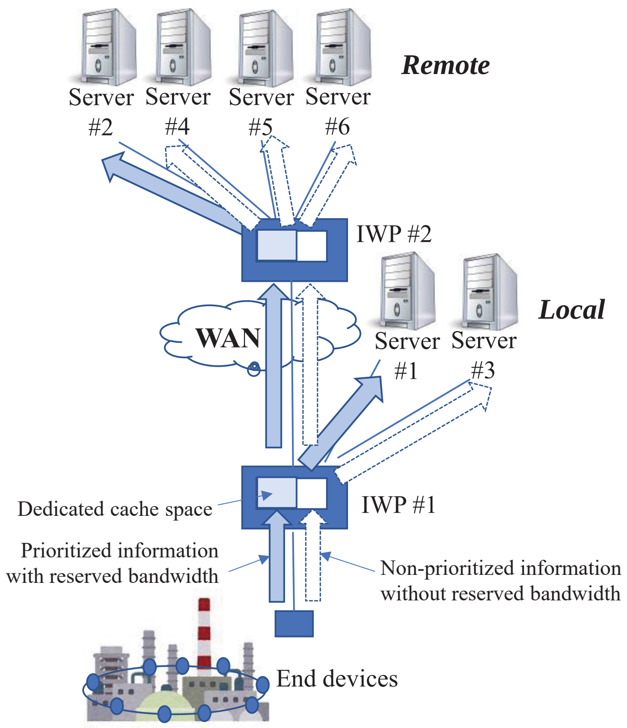


Figure 11 Network configuration with two IWPs

In Fig. 11, two IWPs were deployed in the system. IWP #1 connected two servers: one server for prioritized information and one for non-prioritized information. It also connected IWP #2. IWP #2 connected four servers. Three servers (Servers #4–#6) processed non-prioritized information. Server #2 processed prioritized information. In the links between end devices and IWP #1 and between IWP #1 and IWP #2, bandwidth was reserved for prioritized information. In IWPs #1 and #2, dedicated cache space was provided for prioritized information. This configuration corresponds to the Local-Remote deployment case, e.g., IoT services across WAN and LAN. WAN, e.g., the Internet or dedicated networks, is deployed between IWP #1 and IWP #2. IWP #1 is located at the nearby area of end devices. Servers accommodated in IWP #1 locally monitor information generated from end devices. On the other hand, Servers accommodated in IWP #2 remotely monitor such information.

In these configurations end devices, servers, and IWPs were emulated using Linux PCs with CCNx.

5.2 Numerical Examples

In this performance evaluation, bandwidth was denoted by the number of "unit" which is a virtual time on PC because CPU in PCs cannot emulate a real bit rate. Parameters from the performance evaluation were as follows:

· Link rate (=SIR)	10 Mb/unit
· Reserved bandwidth (=CIR)	3 Mb/unit
· Information block size	4 kB
· Generating rate of Interest for Prioritized information	50/unit

・ Generating rate of Interest for non-Prioritized information	50/unit
・ Dedicated space of cache	1500
・ Total space of cache	4500
・ Update cycle of dedicated space	1 unit
・ Basic policy of cache	LRU

In these parameters, "unit" is the unit of virtual time on the PC.

Some numerical results are shown in Figs. 12–15. In these graphs, the vertical axis denotes the latency in "unit" in the parameter list.

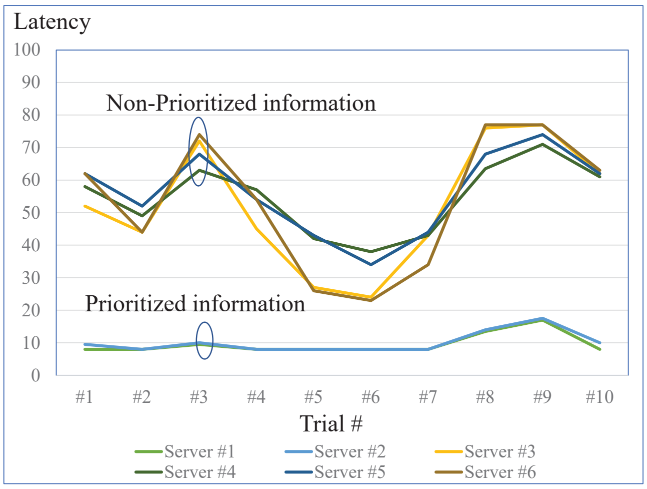


Figure 12 One IWP with cache control

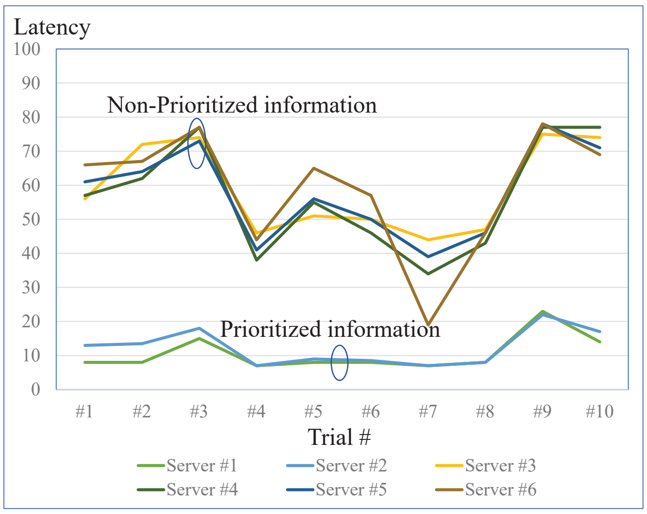


Figure 13 One IWP without cache control

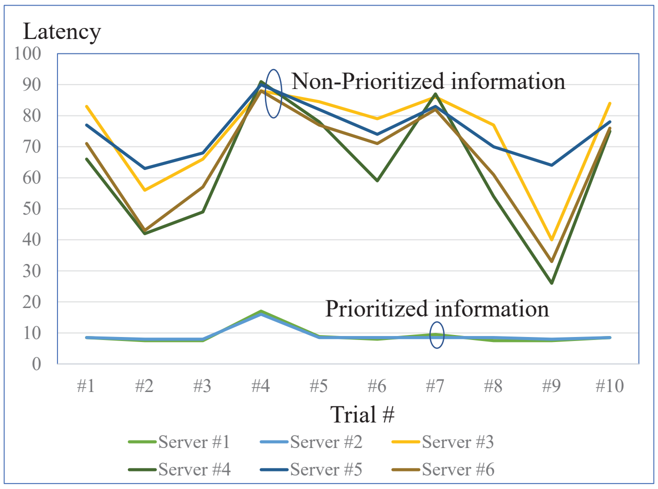


Figure 14 Two IWPs with cache control

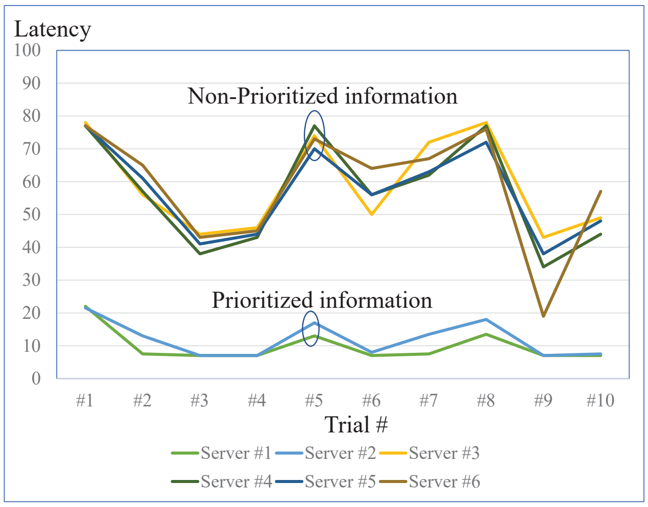


Figure 15 Two IWPs without cache control

These graphs show latency between **Interest** and **Data** in each server. The number of trials was 10 for each condition, as shown in the horizontal axes. The vertical axes show the latency by relative values. It is indicated as Latency denoted by "unit".

In these cases, bandwidth reservation for prioritized information was always activated. Cache control by assignment of dedicated space for prioritized information was activated in the cases displayed in Figs. 12 and 14. It was not activated in the experimental cases of Figs. 13 and 15.

The delay of prioritized information is relatively smaller than the delay of non-prioritized information regardless of cache conditions, i.e., with or without cache. However, behaviors of prioritized information with cache control seem to be more stable than the case of without cache.

Moreover, when the two IWPs case is compared with the one IWP case, delay of prioritized information with cache control is almost the same. It can be concluded that delay of prioritized information is relatively stable independent of the system scale if cache control is activated.

The results indicate that the latency of non-prioritized information is three or four times of prioritized information. In this evaluation, the latency was modelled using virtual time. However, because the propagation delay among transmission links can be negligible, these characteristics are applicable at the real time scale, e.g., millisecond and microsecond orders.

To confirm these characteristics, as characteristics of these examples are highlighted, the confidence interval with 95% in each case is shown in Figs. 16 and 17. In Fig. 17, differential of delay between servers accommodated in IWP #1 and in IWP #2 is small. Especially, in the cache control case, the characteristics of Server #1 and Server #2 are almost the same. Moreover, it is verified that prioritized information exhibits lower latency variation than that exhibited by non-prioritized information. When cache control is activated in each server, the variation in the latency of prioritized information is much smaller than that observed cache control activation. Restricting latency variation is one of important issues in low latency services, especially, industry fields. Hence, it can be concluded that the proposed mechanisms are reasonable.

Therefore, even in the Local-Remote deployment, the delay of prioritized information with cache control can be maintained in low latency except for propagation delay across networks.

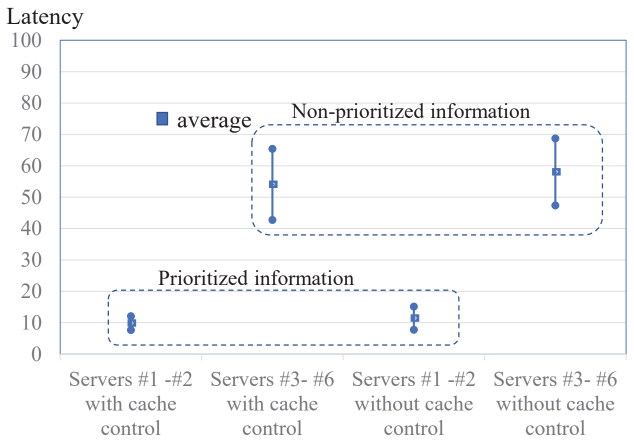


Figure 16 The 95% confidence interval in one IWP

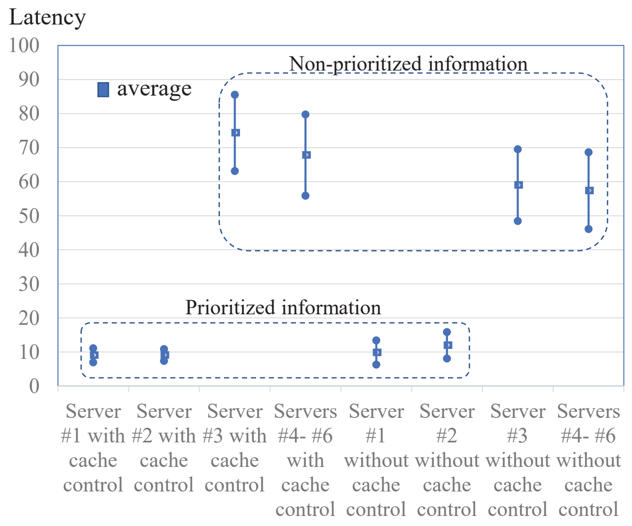


Figure 17 The 95% confidence interval in two IWPs

Performance evaluation can be summarized as follows.

Through these numerical examples, bandwidth reservation and cache control were confirmed effective for transfer of prioritized information including the Local-Remote deployment case. Especially, bandwidth reservation contributed to a reduction of latency. Moreover, cache control provided smaller variation of latency for prioritized information.

6 CONCLUSIONS

This paper proposed transfer mechanisms in IoT communication using ICN technologies. ICN technologies can solve communication problems of IoT presented by the current Internet and provide some advantages in IoT communication. However, security issues, e.g., DDoS attack, must be considered. This paper proposed communication sequences based on CCN to solve this issue.

Moreover, to comply with low latency requirements of IoT services, we proposed traffic control functions, including bandwidth reservation and cache control. Finally, this paper confirmed the advantages of the proposed mechanisms by the emulated system.

Currently, the IoT communication platform is an attractive topic for large-scale deployment. Standard development organizations (SDOs) are addressing this topic. For example, ISO/IEC JTC1/SC41 for IoT and digital twin has promoted this topic [21]. Proposals in this paper will contribute to this activity as detailed mechanisms in the IoT Data Exchange Platform (IoT-DEP), ISO/IEC 30161 series, which are summarized in [1].

REFERENCES

[1] T. Yokotani, and K. Kawai, “Concepts and requirements of IoT networks using IoT Data Exchange Platform toward International standards”, IEEE Conference on Standards for Communications and Networking (IEEE CSCN), ♯1570570960 (2019), DOI: 10.1109/CSCN.2019.8931337, IEEE Xplore.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher and B. Ohlman, “A survey of information-centric networking”, IEEE Communication Magazine, pp. 26-36, Vol. 50, Issue 7 (2012).

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs and R. Braynard, “Networking Named Content,” ACM CoNEXT 2009, pp.1-12 (2009).

[4] T. Yokotani, S. Yamamoto, S. Ohno, K. Sasabayashi and K. Ishibashi, “Survey and comparison of Interworking point routing mechanisms for IoT services in wide area ICNs”, International Conference on Emerging Technologies for Communications (ICETC 2020), D2-4, (2020).

[5] I. U. Din, H. Asmat and M. Guizani, “A review of information centric network-based internet of things: communication architectures, design issues, and research opportunities”, Multimedia Tools and Applications, vol. 78, pp. 30241–30256 (2019).

[6] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos, “Information-centric networking for the internet of things: challenges and opportunities”, IEEE Network, vol. 30, no. 2, pp. 92-100 (2016).

[7] R. Ravindran, Y. Zhang, L. A. Grieco, A. Lindgren, J. Burke, B. Ahlgren and A. Azgin, “Design Considerations for Applying ICN to IoT”, ICNRG, Technical report, draft-irtf-icnrg-icniot-03, https://datatracker.ietf.org/doc/draft-irtf-icnrg-icniot/ (2018).

[8] A. Yokotani, H. Mineno and T. Yokotani, “A proposal on the access control mechanism for real time IoT services using ICN technology”, IoT Enabling Sensing/Network/AI and Photonics Conference 2021 (IoT-SNAP 2021), IoT-SNAP-5-06 (2021).

[9] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste and M. Windisch, “Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture”, pp. 70-78, IEEE Communication Magazine, February (2017).

[10] G. A. Akpakwu, B. J. Silva and G. P. Hancke, “A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges”, pp. 3619 – 3647, Vol. 6, IEEE Access (2018).

[11] D. Buntinas, G. Mercier and W. Gropp, "Implementation and evaluation of shared-memory communication and synchronization operations in MPICH2 using the Nemesis communication subsystem", Journal of Parallel Computing, Vol. 33, No. 9, pp. 634-644 (2007).

[12] K. Oya, K. Tanaka, T. Sato, M. Yoshida and N. Todoroki, “Communication Method of Ring Network for Industrial System”, C-015, Forum on Information Technology 2018 (FIT 2018) (2018).

[13] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth and F. Golatowski, D. Timmermann and J. Schacht, “Survey on Real-Time Communication Via Ethernet in Industrial Automation Environment”, PF-000167, 19th IEEE International Conference on Emerging Technologies and Factory Automation (IEEE ETFA 2014) (2014)

[14] S. Zeng, N. Uzun and S. Papavassiliou, “A dual level leaky bucket traffic shaper architecture for DiffServ networks”, 2001 IEEE Workshop on High Performance Switching and Routing (HSPR 2001) (2001), DOI: 10.1109/HPSR.2001.923607, IEEE Xplore.

[15] T. Maertens , J. Walraevens, H. Bruneel, “A modified HOL priority scheduling discipline: Performance analysis”, European Journal of Operational Research, Vol. 108, pp. 1168-1185 (2007).

[16] H. Li, H. Nakazato, A. Detti, Nicola and B. Melazzi, "Popularity Proportional Cache Size Allocation Policy for Video Delivery on CCN", European Conference on Networks and Communications (EuCNC 2015), pp. 434-438 (2015).

[17] S. Jihoon, R. June-Koo Kevin and J. Sangsu, "Lightweight caching strategy for wireless content delivery networks", IEICE Communications Express, Vol. 3, No. 4, pp. 150-155 (2014).

[18] H. Qian, W. Muqing, H. Hailong, W. Ning and Z. Chaoyi, "In-Network Cache Management Based on Differentiated Service for Information-Centric Networking", IEICE Transactions on Communications, Vol.E97-B, No.12, pp. 2616-2626 (2014).

[19] A. Yokotani, S. Ohzahata, R. Yamamoto and T. Kato, “A Dynamic Cache Size Assignment Method with Bandwidth Reservation for CCN”, 2019 International Conference on Information Networking (ICOIN 2019), P2-15 (2019), DOI: 10.1109/ICOIN.2019.8718174, IEEE Xplore.

[20] "CCNx project”, http://www.ccnx.org.

[21] T. Yokotani, and K. Kawai, “Survey on standardization activities of IoT and proposal of the IoT data exchange platform”, International Conference on Emerging Technologies for Communications (ICETC 2020), IB3-3 (2020).

(Received: October 30, 2021)
(Accepted: September 5, 2022)

Atsuko Yokotani received BE, ME in Tokyo University of Technology and The University of Electro-Communication in 2017, 2019, respectively. Since then, she has joined Mitsubishi Electric Corporation. Currently, she engages development on control networks for self-defense system in Kamakura works. She has been also a doctoral course student in Graduate school of Science and Technology, Shizuoka University from 2020. She obtained an industry paper award in IWIN 2021. She is a member of IEICE.

Hiroshi Mineno received his B.E. and M.E. degrees from Shizuoka University, Japan in 1997 and 1999, respectively. In 2006, he received his Ph.D. degree in information science and electrical engineering from Kyushu University, Japan. Between 1999 and 2002, he was a researcher in the NTT Service Integration Laboratories. In 2002, he joined the Department of Computer Science of Shizuoka University as an Assistant Professor. He is currently a Professor. His research interests include Intelligent IoT systems as well as heterogeneous network convergence. He is a senior member of IEEE, IEICE and IPSJ, a member of ACM and the Informatics Society.

Satoshi Ohzahata received the B.S., M.E., and D.E. degrees from the University of Tsukuba, Ibaraki, Japan, in 1998, 2000, and 2003, respectively. From 2003 to 2007 and from 2007 to 2009, he was a Research Associate in the Department of Computer, Information and Communication Sciences and an Assistant Professor, respectively, at Tokyo University Agriculture and Technology. Since 2009, he has been an Associate Professor at the University of Electro-Communications, Tokyo, Japan. His interests are mobile ad hoc networks, the Internet architecture in mobile environments, and Internet traffic measurement. Dr. Ohzahata is a member of IEEE, ACM, IEICE and IPSJ.

Tetsuya Yokotani received B.S., M.S, and Ph.D. degrees on information science from the Tokyo University of Science in 1985, 1987, and 1997, respectively. He joined the Mitsubishi Electric Corporation in 1987. Since then, he has researched high-speed data communication, optical access systems, home network and performance evaluation technologies of networks mainly in the Information Technology R&D Center. In 2015, he moved to the Kanazawa Institute of Technology as a professor of College of engineering. Since then, he has engaged research and education on networks for various IoT services and has proposed standardization in these related areas. Currently, he is a chair in the IEEE ComSoc the CQR technical committee and a chair-elect in the Hokuriku branch of IEICE. He has also participated in the standardization activities on ITU-T SG15, SG20 and ISO/IEC JTC1 and obtained several awards on these activities. He is a fellow member of IEICE. He is also a member of IEEE ComSoc and IPSJ.

Submission Guidance

About IJIS

International Journal of Informatics Society (ISSN 1883-4566) is published in one volume of three issues a year. One should be a member of Informatics Society for the submission of the article at least. A submission article is reviewed at least two reviewer. The online version of the journal is available at the following site: http://www.infsoc.org.

Aims and Scope of Informatics Society

The evolution of informatics heralds a new information society. It provides more convenience to our life. Informatics and technologies have been integrated by various fields. For example, mathematics, linguistics, logics, engineering, and new fields will join it. Especially, we are continuing to maintain an awareness of informatics and communication convergence. Informatics Society is the organization that tries to develop informatics and technologies with this convergence. International Journal of Informatics Society (IJIS) is the journal of Informatics Society.

Areas of interest include, but are not limited to:

Internet of Things (IoT)
Smart Cities, Communities, and Spaces
Big Data, Artificial Intelligence, and Data Science
Network Systems and Protocols
Computer Supported Cooperative Work and Groupware
Security and Privacy in Information Systems
Intelligent Transportation System
Distributed Computing
Multi-media communication
Information systems
Mobile computing
Ubiquitous computing

Instruction to Authors

For detailed instructions please refer to the Authors Corner on our Web site, http://www.infsoc.org/.

Submission of manuscripts: There is no limitation of page count as full papers, each of which will be subject to a full review process. An electronic, PDF-based submission of papers is mandatory. Download and use the LaTeX2e or Microsoft Word sample IJIS formats.

http://www.infsoc.org/IJIS-Format.pdf

LaTeX2e

LaTeX2e files (ZIP) http://www.infsoc.org/template_IJIS.zip

Microsoft Word™

Sample document http://www.infsoc.org/sample_IJIS.doc

Please send the PDF file of your paper to secretariat@infsoc.org with the following information:

Title, Author: Name (Affiliation), Name (Affiliation), Corresponding Author. Address, Tel, Fax, E-mail:

Copyright

For all copying, reprint, or republication permission, write to: Copyrights and Permissions Department, Informatics Society, secretariat@infsoc.org.

Publisher

Address: Informatics Laboratory, 3-41 Tsujimachi, Kitaku, Nagoya 462-0032, Japan

E-mail: secretariat@infsoc.org

CONTENTS