International Journal of Informatics Society

03/22 Vol. 13 No. 3 ISSN 1883-4566

Informatics Society

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

Hiroshi Yoshiura
Guest Editor of Thirty-ninth Issue of International Journal of Informatics Society

We are delighted to have the Thirty-ninth issue of the International Journal of Informatics Society (IJIS) published. This issue includes selected papers from the Fourteenth International Workshop on Informatics (IWIN2020), which was held online, Sept. 10-11, 2020. The workshop was the fourteenth 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 24 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 IWIN2020 was reviewed in terms of technical content, scientific rigor, novelty, originality and quality of presentation by at least two reviewers. Through those reviews 17 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 were reviewed from the different points of view. This volume includes three 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.

Hiroshi Yoshiura received his B.S. and D.Sc. from the University of Tokyo, Japan in 1981 and 1997. He has been a professor at the Department of Engineering, Kyoto Tachibana University since April 2021. He is also a professor emeritus of the University of Electro-Communications where he had been a professor until March 2021. Before joining UEC, he had been at Systems Development Laboratory, Hitachi, Ltd. He has been engaged in research and development of information security science and technology. He received Annual Best Paper Awards from Information Processing Society of Japan (IPSJ) in 2005 and 2011, and Japan Society of Security Management (JSSM) in 2011, 2016, and 2018, and also received Specially Selected Paper Awards of IPSJ Journal in 2018 and 2020. He received Best Paper Awards of 2006 IEEE IIH-MSP Conference, 2016 IFIP I3E Conference, and 2017 INFSOC IWIN Conference. He is a fellow of IPSJ and is a member of JSSM, The Institute of Electronics, Information and Communication Engineers, The Japanese Society for Artificial Intelligence, The Institute of Systems, Control and Information Engineers, and The Institute of Electrical and Electronics Engineers.

Regular Paper

Executable Counterexample for Java Model Checker

Chellet Marwan Bernard Hassan, Shinpei Ogata, and Kozo Okano

Shinshu University, Nagano, Japan
19w2074f@shinshu-u.ac.jp,{okano, ogata}@cs.shinshu-u.ac.jp

Abstract - Testing is a mandatory task in the development of software. Effectively, ensuring the reliability of software is a major point of work for a developer. Even after software has been tested, it is not uncommon to encounter bugs that can have consequences on the operation of a system. To improve reliability, developers have started using verification methods in addition to testing. The basic method of model checking verifies whether or not the software or a given fragment of program code satisfies properties that can lead to bugs if they are violated. Model checking has already proved to be an effective tool for verifying software, but there are still some inconveniences in using these techniques. When a software model checker finds a violated property, it will signal it by giving a counterexample as output. This usually involves tracing the path from the initial state to the state that reveals the violated property. The counterexamples are usually given in text format and are not always simple to understand as all of the steps of the process are given with a sequence of machine instructions. This paper aims to improve a Java model checker by translating counterexamples into executable Java code that is more understandable for a developer.

Keywords: Model checking, counterexample, Java, simulation, verification

1 INTRODUCTION

Software testing is one of the best ways to get good software reliability. It is practically impossible to use software without testing it beforehand but with all testing methods, it is not rare to miss some cases which can result in bugs during operation. Even a small bug can have real consequences so software safety becomes a priority. Full coverage of an application is hard to achieve even with advanced testing methods. Such methods are also limited because they use the system itself to find errors which makes it difficult to have an overview of the entire program.

To fill the gaps of testing methods, developers also use a verification method called “model checking.” [1][2] This method does not use the system itself, but rather an abstract model of it which is represented by states and transitions. It permits developers to automatically try different scenarios and cases and provides a different field of action than the testing method. Model checking typically works in addition to testing rather than in place of it. Even if software model checking [3] is already recognized as a good way to find errors in software, it is still an underused tool because of its complexity of usability.

One big disadvantage of model checking is the difficulty in treating counterexamples given for a violated property. Unlike the testing method in which such output is usually given in an understandable way by using code, model checking counterexamples are usually presented as sequences of transitions that lead to the error state from the initial state. For software model checking which uses program code as input instead of state machines, the counterexamples are usually represented in a sequence of low-level machine codes due to limitations of the current software model checking tools. For example, Java model checking tools usually take Java Bytecodes instead of Java program code which makes it difficult for people who are not experts in machine language to understand what is being expressed and can be a problem when attempting to localize the bug.

It has been a long time since researchers have tried to study ways to help people to understand the contents of a counterexample [4]. Now we have new technologies and new ways of using counterexamples emerging such as Visual-Studio plugins [5]. One of the most interesting methods reported was a way to make an executable counterexample [6] instead of a complicated list of states in machine language. Our research was especially inspired by the work of Rocha Herbert, Barreto Raimundo, Cordeiro Lucas, and Neto Arilo and their way of creating an executable counterexample for ANSI-C programs [7]. Our paper introduces a method for creating an executable counterexample for Java programs with the Java model checker JBMC [8]. This is done by extracting the necessary data from a counterexample and translating it into lines of Java code to be used in the final version of source code.

This paper is organized as follows: Section 2 introduces the basic concepts and principles of model checking, the model checker we used, and the part of model checking on which we focused. Section 3 explains our proposed method. Section 4 describes our experiments and results. Section 5 presents our observations during the creation of the proposed method. Section 6 describes our future work to improve this proposed method. Section 7 provides all of the existing works related to this proposed method and finally, Section 8 summarizes this paper.

2 MODEL CHECKING

In this section, we introduce the principle of model checking and how it works. After that, we explain our Java model checker choice. The last part focuses on the generation of counterexamples which is the central part of this paper.

2.1 Principle of Model Checking

Model checking has already proven itself as a useful verification method for software [9][10]. The goal of this method is to analyze a system translated into an abstract model to find violated properties. The model is represented as a transition system in the form of an oriented graph: a vertex represents a state of the system and each arc represents a transition, i.e., a possible evolution of the system from a given state to another state. Each state of the oriented graph is labeled by a set of atomic propositions true at this point of execution. The property to check is written by a temporal logic formula. These logical expressions are defined on a set of atomic propositions P or proposition variables. These atomic propositions are combined with a number of logical connectors, including the usual connectors: and, or, not, implication, as well as other operators which are called modalities.

This method works in three phases. First is the modeling phase which takes a real system and translates it into an abstract model. The system becomes a set of states and transitions. States give information about the program such as its variable values. Transitions are used to describe how the system works. The model checker also formalizes the properties to be verified. In the second phase, the model checker evaluates the possible paths of the abstract model to find a violated property. In this work, we are mainly interested in the final phase of analysis. There are three different possible outputs from the second phase. The first possibility is when a property is violated and the model checker traces the path to give a counterexample. The second type of output is when all properties are verified and not violated in which case the model checker reports that no bugs were found in the program. The third type occurs when the software is too complex and the abstract model is too large in which case the model checker runs out of memory in handle it. The model checking principle [1] and its three phases are depicted in Fig. 1.

The model checker we use in this work is a bounded one. The bounded model checking method [11] is a way to use model checking by traversing a finite-state machine with a fixed number of steps k and checking if a property is violated in this bound. The larger the value of k, the better the chances of finding a violation. The principal goal is to be faster and more efficient by giving a limit of the number of steps. This method is also associated with Boolean satisfiability problem (SAT) solvers [12] which, given a propositional logic formula, can determine whether or not there is an assignment of propositional variables that makes a formula true.

2.2 Software Model Checking

This paper is focused on software model checking which is slightly different from general model checking. Effectively the target of software model checking is program code that cannot be transformed into a finite-state model. Instead, code is transformed into transitions written in machine languages. The bounded part is used for program loops where the software model checker unrolls loops with a bound k. Figure 2 is a short explanation of how this works in a while loop with a bound of $k = 3$. The specification of a property in software model checking is also different. It is not represented by a temporal logic formula but by an assertion.

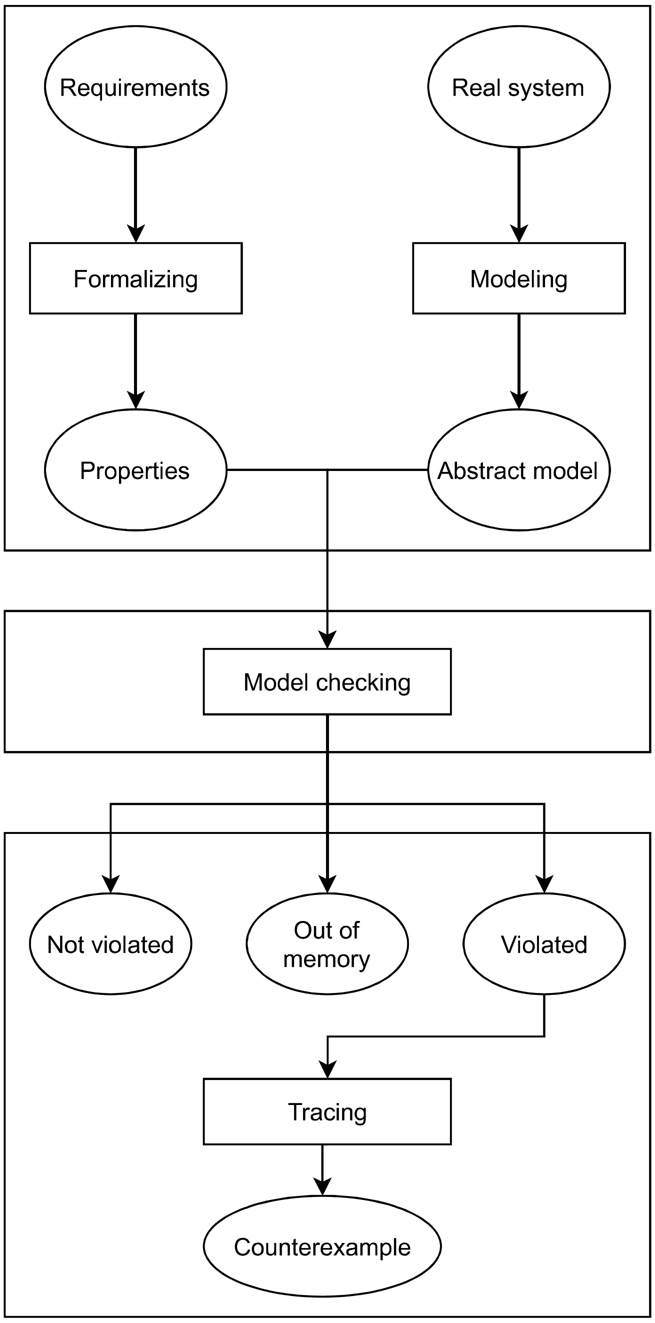


Figure 1: Principle of model checking

2.3 Model Checking with JBMC

JBMC is a bounded Java model checker developed in C++ and is based on the C model checker called CBMC [13]. It is one of the most efficient model checkers for Java programs and takes a Java Bytecode program as input. To be more efficient it also uses an abstract representation of the standard Java libraries called Java Operational Model. This representation was made to simplify the standard Java libraries and speed up the process of model checking. The core of the checker is managed by the CPROVER framework[14]. It is a

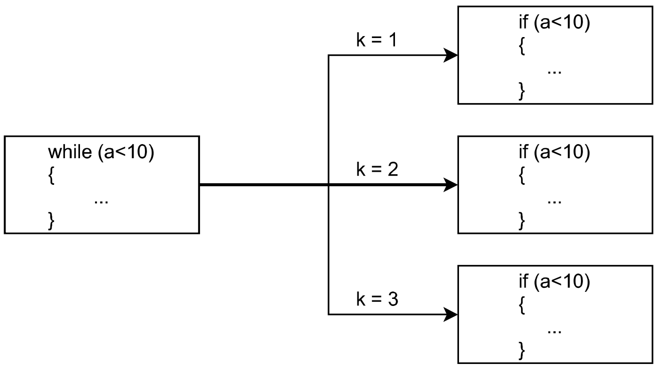


Figure 2: Example of an unrolled loop with a bound of $k = 3$

bounded model checker so the last input will be the number of bounds k that we want to do.

JBMC works in four steps. The first step is to parse the Java Bytecode into a parse tree which corresponds to a translation into a finite-state model. The second step translates the parse tree into a CPROVER control-flow graph representation which is called a GOTO program. The goal of this step is to simplify the Java Bytecode representation and make it easier to analyze with the CPROVER framework. In the third step the checker analyzes the model properties. The last step uses the SAT solver to determine whether or not a property is violated for a given bound k, and returns a counterexample if it is the case.

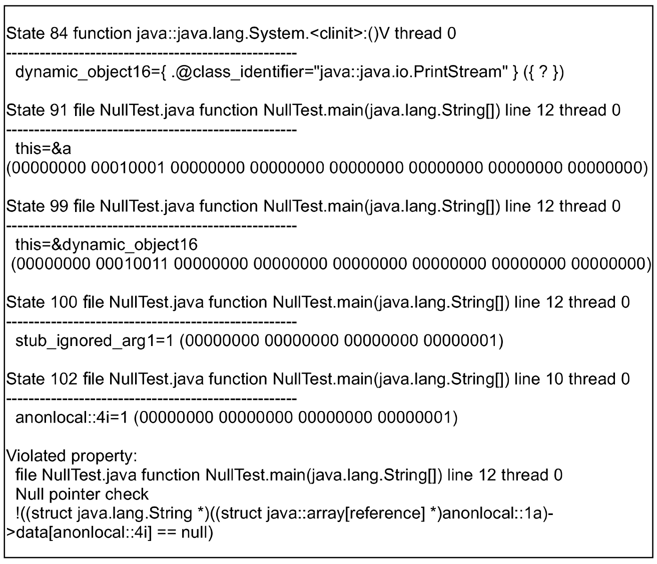
As mentioned above, JBMC uses the CPROVER framework to produce counterexamples. In addition, to verify assertions made by the user, it also covers such properties as Null Pointer Exception, Division by 0, Index out of bounds, etc. Figure 3 illustrates a simple example of a Null Pointer Exception error which can be found by JBMC.

This example is a simple program for printing the values of an array of strings. The variable "v" takes a random value between 1 and 2. If the value is 1, then only the first value of the array "tab" will be printed, and if the value is 2, then it will also print the second value which is null, and throw a Null Pointer Exception.

```
import java.util.Random;

public class NullTest
{
 public static void main(String [] args)
 {
  String[] tab = {"a",null};
  Random r = new Random();
  int v = r.nextInt((2-1)+1)+1;
  for (int i=0;i<v;i++)
  {
   System.out.println(tab[i].length());
  }
 }
}
```

Figure 3: Code with a Null Pointer Exception error



```
State 84 function java::java.lang.System.<clinit>:()V thread 0
----------------------------------------------------
  dynamic_object16={ .@class_identifier="java::java.io.PrintStream" } ({ ? })

State 91 file NullTest.java function NullTest.main(java.lang.String[]) line 12 thread 0
----------------------------------------------------
  this=&a
(00000000 00010001 00000000 00000000 00000000 00000000 00000000 00000000)

State 99 file NullTest.java function NullTest.main(java.lang.String[]) line 12 thread 0
----------------------------------------------------
  this=&dynamic_object16
 (00000000 00010011 00000000 00000000 00000000 00000000 00000000 00000000)

State 100 file NullTest.java function NullTest.main(java.lang.String[]) line 12 thread 0
----------------------------------------------------
  stub_ignored_arg1=1 (00000000 00000000 00000000 00000001)

State 102 file NullTest.java function NullTest.main(java.lang.String[]) line 10 thread 0
----------------------------------------------------
  anonlocal::4i=1 (00000000 00000000 00000000 00000001)

Violated property:
  file NullTest.java function NullTest.main(java.lang.String[]) line 12 thread 0
  Null pointer check
  !((struct java.lang.String *)((struct java::array[reference] *)anonlocal::1a)-
>data[anonlocal::4i] == null)
```

Figure 4: Trace of counterexample from JBMC

2.4 Counterexample

When the model checker finds a violated property in a program, a counterexample is given. This output helps developers localize the bug by giving the behavior which leads to the bug using such information as variable values. As mentioned before, JBMC uses CPROVER which produces counterexamples. The trace example in Fig. 4 is the output given by the model checker after the verification of the code shown in Fig. 3.

From the output, we can easily see which property is violated, but it is difficult to understand how the model checker found the violation and which path we should take to get the same behavior. Every state in the counterexample represents one instruction such as a change of variable value, but this instruction is given by the CPROVER framework which was initially made for C and C++ programs. The problem is that it is difficult to understand these instructions and see what the Java equivalence is.

3 PROPOSED METHOD

This section describes the method proposed in this work for translating counterexamples given by JBMC into Java code. Figure 5 shows the process of the whole system. Step 1 is simply a transformation of Java code into Java Bytecode before sending it to the model checker. Step 2 is the running of the model checker, and if a property is violated we go to Step 3. Step 3 is the treatment of the counterexample and this paper focuses on this work. The last step is the report of the output as a Java code program after the translation of the counterexample.

3.1 Method Objective

The main objective of our proposed method is to support developers who want to use model checking for verifying their software. Since it is an experimental method, the principal goal is to be sure that it is possible to get easily un-

Table 1: Experiment results

	Execution Time	New Assertion	New Instructions	Fusion
Null Pointer	1.03s	OK	OK	OK
Index Out of Bound	0.50s	OK	OK	OK
Division By 0	0.95s	OK	OK	OK
User Assertion	0.88s	NONE	OK	OK

derstandable code from a counterexample produced by the JBMC model checker. To achieve this goal, we take data from a counterexample and put it into the source code in a way that will be clear to the developer who wrote the source code. The main objective will be to use simple code and to analyze it with simple classes. It also has to be adaptive so that we can extend it for more complex software in the future.

3.2 Method Contribution

The contributions of our proposed method are to (1) provide a concrete method to build program code that explicitly shows the source of bugs; and (2) show the effectiveness of the proposed method through examples.

3.3 Step 1: Analysis of Counterexamples

The method proposed in this work is based on a transformation algorithm. The first part adds all new instructions resulting from the counterexample such as variable changes into the source code. The most important task is to understand what property was violated and where it occurred. The last lines of the counterexample give us the data related to the violated property. To help the user find the bug, we create a new line of Java code from this data with an assertion which will fail. Next we find and translate every state of the counterexample into lines of code. More than one state can have the same corresponding line number so in this case, we check if one of our newly created lines corresponds to this line number and update it instead of creating a new line.

3.4 Step 2: Transformation of Bytecode Source Files into Java

The second step of our proposed method is to transform the source file which is in Bytecode into Java code so that it can be used later for debugging. To do this, we use a Java library that transforms bytecode into Java. There are some good libraries such as Procyon, but for this work we chose CFR which does not optimize the code by techniques such as deleting dead code or using variable propagation. This will permit the user to find the code in its original form before the compilation in Bytecode and make it easier to localize the error.

3.5 Step 3: Addition of New Java Code Lines into the New Source File

The last step of the procedure is the fusion of the two previous ones. The goal is to produce Java code that reproduces the behavior of the software at the point where the model checker found the violation of a property. To accomplish this we read the new source file line by line and check to see if there is a new instruction that needs to be added from the counterexample. If we have a new instruction, we delete the old line and replace it with the corresponding one created earlier in Step 1. The variable names from the source file and the counterexample are different, so we have to be sure to preserve the original recognizable variable names and only change their values according to the information in the counterexample rather than simply replacing the whole line. When we reach the line of the violated property, we insert a new line above it with the new assertion created. Figure 6 represents the final result of applying the proposed method to the example in Fig. 3.

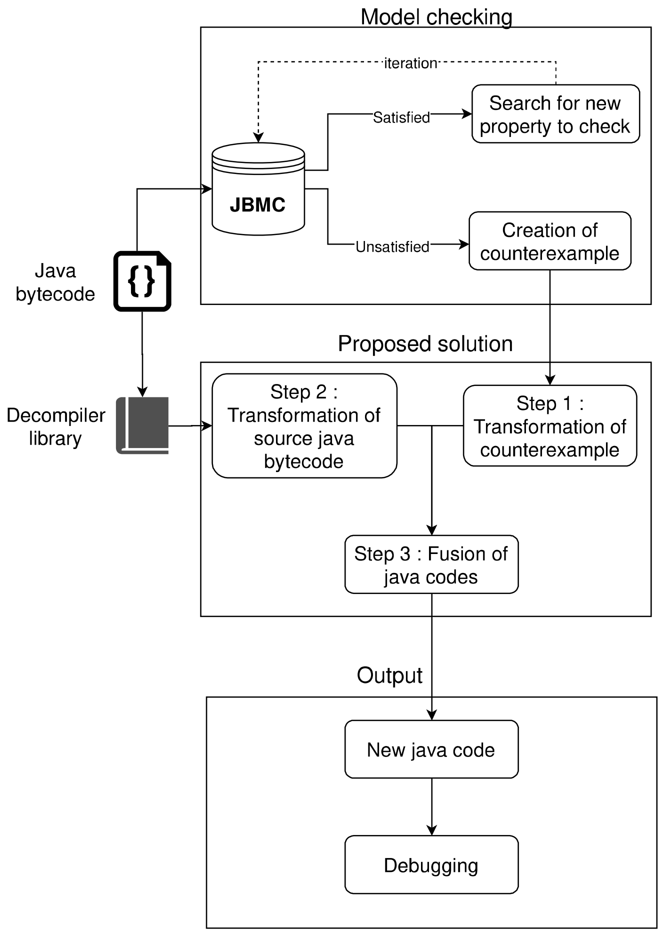

Model checking
iteration
Satisfied
Search for new property to check
JBMC
Unsatisfied
Creation of counterexample
Java bytecode
Proposed solution
Decompiler library
Step 2 : Transformation of source java bytecode
Step 1 : Transformation of counterexample
Step 3 : Fusion of java codes
Output
New java code
Debugging

Figure 5: Proposed method process

```
import java.util.Random;

public class NullTest
{
  public static void main(String [] args
     )
  {
    String[] tab = {"a",null};
    Random r = new Random();
    int v = 2;
    for (int i=0;i<v;i++)
    {
      assert (tab[1] != null);
      System.out.println(tab[i].length()
          );
    }
  }
}
```

Figure 6: New code generated by the proposed method

4 EXPERIMENTS AND RESULTS

To check the efficiency of the proposed method, we tested it with a number of simple programs with different types of errors for which JBMC can find a counterexample. We made one program per error for Null Pointer Exception, Index Out Of Bounds Exception, Division By Zero Exception, and a user assertion. For this experiment we chose a bound of $k = 5$. Table 1 shows the experiment results for each program step by step. Figures 7 to 12 show all source codes and generated codes from the experiment data.

All of the cases were successful and this experiment shows that it is possible to translate a counterexample into Java code. New assertions and the new values of variables are useful in deriving from where an error originates. During this experiment, we realized that it is sometimes difficult to see where the code has changed so it would be a good idea to add comments before new lines of code generated in the output.

```
import java.util.Random;

public class IndexTest
{
  public static void main(String [] args
     )
  {
    int[] test = {1};
    Random r = new Random();
    int v = r.nextInt((2-1)+1)+1;
    for (int i=0;i<v;i++)
    {
      System.out.println(test[i]);
    }
  }
}
```

Figure 7: Source code of an Index Out of Bounds Exception

```
import java.util.Random;

public class IndexTest
{
  public static void main(String [] args
     )
  {
    int[] test = {1};
    Random r = new Random();
    int v = 2;
    for (int i=0;i<v;i++)
    {
      assert (test.length >1)
      System.out.println(test[i]);
    }
  }
}
```

Figure 8: New generated code of an Index Out of Bounds Exception

```
import java.util.Random;

public class DivisionTest
{
  public static void main(String [] args
     )
  {
    int[] test = {1,0};
    Random r = new Random();
    int v = r.nextInt((2-1)+1)+1;
    for (int i=0;i<v;i++)
    {
      System.out.println(10 / test[i]);
    }
  }
}
```

Figure 9: Source code of a Division by 0 Exception

This experiment was done with very simple programs, but it is a good start for the understanding of counterexamples. One Java code is cut into multiple instructions in a counterexample. Even the translation of CPROVER into Java code is complicated. Variable names are also not the same as the source file output. Without this new processing part, the fusion between new Java code instructions and the decompiled source Java Bytecode cannot be done. The next step will be to create an algorithm for finding and sorting all related instructions of a variable from the counterexample within more complex programs.

5 DISCUSSION

Even with our proposed method, the understanding of counterexamples from the Java model checker still seems to

```
import java.util.Random;

public class DivisionTest
{
  public static void main(String [] args
     )
  {
    int[] test = {1,0};
    Random r = new Random();
    int v = 2;
    for (int i=0;i<v;i++)
    {
      assert (test [1]!= 0)
      System.out.println(10 / test[i]);
    }
  }
}
```

Figure 10: New generated code of a Division by 0 Exception

```
import java.util.Random;

public class AssertTest
{
  public static void main(String [] args
     )
  {
    int[] test = {10,15};
    Random r = new Random();
    int v = r.nextInt((2-1)+1)+1;
    assert test[v] ==10;
  }
}
```

Figure 11: Source code of a user assertion

```
import java.util.Random;

public class AssertTest
{
  public static void main(String [] args
     )
  {
    int[] test = {10,15};
    Random r = new Random();
    int v = 2;
    assert test[v] ==10;
  }
}
```

Figure 12: New generated code of a user assertion

be difficult, but it has opened a way for researchers to think about model checking in Java and how to improve it. During this research, we saw that Java model checking does not have many reported solutions as compared to C model checking, but it is still one of the most used languages for software development and requires further studies.

The best future direction will be to have a model checker such as JBMC give developers instructions directly in Java language which can be directly used inside Java code. This step of making counterexamples more understandable is an important one to make model checkers more convenient and useful.

6 FUTURE WORK

As explained above, the method proposed in this paper works with small programs, but it has to be more efficient for use in larger production software. The next step will be to make this method more adaptive to easily add unexpected classes and cases. After that, we can add an implementation for the most used classes including List, Map, etc. This method also has to be adapted for translating software with more than one class which uses a lot of different files. To make the newly generated code more useful, we can directly generate code in a JUnit test which can be a more efficient way to find bugs.

Another innovative way to improve this solution can be to use AI techniques that can directly predict the kind of property violated and suggest ways to resolve it. Employing deep learning by using neural networks can be interesting, but to use this technique we need a large amount of dataset samples before starting.

7 RELATED WORK

The work of improving counterexample generation in model checking is not new ([6], [8], [15]–[19]). One of the first approaches for generating executable counterexamples was made by Dirk Beyer [20]. This paper was inspired by the work of [7] in which a counterexample simplification is done for ANSI-C software by using the ESBMC model checker which also uses the CPROVER framework. All of the understanding of model checking was based on the book Principles of Model Checking [1].

8 CONCLUSION

To conclude, the method proposed in this work was created to help Java developers who are not experts in verification methods to more easily use model checking techniques. It was tested with simple and small programs and has to improve its efficiency to be used with complex software. The new method can still be improved and can open doors for future work such as using AI techniques to make steps in the process automatic and efficient. Model checking, especially for Java programs, is still often underused because of the complexity of use. The method presented in this work can be the beginning for making model checking a common tool for software developers.

ACKNOWLEDGEMENT

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

REFERENCES

[1] C. Baier and J-P. Katoen: "Principles of Model Checking," MIT Press (2008).

[2] E. M. Clarke, O. Grumberg, D. Kroening, D. Peled, and H. Veith: "Model Checking, second edition," MIT Press. (2008).

[3] R. Jhala and R. Majumdar, : "Software model checking," In: ACM Computing Surveys, Vol. 41, No. 4 pp.1-54 (2009).

[4] A. Groce, D. Kroening, and F. Lerda: "Understanding counterexamples with explain," In : International Conference on Computer Aided Verification. CAV 2004, Lecture Notes in Computer Science, Vol. 3114, pp.453-456 (2004).

[5] M. N. Seghir and D. Kroening: "A visual studio plug-in for CProver," In : 2013 3rd International Workshop on Developing Tools as Plug-Ins, TOPI, pp.43-48 (2013).

[6] J. Gennari, A. Gurfinkel, T. Kahsai, J. A. Navas, and E. J. Schwartz: "Executable counterexamples in software model checking," In : Working Conference on Verified Software: Theories, Tools, and Experiments, pp.17-37 (2018).

[7] H. Rocha, R. Barreto, L. Cordeiro, and A. Neto: "Understanding Programming Bugs in ANSI-C Software Using Bounded Model Checking Counter-Examples," In: International Conference on Integrated Formal Methods, IFM 2012, pp.128-142 (2012).

[8] L. Cordeiro, D. Kroening, and P. Schrammel: "JBMC: Bounded Model Checking for Java Bytecode," In: Beyer D., Huisman M., Kordon F., Steffen B. (eds) Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2019, Lecture Notes in Computer Science, Vol. 11429, pp.219-223 (2019).

[9] D. Beyer: "Software verification with validation of results - (report on SV-COMP 2017)," In : Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference. TACAS 2017, Lecture Notes in Computer Science, Vol. 10206, pp.331-349 (2017).

[10] D. Beyer and T. Lemberger: "Software Verification: Testing vs. Model Checking A Comparative Evaluation of the State of the Art," In : 13th International Haifa Verification Conference, HVC 2017, pp.99-114 (2017).

[11] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu: "Bounded model checking," In : Advances in computers, Vol. 58, No. 11, pp.117-148 (2003).

[12] O. Strichman: "Tuning SAT Checkers for Bounded Model Checking," In : Computer Aided Verification, CAV 2000, Lecture Notes in Computer Science, Vol. 1865, pp.480-494 (2000).

[13] E. M. Clarke, D. Kroening, and F. Lerda: "A tool for checking ANSI-c programs," In: 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).

[14] D. Kroening and E.M. Clarke: "The CPROVER User Manual," https://www.cprover.org/cbmc/doc/manual.pdf (2001) (06 July 2021 accessed).

[15] N. Shankar and M. Sorea: "Counterexample-driven model checking," In : Technical Report SRI-CSL-03-04, SRI International Computer Science Laboratory (2003).

[16] T. Ball, M. Naik, and S. Rajamani: "From Symptom to Cause: Localizing Errors in Counterexample Traces," In : Conference Record of the Annual ACM Symposium on Principles of Programming Languages, pp.97-105 (2003).

[17] P. Müller and J. N. Ruskiewicz: "Using Debuggers to Understand Failed Verification Attempts," In : Formal Methods - 17th International Symposium on Formal Methods, FM 2011, pp.73–87 (2011).

[18] D. Kroening, A. Groce, and E. M. Clarke: "Counterexample Guided Abstraction Refinement Via Program Execution," In : Formal Methods and Software Engineering, ICFEM 2004, Lecture Notes in Computer Science, Vol. 3308, pp.224-238 (2004).

[19] K. Rustan, M. Leino, T. Millstein, and J. B. Saxe: "Generating error traces from verification-condition counterexamples," In : Science of Computer Programming, Vol. 55, pp.209-226 (2005).

[20] D. Bayer, A. J. Chlipala, T. A. Henzinger, R. Jhala, and R. Majumdar: "Generating Tests from Counterexamples," In : Proceedings of the 26th International Conference on Software Engineering, ICSE 04, IEEE Computer Society, pp.326–335 (2004).

(Received November 4, 2021)
(Accepted June 9, 2021)

Marwan Bernard Hassan Chellet received his associate degree in computer science from La Rochelle Institute of Technology, France in 2017. One year after, he received his BE in computer science from La Rochelle University, France. Currently, he is a second year student in the master's program at Shinshu Uniersity, Japan majoring in computer science. His current research interests are in verification methods and software model checking.

Shinpei Ogata is an Associate Professor of the Graduate School of Science and Technology in Shinshu University, Japan. He received a PhD from Shibaura Institute of Technology, Japan in 2012. His current research interests include model-driven engineering for information system development. He is a member of IEEE, ACM, IEICE, and IPSJ.

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.

Regular Paper

Building a Robust RTK-GNSS Infrastructure with Seamless Handover and a Multipath Detection Approach

Bhagawan Rokaha[*], Bishnu Prasad Gautam[**] , and Tomoya Kitani[*]

[*]Graduate School of Integrated Science and Technology, Shizuoka University, Japan
[**]The Department of Economic Informatics, Kanazawa Gakuen University, Japan
b-rokaha@kitanilab.org, gautam@kanazawa-gu.ac.jp, t-kitani@kitanilab.org

Abstract - RTK-GNSS is a promising positioning technique to achieve centimeter-level accuracy. In this technique, a stationary base station plays a vital role in correcting the positioning results of a movable user receiver; however, the base station correction signals are often interrupted, or delayed due to single-base line area, hardware biases, environmental factors, and multipath errors. Therefore, we propose three major new components to improve a user receiver's positioning accuracy and precision. The first component detects the status (i.e., healthy or unhealthy state) of the base station through the internet. The second component assigns the most favorable base station from multiple base stations in a seamless approach. The final component detects the multipath signal using a machine-learning classifier model. After analyzing the experimented results, our approach maintained the rover receiver positioning accuracy within the centimeter-level even after the base station handover. Similarly, in multipath detection, around 98% of NLOS and around 95% of the LOS signals are correctly discriminated. By combining all three components, we achieved high reliability of RTK-GNSS positioning in different areas by using continuous correction signals from the base station and considering only the visible satellites.

Keywords: RTK-GNSS, Seamless handover, Web-based monitoring, Reliable infrastructure, Multipath detection

1 INTRODUCTION

Global Navigation Satellite System (GNSS) is an active research area for navigation, mapping, positioning, and many other areas that need monitoring and controlling their location-based services. In the conventional single-point positioning system, the user's position can be instantly determined using a pseudorange between the satellite and the user's receiver. For this, the receivers need a signal from four or more satellites. In this single-point positioning, the positioning accuracy ranges from 10m to 30m, as various factors caused errors in the GPS observation [1]. However, many applications, including autonomous driving and flying, precision agriculture, and weather forecasting, require centimeter-level accuracy, which is called precise positioning. Therefore, we need advanced positioning techniques to provide highly accurate positioning and navigation functionalities in those applications. One of the famous differential positioning systems is the Real-Time Kinematics-Global Navigation Satellite System (RTK-GNSS).

A higher resolution distance information called a phase pseudorange is used instead of the code pseudorange. As shown in Fig. 1, the precise position, also called fixed position, of a rover receiver, i.e., user receiver, is calculated through the received signal from satellites and the correction signal from the base or reference station.

However, rover receiver position accuracy is degraded severely because of the interrupted, delayed, or discontinuous base station's correction data as well as communication link. In this differential system, correction data from the base station is affected by a number of factors: such as coordinate errors, environmental factors, the reflected or diffracted signals, which results in a less accurate position (i.e., within a few meters of accuracy), called a float solution.

Therefore, challenging environments, including snowy, mountain, forest, and urban areas, are crucial for precise positioning in RTK-GNSS. Errors that occurred due to these factors are extremely difficult to solve through differential correction techniques. For instance, the precise positioning applications like the drone carrying medicines and equipment service, aiming to provide medical care to the remote mountain communities and precisely measuring the altitude of the mountain, including Mt. Everest, is the subject of attention in the Himalayan country, Nepal [2], [3]. Similarly, applications like precision agriculture, mapping, and survey, weather forecasting are gradually increasing in many countries.

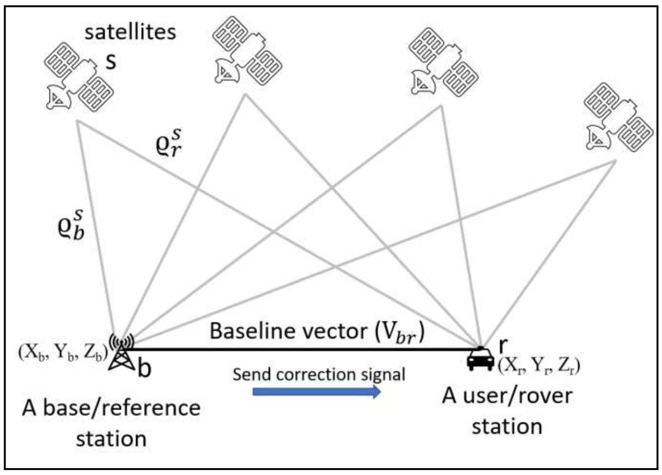


Figure 1: Principle of differential positioning

However, service disruption and false assumptions caused by different error factors in the base station may lead a rover receiver to use unreliable correction information from unhealthy base stations. As a result, it will mislead the rover receiver as well as degrade the reliability and accuracy of the base station data.

Furthermore, the base receiver's differential correction signal is valid for the short-baseline range only, i.e., generally considered the area within 10km. Therefore, the conventional RTK technique is inefficient and cannot ensure the continuity of the GNSS signal for a moving object that may operate beyond a base operating range. Besides, no redundancy of the base station is usually available if the active base station experiences any malfunctioning or hardware bias errors.

On the other hand, satellite positioning is still challenging in urban areas due to signal reflections by buildings or skyscrapers, so-called multipath error. As a result, positioning accuracy is severely degraded.
Therefore, focusing on those errors factor of snowy areas and multipath areas, this paper presents a modality of a robust RTK-GNSS infrastructure that guarantees continuity and reliability for precise positioning.

This paper is divided into seven sections. After the first introductory section, the second section clearly describes the problem statement, where we discuss the necessity of this research work. Section 3 gives a brief overview of the past researches and the preliminary works. Section 4 is an important section, where we discuss the system approach with theoretical dimensions of the research. Section 5 and 6 describe the design, methodology, working principle, characterization, result, and evaluation of this research. Finally, the last section of this paper gives a brief conclusion.

2 PROBLEM STATEMENT

The reliability and continuity are major concerns in RTK-GNSS, even though it is widely used in various applications. Notably, while doing RTK-GNSS experiments in dense snowfall or high buildings areas, various limitations were encountered. Therefore, to address the specific problem through a new approach, the authors mainly focus on three dominant problems that should be addressed for continuous and precise positioning.

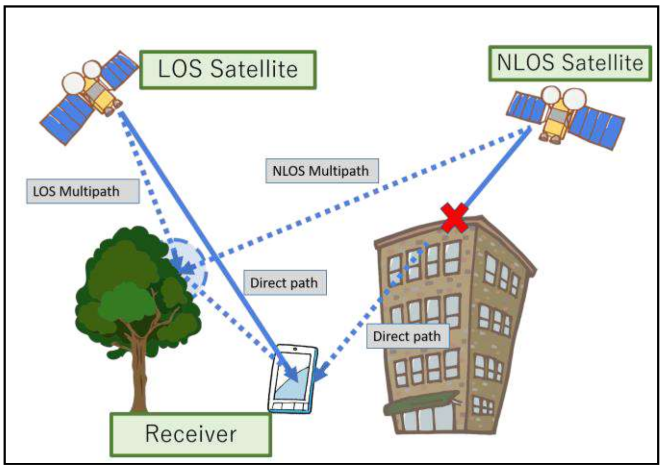


Figure 2: Multipath signals

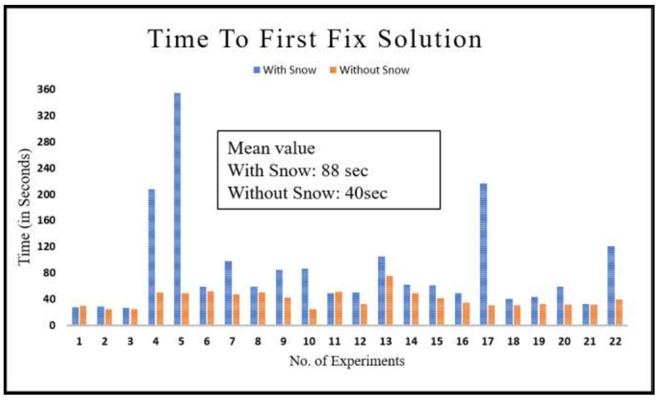


Figure 3: Snow effects in time to first fix solution

2.1 Unknown Status of The Base Station

Any problems or errors in the base station affect the corrective signal used to calculate the rover receiver's precise positioning. For example, in the mountain or snowy area with the uneven landscape, hard frost weather, and chances of heavy snowfall, landslide, earthquake, and volcanic eruption caused significant errors. Mainly, when the bunch of snow covered the base station's antenna, the signal strength was degraded because of the multipath error induced by the snow surface [4]. Also, if the coordinate of the base station changed because of landslides or by other factors, the positioning accuracy of the rover receiver would degrade due to the inaccurate base station's coordinate. Similarly, if the running base station experienced any malfunctioning or hardware errors, no redundancy solution would be available to detect the base site's status from the rover end. In addition, when the base station interrupted by some errors, the recovery time was often several minutes or even a few hours. Hence, the rover receiver ends up using a correction signal from that unstable or unhealthy base station. As a result, positioning accuracy is severely degraded.

On the other hand, if there is no correction signal from the base station, we need to visit the actual field to confirm a base station's state; however, this is not a cost-effective, reliable, and appropriate solution for real-time applications. Therefore, ensuring the data continuity and reliability of the base station under challenging environments is very important.

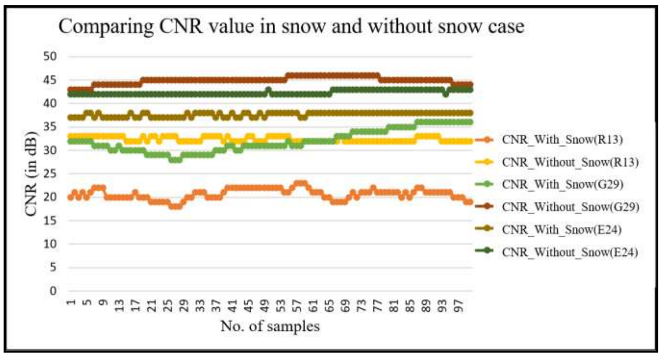


Figure 4: Snow effects in Carrier to Noise Ratio

2.2 Limitation of Base-to-Rover Operating Range

In the conventional RTK-GNSS, the base and the rover station need to operate in the same environmental area (i.e., generally considered 10km from the base station). Beyond this range, distance errors and atmospheric conditions at the base and the rover receiver may significantly vary. These error factors cannot cancel out through differential processing. Therefore, the base station data outage is one of the major concerns, particularly for the moving object. In the past few years, many researchers have proposed a multi-network base station adjustment process for the wide area [5]; however, those methods are challenging to implement in the actual field and for a smooth handover, i.e., handover to another base station without dropped signal. Thus, to provide a continuous correctional signal in a wide area for a moving object, the authors are concerned with this problem.

2.3 Multipath Error on The Rover Receiver

GNSS satellite signals are subject to reflection and diffraction, like any other type of electromagnetic wave. Therefore, in an urban area where the grounds are surrounded by tall buildings, skyscrapers, or trees, the rover receiver often faced multipath error. Notably, the reflected or diffracted signal from those objects causes multipath errors. In general, the receivers receive both direct and reflected (or diffracted) signals. As the multipath signal takes a longer path than the direct signal, an error was caused in pseudorange measurement, which severely degrading the GNSS accuracy to several meters [6].

There are mainly two types of multipath signals: Line of Sight (LOS) multipath signals and Non-Line of Sight (NLOS) multipath signals, as demonstrated in Fig. 2. For LOS multipath error, various signal correlation techniques were proposed to mitigate or minimize the LOS multipath signal by many past researchers.

However, there is no reliable technique for the NLOS multipath. Also, it is more crucial than LOS multipath signals because of the reflected signals. As a result, a broad range of positioning errors happens. Hence, NLOS multipath detection and mitigation techniques are required.

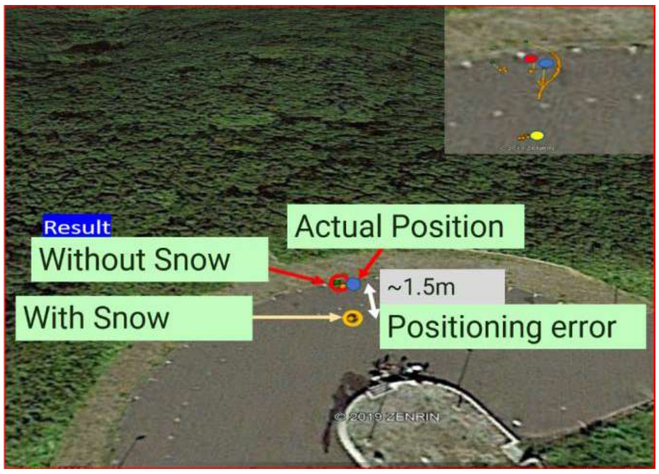


Figure 5: Positioning error caused by Snow accumulation

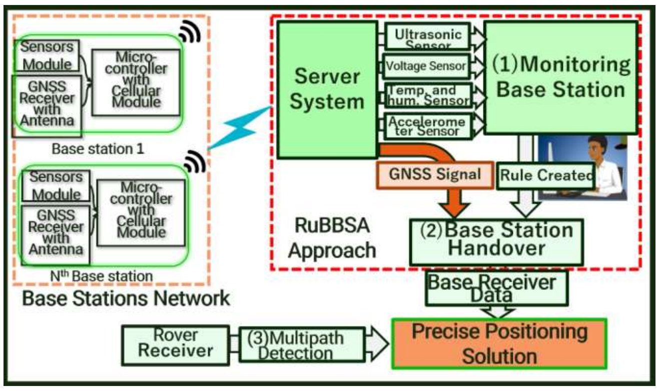


Figure 6: System architecture

In summary, to increase the overall performance and build a robust RTK infrastructure in the mountain and urban areas, multi-base stations with seamless handover mechanisms and NLOS multipath detection mechanisms are essential parameters.

3 BACKGROUND

3.1 Preliminary Research

Our preliminary research proposed a cost-effective and reliable system that overcomes the conventional RTK-GNSS infrastructure to enhance positioning solutions [7].

The goal of that research was to evaluate the low-cost receiver's capability in the harsh receiving condition (such as challenging weather, multipath, obstruction, etc.) and find the major problems that happen in the base site [8]. Therefore, we conducted our experiment in two different weather and geographical regions to demonstrate positioning accuracy, reliability, and feasibility of the system's architecture.

Nonetheless, the major problem encountered while doing an experiment in the heavy snowfall region.

The snow accumulation problem, the signal strength, and time to first fix solution (TFFS) are negatively affected, as shown in Fig. 3. Here, TFFS means the time need to get the first fix solution from the float solution. Similarly, a significant difference was found while comparing the value of Carrier to Noise Ratio (CNR) between snow and without snow state, as shown in Fig. 4. The difference of carrier to noise ratio is more than 8dB because of the reflected or a diffracted signal when the snow height on the antenna is around 15cm.

In the second experiment scenario, we have done our experiment where GNSS signals are often obstructed by buildings leading to reflected and diffracted signals. The observation shows the tendency of a drop in SNR when the receiver is in the multipath environment. SNR measurements are smoother when the receiver is in an obstruction-free environment, as shown in Fig. 5.

We concluded that conventional RTK-GNSS is still insufficient to provide continuous, reliable, and precise positioning in those areas from these experimental scenarios and results. Thus, this research focused on cost-effectively building a robust RTK-GNSS infrastructure.

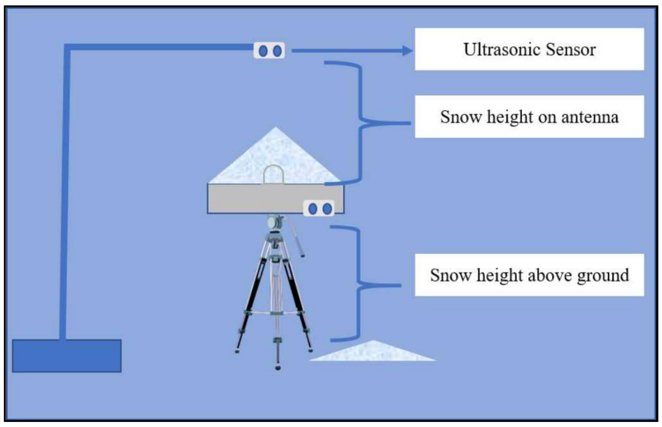


Figure 7: Snow height measurement scenario

3.2 Related Research

Several previous studies enhance the reliability and accuracy of the RTK-GNSS technique using different methods. In RTK positioning, the rover receiver needs to work within a base operating range, which is a major constraint for a moving object that works beyond the operating range. Therefore, many networking techniques that use multiple base stations were practiced in recent years. Some of the networking techniques are Master-Auxiliary (MAC), Virtual Reference Station (VRS), Pseudo-Reference Station (PRS), which operate with multiple base stations and provide precise positioning [9]. Quan et al. proposed a Network RTK system using observations from numerous continuously operating base stations [10]. Similarly, VRS based RTK system was also proposed to provide continuous observations in Malaysia [11]. However, these methods are challenging to implement, need active communication links, demanding control center operation, stability issues, and expensive operating costs. Also, a continuous handover could be the limitation of these Network RTK systems. Generally, accuracy drops to float solution from the fixed solution in the handover process.

Similarly, to mitigate the NLOS multipath error, it is crucial to identify the NLOS signal from all received GNSS signals. A new research stream dealing with multipath detection utilizes an additional sensor, 3D mapping, or image processing technique. Suzuki et al. proposed a fisheye camera and omnidirectional infrared camera techniques to detect multipath signals [12]. However, these detection techniques are affected by weather, light conditions. The method of integrating multi-sensors might be helpful in some conditions but could not solve entirely.

Also, a laser scanner to differentiate visible and invisible satellites was proposed by Maier et al. [13]. Also, multipath detection using 3D maps and Aerial LiDAR data were also practiced in few researches [14]. However, these were complicated to apply for the moving object and challenging to integrate in real-time. Some researchers worked on multiple GNSS signal correlators in a software GNSS receiver; however, it is challenging to design a special correlator in the practical field effectively. Therefore, to address the station-based errors and the base station handover issue, we felt the necessity of reliable, smoothly operable, easily applicable, and cost-effective RTK-GNSS used in different places and scenarios, including Himalayan and snowy regions.

Thus, we have primarily done our research [15] to analyze the system architecture and introduced the handover scheme and multipath detection approach in this research paper.

4 SOLUTION APPROACHES

This section explicitly describes our approaches to solving conventional RTK-GNSS, especially in mountain/snowy areas and urban areas, as shown in Fig. 6. Firstly, to know the base station's status from the rover side (as described in subsection 2.1), we proposed a base station monitoring system such that an unhealthy base station could be detected in the rover end. Secondly, we proposed a seamless handover mechanism with a multi-base network to solve the single base-rover range problem (as described in subsection 2.2). Finally, to solve the multipath error on the rover receiver (described in subsection 2.3), we proposed LOS and NLOS multipath detection mechanisms. The detailed explanations are as follows:

4.1 Detecting an Unhealthy Base Station

The first approach is knowing the base station's status from the rover end in real-time. When different limiting factors obstruct the base station, the correctional signal from that base station consists of many errors. As a result, the positioning solution in the rover receiver degraded. Therefore, to make a proper decision and constant alert for future trouble prevention, we proposed detecting unhealthy base stations through the internet. Different kinds of sensors require in this proposed system. For example, we used ultrasonic sensors and accelerometer sensors, which are used for snow height measurement and antenna movement detection, respectively.

After collecting data on the server, sensors' data are used for two purposes: web-based monitoring and the optimum base station selection process. Here, we proposed a web-based base station monitoring technique to monitor the base station's status manually. Sensors' data are visualized on a web page and monitored from anywhere, provided that an internet connection is available. Similarly, a handover process is needed for the kinematic rover receiver beyond a base station's baseline area.

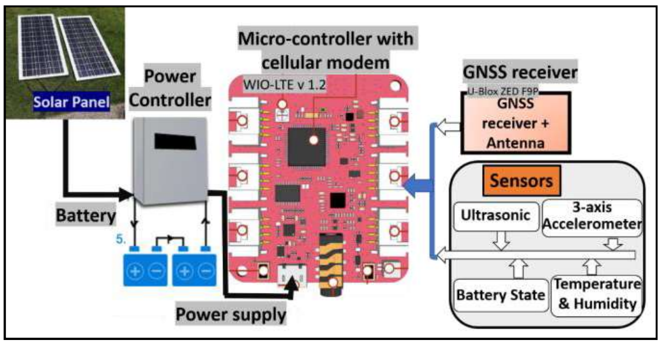


Figure 8: Block diagram of the base station

In such conditions, the optimum base station (i.e., a favorable base station among available base stations) is needed in the rover receiver to calculate precise positioning. Therefore, the base station's sensors data are used with the base-rover range to detect the base station's state. If the particular sensor data is less than the respective threshold value, the base station is considered unhealthy. For instance, the threshold value of snow height is 5cm because the signal strength is decreased faster after that height. For a reliable system, the physical hardware and data processing method also have a significant impact. Therefore, we proposed compact hardware and reliable data processing.

4.2 Seamless Handover

The second component is a processing component, an algorithm specially designed to assign the most favorable base station from the list of multiple base stations in a seamless manner. To achieve precise positioning, we proposed a Rule-Based Base Station Assignment (hereafter, RuBBSA) algorithm. In this algorithm, the rule is based on two major factors: sensors' value and the distance between the rover receiver and the corresponding base station (explicitly explained in section 5.5). The first rule ensures that the sensor's measurement is used to monitor the base station's physical condition, such as snow accumulation, battery power supply outage, and the base station's coordinate. The second rule ensures the operational range of the receiver.

For instance, when the user moves out from the functional baseline area and or conditions when the base station cannot send differential correction information to the rover receiver, this algorithm assigns a new base station. This processing of choosing an optimum base station from multiple base stations is the main target of this algorithm. The next available optimum base station is assigned by the proposed algorithm seamlessly and dynamically.

We used two RTK engines on the rover side for this handover process. In the RTK system, fixed positioning solution is obtained by using carrier-phase measurements rather than just pseudorange. However, the processing of carrier phase measurements is subject to so-called carrier phase ambiguity, an unknown integer number of times the carrier wavelength that needs to be fixed. The ambiguity resolution, which is the crucial factor for precise positioning, is the process of resolving the unknown cycle ambiguities of double-difference carrier phase data as integers. There are mainly three steps to determine ambiguities resolution: estimating float-valued ambiguities, finding the best integer ambiguity set, and validating the best ambiguity set. After validating the ambiguity set, a fixed solution is calculated in the RTK engine. However, when one base station is changed to another base station in the same RTK engine while doing handover, validating the ambiguity set is difficult. Only float-valued ambiguity occurred for few seconds. As a result, the positioning solution is dropped into a float solution. Therefore, to make a handover without losing the fixed positioning solution, two RTK-engines are needed. Thus, the proposed mechanism has two RTK-engines to provide continuous and precise positioning.

4.3 Multipath Detection

In order to increase the positioning accuracy of the rover receiver in a multipath environment, this research also aims to develop a multipath detection technique. Here, multipath detection proposed by using different satellite signal features by differentiating LOS signal and NLOS multipath signal. In the last decade, many methods have been proposed to detect multipath signals. Most of the research focuses on a single-point conventional positioning system by using different algorithms [16], [17].

Only one receiver is used to calculate its own position in single-point positioning; therefore, differentiating LOS signals and NLOS multipath signals are comparatively more straightforward. However, we need to consider the rover and the base station's observation data in the RTK-GNSS technique. Therefore, to classify the LOS and NLOS multipath signals with high accuracy, we proposed a machine-learning-based classifier that can differentiate LOS signals and NLOS multipath signals using significant features values.

Here, the kernel-based support vector machine (SVM) classifier is used to differentiate multipath signals. It is essential to train our data with accurate classification because, based on the training data, a machine learning model learned the features and predicts the output accordingly. Based on the characteristic of CNR, which says that the signal fluctuates under static conditions; thus, the differential CNR value (i.e., the difference between base CNR and a rover CNR value) has been used to detect NLOS signals. These featured values are applied to the training process in machine learning.

5 METHODOLOGY

5.1 Building a Compact Hardware

This research has been conducted by building a base station prototype module consisting of a GNSS receiver with antenna, micro-controller, and sensors network. In the rural and the Himalayan region, the continuous power supply is one of the significant problems. Therefore, a solar energy source- an easily movable, self-sustainable, and reliable power source- is used. A micro-controller called Wio-LTE is used to process GNSS receiver and other sensors data, which is a prototyping development board with LTE(4G) communication version of Wio Tracker (Wireless Input-output) that enables faster IoT GNSS solutions [18]. To monitor each base station's physical condition, we used various sensors modules in respective base stations. For instance, ultrasonic sensors are used to predict the base antenna's snow height, as shown in Fig. 7.

During this process, temperature fluctuation affects the transmitted sound wave from an ultrasonic sensor. Thus, the temperature sensor is used to correct the temperature-related distortion of the measured value. Also, to precisely monitor the base station coordinate, we used a 3-axis digital accelerometer sensor that detects orientation, gesture, and motion in case of natural disasters. Besides that, to check battery voltage level and charging level, a battery state sensor is used. To make a compact hardware system, all sensors are connected to the microcontroller board that consists of a cellular modem, as shown in Fig. 8. All data are sent to our server system in real-time. The base station antenna is fixed correctly in the accurate position.

Similarly, to overcome the multipath errors and fully receive all visible satellite signals, the antenna is placed in an open sky environment such that all satellite signals are received as a Line Of Sight (LOS) signal. This compact infrastructure consumes deficient power. The base station system needs 600mA to 2A current with a 5V power supply at regular communication, making the system operation longer.

Moreover, the system consists of a low-cost receiver and digital sensors. The total cost is around \$1500. In contrast, survey-grade receivers cost around \$10,000, for instance. Therefore, our system can be comparatively cost-effective, movable, and easily installable in regular and challenging weather areas.

5.2 Data Processing and Management System

In our proposed system, collected sensors data from each base station is sent to the server system through the internet connection. Here, we need a proper and continuous communication link between a base station and a user receiver for the real-time application. Therefore, we should select effective communication to provide reliable internet connection continuously. Considering those factors, including operational cost, maintenance cost, and the number of computations needed by the rover and the processing center, we found that a cellular modem is one of the appropriate methods in the mountain region. We used cellular connectivity Soracom, which provides the IoT network platforms [19].

JavaScript Object Notation (JSON) format is used to transmit these data in web-based applications such that those data could be displayed on a web page correctly. The time-series data shows the base station's past conditions and present conditions, such that the changing state is visible and easily compared to the respective sensors' threshold value. Thus, detecting unhealthy base stations is done before assigning a base station in a rover receiver. The web-based graph is plotted on the web browser, which shows that each sensor's data is being updated with time-lapse. This process of monitoring the actual state of data is adequate, especially when the base stations' knowledge is manually needed for the assignment process.

5.3 Determining of an Optimum Base Station

In this section, the mechanism of relative positioning is briefly introduced along with the sorting mechanism to determine the optimum base station.

As shown in Fig. 1, the code pseudorange ρ_b and phase pseudorange ϕ_b at base station b to satellite s measured at epoch t_0 can be modeled by

$$\rho^s_b(t0)=\varrho^s_b(t0)+\Delta\varrho^s_b(t0)+\Delta\varrho^s(t0)+\Delta\varrho_b(t0) \qquad (1)$$

$$\lambda^s\Phi^s_b(t_0)=\varrho^s_b(t_0)+\Delta\varrho^s_b(t_0)+\Delta\varrho^s(t_0)+\Delta\varrho_b(t_0)+\lambda^s N^s_b \qquad (2)$$

Where ϱ^s_b (t_0), $\Delta\varrho^s_b$ (t_0), $\Delta\varrho^s$ (t_0) , $\Delta\varrho_b$ (t_0), N^s_b are the geometric range, orbital errors, satellite-dependent errors, receiver-dependents errors, and phase ambiguity, respectively [20]. Also λ^s is the wavelength, defined as $\lambda^s = c/f^s$, where c is the speed of light and f^s is the frequency of satellite carrier. In relative positioning, the code and phase correction of the base station for the same satellite at base epoch t_0 is calculated as

$$PRC^s(t_0) = \varrho^s_b(t_0 - R^s_b(t_0) \qquad (3)$$

$$PRC^s(t_0)=\varrho^s_b(t_0)-\lambda^s\Phi^s_b(t_0) \qquad (4)$$

Similarly, in the rover receiver 'r', the code pseudorange ρ_r, and phase pseudorange ϕ_r are calculated for the observation epoch t because the range and range rate correction (RRC) referring to the base epoch t_0 are transmitted to the rover receiver in real-time. At r, the pseudorange, carrier phase, and the pseudorange correction (PRC) for the observation epoch t is modeled by

$$\rho^s_r(t) = \varrho^s_r(t) + \Delta\varrho^s_r(t) + \Delta\varrho^s(t) + \Delta\varrho_r(t) \qquad (5)$$

$$\lambda^s\Phi^s_r(t) = \varrho^s_r(t) + \Delta\varrho^s_r(t) + \Delta\varrho^s(t) + \Delta\varrho_r(t) + \lambda^s N^s_r \qquad (6)$$

$$PRC^s(t) = PRC^s(t_0) + RRC^s(t_0)\ (t - t_0) \qquad (7)$$

where (t-t0) is defined as latency. After applying the predicated pseudorange correction $PRC^s(t)$ to the measured pseudorange of the rover receiver, the satellite-dependent bias has canceled out. Also, the base and the rover receiver have highly correlated satellite-receiver-specific biases in a short baseline area. Neglecting these biases, the corrected code and the phase pseudorange are calculated in the rover receiver as

$$\lambda^s\Phi^s_r(t)_{corr} = \varrho^s_r(t) + \Delta\varrho^s_{br}(t) + \lambda^s N^s_{br} \qquad (8)$$

$$\rho^s_r(t)_{corr} = \rho^s_r(t) + PRC^s(t) \qquad (9)$$

Where $\Delta\varrho^s_{br}(t) = \Delta\varrho_r(t)$ - $\Delta\varrho_b(t)$ and $N^s_{br} = N^s_r$ - N^s_b are the difference of phase ambiguities [20]. To determine the coordinate of an unknown point concerning a known point. Thus, the baseline vector between the base and the rover is calculated with corresponding position vectors X_b, and X_r formulated as

$$X_r = X_b + X_{br} \qquad (10)$$

$$V_{br} = \begin{bmatrix} X_r - X_b \\ Y_r - Y_b \\ Z_r - Z_b \end{bmatrix} = \begin{bmatrix} \Delta X_{br} \\ \Delta Y_{br} \\ \Delta Z_{br} \end{bmatrix} \qquad (11)$$

Here, the base point coordinates must be accurately known to calculate the rover receiver coordinate with high precision. In the case of a kinematic rover receiver, it is continuously moving from one place to another. Therefore, the positioning information of the rover receiver is updated in the control unit regularly. Then, the distance between the rover and each base station is calculated and list out all neighboring base stations. The least distance is in the highest priority order. The distance between the rover and all neighbor base stations is calculated as follows

$$\Delta D_{rb} = E \cdot \arccos[(\sin(lat_r) \cdot \sin(lat_b)) + \cos(lat_r) \cdot \cos(lat_b) \cdot \cos(long_b - long_r)] \quad (12)$$

Where lat_r, lat_b, $long_r$, $long_b$ are latitude of a rover, latitude of a base, longitude of a rover, and longitude of a base, respectively. All values in radians. E is the equatorial radius of earth and ΔD_{rb} is the distance between the rover and a base station.

Furthermore, the availability of the base station is also measured through sensors data. For instance, ultrasonic sensors, voltage sensors, and 3-axis accelerometer sensors are used in this research. Besides these sensors, the other sensors can be used based on geographical and environmental conditions. The threshold values need to be entered at the starting time. In this rule of the assignment process, there are the following three cases.

Case I: $S_i \geq S_{TH}$ and $D_i \leq D_{i+1}$; optimum base station

Where S_i is the output of a cumulative function of sensors values, the threshold value of S_{TH} is needed to set after various experiments. Also, D_i is the least distance between the rover, for ith base station. Similarly, D_{i+1} is the distance between the next adjacent base station and rover. In the above scenarios, if the base station satisfies case I, this base station is considered as an optimum base station and assign this base station till the subsequent handover is needed.

Case II: $S_i < S_{TH}$ and $D_i > D_{i+1}$; keep and hold (OK)

If the base station satisfies case II, the base station is considered as an acceptable base station or the next potential base station. Thus, these base stations are kept and hold for the next handover. Handover may require while the rover moved far from the earlier base station and approaches the adjacent base station. In this case, the earlier base station will be dropped off, and the adjacent base station will be handover for the operating base station.

Case III: $S_i < S_{TH}$; remove from the list (NG)

If the base station satisfies case III, it is considered functionless or not a referenceable base station. Therefore, this base station is removed from the list until it satisfies cases I or II. This situation may arise due to various reasons such as due to the accumulation of snow, increase of distance between rover and base station etc.

5.4 Base Station Assignment Process

After successfully determining the optimum base station through sensors' value and distance measurement, the handover mechanism is processed. We need to make configurations for this process such that the correction data and base station coordinate are streamed in both RTK engines.

In the RTK system, the base station sends corrections to the rover via a communication link. This correction signal enables the rover receiver to compute its position relative to the base with high accuracy. Radio Technical Commission for Maritime (RTCM) is the standard format with a binary data protocol for communication. The output stream should be changed in RTCM format to send standard messages and the real base antenna reference point (ARP). Therefore, a resident type application, str2str of RTKLIB [21], is used to input and output stream path. The input command seems as

```
./str2str-intcpsvr://localhost:60021#ubx
-out tcpcli://localhost:52081#rtcm3 -s 0
    msg 1005,1077,1087,1097, 1127 -p
    34.726598357 137.718089538 97.398
```

The data from the TCP server in the u-blox format (i.e., the message format type received by the u-blox receivers and fully configurable with UBX protocol configuration messages) as input stream outputs in the RTCM3 format.

Besides that, the coordinate of the base station (i.e., latitude, longitude, and height of the base station) and RTCM messages are streamed. These multiple signal messages (MSM), as shown in Table 1, are streamed as soon as they are configured for the corresponding GNSS.

In this approach, the base station's coordinates are changed when the base station is assigned dynamically. As every base station consists of a cellular SIM (subscriber identification module), the unique IMSI (International Mobile Subscriber Identity) number is used for the identification of each base station. Here, the base station is assigned as follows:

```
Chdist[RTK-engine number][base station's
              IMSI number]
```

where `chdist` is a command to query the status of a package of a base station to the designated RTK engine; thus, the RTK engine at the rover side receives the observation data from that assigned base station, and finally assigned to the application layer for precise positioning.

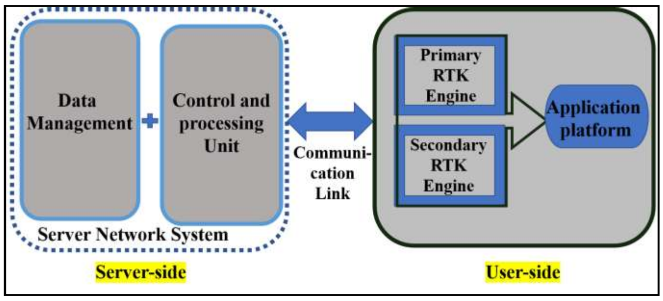


Figure 9: RuBBSA approach

5.5 Base Station Handover Mechanism

This section explains the principle of the RuBBSA algorithm and its approach for seamless handover. There are server backend and user end, as shown in Fig. 9. The server-side server network consists of two primary units: the data management and a control unit. The data management unit is designed to manipulate and manage data. Generally, all physical sensors data, all base stations coordinate data, and real-time differential correction data of corresponding base stations are collected and then manipulated. These data are processed to the central unit called as control and processing unit, where the rule is created to specify the most favorable base station from multiple base stations.

The RTK processing engine is placed on the user side, where the processing of differential correction signals from the base station and positioning observation from the rover receiver is carried. In this proposed system, two RTK engines named primary and secondary RTK engines are used to make a seamless handover operation. The primary RTK engine operates as a default RTK engine that runs until the handover is needed. The assigned base station from a control panel is linked with the primary RTK engine to provide precise positioning in the application layer. The final target of this research is to make a complete autonomous handover system; however, in this research, the server-based handover mechanism is proposed.

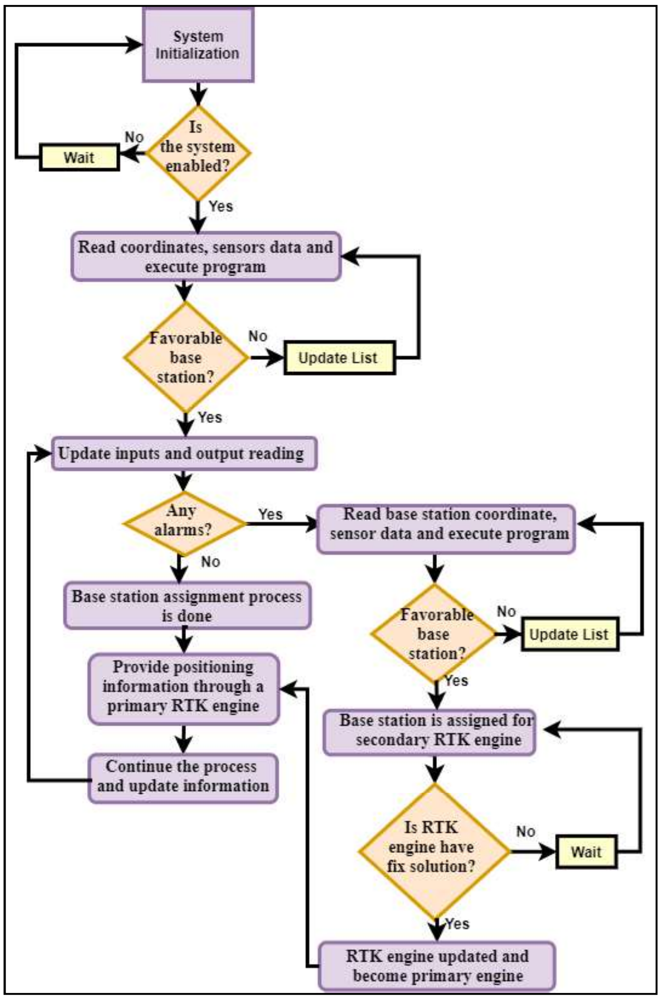

System Initialization
Is the system enabled?
No
Wait
Yes
Read coordinates, sensors data and execute program
Favorable base station?
No
Update List
Yes
Update inputs and output reading
Any alarms?
Yes
No
Read base station coordinate, sensor data and execute program
Base station assignment process is done
Favorable base station?
No
Update List
Yes
Provide positioning information through a primary RTK engine
Base station is assigned for secondary RTK engine
Continue the process and update information
Is RTK engine have fix solution?
No
Wait
Yes
RTK engine updated and become primary engine

Figure 10: Flowchart of RuBBSA approach

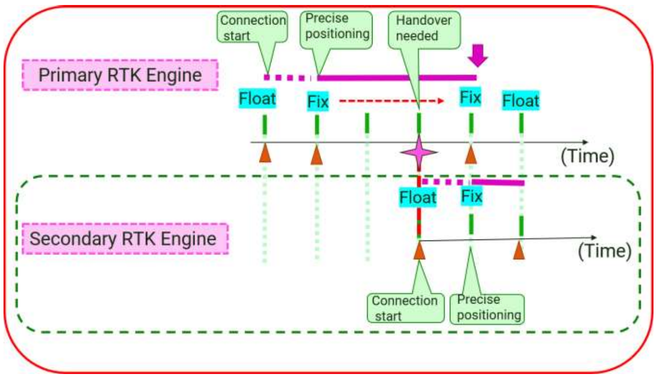


Figure 11: Seamless handover

The flowchart in Fig. 10 explains the mechanism of determining the optimum base station and seamless handover. In this handover mechanism, primary inputs are sensors data (i.e., ultrasonic sensor, accelerometer sensor, and voltage sensor) and coordinates of receivers (i.e., a base and a rover coordinate). From these inputs data, the base station availability has checked using the threshold value. Suppose the base station is a favorable base station based on the input values. In that case, it is considered the optimum base station (also called a favorable base station) and assigned that base station in the primary RTK engine. Here, we introduced an alarm function to check input values continuously. The alarm function is not activated until the currently assigned base station is fulfilling the condition to be optimum.

On the other hand, if the base station is out of the baseline area or the sensor's value is less than the threshold value, the alarm is created, which processes the handover. Thus, the currently assigned base station is replaced by the next adjacent base station through a secondary RTK engine. At that time, a new base station is assigned to the secondary RTK engine because the first base station is still operating in the primary RTK engine. In secondary RTK, the positioning solution is the float for a few seconds; therefore, removing the base station at the primary RTK engine is done after getting the fixed solution in secondary RTK. The removing process is done as

```
rmdist[RTK-engine number][base station's
            IMSI number]
```

where `rmdist` is a command to detach the communication link between the designated base station and RTK engine; after that, the positioning solution is handled from the secondary RTK engine.

Table 1: RTCM message type and description

Message Type	Description
RTCM 1005	Stationary RTK reference station ARP
RTCM 1077	GPS MSM7
RTCM 1087	GLONASS MSM7
RTCM 1097	Galileo MSM7
RTCM 1127	BeiDou MSM7
RTCM 1230	GLONASS code-phase biases

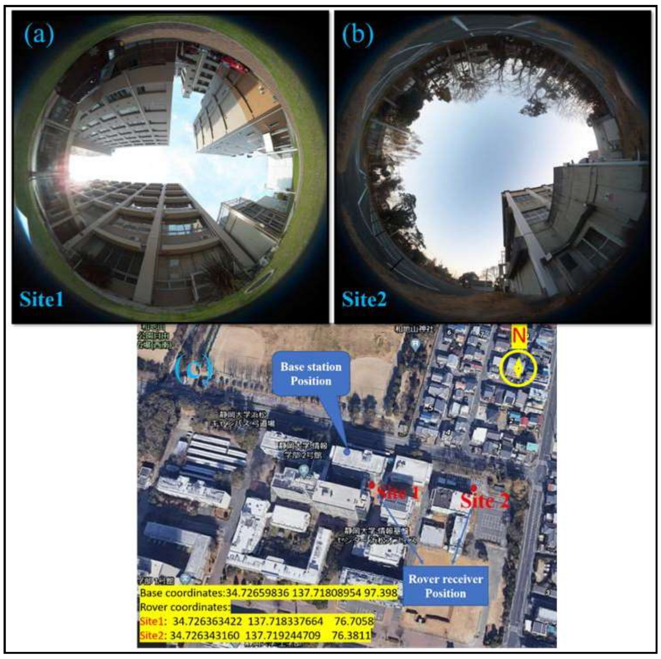

(a)
(b)
Site1
Site2
(c)
Base station Position
N
Site 1
Site 2
Rover receiver Position
Base coordinates:34.72659836 137.71808954 97.398
Rover coordinates:
Site1: 34.726363422 137.718337664 76.7058
Site2: 34.726343160 137.719244709 76.3811

Figure 12: Rover Site location: (a) Site 1, (b) Site 2, (c) Google map

Thus, the positioning solution at the rover receiver is calculated without any interrupted or dropped signal, as shown in Fig. 11. These handover algorithms are written in the C programming language.

5.6 Multipath Detection Using Kernel SVM

The satellite signal consists of both direct and multipath signals. Therefore, the proper classifier is needed to mitigate multipath signals. This research proposed a multipath detection technique by using the classifier method. We have acknowledged that many researchers proposed a multipath signal detection method with different algorithms. They proposed a method such as detecting NLOS using observation data and existing 3D building data [14]. However, the process of detection is complicated and lengthy. Some researchers also proposed multipath detection methods using additional sensors [22]; however, those methods are complicated in differential positioning techniques in the practical field. Therefore, we proposed a practically implementable classifier without adding any additional sensors or hardware devices.

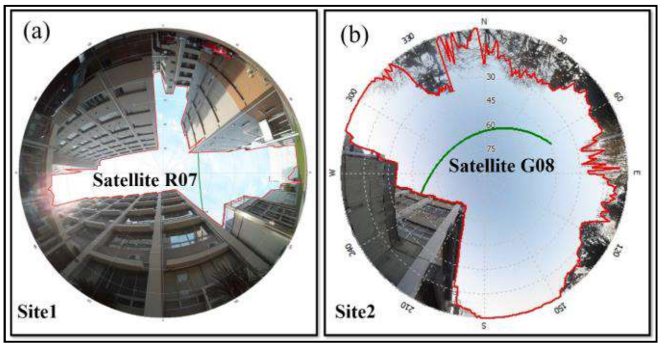


Figure 13: Mask Making; (a) Site 1, (b) Site 2

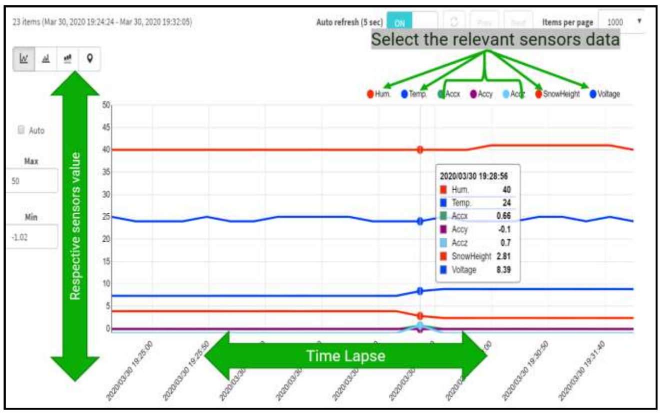


Figure 14: Graphical view of sensors data on webpage

In order to classify multipath and direct signals, we proposed a method that utilizes a classifier based on a machine learning algorithm. We used machine learning algorithms that can deal with linearly separable and non-separable data, called the kernel-based support vector machine (SVM). The reason behind this classifier is that the satellite signal may not always be linear. Thus, the data might not classify with a simple linear method to discriminate correctly.

Therefore, we need an algorithm that can deal with higher dimensions to make inseparable to separable form. So, we used kernel SVM in this work.

The kernel SVM is a supervised machine learning algorithm mainly used for classification purposes because it tries to learn similarities between datasets, and those become support vectors. Those support vectors are the data points that define the position and the margin of the hyperplane. The optimum hyperplane is the one that maximizes the margin, under the constraint that each data point must lie on the right side of the margin. Thus, only the support vectors are enough to make a classification. Here, multiple features data are used while training the classifier. The details of the multipath detection using kernel SVM, and features data are as follows.

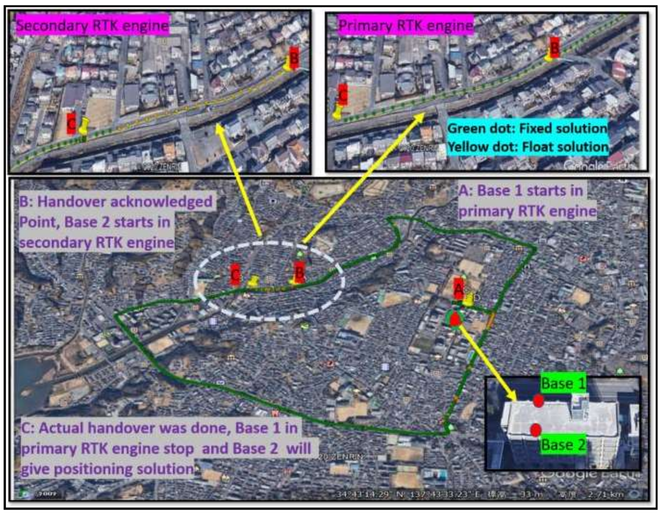

Secondary RTK engine
Primary RTK engine
Green dot: Fixed solution
Yellow dot: Float solution
B: Handover acknowledged Point, Base 2 starts in secondary RTK engine
A: Base 1 starts in primary RTK engine
C: Actual handover was done, Base 1 in primary RTK engine stop and Base 2 will give positioning solution
Base 1
Base 2

Figure 15: Experimental scenario for handover process

5.6.1 Differentiate CNR

We used signal strength as one of the major features data. In GNSS signal, correlations are essential for receivers to synchronize with the incoming signal, generate GNSS observables data, and retrieve the navigation message. Therefore, satellite signal strength is related to the magnitude of the correlation peak. In general, the signal strength of the direct signals is stronger than that of the reflected or diffracted signals. Even in a single GNSS, the signal strength called Signal to Noise Ratio (SNR) is widely used to exclude the multipath signals. Similarly, the signal strength called Carrier to Noise Ratio (CNR) is used in the RTK system. However, only considering the CNR value of the rover receiver is not practical because in some cases, when the satellites are very close to the mask line, we could not differentiate easily through fisheye image only. At that moment, the signal strength of the NLOS signal may higher than the LOS signal due to random errors. Therefore, we used the signal strength difference between the base and the rover receiver to correct those signals. Here, differentiate CNR is calculated by subtracting rover receiver's CNR with base station's CNR.

5.6.2 Elevation Angle

The signal strength is also dependent on the satellite elevation angle. The elevation angle of the antenna of a ground receiver has a peak value when the satellite is just above it and gradually decreasing. After some time, it reaches an elevation angle of zero. In the GNSS signal, as the elevation angle increases, the received signal strength keeps increasing. The signal strength is maximum when the satellite is just above the antenna. Previous researchers, Sheng-Yi Li et al., derived the analytic and mathematical relation between elevation angle and signal strength in their research work [23]. Therefore, we used elevation angle as one of the crucial features.

5.6.3 LOS and NLOS State

We also need to know the LOS and NLOS signals to train our machine learning model with these features' matrix. Therefore, a Fisheye camera is used to take a fisheye view image, which has finally used to determine the satellite LOS and NLOS state. Here, to find a distinct state between LOS and NLOS signals, the following procedures are processed.

A. GNSS data collection

First of all, we need to collect data from the base and the rover receiver. Here, A rover receiver is placed in the multipath environment to receive both direct and multipath signals. A base station is placed in an open sky area, such as the base rover, for direct signals only.

B. Analysis of Fish-eye View Images

After the data collection process, we need to distinguish the signal state, 0 for NLOS and 1 for LOS signal, of the rover receiver to use as a training data set. Therefore, the fisheye camera is used to capture the sky view from the rover receiver. Fisheye image and position of rover station are shown in Fig. 12. To estimate satellites' orientation (both NLOS and LOS satellites), we need to make a mask that differentiates an obstacle in a fisheye image. Thus, we need to adjust the azimuth of an image with an antenna by using an open-source platform called RTKLIB [21]. After that, the process of masking is carried out with the corrected binarized image, as shown in Fig. 13. The red line is a mask line that differentiates the obstacle's clear sky view and presence. In our case, obstacles are buildings and trees.

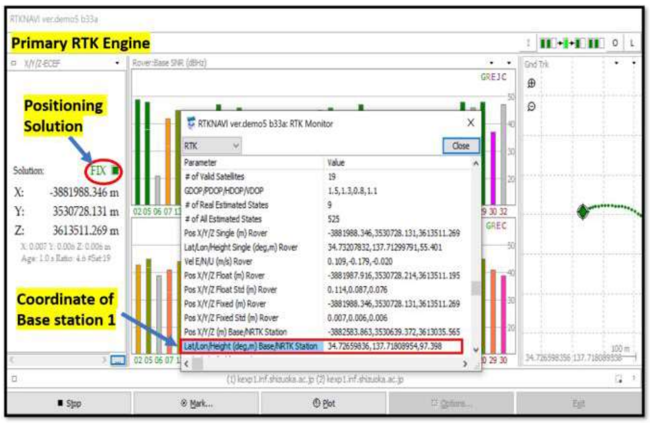
C. Extraction of NLOS features and Labeling

After that, the position of the satellite is estimated. The satellites found in a clear sky area are considered LOS satellites and marked as LOS signals. Similarly, the satellites which are found other than clear sky area are considered NLOS satellites.

Furthermore, the receiver's signal strength is varied by the satellite elevation angle due to differences in path loss and the antenna gain patterns. Therefore, the elevation angle is used as the next feature value while training the classifier.

5.6.4 Outline of Kernel-SVM

In the machine learning process, the data labeling process is essential to improve accuracy and efficiency. The main challenge is to decide which features data are more responsible and make the overall performance of a predictive model. Our model uses the matrix of features, i.e., elevation angle and differentiate CNR value, with dependent variable vector, i.e., the NLOS and LOS state by the fisheye image as a features data. Here, LOS and NLOS states are predefined target attributes used for training the algorithm that we will predict satellite states from future satellite data. We mainly focused on features data, training data ratio, and kernel tricks to make the correct label for learning data. First, we chose responsible features data, then we classified train and test set data in maximum performance ratio, i.e., 80% of our data are used for the training process, and the rest are for the test process. Finally, we trailed with different algorithms such as decision tree, naïve Bayes, K-nearest etc., algorithm; however, we found the best result in the Gaussian radial basis function (RBF) kernel trick technique. SVM uses a Gaussian radial basis function (RBF) kernel trick technique to transform the input data in this classifier approach.

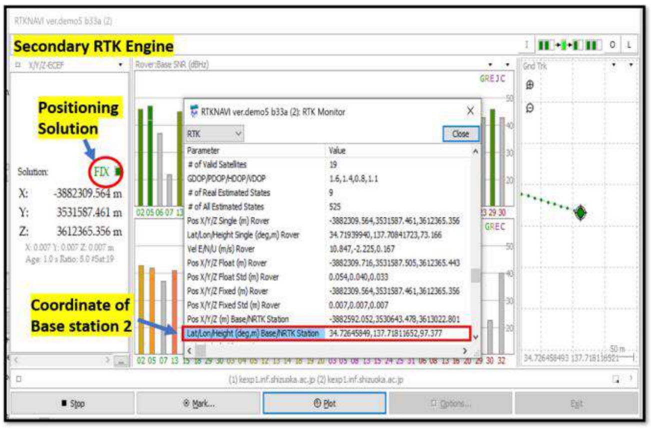


Solution: FIX
X: -3881988.346 m
Y: 3530728.131 m
Z: 3613511.269 m
X: 0.007 Y: 0.006 Z: 0.006 m
Age: 1.0 s Ratio: 4.6 #Sat:19

Parameter	Value
# of Valid Satellites	19
GDOP/PDOP/HDOP/VDOP	1.5,1.3,0.8,1.1
# of Real Estimated States	9
# of All Estimated States	525
Pos X/Y/Z Single (m) Rover	-3881988.346,3530728.131,3613511.269
Lat/Lon/Height Single (deg,m) Rover	34.73207832,137.71299791,55.401
Vel E/N/U (m/s) Rover	0.109,-0.179,-0.020
Pos X/Y/Z Float (m) Rover	-3881987.916,3530728.214,3613511.195
Pos X/Y/Z Float Std (m) Rover	0.114,0.087,0.076
Pos X/Y/Z Fixed (m) Rover	-3881988.346,3530728.131,3613511.269
Pos X/Y/Z Fixed Std (m) Rover	0.007,0.006,0.006
Pos X/Y/Z (m) Base/NRTK Station	-3882583.863,3530639.372,3613035.565
Lat/Lon/Height (deg,m) Base/NRTK Station	34.72659836,137.71808954,97.398

(1) kexp1.inf.shizuoka.ac.jp (2) kexp1.inf.shizuoka.ac.jp

Figure 16: Result in primary RTK engine



RTKNAVI ver.demo5 b33a (2): RTK Monitor

RTK

Parameter	Value
# of Valid Satellites	19
GDOP/PDOP/HDOP/VDOP	1.6,1.4,0.8,1.1
# of Real Estimated States	9
# of All Estimated States	525
Pos X/Y/Z Single (m) Rover	-3882309.564,3531587.461,3612365.356
Lat/Lon/Height Single (deg,m) Rover	34.71939940,137.70841723,73.166
Vel E/N/U (m/s) Rover	10.847,-2.225,0.167
Pos X/Y/Z Float (m) Rover	-3882309.716,3531587.505,3612365.443
Pos X/Y/Z Float Std (m) Rover	0.054,0.040,0.033
Pos X/Y/Z Fixed (m) Rover	-3882309.564,3531587.461,3612365.356
Pos X/Y/Z Fixed Std (m) Rover	0.007,0.007,0.007
Pos X/Y/Z (m) Base/NRTK Station	-3882592.052,3530643.478,3613022.801
Lat/Lon/Height (deg,m) Base/NRTK Station	34.72645849,137.71811652,97.377

(1) kexp1.inf.shizuoka.ac.jp (2) kexp1.inf.shizuoka.ac.jp

Figure 17: Results in secondary RTK engine

As a result, the optimal bounds of the target classes are obtained to classify nonlinear data, which we consider the right label for our system. Also, feature scaling is used to standardize datasets. The detailed working environments and results are described in the next section.

6 RESULTS AND DISCUSSIONS

To test the proposed system's performance, the base station was set up and tested in the practical field. A set of task actions and obtaining results are described in the following sub-section.

6.1 Discussion of Monitoring System and Its Performance

Base stations are equipped with digital sensor network systems that were deployed throughout all base stations. We have considered three significant problems in the mountain and snowy regions that affect the base station conditions. These problems are (1) snow accumulation, (2) natural disasters, and (3) power outage problems. Various cost-effective and easily applicable sensors are used to address these problems concretely. To check the monitoring system's performance, we have done our experiment in the dense snowy placed named Wakkanai, Hokkaido, where there was heavy snowfall, dense fog, and cold weather. We have done that experiment to check the performance of the prototype on the snowy area that consists of multiple sensors with a cellular RTK receiver. The main goals of this experiment are (1) remotely monitor the base stations' conditions (such as snow level on antenna, antenna's orientation, power supply status, etc.) in real-time, (2) send the satellite signal of the base receiver to the server system using a cellular network.

In our system, the microcontroller works with a sketch file that creates a data feed from the ultrasonic sensor, accelerometer sensor, and voltage sensor. Here, an ultrasonic sensor provides snow height levels. Similarly, the accelerometer uses for detecting the orientation, gesture, and motion of the antenna. Those are the ground data that we used to monitor the base station. Similarly, the GNSS receiver, which is supposed to work as a base station, is embedded with a cellular LTE model to send the satellite signal to the server using a cellular network. Those sensors' data are sent to web-based applications and displays on a web page correctly.

These data are received as byte-type data. Therefore, byte type data is decoded in Harvest's GUI and displayed as a column of primarily processed data. These data can be saved to the local file or own server system. Finally, those data are displayed in graphical form were shown in Fig. 14. Also, the sensors' data are used to detect the healthy or unhealthy state of the base station.

As a result, the optimum base station could be assigned to a rover receiver. From this experiment, we checked hardware and software performance regarding sensors-based base stations. We concluded some bits of knowledge such as sensors accuracy and its real-time performance.

First of all, our monitoring system in the snowy region is perfectly worked. We were able to collect data and monitor it in real-time through the sensors. Secondly, the wireless data collection using the cellular network to server system is smoothly done for both cases, sensors and GNSS signal.

6.2 Discussion of Seamless Handover and Its Performance

In order to make a robust RTK infrastructure, we need to address the networking mechanism of the base station. Therefore, we proposed a seamless handover mechanism between two or more base stations in network cellular RTK. After the monitoring system, proposed algorithm, its practical use, and performance evaluation were done with static and kinematic rover receivers, as shown in Fig. 15.

We tested the handover process, system's functionality, and performance concerning the RTK accuracy. To test the performance of the proposed algorithm, we have done our field test experiment in Hamamatsu, Japan, where the surrounding environment contains tall buildings and dense traffic. In our experimental kinematic scenario, two static base stations and one kinematic rover receiver are used. For this experiment, the base station assignment process was computed manually through a server system.

At first, we have connected all base stations in the control server system where available base stations are displayed with their unique identity. As every base station consists of a cellular module as an internet provider, the unique IMSI (International Mobile Subscriber Identity) number is used for base station identity. Basically, a base station is assigned to the primary RTK engine.

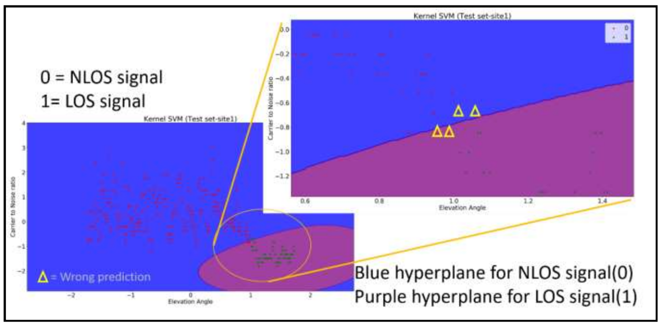


Figure 18: Experimental results of site 1

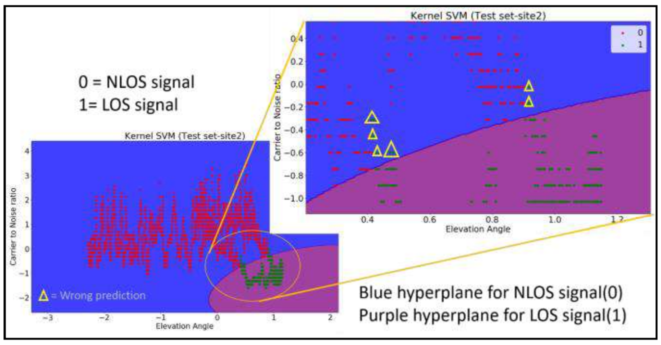


Figure 19: Experimental results of site 2

This selected base station started a connection to the rover receiver and got a fixed solution after a few seconds, as shown in Fig. 16. After getting a fixed solution in the RTK engine, the positioning information is ready to use by applications. At the time of handover, the new base station is assigned through the server system to the user end. The communication link is established to the secondary RTK engine because the first base station is still operating in the primary end.

In this case, the secondary RTK engine receives correctional information from a base station and provides the fixed solution, as shown in Fig. 17. After getting a fixed solution in the secondary RTK engine, the primarily selected RTK engine went to ideal mode. The secondary RTK engine starts its positioning solution and becomes the default RTK engine. An open-source program package of RTKLIB library with a program package is used as RTK engines.

In our experiment, we used two base stations at a fixed location and near each other. However, those two base stations might not be consistent; thus, the precise positioning solution could be different in the primary RTK engine and the secondary one. Therefore, to make a seamless handover on that condition, we need to calculate the positioning error in both RTK-engine, such that at a point, the positioning error is conceding and become minimum; thus, handover could be done continuously. For instance, if we plot the positioning solution of the primary RTK-engine with the positioning solution of the secondary RTK-engine, the difference in positioning solution can be calculated and vice versa.

Table 2: Processing components

Name	Description
Features' data	Differential CNR, elevation angle, azimuth angle, distinct state of NLOS(0) and LOS(1)
Programming language	Python (V. 3.7.7)
Libraries	NumPy (v. 1.16.4), Matplotlib (v. 3.1.0), and Pandas (v. 0.24.2)
Test set value	0.20
kernel	Radial basis function (RBF)
Computer environment	Windows 10, 16GB RAM, i7-8550

From this experiment and result analysis, we concluded that the seamless handover is possible with two RTK engines in a simply smart way. If we combine both the monitoring and seamless handover mechanism, this development may upgrade the performance of network RTK to a new level.

6.3 Multipath Detection and Its Performance

The classification results of the kernel SVM-based classifier are shown in Fig. 18 and Fig. 19. We have chosen those satellites for both experiment areas, which had changed their position from being LOS to NLOS or vice versa. Observation data from the rover receiver and base receiver is converted into an excel file to prepare a dataset for training. Data and time, satellite number, azimuth, elevation angle, differential CNR and the distinction state for LOS and NLOS signal are used as the training dataset. Here, the distinction set is marked 0 for NLOS and 1 for the LOS satellite. The processing components are shown in Table 2. To split the dataset into the train and test set, we used a test set value of 0.20 (i.e., 80% sample data are used for the training set, and 20% existing data is set for the test set).

After that, feature scaling is applied to normalize the features data (i.e., independent variable) with a particular range and also helps in speeding up the calculations in an algorithm. After that, we applied the kernel SVM model is created and applied to the training data set. The experimental results in Site 1 and Site 2 are described in the following section.

A. Experimental results in Site 1

After sufficiently train our kernel SVM model, the test experiment is done for an hour of data (excluding observation data for the same elevation angle). The result of the classifier is shown in Fig. 18. To analyze the result, we used a graphical view and confusion Metrix. We found that 98% of NLOS data and 90% of LOS signals are predicted correctly.

B. Experimental results in Site 2

Similarly, the experiment is carried out on Site 2. For this experiment, three hours of data are used to train the model. The experimental result is quite improved while using all experiment data (i.e., every CNR value is used regardless of the same elevation angle) shown in Fig. 19. After analyzing the results through the confusion matrix, we found that 99% NLOS and 97% LOS signals are predicted correctly. Also, multiple experiments were conducted to analyze the accuracy of the prediction. On average, 98% NLOS and 95% LOS signals are predicted accurately.

7 CONCLUSION

This research has presented the novel perspective to build a reliable RTK infrastructure that is applicable in snowy/mountain and urban areas. We proposed three new components into the system to address the issues of those areas, especially targeting the reliability and higher positioning accuracy for movable objects.

Firstly, we practically implemented a mechanism of detecting an unhealthy base station in order to monitor its availability. We have practically checked the performance of sensors embedded cellular RTK base station in Hokkaido, Japan. We used different sensors in the base site to monitor its status through the internet. Those data are used for web-based monitoring purposes, such that a base station's state (i.e., healthy, or unhealthy) is easily noticeable on the user side. Experimental results confirmed that the base station's availability is regularly ensured in real-time.

Secondly, we proposed an algorithm to assign the optimum base station from multiple base stations in order to provide continuous and high accurate positioning for a portable rover receiver. The actual field experiment was done in a network RTK system to explore a seamless handover mechanism in Hamamatsu, Japan. The concept of assigning optimal base station is based on two factors: base-rover distance and sensors value. For the seamless handover, we proposed the usage of two RTK engines on the user side, such that the positioning accuracy was maintained at centimeter-level before and after handover.

Thirdly, the multipath detection model is proposed as a final component of our robust infrastructure. A new method of distinguishing the LOS and the NLOS multipath signals was developed to improve RTK-GNSS positioning accuracy in urban environments. A classifier based on the kernel SVM technique is proposed using receiver signal strength and its elevation angle. As a result, around 98% of the NLOS multipath signals and 95% of the LOS signals were correctly classified. From the experimental results, we have confirmed that the proposed technique can effectively predict future CNR and helps to mitigate multipath signals.

By combining these three components' results, we have confirmed that our approaches significantly impact building a robust RTK-GNSS infrastructure for continuous and precise positioning. Also, we have verified that continuous correctional signals and precise positioning in challenging environments can be achieved from our method for a movable object.

ACKNOWLEDGMENT

This research work was supported by JSPS KAKENHI Grant Number JP17H01731.

REFERENCES

[1] P. P. Das and S. Nakamura, "Analysis of GPS Single Point Positioning and Software Development," *Iject*, vol. 7, no. 1, pp. 34–40 (2016).

[2] U. Pudasaini, "Drones Optimized Therapy System (DrOTS): Use of Drones for Tuberculosis Diagnosis in Nepal," *Int. J. Hum. Heal. Sci.*, vol. Sup. Issue, p. 14 (2019).

[3] D. Jonathan, "Trimble: Mapping Everest," *Int. Work. Meas. Height Sagarmatha (Mt. Everest) GNSS Appl.*, pp.1-14 (2017).

[4] T. Yoshihara, H. Motoyoshi, T. Sato, S. Yamaguchi, and S. Saito, "GAST-D integrity risks of snow accumulation on GBAS reference antennas and multipath effects due to snow-surface reflection," in *International Technical Meeting*, pp.112–120 (2013).

[5] H. He *et al.*, "VRS technology based on multi-base station network and its error analysis," in *Joint Conference of SPAWDA 2009 and 2009 China Symposium on Frequency Control Technology*, pp. 288–292 (2009).

[6] G. Zhang and L. Hsu, "Adaptive GNSS/INS integration based on supervised machine learning approach," in *International Symposium on GNSS*, pp. 1–26 (2017).

[7] B. Rokaha, B. P. Gautam, and T. Kitani, "Building a Reliable and Cost-Effective RTK-GNSS Infrastructure for Precise Positioning of IoT Applications," in *12th International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pp.1–4 (2019).

[8] J. Jackson, B. Davis, and D. Gebre-Egziabher, "A performance assessment of low-cost RTK GNSS receivers," in *IEEE/ION Position, Location and Navigation Symposium (PLANS),* pp.642–649 (2018).

[9] Y. Du, G. Huang, Q. Zhang, Y. Gao, and Y. Gao, "A new asynchronous RTK method to mitigate base station observation outages," *Sensors*, vol. 19, no. 15, p. 3376 (2019).

[10] Y. Quan, X. Meng, L. Yang, and S. Stephenson, "Network RTK GNSS Quality Assessment," in *European Navigation Conference*, pp. 1–13 (2013).

[11] S. A. H. Sulaiman, M. A. Mustafar, T. A. T. Ali, M. A. Abbas, and H. Z. M. Shafri, "Practical accuracy of VRS RTK outside the Malaysian Real Time Kinematic Network (MyRTKnet)," in *5th International Colloquium on Signal Processing and Its Applications*, pp. 395–399 (2009).

[12] T. Suzuki and N. Kubo, "N-LOS GNSS signal detection using fish-eye camera for vehicle navigation in urban environments," *27th Int. Tech. Meet. Satell. Div. Inst. Navig. ION GNSS 2014*, vol. 3, pp. 1897–1906 (2014).

[13] D. Maier and A. Kleiner, "Improved GPS sensor model for mobile robots in urban terrain," in *EEE International Conference on Robotics and Automation*, pp. 4385–4390 (2010).

[14] A. Sahni, K. Nakaaki, and T. Kitani, "Enhancing 3D Maps using RTK-GNSS to help improve the Position Solution in Urban Areas," *Proc. Int. Work. Informatics*, pp. 63–68 (2019).

[15] B. Rokaha, B. P. Gautam, and T. Kitani, "Enhancing RTK-GNSS Infrastructure in Snowy and Mountain Region Through Rule-Based Base Station Assignment Approach Graduate School of Integrated Science and Technology , Shizuoka University , Japan Department of Economic Informatics , Kanazawa Gakuen Univ," in *International Workshop on Informatics (IWIN2020)*, pp. 101–108 (2020).

[16] Y. Quan, L. Lau, G. W. Roberts, X. Meng, and C. Zhang, "Convolutional neural network based

multipath detection method for static and kinematic GPS high precision positioning," *Remote Sens.*, vol. 10, no. 12 (2018).

[17] L. T. Hsu, "GNSS multipath detection using a machine learning approach," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 1–6 (2018).

[18] Faire Shenzhen, "Wio LTE Cat.1," *the first maker faire in China*, 2012. [Online]. Available: http://wiki.seeedstudio.com/Wio_LTE_Cat.1/.

[19] Amazon AWS veterans and Telco engineers, "Soracom: IoT cellular Connectivity Provider." [Online]. Available: https://www.soracom.io/.

[20] B. Hofmann-Wellenhof, *Elementary Mathematical Models for GNSS Positioning* (2018).

[21] T. Takasu, "RTKLIB ver. 2.4.2 Manual. Tokyo University of Marine Science and Technology; Tokyo, Japan" (2013).

[22] T. Suzuki, M. Kitamura, Y. Amano, and T. Hashizume, "Multipath mitigation using omnidirectional infrared camera for tightly coupled GPS/INS integration in urban environments," *24th Int. Tech. Meet. Satell. Div. Inst. Navig. 2011, ION GNSS 2011*, vol. 4, pp. 2914–2922 (2011).

[23] L. E. O. S. Systems, S. Li, and C. H. Liu, "An Analytical Model to Predict the Probability Density Function of Elevation Angles for," vol. 6, no. 4, pp. 138–140 (2002).

(Received November 11, 2020)
(Accepted September 6, 2021)

Bhagawan Rokaha received his bachelor's degree in Electronics and Communication Eng. from the Department of Eng., Kathmandu Eng. College, Tribhuvan University. He has completed his Master's degree in computer science from Shizuoka University and joined Mitsubishi Electric in April 2021. He is currently a member of Kitani laboratory, Shizuoka University, conducting the research on improvement of RTK-GNSS and the other GPS-related subjects. He is also planning to do a Ph.D. in the near future. His research interest includes Intelligent Transport System, Cellular Network-RTK, Next Generation Train Control and Monitoring System (NG-TCMS), and Automotive Software Architectures. He is a member of IPNTJ, and Sakura Science Club.

Bishnu Prasad Gautam received his bachelor's degree from Wakkanai Hokusei Gakuen University. He received his Master's degree and Ph.D. in Computer Engineering from Shinshu University. He is currently working as Assoc. Prof. at Kanazawa Gakuin University, and he is a member of IEEE, IPSJ, and IAENG. His current research interest includes Sustainable Computing, Network Architecture, Network Security, and IoT.

Tomoya Kitani was born in 1979. He received his associate degree in Engineering from Nara National College of Technology in 2000, Master's degree in Information Science and Technology, and Ph.D. from Osaka University in 2004 and 2006, respectively. He is currently an associate professor at the College of Informatics, Academic Institute, Shizuoka University. His research interests include Intelligent Transport Systems for Motorcycles, Global Navigation Satellite Systems, Computer Networks, and Embedded Systems. He joined the Information Processing Society of Japan (IPSJ) in 2003. He is a committee of the Special Interest Group of Intelligent Transport Systems and Smart Community (SIGITS) of IPSJ. He is also a committee of the Motorcycle Dynamics Division, Society of Automotive Engineers of Japan (JSAE). He is a member of the IEEE, IPSJ, IEICE, and JSAE.

Regular Paper

Evaluation of Autonomous Control of Server Relocation for Fog Computing Systems

Kouki Kamada[†], Hiroshi Inamura[‡], Yoshitaka Nakamura[‡]

[†]Graduate School of Systems Information Science, Future University Hakodate, Hakodate, Japan
[‡]School of Systems Information Science, Future University Hakodate, Hakodate, Japan

Abstract - Fog computing, which extends the paradigm of cloud computing to the edge of networking, has been proposed, and its research has been active. In the field of networking, research on Content Centric Networks (CCN) has been conducted. CCN have been shown to be able to handle cached content naturally within the network, reducing traffic and latency. However, in today's Internet, dynamic content with dynamic services is indispensable. A system that capable of handling dynamic services is desired by incorporating the way of handling computational resources in fog computing into CCN. In this paper, an autonomous control scheme of server relocation for fog computing systems is proposed. We study the optimizing quality of service by allowing services running on the network to be dynamically relocated. Our ns-3 simulations show the fairness between users achieved and the reduction of the average response time on non-uniform computer resources with three use cases.

Keywords: Contents Centric Network, Fog Computing, In-Network Caching, Server Relocation

1 INTRODUCTION

The number of IoT devices, which is 27.4 billion as of 2017, is expected to increase to about 40 billion by 2020[1]. For these large volumes of data generated by IoT devices, processing-intensive architectures such as cloud computing do not take advantage of the processing power of the edge and the latency from the point of data generation to the remote data centers cannot be ignored. Therefore, fog computing, which extends the paradigm of cloud computing to the edge of the network, has been proposed and actively studied[2].

In the field of networking, research on CCN (Content Centric Networks) such as NDN (Named Data Networking) has been carried out instead of the conventional IP address-based architecture[3]. It has been shown that CCN can naturally handle cached content in the network by using location- independent content as an identifier, which capable of reducing traffic and latency.

We proposed an autonomous control of server relocation for fog computing systems[4]. In addition, we improved the autonomous control of server relocation to transfer services on the fog network where the processing capacity is heterogeneous, so that the service transfer is commensurate with the required processing capacity[5].

In this paper, to optimize end-user QoS, we control server transfers in a fog computing environment to achieve both shortening of the average response time and fairness between users. For this purpose, we set up three use cases to examine the fairness between users in uniform computer resources, the realization of shortening the average response time in heterogeneous computer resources, and the realization of fairness between users and shortening the average response time in heterogeneous computer resources, respectively.

2 RELATED WORKS

2.1 Fog Computing

In fog computing, the delay time for execution is reduced by selecting and transporting the points necessary for the execution process. For example, in wireless sensor and actuator networking, simple processing should be performed at intermediate nodes, such as fog nodes, before the data collected by sensor nodes are transferred to the cloud. The appropriate placement of the processes at the intermediate nodes improves the command response time for actuation, compared with it is executed at cloud. There is another technique called code-offloading[6]–[9]. Code-offloading is the optimal use of resource-constrained mobile devices. This technology aims to improve the energy efficiency and execution speed of applications. In a mobile application, a part of the application code will be offloaded to the node on fog that has more computational resources to execute the code, rather than running on mobile devices. With the decision, it is possible to save resources such as batteries in mobile devices. In fog computing, the optimal allocation of computational resources is a focus, but there has been no discussion on the optimal placement of content.

Code Bubbling Offload System F. Berg et al.[9] focused on the fact that only two or relatively simple, restrictive system models consisting of three devices have been considered in previous code-offloading techniques. They argue that n-tier architectures become interrelated when multiple classes of various highly distributed resources take advantage of code offloading. For example, the smartwatch (tier 1) connects to the smartphone (tier 2) via Bluetooth. Its smartphone connects via Wi-Fi to a car (tier 3) connected to an edge server (tier 4). In addition, an edge server is connected to a server in a data center (tier 5) across 4G mobile communications and fixed networks. Heterogeneous devices in such an n-tier system differ greatly in terms of energy and computational resources. In this example, they have tiers 1 to 5, but the complexity of the tiers will increase typically from very limited to virtually unlimited. Therefore, they targeted an n-tier environment containing highly distributed heterogeneous resources with different performance characteristics and cost relationships code offload system proposed CoBOS (Code Bubbling Offload System). This proposal includes a concept called code bubbling.

Code bubbling[9] moves code dynamically and adaptively towards more powerful and more distant tiers, enabling an efficient and scalable code-offloading in n-tier environment. CoBOS decreases the energy consumption by 77% and the execution time by 83% for code-offloading in an n-tier environment.

This research aims to optimize the execution point in mobile applications by offloading the code components of the application to a stronger tier at runtime. Therefore, we thought that we could optimize the execution point of the services by considering an architecture that executes the services running in the cloud closer to the user.

2.2 CCN

V. Jacobson et al.[3] proposed a CCN that does not use the traditional IP addressing architecture and Two types of CCN messages, Interest and Data, are used in the CCN communication. This is done by a protocol that is based on the Messages can be sent and received through the FIB(Forwarding Information Base), CS(Content Store), PIT (Pending Interest Table) to send the data back to the requester, three main data structures are used. Using these data structures, CCN exchange messages between Interest and Data. The result retains the simplicity and scalability of IP but offers much better security, delivery efficiency, and disruption tolerance. In this way, CCN put content closer to the user, which allows static contents to be disseminated. However, to treat the running system, we need to care the internal state to continue the process, it is not possible to handle in the same way to provide dynamic content and services.

There is research on cache efficiency in CCN and how to route Interest packets efficiently[10]. These studies have been discussing the treatment of static content and how efficiently distributed content can be considered as transparent, and there is no discussion on how dynamic services can be distributed and deployed on the network.

2.2.1 Service over Content-Centric Routing

In host-oriented communication, technologies such as replication of content and services, caching services, load balancing, and routing of content requests have been enabled by the introduction of applications (e.g., CDN and P2P), but the cost of managing and operating the network is high. Two paradigms, CCN and SCN (Service Centric Network)[11], are used to solve these problems.

If content and services are treated separately, it is not possible to show the relationship between content and services. In practice, services generate new content or perform various functions on existing content, just as some services perform various functions on existing content, but despite the deep relationship between content and services, there is a problem of creating a content-service divide and the gap between CCN and SCN needs to be bridged. Therefore, S. Shanbhag et al. proposed Services over Content-Centric Routing (SoCCeR)[10]. The SoCCeR has added a new ant colony optimization[12] based service routing control layer on top of the CCN to manipulate the routing table for Interest messages. Without affecting the content request and retrieval capabilities of the CCN, they show that they can add SCN capabilities, service requests selectively to service instances with lighter loads and is highly responsive to network and service state changes. However, there is no discussion of how to deploy distributed service instances on the network.

2.2.2 PiGeon

A platform for service orchestration is proposed that addresses the challenges of CCNs in delivering dynamic content and services. M. Selimi et al. propose a Docker container-based PiGeon platform that extends CCN to seamlessly deliver, cache, and deploy at the network edge[13]. PiGeon consists of three types of components: a service controller that monitors the network topology and resource consumption of nodes for service deployment, a service execution gateway for executing service instances at the network edge, and a forwarding node that forwards user requests to nearby caches and other content. The service provider uploads the service to the service controller as a prelude, and the service controller uses the decision engine to decide when and where to place the service in a service algorithm based on the monitoring data.

The PiGeon platform has a service controller as a single point of failure. If a service controller is isolated from the network, the service is not relocated. It is necessary to register the service execution gateway with the service controller in advance, and the monitoring information must be sent periodically from the service execution gateway to the service controller. In medium-sized networks, such as the 80-node experiment in their previous work[14] and the 32-node network in their study, the effect is more than the traffic of monitoring information, but in large networks, the traffic of monitoring information is rapidly increasing and the scalability to the network is low.

2.2.3 Necessity of Fog Computing and CCN Integration

In order to solve the problems we have seen from the research mentioned so far, it would be useful to consider an architecture that allows us to control the deployment of services running in the cloud and dynamically redeploy them as needed. For example, it may be possible to optimize the point of execution of services by running them closer to the user.

Since the CCN is based on the idea of replacing the current TCP / IP with the CCN, there are several discussions on static content caching schemes. However, in today's Internet, which is created by real-world TCP / IP, dynamic content with dynamic services is essential. For example, there is a web page that authenticates the user and displays information appropriate for the user. We wondered if a static content caching scheme is not enough to replace the current Internet with a CCN because of the large amount of these dynamic contents. In the study of fog computing, there is little discussion on the issue of how to place data on a fog network. Therefore, we believe that the problems in the research fields of fog computing and CCN can be mutually resolved by incorporating the way computational resources are handled in the CCN, as introduced in 2.1, into the CCN.

3 CHALLENGE

We consider optimizing quality of service by allowing services running on the network to be dynamically relocated. In this paper, we focus on response time as seen by the client as quality of service. We assume that the response time is expressed as the sum of the network transmission delay and the processing time at the server. When considering the response time, we need to optimize the system in two ways: fairness of the response time among clients and minimization of the processing time. An example is shown and discussed below.

3.1 Use Case that Require Fairness in Delay Times

There is a need for autonomous resource allocation that satisfies the fairness of delay times for participants. For example, in an Internet conferencing system, media quality for all participants may not be maintained if the server is in a single location for clients distributed in different locations on the network with different latency. Therefore, there is a need for autonomous resource allocation that satisfies the equity of delay time for participants.

3.2 Use Case that Require Minimize Processing Time

We assume that the system is available with a dedicated purpose-specific unit, such as TPU (Tensor Processing Unit)[15], to enable machine learning at the edge. Installing a purpose-specific unit for every node may be cost-prohibitive. Therefore, in order to utilize the sparsely placed purpose-specific unit, it is necessary to discover the node with the unit from the topology and to transfer the execution point to the node.

3.3 Assumptions and Requirements for Autonomous Control of Server Relocation System

We need a system that aims at fairness in average response time and shortening of service execution time among users simultaneously. We define the service response time which is the sum of the network latency between the client and server and the processing time of the service.

In addition, the system should reduce the average response time on non-uniform computer resources. For machine learning applications, where the execution time varies greatly depending on the processing performance, the processing time of the service becomes a bottleneck due to the processing performance. Therefore, by monitoring the processing performance of each node and transferring services based on the predicted service response time, services can be transferred to nodes with appropriate processing capacity.

In this system, we assume that the client has a fixed position in the network because it communicates directly with the sensor and user. On the other hand, since servers providing services are arbitrarily located in the fog/cloud, we assume that it is possible to relocate the server by transferring the state of service execution to obtain the necessary resources to execute a process.

4 PROPOSAL FOR AUTONOMOUS CONTROL OF SERVER RELOCATION SYSTEM

In order to optimize QoS for end users, we propose the autonomous control of server relocation for fog computing to reduce both the average response time and the fairness between users.

4.1 The Functions Required by the System

This system collects information about the computing environment of the surrounding nodes, searches for a candidate node that can minimize the service response time, and transfers the server to the selected node.

In searching for candidate nodes, it is not realistic to assume global knowledge across different computing environments such as fog and cloud. As a reasonable scope of search, we assume a routing topology of CCN interest messages when the server is regarded as a resource. It collects PCEL (Available Processing Capacity and Estimated Latency) management information for each node on the path where a message arrives and determines the server transfer based on this information.

Compared with the related studies mentioned in Section 2.2.2, the advantages of our system are as follows. From this system collects information via service communication and the service execution point makes the relocation decision, the system has high availability, including the fact that the single point of failure for each service moves through the fog network and does not affect other services. By this system loads monitoring information on a node via service communication, it is not necessary to register the system at a single location simply by installing it on the node. Also, since the monitoring information is superimposed on the communication of the service, there is no increase in traffic to monitor the network status and the network is scalable.

The following sections describe the main components of the proposal, the estimation of the service processing time, the collection of PCEL information on the message arrival route, and the algorithm for selecting candidate nodes.

Service processing time

C which is the processing power of a node and L which is the amount of processing of the requested service executed by the node are represented as a two-dimensional vector to decompose the processing power of the node into CPU C

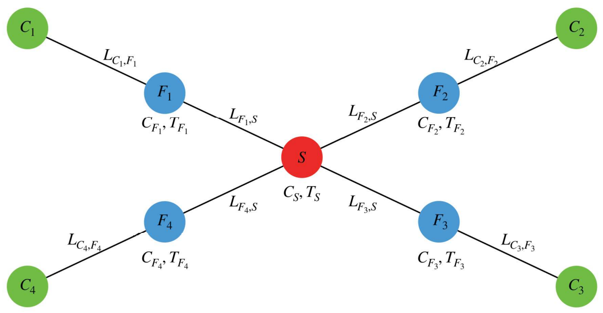


Figure 1: System Configuration Example

and the purpose-specific unit T, respectively. The processing capacity of a fog node is represented by (C_C, C_T), and the amount of processing required for service execution is defined as (L_C, L_T). Based on these processing capacity and processing volume, the service processing time T_{est} is defined as follows.

$$T_{est}(C_C, C_T, L_C, L_T, \alpha) = \begin{cases} \frac{1}{C_C}(L_C + \frac{L_T}{\alpha}) & (C_T = 0) \\ \max(\frac{L_C}{C_C}, \frac{L_T}{C_T}) & (otherwise) \end{cases}$$

However, the CPU can perform the processing required for the purpose specific unit. The α that express ratio is set to 5 for use in later evaluation.

Collecting PCEL information on the message arrival path

In order to treat all the nodes on the path from each client node to the server as candidates for transfer, it is necessary to collect information on the delays between the links traversed and the processing capacity of the nodes traversed. These are called the route PCEL information. In our system, PCEL information is added to the request message at the node that passes by the time the request message reaches the server node, and it is transmitted to the server.

For example, when our system is used as shown in Fig. 1, the following parameters are added to the request message of C_1. In Fig. 1, S is the server node, F is the fog node, and C is the client node.

- L_{C_1,F_1}, which is the communication delay of the link from C_1 to fog node F when a request message from client node C_1 goes through each fog node
- $L_{F_1,S}$, which is communication delay of the link from server node S to F_1
- C_{F_1}, which is the CPU processing power of F_1
- T_{F_1}, which is the processing power of the purpose-specific unit of F_1

This added PCEL information can be acquired by S from the request message. Similarly, S is able to obtain information on the route to and from all clients from the PCEL information attached to the request messages from C_1 to C_4, which are all participating client nodes.

Candidate node selection algorithm

The server selects candidate nodes for transfer from the PCEL information appended to the request message received from the client By using the algorithm shown in Algorithm 1. Algorithm 1 calculates the average and standard deviation of the service response time of the participating clients based on the information that the server shown in Fig. 1 can obtain from the request message. The value is the sum of the service response time multiplied by $1 - R_{std}$ and the standard deviation multiplied by R_{std}, based on R_{std}, which specifies how much importance is placed on the fairness between clients and users. Then, we find the node whose evaluation value is at a minimum.

Algorithm 1 Find Candidate Node

```
Require: $L_{All}$:List of $L$ on the Path
Require: $L_{(F_i,C_j)}$:List of $L$ on the Path between $F_i$ to $C_j$
Require: $F_{All}$:List of $F$ on the Path
Require: $C_{All}$:Clients connected to the service
Require: $L_C$:CPU processing capacity during service execution
Require: $L_T$:Purpose specific unit throughput during service execution
Require: $node.C_C$: CPU processing capacity of the node
Require: $node.C_T$: Purpose specific unit capacity of the node
Require: $R_{std}$:Ratio of importance to the standard deviation
Require: $S_{F_i,C_{All}}$:Standard deviation of service response time between $F_i$ to $C_{All}$
Require: $\alpha$:Coefficient that represents the ratio when the CPU can handle the amount of processing of the target-specific unit
Ensure: $MinNode$ is candidate node
  $MinCost \leftarrow \infty$
  for all $node$ in $F_{All}$ do
    for all $client$ in $C_{All}$ do
      $Latency_{node,client} \leftarrow \sum L_{node,client} * 2$
      $Latency_{sum} \leftarrow Latency_{sum} + Latency_{node,client}$
    end for
    $Latency_{ave} \leftarrow \frac{Latency_{sum}}{C_{All}.length}$
    $Latency_{var} \leftarrow \frac{1}{C_{All}.length} \sum_{i=1}^{C_{All}.length} (Latency_{Ave} - Latency_{node,C_i})^2$
    $ServiceRTT \leftarrow T_{est}(node.C_C, node.C_T, L_C, L_T, \alpha) + Latency_{ave}$
    $R_{RTT} \leftarrow 1 - R_{std}$
    $COST \leftarrow ServiceRTT * R_{RTT} + S_{node,C_{All}} * R_{std}$
    if $MinCost > Cost$ then
      $MinCost \leftarrow Cost$
      $MinNode \leftarrow node$
    end if
  end for
  return $MinNode$
```

4.2 Functions of the Node

The movement of the proposed system is shown in Fig. 2. This system is assumed to operate at the session layer of all

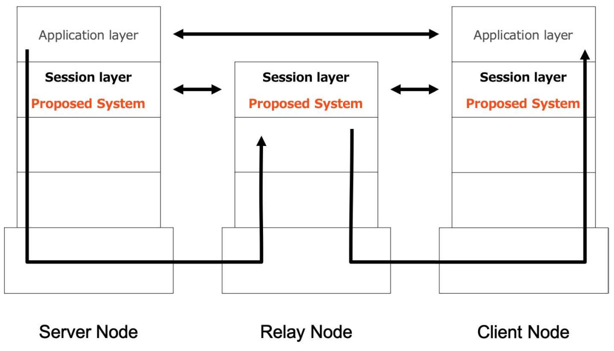


Figure 2: Autonomous Control of Server Relocation System

participating fog nodes. It is preferable that the change in the point of service execution is done transparently to the client and server. In order to implement the collection of PCEL information and transparent addition to the request message, it is convenient to work at the session layer in the seven-layer model. Since management actions such as service transport are operated by resources below the transport layer, they must be located in the upper layer where they are visible, and the session layer is lower than the application layer where clients and servers are running. By relaying communication between server nodes and client nodes at the application layer, the server relocation system at the session layer of server nodes, relay nodes and client nodes share information about resources available at each node. Based on the collected information, it realizes the selection of service execution points and resource allocation.

The proposed system consists of three types of nodes. The proposed system consists of multiple connections: a cloud node that has the contents necessary for service execution and has high processing power, a middle-class fog node that has medium processing power and can communicate with end devices with relatively low latency, and a client node that is an end device such as a smartphone that participates in the server. Based on the client-server communication model, cloud nodes and fog nodes play the role of servers, and leaf nodes play the role of clients. The server monitors the communication status of participating clients and decides whether it should autonomously play the role of the server or delegate the role of the server to other fog nodes based on the communication status. The delegated fog node takes over the role of the server. In this way, we try to optimize the server relocation of the server's role. This is an attempt to reduce the service response time.

There is each fog node has an in-network resource monitoring function, a candidate selection function and a service transfer function. In this section, each function is explained. The autonomous control of server relocation system at each node operates at the session layer to superimpose management information on the communication messages between the server and the client to realize the resource monitoring function in the network. At the same time, it constantly monitors changes in the resources in the network, and if a change is observed, it executes the candidate selection function and, if it is judged to be necessary, it executes the service transfer function to the selected node to optimize the server relocation.

4.2.1 Network Resource Monitoring Function

The in-network resource monitoring function monitors the currently used network resources and collects information to decide whether or not to transfer the service. Specifically, this function monitors and records network information, such as the delay between client nodes participating in the service and the network information, as well as the CPU and memory resources of the fog nodes themselves, while the fog nodes on which this system is installed are deploying the service. As described in the previous section, the PCEL(available Processing Capacity and Estimated Latency) management information described below is included in the normal communication message exchange between the client and server, and information on relay nodes that exist on the route between the client and server is collected.

4.2.2 Candidate Selection Function

The candidate selection function is called when there is a change in the information collected by the in-network resource monitoring function. This function determines whether the service should be transferred. In addition, if it is to be transferred, it will be based on the information collected by the in-network resource monitoring function. Select the appropriate candidate nodes. After the consignee is determined by this function, the server is consigned by the service transfer function.

4.2.3 Service Transfer Function

This function transfers services to the nodes selected by the candidate selection function. The node that is the current server is the candidate server for the new candidate fog node specified by the candidate selection function. The information about the data required for the service is sent with a message informing the user that the service is being entrusted. The server candidate fog nodes are now in the process of gathering all the necessary data and are ready to take on the server. It sends a message informing the user that it can be entrusted to a fog node that is a server of When the fog node, which is currently the server, receives it, it sends a message to a client node that has joined the server It sends a message notifying the new destination to In this way, the service is transferred.

4.2.4 Overall Flow to Optimize Server Placement

We summarize the optimization process explained so far. The client sends a request to the server . The server stores information associated with the request by means of an in-network monitoring function. Using the candidate selection function, we select candidate nodes from the accumulated information. After a candidate node is selected, it sends a message to the candidate node that it will transfer the service. A fog node that receives a message to transport a service confirms that no other service is established at its own node and starts collecting data for service execution, while at the same time sending a message to the node from which the service is being transported to inform it that the service is being prepared. When a

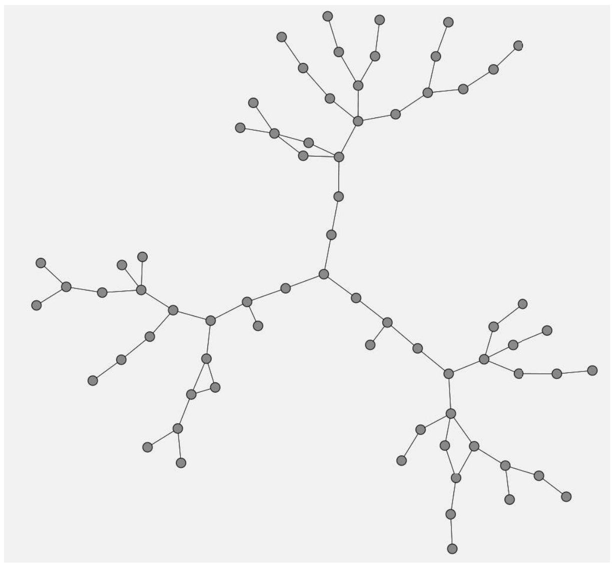


Figure 3: The network topology used in the experiment

fog node completes its data collection, it starts the service and sends a message to the source node telling it that it is ready. The server node that receives the ready message announces the new server to its current clients. The client receives information about the new server and changes the destination of the request to the new server. A client that joins from the middle of the process first sends a request to the original server node, receives information on the current server, and joins the service based on that information.

5 EVALUATION

We defined three use cases to show three aspects: fairness between users on uniform computer resources, reduction of average response time on heterogeneous computer resources, and fairness and reduction of average response time between users on heterogeneous computer resources.

5.1 Simulation Environment

For the network topology, we used the topology generated by BRITE[16], which is a topology generator as shown in Fig. 3. A county of three AS-equivalent nodes was prepared, with about 20 Fog nodes in each AS, and in each simulation, one AS was treated as a cloud environment and two AS were treated as a fog network with clients connected to it. Table 1 shows the parameters used in the simulation. The amount of content cache space owned by each node is also determined by the This was done assuming that the system has enough space to cache all the necessary data.

5.2 Use Case 1: Internet Conference

Use case 1 assumes a multi-point Internet conferencing system to show fairness among users with uniform computer resources. In the Internet conferencing system, the server mixes media data received from all connected terminals and distributes them as a single stream to all terminals. The goal

Table 1: Simulation Parameters

Parameters	Value
Cache Algorithm	LRU
Data Rate	10Mbps
Delay	1ms
Simulation time	100s
Server's C_C	100.0
Server's C_T	100.0
Dedicated Unit's C_C	20.0
Dedicated Unit's C_T	50.0
Regular Unit's C_C	20.0
Regular Unit's C_T	0

is to keep media quality fair in situations where geographically distributed participants connect to the system. Simulate the behavior of a server moving to the optimal location for clients distributed in different locations on the network with different latency times.

Simulation Scenario Multiple meetings were defined and the participants of each meeting were placed in the same AS, and the servers of all the meetings were placed together in a different AS than the AS in which the clients were participating. The processing capacity for conducting the conference and the processing capacity of each node were assumed to be constant. The results were compared with the case in which no transfer was performed.

5.3 Use Case 2: Machine Learning

In Use Case 2, we assume a service that uses machine learning on a fog network to demonstrate the realization of shortening the average response time on heterogeneous computational resources. For efficient processing of machine learning, it is better to have a dedicated purpose specific unit such as a TPU (Tensor Processing Unit). However, installing a purpose-specific unit for every fog node may be cost-prohibitive. Therefore, the cost problem is solved by using nodes with dedicated units as part of the fog network. In the fog network, nodes with normal CPU only and nodes with specialized units are mixed together, and the processing power is not uniform in the vicinity of a point. In this situation, we aim to minimize the service response time by transferring the service execution point from the processing accelerated by a purpose-specific unit, such as machine learning, to nodes with appropriate processing power. Experiments in this use case show that a certain percentage of nodes with purpose-specific units such as TPU are randomly placed on the fog network, and the average service response time can be reduced as expected in the entire network.

Simulation Scenario The following simulation scenarios were prepared to achieve the purpose of the experiment. Multiple clients connected to the network are available with servers in the distant cloud and fog nodes with various processing capabilities. Clients make service requests that be able to take advantage of the processing power of purpose-specific units such as the TPU. In this experiment, we show that the service response time can be minimized by arranging about % of all fog nodes. For that purpose, the ratio of nodes equipped with a dedicated unit in the fog network was changed from 0% to 100% in 20% increments, and an experiment was performed 100 times in which the arrangement was randomly changed.

5.4 Use Case 3: Video Delivery Service

In the third use case, we assumed a video delivery service to demonstrate the realization of fairness among users and reduction of average response time on heterogeneous computational resources. In some cases, you may want to cut out noteworthy parts of the video, overlay the board, or overlay chroma-keyless CG. When we want to perform such advanced video delivery, we need computer resources for video delivery. In addition, it is required that the communication delay must be low for the participants to comfortably watch videos. As in Use Case 2, it is not practical to deploy computational resources for video delivery to all nodes due to cost issues. Therefore, optimizing the execution point requires both processing time and communication delay, which are affected by processing performance. The results are compared with the case where simple average service response times are taken into account.

Simulation Scenario The video distributor sends the video to the server. The client receives the delivery stream processed by the server. While transferring to nodes with processing power suitable for video processing, the delay in video delivery to participants is reduced to such a node. When new participants are added from a different AS than the one that attracts video distributors and initial participants, the processing power and participants were observed transferring based on the fairness of the communication delay. New participants are timed to join the video feed at 50 seconds from the start of the simulation, such as Simulations were performed.

6 RESULTS AND DISCUSSION

The results and discussion of the experiments conducted for each use case are described, respectively.

6.1 Use Case 1: Internet Conference

In this use case experiment, we achieved fairness between users on a uniform computational resource. The experimental results for Use Case 1 are shown in Fig. 6 and Fig. 7. Figure 6 shows the change in service response time when the proposed system is not used, and Fig. 7 plots the change in service response time against time when the proposed system is used. The users of the proposed system are gradually transferred to the one with less network latency.

At the timing of the start of the simulation, the two conference streams are shown in Fig. 4. It is sent from different AS to a single AS, and the network traffic is aggregated. In the network after the transfer, the servers are transferred within each AS, as shown in Fig. 5, and the two conference avoids the aggregation of meeting traffic.

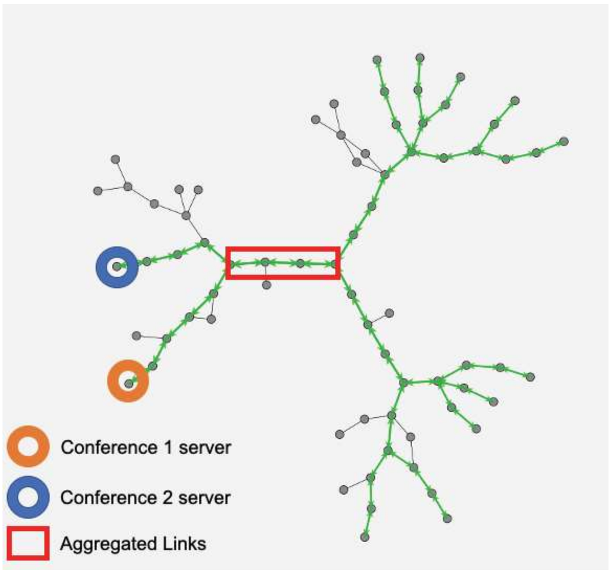


Figure 4: Use case 1: Network status before the transfer begins

In Figure 5, the fairness between users depends on which node on each communication path the server will be transferred to. It is possible to achieve the fairness required by each application by adjusting the R_{std} used in the algorithm 1 for destination determination. Figure 8 shows the trend of the mean and standard deviation of the service response time for a single conference among the results of the selecting candidate nodes for transfer, where the value of R_{std} is set to 0 and only the mean of the service response time is important. Figure 9 sets the value of R_{std} as 0.7 and the node selection with a weighted standard deviation of 70% of the response time, showed the average service response time for the same meeting as in Fig. 8. Comparing the two figures, we can confirm that the result of Fig. 9, weighted at 70%, is fairer (i.e. smaller deviation) than the result of Fig. 9, which shows the fairness of the transfer between users. We are able to confirm that the fairness of the transfer between users is maintained. In this way, we achieved fairness among users with uniform computer resources by performing transfers to shorten the service response time and adjusting the parameters to meet the requirements of the application.

6.2 Use Case 2: Machine Learning

In this use case, we have shown the realization of reduced average response time on heterogeneous computational resources. Figure 10 shows the average service response time of 100 simulations with the TPU placement probability varying from 0% to 100% in 20 percent increments. In the worst case with the TPU node placement rate set to 0%, transfers do not occur because the service response time is shorter if the service continues to be processed at the initial server node than if it is transferred to the fog node. When the placement rate of TPU nodes is increased, transfers occur to TPU nodes and the processing capacity is uneven The service response in Fig. 10 shows that the optimization of the execution point on the fog network is correctly done This can be seen by looking

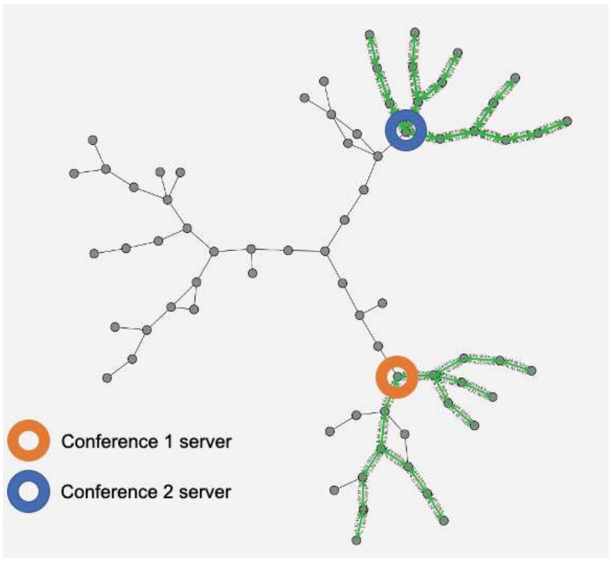


Figure 5: Use case 1: Network status after transfer

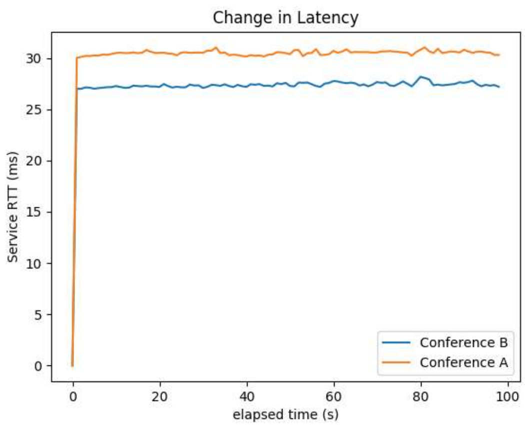


Figure 6: Use case 1: Changes in service response time if the proposed system was not used

at the changes in time. We have shown that this reduces the average response time for non-uniform computer resources.

6.3 Use Case 3: Video Delivery Services

In Use Case 3, we showed how to achieve fairness between users and reduce the average response time on non-uniform computational resources. A simulation with a 20% probability of deploying a dedicated unit to process the video. Figure 11 shows the mean and standard deviation of the service response time after 100 trials. In the period from 0 to 50 seconds in Fig. 11, when only the clients stuck in one AS are connected to the video delivery server, we are able to see that transfer reduces the standard deviation value, i.e., the fairness among the participating clients, but the mean of the service response time can be reduced significantly, and therefore transfer is used. In addition, after 50 seconds of simulation time, a new client of another AS began to connect to the video delivery server. Temporarily, the values of the mean and standard deviation of the service response time are increasing. However, we found a fog node where both the standard deviation

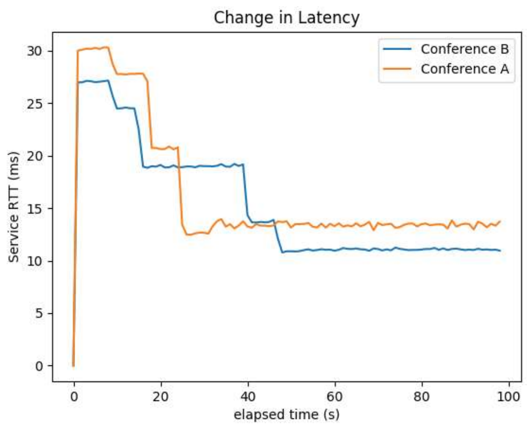


Figure 7: Use case 1: Changes in service response time when using the proposed system

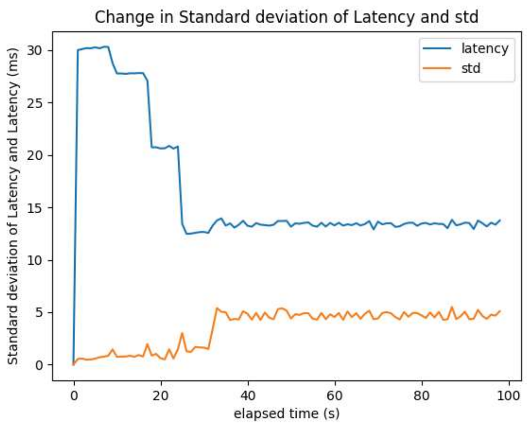


Figure 8: Mean and standard deviation of service response time for conference A when R_{std} is set to 0%

and service response time values can be reduced by about 75 seconds of simulation time, and we are capable of seeing a gradual change in the transfer.

6.4 Discussion

In this section, we discuss the validity and constraint of the design of the system in the view point of flexibility to scale and response to changes in available resources.

For scalability to the size of network, the proposed system does not require aggregation of information for the entire network nodes. The server node executing the service determines the service transfer based only on the PCEL information of each client and the relaying node on their path. In this design, we limit the discovery of the available resource in node on the message path. Our evaluation shows the effectiveness, however the result of the service transfer may be sub-optimal.

In terms of service scalability, the proposed system improved fairness and response time by controlling the placement of a single server instance. We have not considered the case to utilize multiple instances to scale.

We discuss the network change issues from the perspective

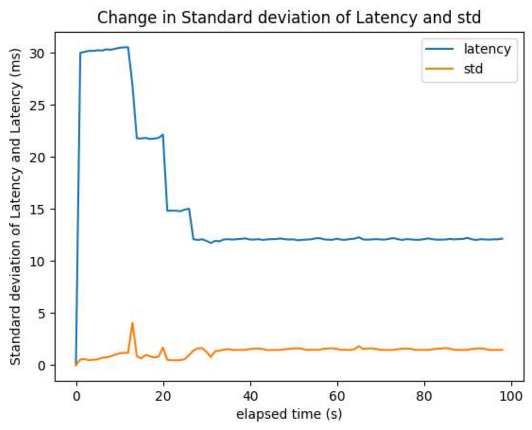


Figure 9: Mean and standard deviation of service response time for conference A when R_{std} is set to 70%

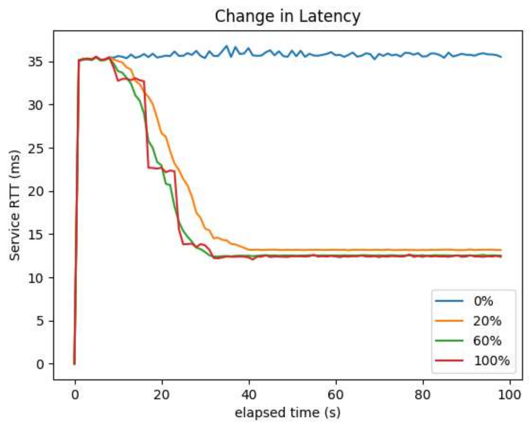


Figure 10: Changes in the average service response time as a percentage of TPU nodes deployed

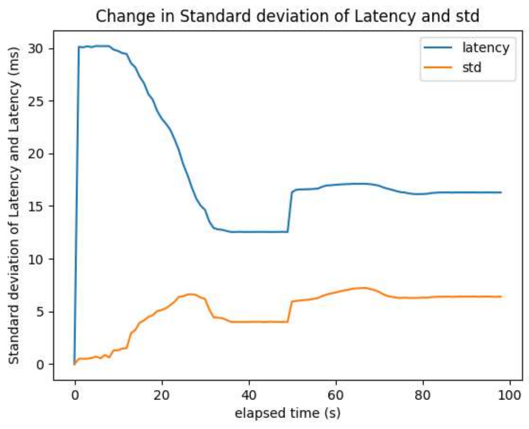


Figure 11: Use case 3: Mean and Standard Deviation of Service Response Time

of adapting to changes in paths and available resources.

Regarding the change of the path due to the reflection of the network topology etc., proposed system assumes that the interest message of the base CCN mechanism follows, and has no other mechanism. To estimate the available resources, PCEL information is piggybacked on the interest message and collected at the server node. From these, it can be said that the possibility of immediate adaptability to changes in the path of this system and changes in available resources depends on the frequency of exchange of interest messages.

If there is high frequency of exchange, it is possible to estimate precisely than when there are few. The evaluation scenarios discussed in this paper, target applications inherently assume stable frequency of sending and receiving messages. In contrast, if the use case only interacts with clients infrequently, they may not be able to keep up with changes in paths and available resources. It may be necessary to increase the volume of message flow for the resource monitoring purpose.

In summary, the situations in which this system is considered applicable are follows. This system makes it possible to relocate servers by utilizing frequent exchanges of messages between clients and servers in response to changes in the network. While this system is scalable to the size of the network because it does not require aggregation of information for the entire network, there is a limitation to the service scalability.

7 CONCLUSION

Fog computing, which extends the cloud computing paradigm to the edge of the network, has been proposed and is being actively researched. In the field of networking, there is research on CCN that use location-independent content as identifiers instead of the traditional IP address-based architecture. So far, we have proposed a system that aims at fairness in response time and shortening of service execution time between users, respectively. Therefore, in this study, we proposed the autonomous control of server relocation for fog computing systems to optimize QoS for end users, which achieves both shortening the average response time and fairness between users. To this end, the system achieves inter-user fairness on uniform computer resources, reduction of average response time on non-uniform computer resources, and we tested the fairness between users and the reduction of the average response time on non-uniform computer resources by setting up three use cases for each.

In the future, we will consider further expansion of the proposed system based on the use cases mentioned in this article, and work to enhance the usefulness of the proposed system.

REFERENCES

[1] Ministry of Internal Affairs and Communications, Japan: *The 2018 White Paper on Information and Communications in Japan*, Japanese Government (2018).

[2] Bonomi, F., Milito, R., Zhu, J. and Addepalli, S.: Fog Computing and Its Role in the Internet of Things, *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pp. 13–16 (2012).

[3] Jacobson, V., Smetters, D. K., Thornton, J. D. et al.: Networking Named Content, *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pp. 1–12 (2009).

[4] Kamada, K., Inamura, H. and Nakamura, Y.: A Proposal of Autonomous Control of Server Relocation for Fog Computing Systems(In Japanese), *IPSJ SIG Technical Report*, Vol. 2018-MBL-89, No. 1, pp. 1–5 (2018).

[5] Kamada, K., Inamura, H. and Nakamura, Y.: Autonomous Control Scheme of Server Relocation for Non-Uniform Computing Capacity in Fog Networks(In Japanese), *DICOMO2019*, Vol. 2019, pp. 1204–1211 (2019).

[6] Cuervo, E., Balasubramanian, A., Cho, D.-k. et al.: MAUI: making smartphones last longer with code offload, ACM Press, p. 49 (2010).

[7] Chun, B.-G., Ihm, S., Maniatis, P. et al.: CloneCloud: elastic execution between mobile device and cloud, ACM Press, p. 301 (2011).

[8] Kosta, S., Aucinas, A., Hui, P. et al.: ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, *2012 Proceedings IEEE INFOCOM*, pp. 945–953 (2012).

[9] Berg, F., Dürr, F. and Rothermel, K.: Increasing the Efficiency of Code Offloading in N-tier Environments with Code Bubbling, *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS 2016, ACM, pp. 170–179 (2019).

[10] Shanbhag, S., Schwan, N., Rimac, I. and Varvello, M.: SoCCeR: services over content-centric routing, *Proceedings of the ACM SIGCOMM workshop on Information-centric networking - ICN '11*, ACM Press, p. 62 (2011).

[11] Wolf, T.: Service-Centric End-to-End Abstractions in Next-Generation Networks, *Proceedings of 15th International Conference on Computer Communications and Networks*, IEEE, pp. 79–86 (2006).

[12] Dorigo, M. and Birattari, M.: Ant Colony Optimization, *Encyclopedia of Machine Learning* (Sammut, C. and Webb, G. I., eds.), Springer US, pp. 36–39 (online), https://doi.org/10.1007/978-0-387-30164-8_22 (2010).

[13] Selimi, M., Navarro, L., Braem, B., Freitag, F. and Lertsinsrubtavee, A.: Towards Information-Centric Edge Platform for Mesh Networks: The Case of City-Lab Testbed, *2020 IEEE International Conference on Fog Computing (ICFC)*, IEEE, pp. 50–55 (online), doi10.1109/ICFC49376.2020.00016.

[14] Selimi, M., Lertsinsrubtavee, A., Sathiaseelan, A., Cerdà-Alabern, L. and Navarro, L.: PiCasso: Enabling information-centric multi-tenancy at the edge of community mesh networks, Vol. 164, p. 106897 (online), doi10.1016/j.comnet.2019.106897.

[15] Jouppi, N. P., Borchers, A., Boyle, R. et al.: In-Datacenter Performance Analysis of a Tensor Processing Unit, *Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17*, ACM Press, pp. 1–12 (2017).

[16] Medina, A., Lakhina, A., Matta, I. and Byers, J.: BRITE: An approach to universal topology generation, *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE, pp. 346–353 (2001).

(Received November 14, 2021)
(Accepted October 1, 2021)

Kouki Kamada received his B.E. and M.E. degrees in information science from Future University Hakodate, Japan in 2019 and 2021. His research interests include cloud infrastructure and mobile computing. He currently works in Cybozu, Inc.

Hiroshi Inamura is a professor of School of Systems Information Science, Future University Hakodate, since 2016. His current research interests include mobile computing, system software for smart devices, mobile and sensor network and their security. He was an executive research engineer in NTT docomo, Inc. He received B.E., M.E. and D.E. degree in Keio University, Japan. He is a member of IPSJ, IEICE, ACM and IEEE.

Yoshitaka Nakamura received B.E., M.S., and Ph.D. degrees from Osaka Univer-sity in 2002, 2004 and 2007, respectively. He is currently an associate professor at the Faculty of Engineering, Kyoto Tachibana University. His research interest includes infor-mation security and ubiquitous computing. He is a member of IEEE, IEICE, 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)	Intelligent Transportation System
Smart Cities, Communities, and Spaces	Distributed Computing
Big Data, Artificial Intelligence, and Data Science	Multi-media communication
Network Systems and Protocols	Information systems
Computer Supported Cooperative Work and Groupware	Mobile computing
Security and Privacy in Information Systems	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^{TM}

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

Publisher

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

E-mail: secretariat@infsoc.org

CONTENTS