Wireless Communications and Mobile Computing

Parallel and Distributed Computing: Algorithms, Programming, Applications and Technologies

Lead Guest Editor: Yong Zhang
Guest Editors: Li Ning, Peter Yung H. Tsin, Vincent Chau, and Vassilis Zissimopoulos

Parallel and Distributed Computing: Algorithms, Programming, Applications and Technologies

Parallel and Distributed Computing: Algorithms, Programming, Applications and Technologies

Lead Guest Editor: Yong Zhang
Guest Editors: Li Ning, Peter Yung H. Tsin, Vincent Chau, and Vassilis Zissimopoulos

Contents

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8478076, 17 pages
https://doi.org/10.1155/2021/8478076

Research Article

Debugging of Performance Degradation in Distributed Requests Handling Using Multilevel Trace Analysis

Naser Ezzati-Jivan,[1] **Houssem Daoud**,[2] **and Michel R. Dagenais**[2]

[1]*Brock University, St. Catharines, Ontario, Canada L2S 3A1*
[2]*Polytechnique Montreal, Montreal, Quebec, Canada H3T 1J4*

Correspondence should be addressed to Naser Ezzati-Jivan; nezzati@brocku.ca

Received 6 May 2021; Revised 29 June 2021; Accepted 16 October 2021; Published 16 November 2021

Academic Editor: Yong Zhang

Root cause identification of performance degradation within distributed systems is often a difficult and time-consuming task, yet it is crucial for maintaining high performance. In this paper, we present an execution trace-driven solution that reduces the efforts required to investigate, debug, and solve performance problems found in multinode distributed systems. The proposed approach employs a unified analysis method to represent trace data collected from the user-space level to the hardware level of involved nodes, allowing for efficient and effective root cause analysis. This solution works by extracting performance metrics and state information from trace data collected at user-space, kernel, and network levels. The multisource trace data is then synchronized and structured in a multidimensional data store, which is designed specifically for this kind of data. A posteriori analysis using a top-down approach is then used to investigate performance problems and detect their root causes. In this paper, we apply this generic framework to analyze trace data collected from the execution of the web server, database server, and application servers in a distributed LAMP (Linux, Apache, MySQL, and PHP) Stack. Using industrial level use cases, we show that the proposed approach is capable of investigating the root cause of performance issues, addressing unusual latency, and improving base latency by 70%. This is achieved with minimal tracing overhead that does not significantly impact performance, as well as $O(\log n)$ query response times for efficient analysis.

1. Introduction

When performance degradation occurs within a distributed system, it can have multiple causes. For instance, it may be caused by insufficient system resources, a problem in the network layer, a bug in the software code within a connecting node, incorrect input data, or the misconfiguration of one of the active modules or nodes. The situation gets more serious when we notice that it is not easy to locate the problem in the system, as it is running continuously in parallel with other software and machines that cannot be stopped for debugging.

Therefore, most issues within distributed systems are hard to recognize and investigate. Active monitoring of distributed system execution using runtime information can be helpful in this matter [1–3]. The runtime execution data, which is usually collected by logging and tracing tools, can help monitor the actual executions of systems, detect possible runtime problems, and hopefully pinpoint their root causes. Tracing is a method that consists of collecting execution logs from a system at runtime [4]. Unlike profiling, which usually provides statistics about a time range, tracing can display the state of the system at various levels, including the active processes, running system calls, function call stack, network usages, and active elements of disk queues at different time points, e.g., when a latency problem is detected in the system [5].

There are already trace-based solutions to debug performance problems of distributed systems [5–7], however, extracting the root cause of a performance problem requires much more depth than these solutions can provide since they mainly rely on a single level of data, either kernel, user-space, or network level [8]. Conversely, we present a trace-oriented solution that monitors the system in several layers (from the application layer to the driver layer), to improve visibility. The proposed solution uses trace data gathered from Linux Trace Toolkit Next Generation (LTTng), a low-overhead kernel and user-space tracer [4]. The traces collected from the different components of the distributed system are synchronized together based on an event matching algorithm so that all events are related to a unified clock. Our approach is generic and can be used to debug different distributed system issues, from problems in web servers to network file systems. However, our focus in this paper is mostly on web servers (distributed or centralized) and more specifically the Linux, Apache, MySQL, and PHP (LAMP) Stack. We instrument the different LAMP stack modules and provide tracing probes to collect runtime information from the web server, application server(s), database server(s), and from the client(s).

The collected trace is analyzed using a stateful top-down approach using a historical data-store that is built while the trace is first read. This data-store keeps track of the state of the system throughout execution and includes metrics from the various system layers which it measures, aggregates, and organizes. Its purpose is to give users an overview of the system and to help pinpoint any bottlenecks or performance misbehaviors. This data-store uses a tree structure with search complexity of $O(\log\ n)$ which is efficient for recovering the desired analysis data, without having to reread the substantial trace events every time the user selects a new area or system resource (e.g., a CPU, a process, or a virtual machine) to analyze. The database can be used to compare metric values from different points of execution to determine if there is misbehavior (e.g., an unexpectedly high response time or resource overload) at any time during execution. Upon discovery of problematic symptoms in the first overview step, the next analysis phase is to dig deeper into the problem using the information gathered and analyzed from the other layers, to investigate and detect the root cause of the problem.

The main contributions of the paper are

(i) A generic, top-down, and multi-level analysis solution to detect and locate performance bottlenecks of distributed systems

(ii) Using an optimized data structure to store the state of the system for the trace duration, which makes the analysis much more interactive afterward

(iii) A unified data collection approach to trace the different components of the LAMP stack (Linux, Apache (web server), MySQL (database server), and PHP (application server)) to efficiently collect the required run time execution data for a whole-chain web request analysis

The remainder of the paper is organized as follows: we discuss the related work of distributed system and web server performance analysis in Section 2, followed by the architecture of our performance analysis method in Section 3. In Section 4, we highlight the usefulness and functionality of our work through some real-world use-cases. We evaluate the overhead of our proposed method in Section 5 and conclude with a look at some interesting future work.

2. Related Work

A distributed system is a system whose processes and threads are running on multiple computers with different configurations. The possible incompatibility in the runtime environments makes the debugging of performance problems more difficult. Runtime execution data, collected by tracing tools, can help monitor the system execution, detecting performance problems, and uncovering their root causes.

Tracing is a method that consists of collecting execution logs from a system at runtime [9, 10]. The payload of a trace event usually contains the event name, the ID of the processor on which the event is executed, the timestamp, and the arguments. A trace event can be a system call, a function call, a signal, an IRQ, etc. Unlike debugging, where the program is executed step by step to get its current state, tracing collects data during the execution, and the trace file is often analyzed offline. The overhead of tracing should be minimal to preserve the normal behavior of the system.

Linux Trace Toolkit Next Generation (LTTng) [4] is an open-source Linux tracing tool initially developed by DORSAL (Distributed Open Reliable Systems Analysis Lab (DORSAL) http://www.dorsal.polymtl.ca) to provide very low-overhead tracing capabilities. It is packaged as an out-of-tree kernel module, and it is available in all major Linux distributions. LTTng supports kernel and user-space tracing, which is useful to correlate high-level application events with low-level kernel events to support a multilevel trace-based analysis. Execution traces can be used to study the runtime behavior of software applications. Daoud and Dagenais [11] proposed a trace-based framework to provide detailed workload characterizations of Java virtual machines. Surveys of trace-based dynamic analyses methods including different applications of multilevel execution traces in software analyses are presented in [10, 12].

Some previous works have applied kernel tracing to aid in distributed system performance monitoring. For instance, in [13], vectors representing system call sequences are created and used along with machine learning classification algorithms to identify anomalies. This work demonstrates how valuable kernel tracing can be in the identification of CPU or memory-related problems. Furthermore, in [14], Ates et al. utilize tracing methods to identify regions of high performance variability for the purposes of automatically enabling additional instrumentation. By including user-space and network level trace data, one can go beyond performance degradation identification and conduct precise root cause analysis within a distributed system.

Google Dapper [15], a tracing framework for distributed systems, works by instrumenting the RPC (remote procedure call) libraries to trace distributed requests. In this system, a unique identifier is assigned to each request at the entry point and is used to follow the request through the whole distributed system. This unique ID is used to temporarily join and relate the components interacting to serve the request. At the visualization time, a request is drawn recursively in a trace tree timeline, including spans and arrows, where each span includes the basic unit of work and arrows display the communication and interactions between spans. While Dapper is an excellent tool to break down a request into all its components, it does not offer much information to analyze the problem's root causes.

Pinpoint [6] monitors call request paths and clusters them into success and failure groups, to find latency-anomalous components. However, the fault may come from the operating system and not the software components, in which case this tool will not offer interesting insight.

Spectroscope [7] is aimed at diagnosing performance changes by comparing request flows. It applies root cause analysis by comparing the way the system services requests from different normal and problematic periods. Spectroscope extracts critical paths of requests from the normal and problematic parts of the system execution and groups the similar paths into a cluster using k-mean algorithms. The clusters correspond to different types of requests and the critical paths within a cluster show different ways of handling the same type of requests. They use data from only a single layer, whereas our approach uses several.

TraceCompare [5] uses a similar idea but leaves the selection of normal and slow requests to users, where they can see all requests in side-by-side distributions and can choose any two areas to compare. TraceCompare [5] extracts critical paths of requests and converts them to an enhanced calling context tree (ECCT). It then extracts a differential flame graph from the various requests to compare the differences. By comparing two different requests, a user can get the detailed differences of their executions and possibly figure out the reasons for the latency of the slow request. Although it uses traces from several levels, it depends on comparing various executions, which is different from our proposal.

In this paper, we address the runtime issues which may occur in the different modules, nodes, or layers of distributed systems (i.e., LAMP stack including a web server, a database server, and an application server). We collect trace data from multiple levels (application, system call, disk block layer, and network layer). The data from each layer has its specificity and should be processed based on the characteristics of that layer. A comparison of using user-space and kernel-space trace data in software anomaly detection is presented in [16]. The separate analysis of different layers of kernel data is studied in [10], while the processing of system call traces is reviewed in [17]. Another important layer is the physical storage (disk) layer. The performance of disk operations may directly affect the performance of applications. Different applications, like web and database servers, usually provide a user-space caching mechanism to reduce disk accesses, but it is still impossible to hold all the required data in memory, and accesses to disk are necessary at some point. The storage subsystem itself is composed of different layers: the virtual file system (VFS), the file system (ext4, ext2, btrfs, etc.), the block layer, and the disk driver. A user-space application requests I/O operations through system calls, and these requests go down through the layers until they reach the disk drive where they get processed. Daoud and Dagenais [11] proposed a comprehensive Linux tool to recover high-level storage metrics from low-level trace events collected at different layers of the storage subsystem. Similar tools like Oracle ZFS Storage Software [18] and IBM XIV [19] are available in other operating systems.

Many storage debugging tools can be used to debug storage issues. The traditional tools like *iostat* and *iotop* provide information about disk activity, and the processes causing this activity, by parsing the kernel statistics available in the *proc* file system. This information is useful to monitor disk activity, but it cannot be used to debug difficult problems. Block-level tracing provides a more precise solution to debug disk performance. Tracers like *Blktrace* [20] provide low-level information about I/O requests and the behavior of the disk scheduler, but this information is usually very detailed and difficult to analyze manually. Visualization tools like *Seekwatcher* [21], IOprof [22], and BTT [23] can be used to read the trace files generated by *blktrace* and to generate storage metrics from them. However, the visibility of *blktrace* is limited to the block layer, and it does not cover the other layers of the storage subsystems.

All the above studies review different aspects of trace-based analysis. However, they lack a unified way to analyze multilevel multinode trace data, which we address. We propose a solution to collect execution data from different layers and different sources, process them, extract the required information, and place them in a common data store to use in a posteriori analysis phase. This analysis synchronizes the data and performs a global analysis based on the common features (i.e., clock time and process names) between the data at different layers and from different sources.

3. Architecture

A distributed system is composed of different computers interconnected through the network. To ensure a comprehensive analysis, it is important to collect traces at different levels: user-space level, kernel level, and network level. The collected information is then sent to an automated tool for unified analysis. The general architecture of the proposed solution is presented in Figure 1. The next section will provide a detailed description of each of those components.

3.1. Data Collection

3.1.1. User-Space Tracing. User-space tracing is a technique to collect program runtime data by probing the different points of application code. It can be used for anomaly and fault localization in software code. Developers can embed different probes (tracepoints) in the important parts of the software code to get execution logs at runtime when the execution reaches them. Then, by analyzing these logs, users can understand what is actually happening in the different parts of the software.

However, it is also possible to trace a user-space application without changing its source code. To do so, the core library is replaced with a wrapper library and the program

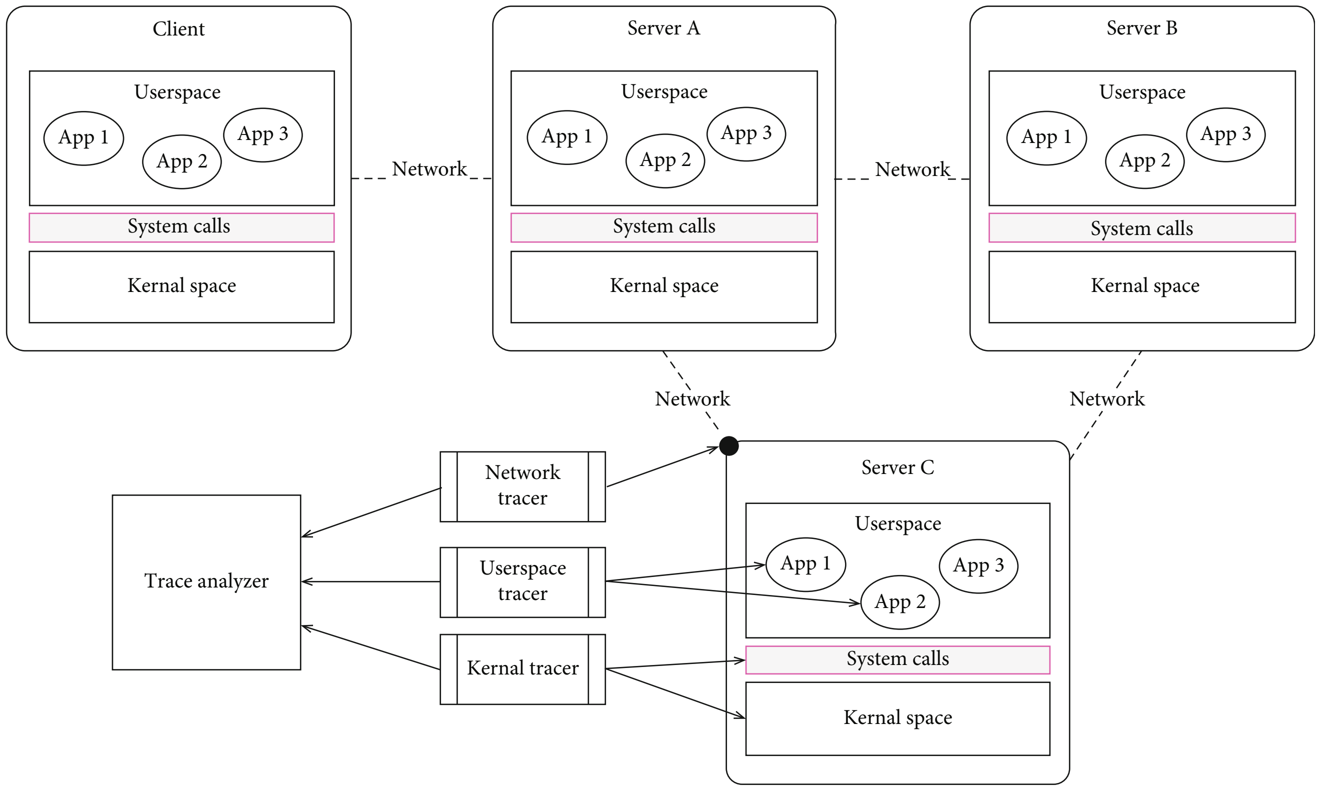


Figure 1: General architecture of the proposed multilevel analysis.

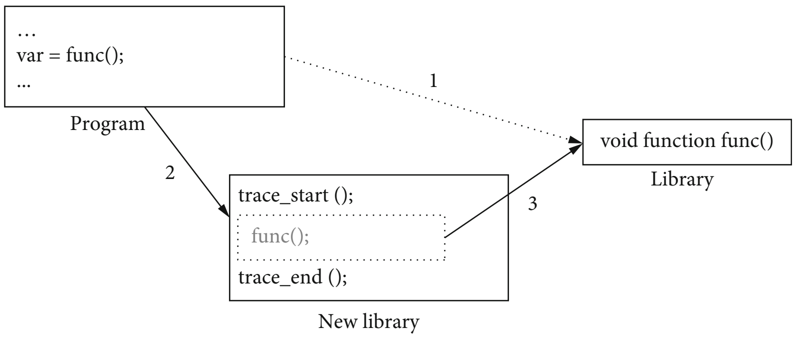


Figure 2: User-space tracing using the wrapping method.

calls the functions from the new wrapper library using LD_ PRELOAD or other interception techniques. The wrapper library in turn calls the functions from the original library but includes some tracing before and after. Figure 2 shows this technique. Arrow number 1 shows the original path between the program and the library. Arrows 2 and 3 show the new calling path going through the wrapper library.

In the first phase of the technique proposed in this paper, the core of the LAMP stack (Linux, Apache, MySQL/MariaDB, and PHP) was changed to support tracing. A few tracepoints have been added to the libraries of the different modules to ensure that all important aspects of the LAMP stack are covered. Using this modified core of the LAMP stack, users can trace their web applications without needing to change their program source code.

For Apache, we have written a module to trace Apache requests. For each request, it records the start and end times, in addition to all the details about each request (e.g., client IP, file accessed, method, and user-agent).

In PHP, the core functions are changed to enable tracing at the request level, function call level, and code execution level. For each request, it records the start and end times and details such as client IP, files accessed, methods, and ports. For each function call, it records the filename, function name, class name, and status of the function execution. For each execution, it records the line number, file and function names, and execution status. These LAMP extensions (both the Apache module and the PHP extension) are open-source (https://github.com/naser/) and available for public use.

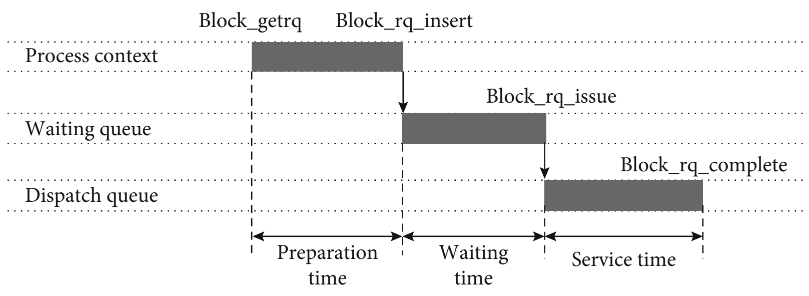


Figure 3: Request latency.

MariaDB (a fork of MySQL) is also instrumented to trace database connections, commands, and queries. Using these tracepoints, it is possible to gather precise runtime information about query execution, query response time, latency, or time elapsed in different parts of the query execution steps (e.g., query parsing, query caching, and query real execution).

What is important to note is that when the tracepoints have not been chosen for tracing or are disabled, they do not execute any tracing code, so they impose no additional overhead. When they are enabled, they may incur some very low overhead. We measure this additional overhead cost in Section 5. Using the mentioned tracepoints, we can debug the whole LAMP stack. For each request, we can give the exact time elapsed in each part of the stack and what fraction of the request time was spent in Apache, PHP, or the database. This means we can locate the potential bottlenecks of any request.

3.1.2. Kernel Tracing. System calls form a boundary between the user-space and the kernel-space. They are used by the applications to interact with the operating system and hardware resources. Tracing the system call layer provides useful information, which can be used to analyze the behavior of the system and to evaluate its performance. Detecting problematic calls is done by tracing the entry and exit of each system call and calculating its duration. In the context of our analysis, we need to monitor all system calls related to I/O operations (opening and creating files, write/read operations, reposition file offset, duplicate file descriptor, etc.).

System call events provide limited visibility about what is happening inside the storage subsystem. For example, a *READ* system call may return quickly because the data was found in the page cache, not because the disk offers good performance. To get a wider visibility, it is important to trace lower-level events that give more precise information about storage activity. The three important components to be considered in a deep storage performance analysis are the file system, the page cache, and the block layer.

The file system is the key component of the operating system responsible for storing and retrieving data from storage. It presents the data to the user-space applications in terms of files and internally keeps the match between each file and its physical location on disk. Typical file system operations are creating, reading, writing, and deleting a file. System calls are the entry points to the file system. For example, a read system call is directly linked to the *read* handler defined by the file system to *struct file_operations*. In addition to the basic operations, the file system uses many advanced mechanisms to ensure the coherence and security of the data. Some interesting file system events that can be used for a detailed analysis include inode creation and deletion, starting and processing writing operations, and when the file system journal starts to commit an operation.

The page cache is used to store data in the main memory, to minimize disk I/O operations. The file system always tries to recover the required data from the page cache before issuing a read request to the disk. The same happens for writing operations: the data is written to the page cache before being transferred to the disk asynchronously by the write-back mechanism. In Linux, the page cache is represented as a Radix Tree, since it provides a good search complexity. It is possible to evaluate the page cache lookup efficiency by tracing the entry and exit of *find_get_page*. However, in our analysis, we decided to generate a trace event only when a cache miss happens. The reasoning behind this decision was twofold. First, tracing all cache lookup operations adds too much overhead to the system. Second, the lookup time is insignificant compared to the cache miss handling time.

The block layer is the operating system component responsible for I/O request management. By instrumenting the block layer, we can get very precise information about I/O requests that are being processed, and how they are managed by the disk scheduler. We follow the I/O request and collect trace events from creation, through the waiting and dispatch queues, to completion by the disk drive.

The request latency can be broken into three main parts, as shown in Figure 3. The preparation time is the time needed to create the request data structure. The waiting time is the time during which the request resides in the waiting queue. The service time is the time taken by the disk drive to handle the request.

3.1.3. Kernel Modules Tracing. The Linux kernel offers many helper functions to facilitate driver development. For example, disk drivers typically use the exported functions of the block layer to manage requests and waiting queues. However, information may be needed from inside the driver and, for that, additional instrumentation is required. In the case of storage devices, it is important to know exactly when the interaction with the hardware happens. Four tracepoints are used for this:

(i) *scsi_dispatch_cmd_start* is executed when the request is sent to the disk controller

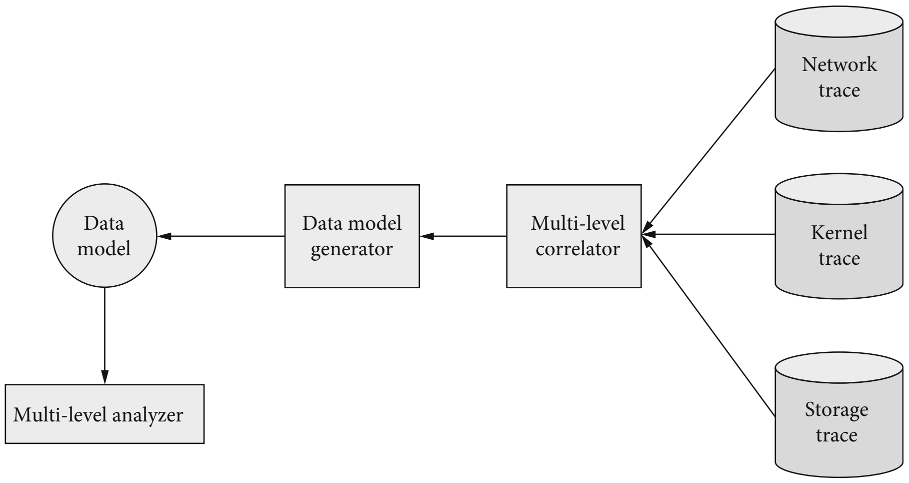


Figure 4: Data analyzer architecture.

(ii) *scsi_dispatch_cmd_error* is executed if the request delivery has failed

(iii) *scsi_dispatch_cmd_done* is executed if the command was handled correctly by the disk drive

(iv) *scsi_dispatch_cmd_timeout* is executed if the disk drive does not respond in time

3.1.4. Network Tracing. Network communication is an important part of a distributed system. Latency in the transmission of network packets has a direct impact on the performance of the system. In such cases, the application must be blocked until the required information is transmitted through the network. To get precise information about network latency, we based our analysis on many network-related system calls such as *connect()*, *accept()*, and *shutdown()*.

Network tracing is also used to synchronize traces between different machines. LTTng recovers event timestamps using the monotonic clock provided by the kernel of the traced machine. The absence of a global clock analysis in a distributed system is challenging and may even cause messages to seemingly be received before being issued (message inversion). The convex hull algorithm was proposed by Jabbarifar et al. [24] to solve this problem. The idea is to use event matching between different hosts to unify the clock sources. The couple {inet_sock_local_out, inet_sock_local_in} can be used for synchronization in the case of a distributed system because one event triggers the other and they share a common ID in their payload.

3.2. Data Analysis. Trace events usually contain very detailed information about the runtime behavior of the system, which is essential in the detection of hard problems. However, trace files are often very large, and pinpointing a problem inside them is not an easy task. An automated analysis system should be used to read the trace and generate higher-level, easier to understand information.

Data-driven abstraction is an abstraction method that consists of generating compound events by grouping low-level trace events. Ezzati-Jivan and Dagenais [10] proposed an approach that aims to summarize trace files based on a pattern library. The idea is to replace semantically-related trace events with a compound event like SEND_NETWORK_PACKET or WAIT_ON_MUTEX. This method improves the legibility of the trace but the oversimplification may hide important details, making the detection of some problems impossible. In the case of distributed systems, the problems are usually at a very low level; a deeper analysis is therefore required.

Metric-based abstraction is another approach that helps to identify problems without reducing the precision of the analysis. A metric is computed from one or many trace events, and a hierarchy of metrics can be defined so that a high-level metric is computed from lower-level metrics. Performance metrics are very useful for an efficient top-down approach. This technique is useful in our case: the troubleshooter can pinpoint the origin of the problem by looking at different metrics provided from the different layers. It starts by monitoring high-level metrics, and, when something unexpected happens, it goes down to see what happens at a deeper level of detail.

3.2.1. Proposed Architecture. In this section, we propose a generic architecture to process and analyze trace events provided from different layers. This architecture is illustrated in Figure 4.

3.2.2. Multilevel Correlator. It is the component responsible for getting the trace data from the different layers and synchronizing it. If the trace events are gathered from the same machine, the synchronization is based on the event timestamps. LTTng follows the monotonic time-base of the Linux kernel to reliably recover the exact time of each event [4].

The Linux kernel has two internal clocks: CLOCK_REALTIME and CLOCK_MONOTONIC. CLOCK_REALTIME (kernel function do_gettimeofday) tracks the wall clock and gets updated and corrected by the Network Time Protocol (NTP) daemon and is not monotonic. CLOCK_MONOTONIC represents the absolute elapsed wall-clock time since the machine boot time (which is an optional fixed prior time point). CLOCK_MONOTONIC, therefore, monitors more closely the hardware timers and does not jump in time. CLOCK_MONOTONIC is generally used for tracing, since it

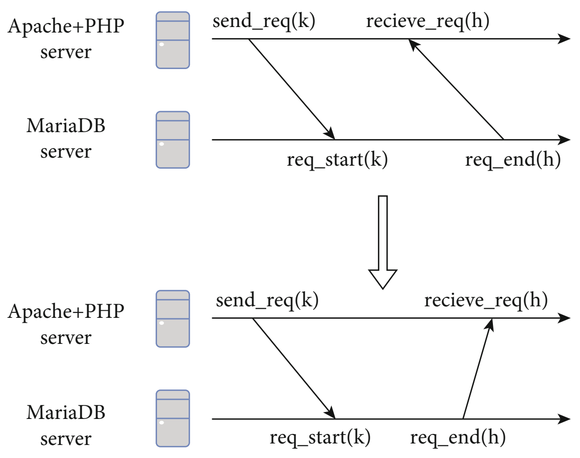


Figure 5: Trace synchronization in a distributed system.

is precise, has a fine granularity, and is most consistent across small and large time intervals. Moreover, the fact that it is monotonic avoids the ambiguity present for timestamps with CLOCK_REALTIME, when the clock is adjusted backwards.

LTTng follows the second approach, like CLOCK_MONOTONIC. However, to be safe from integer arithmetic imprecision, it directly reads and outputs the CPU timestamp counters which include the most accurate CPU calibration frequency and leaves the other integer arithmetic operations to the trace postprocessing analysis tools like Trace Compass or Babeltrace.

This time-base guarantees a partial ordering of events, even if the events are generated by different CPUs. The synchronization is more complex if the traces are collected from different machines. In this case, the causality between events is the only way to get a partial ordering, which is enough for the analysis. The fully incremental convex hull synchronization algorithm [24] is used to solve the problem of trace synchronization in distributed systems. The algorithm is based on matching an event a from one trace and an event b from the other. The couple {a,b} must satisfy the following requirement:

(i) a triggers b

(ii) a and b must have a common value in their payload

(iii) Every event b must have a corresponding event a in the other trace

For example, if PHP and MariaDB are installed on different machines, the traces can be synchronized using the matching events {send_req,req_start} and {req_end, receive_req}, as shown in Figure 5.

3.2.3. Data Model Generator. Detecting a performance bottleneck is a complex operation. To analyze the trace, users have to navigate horizontally by selecting different time regions and vertically by zooming in on regions of interest. Computing the metrics every time the user selects a new time range is a very slow operation because it requires rereading the same events again and again to generate the statistics.

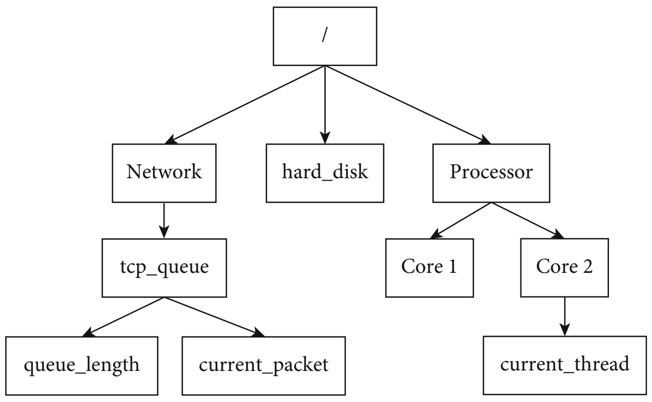


Figure 6: Attribute tree.

To avoid this extra computation, we decided to use a stateful approach in which the state of the system is kept in an incremental database when the trace is read for the first time. Every time a new time range is selected, the metrics can be quickly retrieved from the database, without reading the trace file.

Using a memory-based data-structure like Segment-tree or R-tree is not a viable solution, since trace files may be huge and the model generated cannot be kept in memory. On the other hand, disk-based solutions like relational databases are not tailored to tracing and provide a very slow response time in that context, as shown in Section 5.3.

Here, we use the modeled state system a custom-build database to keep track of the execution states of the different components of the system during its execution. The main components of the modeled state system are the attribute tree and the state history tree. The attribute tree represents the resource hierarchy of the system, and each node can be accessed using an absolute or relative path. A sample of the attribute tree used in our analysis is shown in Figure 6. For example, we can easily access the queue length of the TCP stack using the path/network/tcp_queue/queue_length.

The state history tree saves the state of the system on the disk in terms of nodes. A node is defined by a key, a time range, and a state, where the key is the path of the system component in the attribute tree. States are added continuously into a node until it is full and a new sibling is created, as shown in Figure 7. The state history tree offers a fast query time ($O(\log\ n)$ where n is the number of nodes), making it very convenient for an interactive analysis when a big trace is involved.

Formally, every event EV is a tuple like $(r1, \cdots, rn)$ which represents an interaction between a set of system resources, e.g., processors, files, disks, processes, and network sockets at a specific timestamp ti, and with one or more output values like $v_i \in N$.

$$\begin{cases} \mathrm{EV} = \left\{(t_i, r_1, \cdots, r_n, v_1, \cdots, v_m) \mid t_i \in T, r_j \in \mathrm{SR}, v_k \in N\right\}, \\ T\{t \mid t \in N\}, \\ \mathrm{SR} = \mathrm{SystemResources}. \end{cases} \tag{1}$$

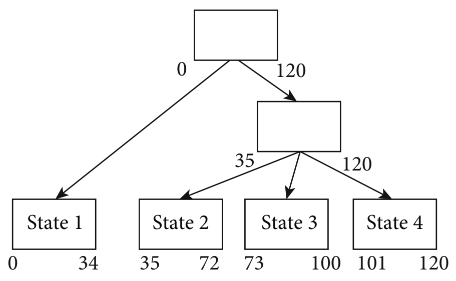


Figure 7: State history tree.

For example, an event like (t1, p1, read, fd1, cpu0, 100) shows that at time t1, process p1 reads 100 bytes from a file associated with the file descriptor fd1, while running on cpu0.

A trace TR may be seen as a set of ordered events. Events in a trace are ordered by their timestamp, and no two distinct events have the same time value:

$$\mathrm{TR} = \left\{ e_i \mid e_i \in \mathrm{EV}, i < j \longrightarrow t_{e_i} < t_{e_j} \right\}. \quad (2)$$

An abstract event, which we also call "State Change," is a high-level event that is built by grouping some raw level events. Each state has a duration, a key, and a value associated with that attribute for the given time duration. Abstract events (state changes) can be used to denote high-level concepts like an active network connection, a process blocked, a process running, and a CPU preempted. In summary, each state change is associated with time duration and contains a key and a value. For instance, the state change of "an active network connection" includes a key (i.e., the network connection socket id or the source and destination IP addresses), a value (i.e., active), and time duration (e.g., t1 to t2) during which the state value is valid for the key.

$$\begin{cases} \text{Time Duration}\mathrm{TD} = \left\{ \left[t_i, t_j\right] \mid t_i, t_j \in N, t_i < t_j \right\}, \\ \mathrm{SV} = \left\{ (td_i, at_i, v_i) \mid td_i \in \mathrm{TD}, at_i \in \text{Attributes}, v_i \in N \right\}, \\ \text{State Database } SD = \left\{ sv_i \mid sv_i \in \mathrm{SV} \right\}. \end{cases} \quad (3)$$

As shown in the above equations, each state value sv_i includes a time range, an attribute, and a value. The attribute is a way to describe an aspect of a system resource. For example, an attribute fd shows a file descriptor (i.e., an aspect) of a file (i.e., a system resource). Other examples are the id of a process or a thread, the CPU core number, the address of a socket, the parent PID of a process, and so on.

Analysis AN is a function from the trace events TR to the state values database SD.

$$\text{Analysis AN} : \mathrm{TR} \longrightarrow \mathrm{SV}. \quad (4)$$

For instance, the analysis to extract the CPU utilization of each process $(\mathrm{CPU}A(p_i))$ may contain data like the following:

$$\begin{cases} \mathrm{TD} = \left\{ [ti, tj] \mid ti, tj \in N, ti < tj \right\}, \\ \mathrm{PR} = \left\{ p_i \mid p_i \in \text{System Processes} \right\}, \\ \mathrm{CU}_s = \left\{ (td_i, p_i, v_i) \mid tdi \in \mathrm{TD}, p_i \in \mathrm{PR}, v_i \in N \right\}, \\ \mathrm{CPU}A(p_i)\colon \mathrm{TR} \longrightarrow \mathrm{CU}_s. \end{cases} \quad (5)$$

Each analysis (like the analysis CPUA in the previous example) has a set of different mapping rules to convert events to state values. Depending on the analysis, the type of input events, and their effect on the state of the system attributes, the rules can take the form of a simple "if-then-else" or a complex transition pattern. The rules can be defined in the analysis tool source code, in the form of JAVA code, or can be specified dynamically using XML patterns. The possibility to define the analysis rules in XML form allows defining complex transitions from the trace events to states, to fulfill the requirements of specific use cases and problems.

$$\mathrm{RL} = \left\{ (e_i \longrightarrow sv_i) \mid e_i \subset \mathrm{TR}, sv_i \subset \mathrm{SD} \right\}, \quad (6)$$

$$\mathrm{AN} = \left\{ r \mid r \in \mathrm{RL} \right\}. \quad (7)$$

In the same way, we wrap all the analysis belonging to the same host into a container called a model. A model is a set of analyses, for a specific system, which can be used to reason about the underlying system from different points of view. A Model Cloud, in turn, is the superset of all existing models: a container for all analyses of all tracing systems (Figure 8).

$$\text{Model Cloud MDC} = \left\{ an_i \mid an_i \in \mathrm{AN} \right\}, \quad (8)$$

$$\begin{cases} \mathrm{AS} : \dfrac{\text{Attributes of a specific system}}{\text{host}}, \\ \mathrm{MDC} = \left\{ an_i \mid an_i \in \mathrm{AN}, A_a n_i \in \mathrm{AS} \right\}, \\ \text{Model } \mathrm{MD}_s \subset \mathrm{MDC}. \end{cases} \quad (9)$$

Using the proposed model, users may aim to trace a distributed system, spanning different nodes (virtual machines or physical hosts). After tracing all machines, analyses for each are performed and a model for each is built. All such models together constitute a Model Cloud. The generated Model Cloud is then used in the analysis phase to render user-friendly views.

3.2.4. Multilevel Analyzer. As mentioned earlier, analyses for a model can be defined using JAVA code, recompiled into the tool before opening the trace, or can be dynamically defined using XML language patterns. The resulting model can be huge if many distributed nodes are involved, and opening all of them may slow down the trace analysis. Therefore, the analyses are partitioned into two main categories: base and subsidiary. The base analyses are those that are required to render the basic views (like control flow, histogram, etc.) and are executed when the trace is first opened. The subsidiary analyses, however, are not opened with the trace and are executed only upon a specific user request.

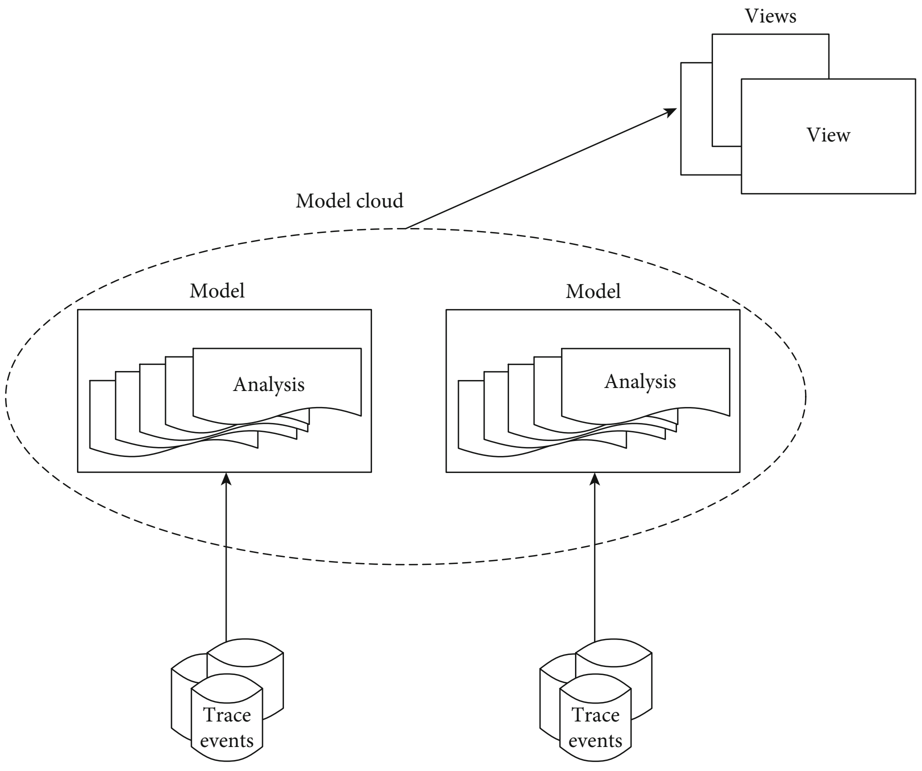


Figure 8: Trace events to analysis to model to Model Cloud.

This prevents the system from being slow when the trace is opened for the first time. Users can nonetheless see a list of the available analyses and can execute them whenever they want.

3.2.5. Generic Detection Algorithm. As explained above, the data collected from different sources is synchronized using timing and causality relationships with the multilevel correlator, and then modeled and saved in the State History database by the Data Model Generator. The multilevel analyzer recovers the data from the generated data model and uses an advanced algorithm to help the user detect performance bottlenecks. Bottlenecks can reside in any system layer. For example, in the case of a web server, the latency can reside in the web engine, the database, the network, the storage subsystem, etc. A good strategy is to start by analyzing the high-level layers and then go deeper if a problem is detected. The generic detection algorithm used by our tool is described in Algorithm 1 and shown graphically in Figure 9 to find latency problems in the context of a web server.

This algorithm is based on a top-down approach. It starts by computing high-level metrics from the user-space tracepoints. If an anomaly is detected, it is important to know if the problem resides in the application itself or if it is caused by the environment. An anomaly is detected if a web request takes longer than the normal distribution. We collect statistics about the number of requests, the latency, and the type of requests, and we use that as a baseline with which we compare. The moment we detect a request that took a longer time than normal, we consider it as abnormal and we use the proposed analysis for further investigation. To this end, the tool looks at the user-kernel boundary and checks if there is a specific system call that caused the latency. In such a case, we can use the kernel system calls to uncover the deeper reason for the latency.

In Figure 9, we describe in more detail how a problem is detected in the context of a web server. If a web request is slow (slower than a threshold), it can be because of the user-space logic (e.g., a long loop in the application code); in this case, no system calls are involved. However, the slowness may also be caused by a long system call, for example, a read syscall. In this case, the two main reasons are possible. The file is big and needs a lot of time to be read or the file is small, but there is disk contention, which makes it slow to read. To uncover the root cause, kernel events are needed, in addition to system calls. Figure 9 describes the decision process for this detection.

The procedure shown in Algorithm 1 covers system calls and other types of events at several levels of the execution. The term multilevel here means the different LAMP stack layers (Apache/PHP/MySQL), and also user-space/system calls/kernel. An operation can be slow because many computations have to be done on the CPU, or because it is reading a big file, or because it is waiting for a network packet, etc. The only way to know the reason for the latency is to have full visibility about what is happening in the system and to use the provided algorithm to narrow down the reason for the slowness.

The proposed method is implemented in Trace Compass [25] (an open-source trace viewing and analysis tool), under the name of Trace Compass Incubator and is available to the public [25].

```
**Input:**
User-space Tracepoints
System calls
Kernel Tracepoints
Events relationships map
Output: Problem
Compute high level metrics from user-space tracepoints
If user-space latency detected
    Find the system calls that caused the latency
    If system calls latency detected then
        Get the kernel events related to the problematic system calls
        If kernel anomaly detected
            Problem ⟵ Kernel
        Else
            Problem ⟵ System calls
        End
    Else
        Problem ⟵ User – space
    End
Else
    Problem ⟵ NONE
End
```

Algorithm 1: Generic detection algorithm.

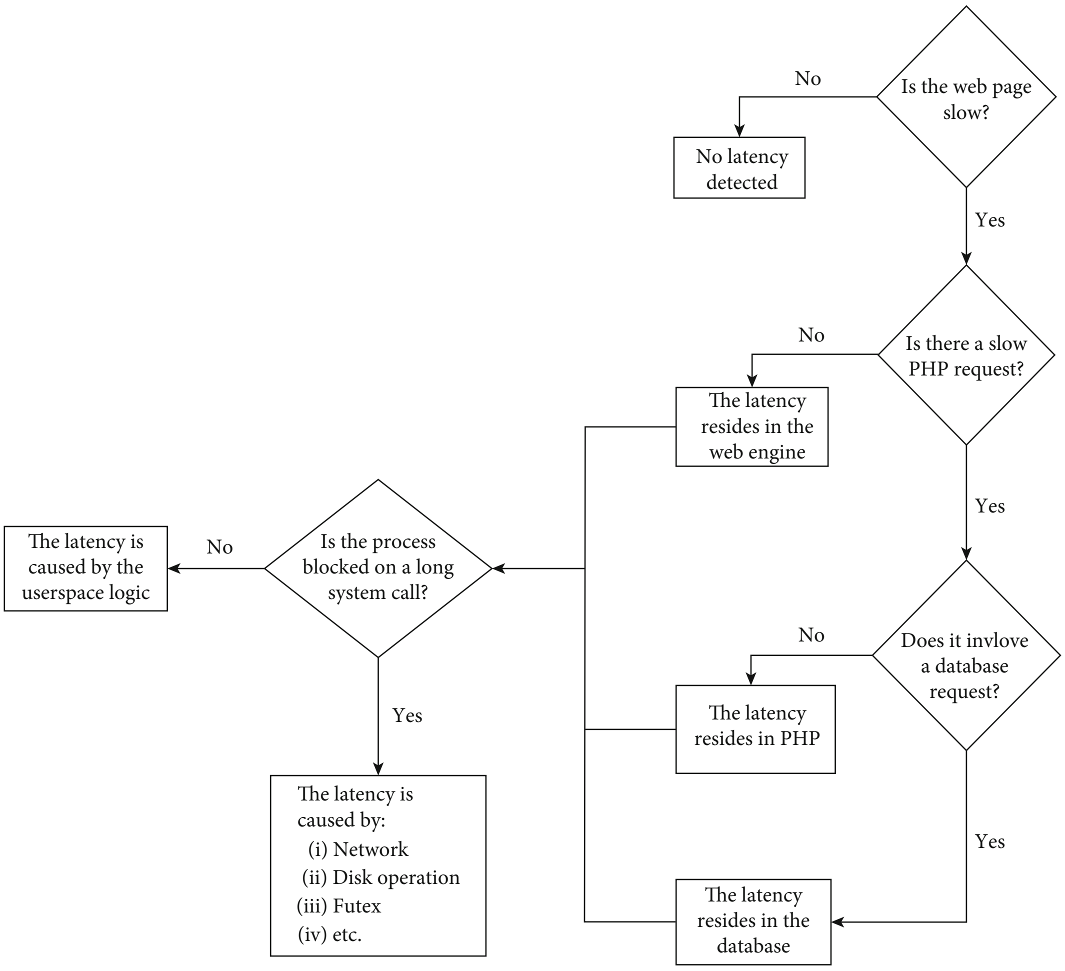

Is the web page slow?
No
No latency detected
Yes
Is there a slow PHP request?
No
The latency resides in the web engine
Yes
Does it invlove a database request?
No
The latency resides in PHP
Yes
The latency resides in the database
Is the process blocked on a long system call?
No
The latency is caused by the userspace logic
Yes
The latency is caused by: (i) Network (ii) Disk operation (iii) Futex (iv) etc.

Figure 9: Decision diagram.

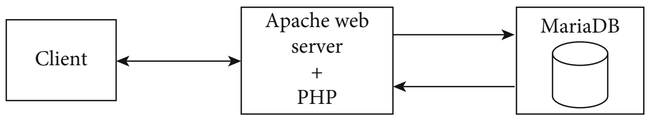


Figure 10: Web server architecture.

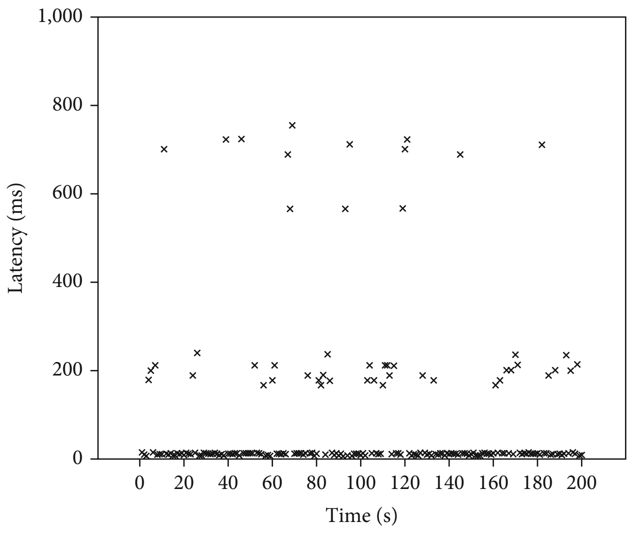


Figure 11: Web server latency.

4. Use Cases

The architecture we are proposing is generic and can detect performance problems in distributed systems, regardless of their precise architecture. It can be a distributed database, a computing cluster, a distributed file system, a distributed web server, etc. In this section, we present a detailed use case where the proposed solution was used to find some performance issues, difficult to detect using other monitoring tools.

The users of a web server complained about some long response times. The architecture of the server was as follows: a first machine runs Apache 2.4.23, and PHP 5.4, and is connected to a remote database server running MariaDB 10.2 (shown in Figure 10).

Our objective is to answer two main questions:

(i) *Base Latency*. Why is the web site always slow?

A web page is supposed to be processed in 20 ms (20 ms being the average response time of other similar web sites on the same hardware), but in our case, the processing takes at least 50 ms.

(ii) *Unusual Latency*. Why is the response time much slower sometimes, but fine at other times?

The processing time is sometimes 10x more than usual.

4.1. Base Latency. A website may be slow for different reasons. It may be a misconfiguration or a problem on the web server, application server, database layer, network layer, or in the host machine operating system. Therefore, finding the root cause without having precise runtime information from each module would be impossible or very difficult. In this use case, we investigated different areas with our proposed tool and identified the root cause of the problem.

We had problems with a website running the MediaWiki (https://www.mediawiki.org) engine, taking 10 to 15 seconds of loading time. Apache logs were only showing that the response times are high, without helping us to pinpoint the problem. MariaDB slow query logs were also showing nothing.

Using our proposed method, we can see the different layers, having enough details at each level to investigate problems. We were able to see the response time for each request, including the portion clasped in each module separately. From this view, we found that the application layer (i.e., PHP) seems to be the bottleneck (or at least the first place to investigate) because it consumed the largest portion of the response time. This leads us to look at the PHP level to see if there is a problem.

Looking at the top function calls for the problematic server showed that all requests go to functions called wp_∗ which are functions from an installed WordPress. It was surprising to see WordPress here because the Mediawiki engine is a completely separate application and does not need any other application like WordPress. More investigations in the analysis data show that a WordPress module is installed to unify the authentication of two related sites, one running Mediawiki and the other WordPress, both being based on PHP sessions.

In other words, each MediaWiki request keeps checking the PHP session to see if there is an authenticated user in WordPress, to let them sign into the Mediawiki engine, which was taking around 70% percent of each request. By changing the module source code and commenting out the part that was continuously checking the session, we could confirm that the problem was because of this module.

Moreover, our tool helped us to uncover problems in other layers as well, like the database layer. We saw that some requests spent around 50% of their time in the database layer. We used our MySQL layer data to dig deeper into these requests.

We found a database connection that had some update queries that take time and look costly. Expanding on this view, we can easily see which queries take a longer time than others. Looking at the queries, we noticed that the update queries are for gathering statistics about the WordPress component of the site. By looking at the installed WordPress plugins, and disabling the statistics plugin, we could solve the problem of these requests as well.

The above use cases, which are only a few samples out of many possible analyses with our tool, show that, with the proposed methods and tools, we can quickly see the modules, requests, pages, and database tables involved in the problem. It then becomes relatively easy to understand the situation and to solve any unexpected behavior by disabling the problematic nonrequired default modules, or by modifying the configuration that had seemed interesting at first but added much latency.

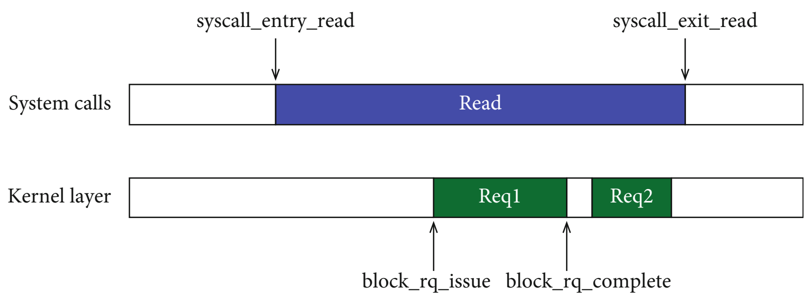


FIGURE 12: Cache miss ratio.

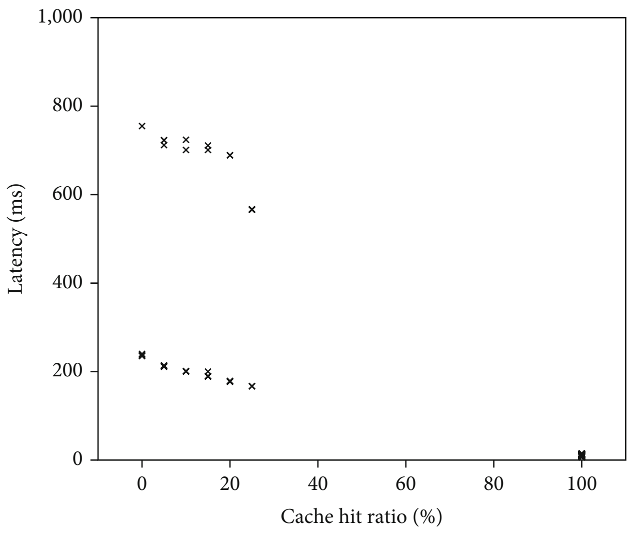


FIGURE 13: Correlation between response time latency and cache hit ratio.

4.2. Unusual Latency. In the previous use case, after detecting and solving the latency problem, we were able to get a much better response time. To ensure that everything was working as expected, we used ApacheBench to stress the website. A scatter chart of response times is presented in Figure 11.

The response time of the server is unstable. Most web requests are processed in 10 ms. However, in some cases, the processing takes much more time. Figure 11 shows three main categories of requests:

(C1) A latency of 10 ms. Most requests belong to this category.

(C2) A latency of (180 ms, 220 ms).

(C3) A latency of (600 ms, 800 ms).

The user-space trace provided by the LAMP tracing module is not enough to understand this problem. More precise information from the operating system is needed for a deeper analysis.

One of the major factors that can affect the speed of the website is the storage device access, whether to process database queries or to load static web pages. Disk caching plays a major role in reducing the frequency of disk accesses, by keeping the required data in memory. Many web servers and database engines implement a user-space cache, and others rely on the operating system cache. User-space caching is generally more efficient since the caching mechanism is tailored by the database itself.

Our tool can generate precise information about cache hits/misses that happen in the context of the webserver, based on system calls and block events. The computation is performed as follows: a read operation starts with the event syscall_entry_read and ends with syscall_exit_read. If all the required data is found in the cache, no disk operation is observed. Otherwise, block device requests will be created to recover the missing information from the disk. The miss ratio is the amount of data read from the disk divided by the amount returned by the *read* system call. For example, the miss ratio of Figure 12 can be computed as follows:

$$\text{MissRatio} = \frac{\text{Size}(\text{Req1}) + \text{Size}(\text{Req2})}{\text{Size}(\text{READ})}. \tag{10}$$

And therefore,

$$\begin{aligned}\text{HitRatio} &= 1 - \text{MissRatio}\\ &= \frac{\text{Size}(\text{READ}) - \text{Size}(\text{Req1}) + \text{Size}(\text{Req2}))}{\text{Size}(\text{READ})}.\end{aligned} \tag{11}$$

Figure 13 shows that there is a clear correlation between the response time and the cache hit ratio. Two cases, with respect to Figure 14, are to be considered:

(1) *Web Requests That Are Fully Served from the Cache.* These requests are the fastest ones and they correspond to category (C1). All the data required is present in the cache and no disk access is required

(2) *Web Requests That Are Partially or Totally Served from the Disk (Cache Hit Ratio between 0% and 30%).* These requests correspond mainly to categories (C2) and (C3).

The cache hit analysis was useful to understand the origin of the slowness of some requests. However, the difference between categories (C2) and (C3) is still unknown. Why do (C3) requests take more time to get the required information from the disk? To answer this question, a more detailed analysis at the level of the storage subsystem should be performed. It is important to understand why an I/O request is sometimes slower than usual. Our tool gives very detailed information about the I/O scheduler, such as the length of the waiting queue and the identity of the processes

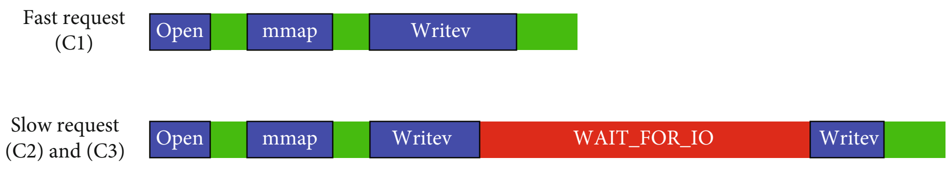


Figure 14: Difference between fast and slow requests.

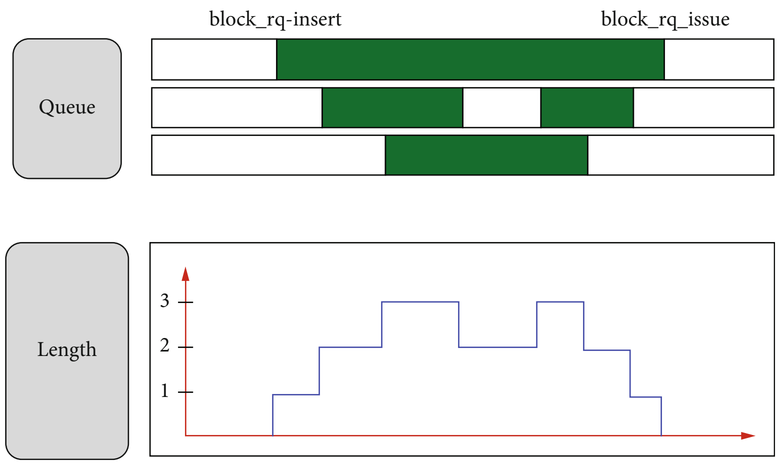


Figure 15: Queue length computation.

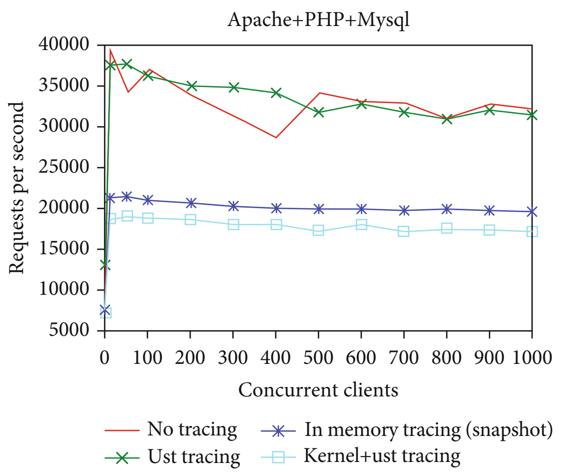


Figure 16: Comparison of kernel and user-space tracing costs.

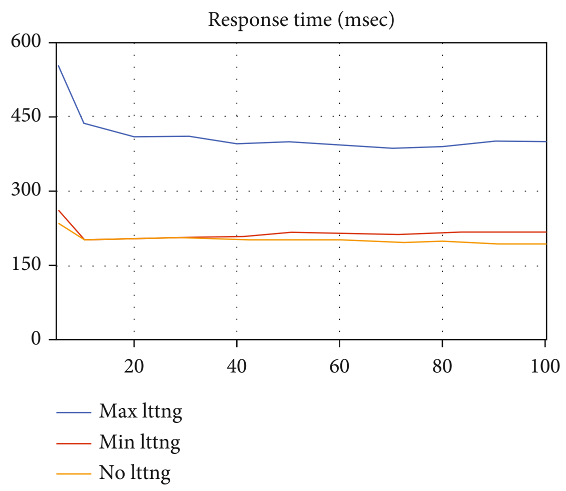


Figure 17: Performance of user-space tracing with 65 million lines of code.

generating I/O requests during a selected time range. Computing the waiting queue is done based on the events *block_rq_insert* and block_rq_complete, as shown in Figure 15.

Our tool was able to show that the difference between (C2) and (C3) is that (C3) requests are processed when the disk drive is busy processing other requests using a process called *backup.sh*. This process is copying server files to a backup medium for recovery purposes in case of failure.

5. Evaluation

It is crucial to minimize the overhead of tracing to avoid having biased results. A high overhead can change the normal behavior of the system, which makes the solution unusable in production systems.

The analysis cost is another important factor that must be taken into account. Building the state system database must be performed in a reasonable time to improve the user experience. The trace files are often very large and inspecting them must be done efficiently.

This section shows the different tests realized to ensure that our tool has low overhead under different circumstances.

5.1. Environment. The tests are executed in a machine with the following configuration.

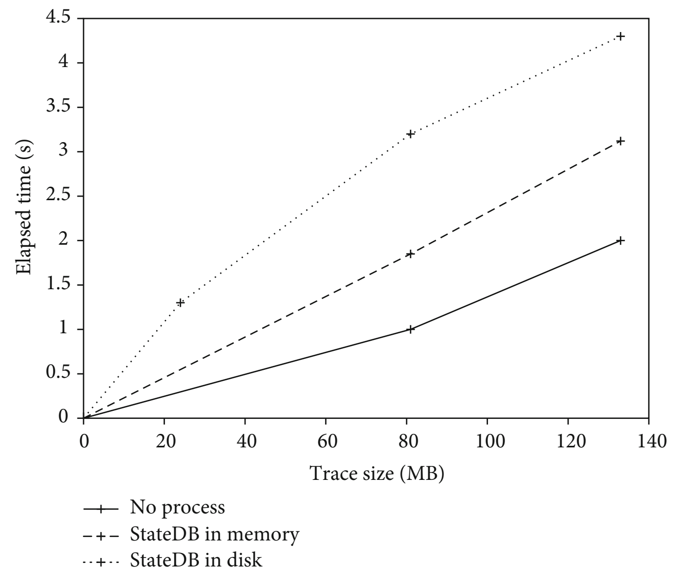


Figure 18: Comparison of state database construction cases.

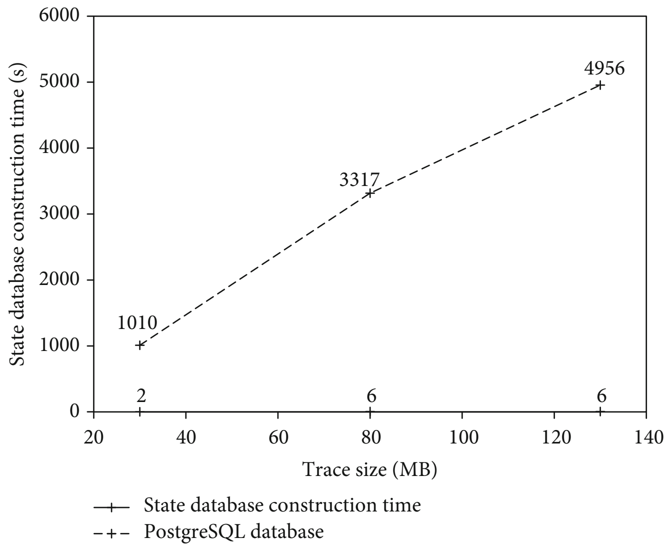


Figure 19: Comparison of construction time against PostgreSQL.

(a) Hardware configuration:

(i) Intel i7-4790 CPU @ 3.60GHz

(ii) 32 GB RAM

(b) Software configuration:

(i) Linux kernel version 4.4

(ii) LTTng 2.7

(iii) Apache 2.4.23, PHP 5.4, and MariaDB 10.2

5.2. Tracing Cost. The performance analysis is achieved using a real configuration composed of an Apache Web server and a MariaDB database. The workloads applied are generated using *ab* (ApacheBench), and the traces are collected using LTTng 2.7. ApacheBench is used to simulate the behavior of many clients navigating throughout the site. The experiment is performed using different numbers of clients (between 1 and 1000) with the following configurations:

(i) *No Tracing*. The tracing is disabled

(ii) *Required Events*. Only events required for the analyses are activated

(iii) *All Events in Memory*. All kernel and user-space events are activated and the trace is kept in memory

(iv) *All Events*. All kernel and user-space events are activated and the trace is written to the disk

The results of the experiment are presented in Figure 16.

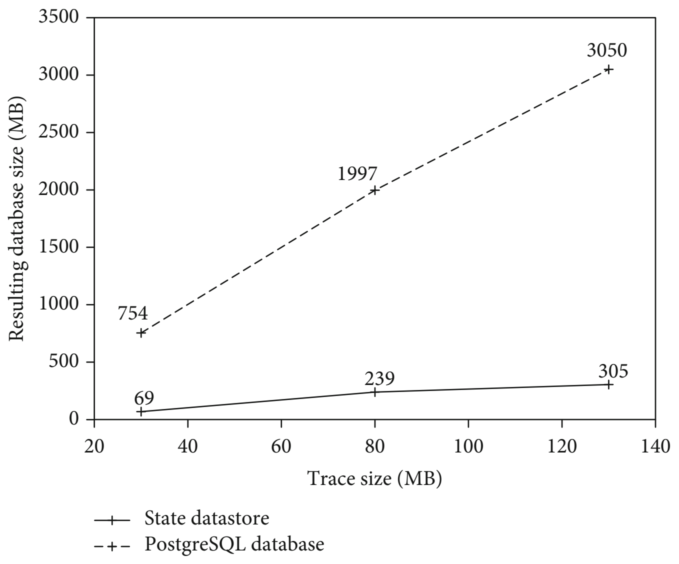


Figure 20: Comparison of disk usage against PostgreSQL.

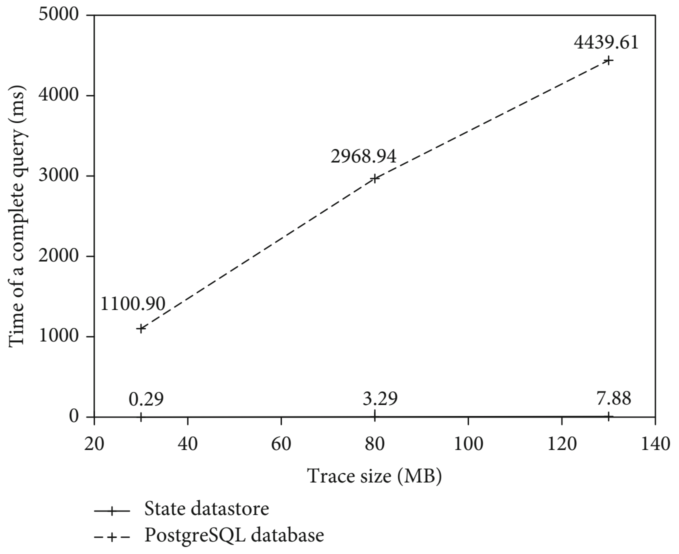


Figure 21: Comparison of full query time against PostgreSQL.

The graph shows that enabling the required events does not have a significant impact on the website's performance. The server can process about 30000 requests per second in both cases. The impact of tracing becomes significant if all tracepoints (including kernel tracepoints) are activated. Kernel tracing appends a lot of system details to the trace and increases the overhead. In the case where the kernel tracing is enabled, the processing speed goes down to 21000 if the trace is written in memory, and 19000 if it is written to the disk. Writing data to the disk does not add much overhead, because it is done asynchronously using a separate process. Please note that in Figure 16, for some points, the ust-tracing acts slightly better than the no-tracing case. This should not have happened if they were running under identical conditions. However, there is always some variability in operating system execution (interrupts, scheduling, memory page faults...). In that case, since the overhead of tracing a relatively small number of ust events is almost negligible, this overhead is difficult to measure, being smaller than the operating system variability.

The experiment is conducted on a Drupal website where serving a web request requires executing 30 k lines of code on average. We repeated the test on a benchmark of 65 million lines of code, to test if executing more lines of code will make a big difference. Figure 17 shows the response time in three different cases: (1) when all UST trace events are enabled, (2) when all trace events are enabled except the PHP line-execution events (ust_PHP:execution_start, ust_PHP:execution_end), and (3) tracing is disabled.

As shown in Figure 17, when there are 65 million lines of code (which is very rare for a web application), the response time is twice as long with tracing, compared to when there is no tracing (or minimum tracing). This is predictable because there are two PHP events activated and triggered for the execution of each PHP line of code, which provides very detailed information but increases the response time. However, when the line execution events are disabled (i.e., the min-LTTng case), the response time becomes almost the same as when tracing is disabled, showing that UST tracing does not impose a significant performance overhead.

5.3. Analysis Cost. As mentioned earlier, the proposed technique uses a tree-based data structure to store the system state, extracted from the raw trace events, at different time points, to be used later in the analysis phase. Since the state database offers a fast query response time ($O(\log n)$), locating the required data within the (usually considerable) trace data becomes faster, leading to efficient analysis and faster graphical rendering. In this section, we evaluate the costs of using this database and compare it to a spatial database like PostgreSQL, which can store multidimensional data on the disk, enabling it to store state intervals for large trace files.

First, we look at the time needed to extract the required state from the trace events and construct the state database. Figure 18 shows the time required to construct the attribute tree and state database. It compares the different cases. Not surprisingly, the fastest time is for the case where the system only reads the trace events without parsing and analyzing them. The second fastest case is when the system reads the traces, analyses them, extracts states but stores them only in memory and not on disk, therefore, no disk I/O is involved. The worst case is when the state database is built and written to disk.

As Figure 18 shows, the I/O required to write the database to disk is the most time-consuming part.

We then compare the construction time of the state database using our solution, with the one using the PostgreSQL database. Figure 19 shows that PostgreSQL takes a much longer time than the state database, especially with larger trace files.

We also compared the disk usage and the full query time between the two databases (Figures 20 and 21). PostgreSQL requires more disk space to save the data and takes more time to serve full queries.

The results above show that even if PostgreSQL provides highly optimized mechanisms to manage generic data sets, it cannot beat the performance of a special-purpose database such as the state database. The state database is tailored to trace files, and it stores the data in ways that make it quickly accessible while using minimal disk space.

6. Conclusion

We have presented a unified analysis method for studying trace data gathered from different layers and sources. The concept is to analyze the collected data, based on the relations between the layers (timestamp, process id, etc.). Our solution extracts the required information from the raw input data and stores it in a multidimensional data store where it can then be analyzed.

This solution has been evaluated by real-world web applications experiencing performance degradation (i.e., MediaWiki, Drupal, and WordPress). For instance, when a website is running slow or when there is an unexpected latency. The solution uses trace data from multiple layers and helps uncovering the root cause(s) of the problem.

Since the proposed solution works by gathering runtime data from different layers; which contains valuable information about the different aspects of the running system; it can be utilized to investigate a wider variety of problems, including network attacks and host-based anomalies. This work would also benefit from a multilevel visualization display of the analyses data. Both will be investigated as future work.

Data Availability

All the data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We would like to gratefully acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC), Prompt, Ericsson, and EfficiOS for funding this project.

References

[1] B. C. Tak, C. Tang, C. Zhang, S. Govindan, B. Urgaonkar, and R. N. Chang, "vpath: precise discovery of request processing paths from black-box observations of thread and network activities," in *In Proceedings of the 2009 Conference on USENIX Annual Technical Conference, USENIX'09*, Berkeley, CA, USA, 2009.

[2] E. Thereska, B. Salmon, J. Strunk et al., "Stardust: tracking activity in a distributed storage system," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, pp. 3–14, 2006.

[3] L. Zhang, D. R. Bild, R. P. Dick, Z. M. Mao, and P. Dinda, "Panappticon: event-based tracing to measure mobile application and platform performance," in *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, Montreal, QC, Canada, 2013.

[4] M. Desnoyers and M. R. Dagenais, "The lttng tracer: a low impact performance and behavior monitor for gnu/linux," *In OLS (Ottawa Linux Symposium)*, vol. 2006, pp. 209–224, 2006.

[5] F. Doray and M. Dagenais, "Diagnosing performance variations by comparing multi-level execution traces," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 462–474, 2016.

[6] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: problem determination in large, dynamic internet services," in *Proceedings International Conference on Dependable Systems and Networks*, pp. 595–604, Washington, DC, USA, June 2002.

[7] R. R. Sambasivan, A. X. Zheng, M. De Rosa et al., "Diagnosing performance changes by comparing request flows," in *In Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, NSDI'11*, pp. 43–56, Berkeley, CA, USA, 2011.

[8] D. J. Dean, H. Nguyen, P. Wang, X. Gu, A. Sailer, and A. Kochut, "Perfcompass: online performance anomaly fault localization and inference in infrastructure-as-a-service clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1742–1755, 2016.

[9] M. Bligh, M. Desnoyers, and R. Schultz, "Linux kernel debugging on google sized clusters," in *Proceedings of the Linux Symposium*, pp. 29–40, Ottawa, Ontario, Canada, 2007.

[10] N. Ezzati-Jivan and M. R. Dagenais, "Multi-scale navigation of large trace data: a survey," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, article e4068, 2017.

[11] H. Daoud and M. Dagenais, "Multilevel analysis of the java virtual machine based on kernel and userspace traces," *Journal of Systems and Software*, vol. 167, article 110589, 2020.

[12] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 684–702, 2009.

[13] I. Kohyarnejadfard, D. Aloise, M. R. Dagenais, and M. Shakeri, "A framework for detecting system performance anomalies using tracing data analysis," *Entropy*, vol. 23, no. 8, p. 1011, 2021.

[14] E. Ates, L. Sturmann, M. Toslali et al., "Association for Computing Machinery," in *An automated, crosslayer instrumentation framework for diagnosing performance problems in distributed applications. In Proceedings of the ACM Symposium on Cloud Computing, SoCC '19*, pp. 165–170, New York, NY, USA, 2019.

[15] B. H. Sigelman, L. Andr'e Barroso, M. Burrows et al., *Dapper, a large-scale distributed systems tracing infrastructure*, Technical report, Google, Inc., 2010, https://research.google.com/archive/papers/dapper-2010-1.pdf.

[16] S. S. Murtaza, A. Sultana, A. Hamou-Lhadj, and M. Couture, "On the comparison of user space and kernel space traces in identification of software anomalies," in *2012 16th European Conference on Software Maintenance and Reengineering*, pp. 127–136, Szeged, Hungary, March 2012.

[17] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Graph summarization methods and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–34, 2018.

[18] B. Gregg, "Visualizing system latency," *Communications of the ACM*, vol. 53, no. 7, pp. 48–54, 2010.

[19] O. Rodeh, H. Helman, and D. Chambliss, "Visualizing block io workloads," *ACM Transactions on Storage*, vol. 11, no. 2, 2015.

[20] A. D. Brunelle, *Block i/o layer tracing:blktrace*http://duch.mimuw.edu.pl/~lichota/09-10/Optymalizacja-open-source/Materialy/10%20-%20Dysk/gelatoICE06aprblktracebrunellehp.pdf.

[21] C. Mason, *Seekwatcher*http://oss.oracle.com/~mason/seekwatcher.

[22] B. Donie, *Ioprof*https://github.com/01org/ioprof.

[23] A. Brunellehttps://usermanual.wiki/Document/bttmanual.1495776143/view.

[24] M. Jabbarifar, M. Dagenais, and A. Shameli-Sendi, "Online incremental clock synchronization," *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 1034–1066, 2015.

[25] *Eclipse trace compass*https://www.tracecompass.org.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 7490689, 12 pages
https://doi.org/10.1155/2021/7490689

Research Article

Energy-Efficient Resource Allocation for NOMA-Enabled Internet of Vehicles

Xin Chen,[1] Zhuo Ma,[1] Teng Ma,[2] Xu Liu,[2] and Ying Chen[1]

[1]*School of Computer Science, Beijing Information Science & Technology University, Beijing, China*
[2]*School of Automation, Beijing Information Science & Technology University, Beijing, China*

Correspondence should be addressed to Xin Chen; chenxin@bistu.edu.cn

Received 7 May 2021; Revised 10 August 2021; Accepted 25 August 2021; Published 16 September 2021

Academic Editor: Yong Zhang

With the rapid development of Internet of vehicles (IoV) technology, the distribution of vehicles on the highway becomes more dense and the highly reliable communication between vehicles becomes more important. Nonorthogonal multiple access (NOMA) is a promising technology to meet the multiple access volume and the high reliability communication demands of IoV. To meet the Vehicle-to-Vehicle (V2V) communication requirements, a NOMA-based IoV system is proposed. Firstly, a NOMA-based resource allocation model in IoV is developed to maximize the energy efficiency (EE) of the system. Secondly, the established model is transformed into a Markov decision process (MDP) model and a deep reinforcement learning-based subchannel and power allocation (DSPA) algorithm is designed. An event trigger block is used to reduce computation time. Finally, the simulation results show that NOMA can significantly improve the system performance compared to orthogonal multiaccess, and the proposed DSPA algorithm can significantly improve the system EE and reduce the computation time.

1. Introduction

With the rapid development of vehicle wireless communication technology, Internet of vehicles (IoV) has a broad development prospect [1]. Among the various applications generated by IoV, security applications are undoubtedly of the highest priority because they impact on the safety of the vehicles directly [2]. Vehicle-to-Vehicle (V2V) communication as a key technology in intelligent transportation system (ITS) that could meet the strict latency and reliability requirements of safety applications has attracted continuous academic attention [3].

V2V communication is aimed at communicating directly between vehicles with extremely low latency and ultrahigh reliability, which could guarantee the quality of service (QoS) requirements of security applications [4]. In general, device-to-device (D2D) communication provides the principle of direct propagate information between adjacent devices, which could greatly reduce latency and transmission energy consumption. Therefore, D2D technology is commonly used as the basis for V2V communication. That is why the 3rd Generation Partnership Project (3GPP) developed V2V communication principles based on D2D technology [5] in the long-term evolutionary (LTE) system. However, it has been shown that the QoS requirements of V2V communication cannot always be guaranteed under this principle. The reason is D2D communication following this principle is based on orthogonal multiple access (OMA) [6], a technology that does not make full use of spectrum resource and has difficulty in solving interference problems due to the increase in vehicles. When vehicles are deployed densely, IoV system would suffer from severe congestion, which affects the performance of the system.

Such problems have been solved with the rise of 5th generation (5G) mobile networks. 5G introduces nonorthogonal multiple access (NOMA) technology that allows a resource block to be assigned to multiple users, thus greatly expanding the amount of access to the network [7]. In some cases, such as uplink communication intensive scenarios, NOMA-enabled system has a significant performance improvement compared to OMA system. The cost of extended access is that NOMA actively introduces interference information and requires reducing the impact of interference by successive interference cancelation (SIC) techniques [8]. Compared to OMA system, NOMA is more complex to decode at the receiver side, but after adopting SIC and other technologies, it is beneficial to the whole system performance. SIC technology decodes the received signal level by level and removes it after successful decoding to reduce the interference to the undecoded signal. In NOMA-enabled IoV system, the performance of V2V communication can be significantly improved.

Due to its advantages over OMA, NOMA is widely used in ultradense network (UDN), mobile edge computing (MEC), IoV, and other environments [9, 10]. Currently, NOMA has great potential to expand network access and improve network performance, but there are still some issues that need to be addressed. There have been many works introducing NOMA technology for resource allocation and interference management. In these works, the optimization of system throughput and the QoS requirements for V2V communication have been mainly considered. However, NOMA extends the number of user accesses through channel multiplexing, which increases the difficulty of channel allocation. In addition, the power allocation scheme becomes more complex due to the interference introduced by NOMA, and the overall system power consumption should be considered in the resource allocation scheme. Besides, literature [11] has analyzed the SIC technique and pointed out that, due to the complexity of implementation, normally two users could share the same subchannel at most.

To solve the above problem, we study the resource allocation problem for high energy efficiency (EE) in IoV systems. We describe the scenario of the NOMA-enabled IoV system and present the resource allocation problem for maximizing the system EE. Due to the complexity of the system and the high computational dimensionality of the direct solution, we transform the optimization problem into a Markov decision process (MDP) and use deep reinforcement learning (DRL) method to solve it. The main contributions of this paper are as follows:

(i) We investigate the problem of resource allocation in IoV system. The NOMA technology is introduced to meet the demand for multivehicle access, and the implementation of uplink SIC technology is presented. By allocating the channel and power resources of vehicles, we propose an optimization goal of maximizing the system EE

(ii) We transform the optimization goal into a problem of resource allocation strategy based on MDP and propose a DRL-based subchannel and power allocation (DSPA) algorithm to solve it. Specifically, the deep Q network (DQN) method is used to solve the subchannel selection and the deep deterministic policy gradient (DDPG) method is used to solve the power allocation problem. The event trigger block is used to reduce the computation time

(iii) We simulate and analyze the designed algorithm. The simulation results show that the performance of the NOMA-enabled IoV system is more suitable for multiple vehicle access situations than OMA, and the DSPA algorithm can effectively enhance the system EE and reduce the computation time

The rest of this paper is organized as follows. In Section 2, we analyze the work related to this paper. The system model and problem formulation are given in Section 3. In Section 4, we transform the optimization problem into an MDP model and design the DSPA algorithm for solving it. In Section 5, the proposed resource allocation method is simulated and analyzed. Section 6 is the conclusion.

2. Related Work

Due to the variability of QoS requirements for vehicle users, the resource allocation problem in vehicle networks has attractive research value and has received extensive attention from researchers for years [12, 13]. Since the high speed movement of vehicles in IoV makes it difficult to obtain accurate and fast channel change information, Guo et al. [14] obtained the time delay of V2V link in steady state based on Markov process and determine the optimal transmit power for each possible spectrum and finally allocated the spectrum resource by dichotomous matching method to maximize the system data transmission rate. Chen et al. [15] developed an online network slice resource allocation strategy that can meet the demand for QoS requirements for IoV applications and maximize system capacity. Liang et al. [16] designed a multi-intelligent DQN algorithm to allocate spectrum and power for each V2V link and maximize the total system throughput. Yang et al. [17] studied the design frame structure for V2V communication in IoV and proposed a semipersistent frame scheduling algorithm, which greatly meets the needs of V2V communication.

Resource allocation for IoV system can also be combined with MEC. Chen et al. [18] considered the dynamics of computational task arrival and wireless channel state in the MEC scenario and jointly optimized task and computational resource allocation to minimize system energy consumption while guaranteeing the upper limit of queue length. Zhao et al. [19] studied the collaborative offloading strategy of edge clouds in IoV and designed a distributed computational offloading and resource allocation algorithm to optimize the joint benefits of offloading and resource allocation. The problem of joint allocation of spectrum, computation, and storage resources in MEC-based IoV was studied by Peng et al. [20]. Since the problem has a high computational complexity, the authors transformed the problem using reinforcement learning (RL) method and solved it with a hierarchical learning architecture to obtain the optimal resource allocation decision.

By introducing NOMA technology in IoV scenery, the system performance will be further improved. Di et al. [21] proposed a resource allocation scheme in IoV broadcast scenarios, using NOMA to reduce latency and improve data acceptance probability. The main idea of this scheme is a centralized channel selection strategy and a distributed power allocation strategy. The packet reception probability is significantly improved by this scheme. Liu et al. [22]

studied the optimal power allocation problem in broadcast and multicast transmission schemes in half-duplex NOMA-based IoV scenarios and proposed a bifurcation-based power allocation algorithm that significantly improves the system throughput compared with the OMA scheme.

3. System Model and Problem Formulation

3.1. System Model. We consider a multivehicle highway scenario where one base station is located at the center and the radius of the base station coverage is D, as shown in Figure 1. The time domain is uniformly divided into multiple time slots, and the length of each slot is τ. We denote m as the index for the m-th moving vehicle on the highway where $m \in \{1, 2, \cdots, M\}$, and the maximum travel speed of the vehicle is $v_{\max}$. At each time slot t, there are N $(N < M)$ vehicles that send the required security information to the surrounding vehicles within its communication range through up to one subchannel. Such communications are based on V2V communication, and this transmission vehicles are denoted as VT user; the set of all VT users is $\mathcal{N}$. During each time slot t, the number of VT users $|\mathcal{N}^{(t)}|$ obeys a Poisson distribution

$$\Pr\left\{\left|\mathcal{N}^{(\mathrm{t})}\right| = n\right\} = \frac{(\alpha_{\mathrm{VT}}\tau)^n}{n!} \exp(-\alpha_{\mathrm{VT}}\tau), \quad n = 0, 1, 2, \cdots, \tag{1}$$

where α_{VT} denotes the arrival intensity of VT users in terms of VTs per second.

A right-angle coordinate system is established with the base station as the origin, and the position of each vehicle is denoted by (a_m, b_m). All vehicles are traveling in one direction with speed v_m, and the coverage radius of V2V communication is $d_{\max}$. The total available bandwidth for the D2D communication is W_{all} and is divided equally into K nonorthogonal subchannels, each bandwidth $W = W_{\mathrm{all}}/K$.

Due to the dense vehicle coverage, when multiple VT users send messages through the same subchannel simultaneously, the receiving vehicles (denoted as VR) located in the common coverage area of these VT users may receive messages with large interference. NOMA allows multiple vehicles to transmit information through the same channel simultaneously, and the VR users use SIC technology to decode the received information and reduce the cochannel interference.

We denote $\mathbb{N}_l$ as the set of all VT users that can be received by the receiving vehicle VR$_l$, i.e., $\mathbb{N}_l = \{1 \le n \le N \mid d_{n,l} \le d_{\max}\}$, where $d_{n,l} = \sqrt{(b_n - b_l)^2 + (a_n - a_l)^2}$ is the distance between VT$_n$ and VR$_l$. In time slot t, the signal received by the receiving vehicle VR$_l$ on subchannel k (SC$_k$) is

$$y_{l,k}^{(t)} = \sum_{n\in\mathbb{N}_l} \alpha_{n,k}^{(t)} \sqrt{p_{n,k}^{(t)}} h_{n,l,k}^{(t)} s_n^{(t)} + z_l^{(t)}, \tag{2}$$

where $\alpha_{n,k}^{(t)}$ is a binary variable that indicates the subchannel selected by VT$_n$. Specifically, $\alpha_{n,k}^{(t)} = 1$ if VT$_n$ transmits through SC$_k$, and $\alpha_{n,k}^{(t)} = 0$ otherwise. $p_{n,k}^{(t)}$ is the transmitted power of VT$_n$ in time slot t, $s_n^{(t)}$ denotes the modulation symbol, and z_l^t represents the additive white Gaussian noise (AWGN) for VR$_l$ which obeys the complex Gaussian distribution with variance σ_l^2, that is, $z_l^{(t)} \sim \mathscr{CN}(0, \sigma_l^2)$. $h_{n,l,k}^{(t)}$ denotes the coefficient of SC$_k$ from VT$_n$ to VR$_l$. Specifically, $h_{n,l,k}^{(t)} = g_{n,l}^{(t)} \cdot \mathrm{PL}^{-1}(d_{n,l}^{(t)})$, where $g_{n,l}^{(t)}$ denotes Rayleigh fading channel gain, and $\mathrm{PL}^{-1}(d_{n,l}^{(t)}) = \beta(d_{n,l}^{(t)})^{-\varphi}$ represents the path loss function with the shadowing component β and the power decay exponent φ.

We map the mobility of the vehicle to the change in the position of the vehicle. Since there is short length of time slots, it can be assumed that the position of the vehicles in time slot t does not change, so the distance of any two vehicles $d_{m,m'}$ remains constant in time slot t. The position of the vehicle needs to be recalculated at the beginning of the time slot $t+1$. According to Equation (2), the distance between vehicles is further mapped to the change in channel gain, so we assume that the channel gain also remains within one time slot, while it changes in the adjacent time slots. Thus, the SINR between VT$_n$ and VR$_l$ over SC$_k$ in time slot t without SIC technology can be expressed as

$$\Gamma_{n,l,k}^{(t)} = \frac{p_{n,l}^{(t)}\left|h_{n,l,k}^{(t)}\right|^2}{\sigma_l^2 + \sum_{n'\in\mathbb{N}_l, n'\neq n} p_{n',l}^{(t)}\left|h_{n',l,k}^{(t)}\right|^2}, \tag{3}$$

where $\sigma_l^2 = E[|z_l^{(t)}|^2]$ is the noise power on SC$_k$ and $|h_{n,l,k}^{(t)}|^2$ is the channel gain. The data rate of SC$_k$ between VT$_n$ and VR$_l$ without SIC technique can be expressed as

$$R_{n,l,k}^{(t)} = W \cdot \log_2\left(1 + \Gamma_{n,l,k}^{(t)}\right). \tag{4}$$

In the uplink NOMA system, the superimposed signal $y_{n,l}^{(t)}$ received by VR$_l$ needs to have a certain clarity between the different signals in order to eliminate interference. Since the channels between each VT$_n$ and VR$_l$ are different, the signals sent by each VT user in the uplink experience a different channel gain. Therefore, among the superimposed signals $y_{n,l}^{(t)}$, the VT user with the best channel quality may have the strongest received power, and VR$_l$ decodes this VT signal first, i.e., the decoding order of VR$_l$ is from VT users with good channel quality to those with poor channel quality. Otherwise, it has to allocate higher power for VT users with poor channel quality to improve their received power, which will reduce EE. Assuming that there are N VT users sending messages to VR$_l$ over SC$_k$ and the order of the channel gains between each VT user and VR$_l$ is

$$\left|h_{1,l,k}^{(t)}\right|^2 \ge \left|h_{2,l,k}^{(t)}\right|^2 \ge \cdots \ge \left|h_{n,l,k}^{(t)}\right|^2 \ge \left|h_{n+1,l,k}^{(t)}\right|^2 \ge \cdots \ge \left|h_{N,l,k}^{(t)}\right|^2. \tag{5}$$

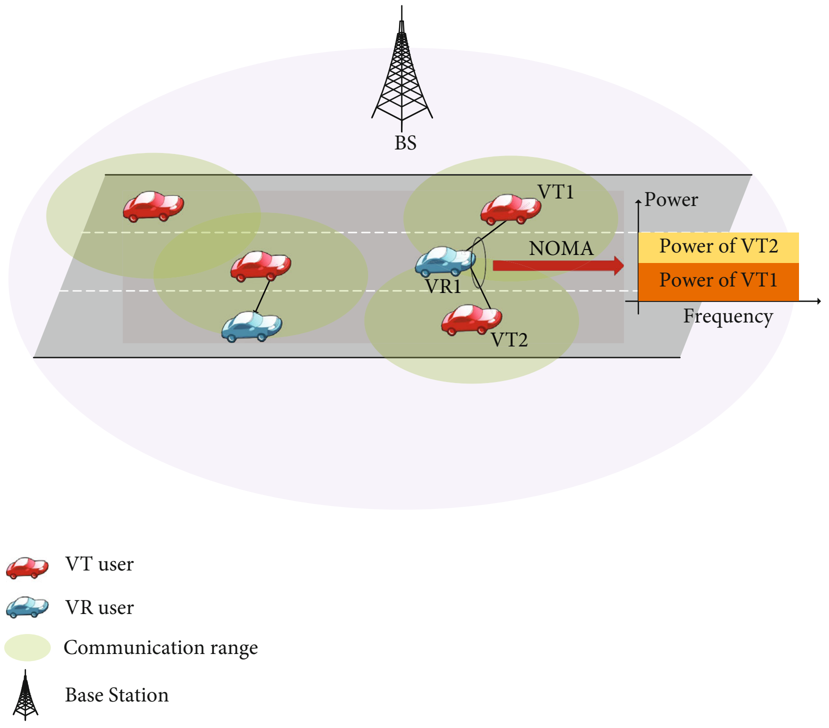


Figure 1: NOMA-based IoV system scenario.

According to the SIC decoding rules, VR_l firstly decodes VT users with $n' < n$ and eliminates $\mathrm{VT}_{n'}$ interference symbols when decoding VT_n, but not eliminate $\mathrm{VT}_{n''}(n'' > n)$ interference symbols. Therefore, the SINR between VT_n and VR_l over SC_k in time slot t with SIC technology can be expressed as

$$\widetilde{\Gamma_{n,l,k}^{(t)}} = \frac{p_{n,l}^{(t)}\left|h_{n,l,k}^{(t)}\right|^2}{\sigma_l^2 + \sum_{n' \in \mathbb{N}_l'} p_{n',l}^{(t)}\left|h_{n',l,k}^{(t)}\right|^2}, \quad (6)$$

where $\mathbb{N}_l' = \{n' \in \mathbb{N} \mid |h_{n',l,k}^{(t)}| < |h_{n,l,k}^{(t)}|\}$ represents a set of interfering VT users.

Considering the QoS requirements of VT users, VR_l can successfully decode the information delivered by VT_n through subchannel SC_k which also needs to satisfy the transmission rate $R_{n,l,k}^{(t)}$ not below the rate threshold, i.e., $R_{n,l,k}^{(t)} \geq R_{\min}$. Otherwise, VR_l will not be able to decode the information. We assume that the transmission rate $R_{n,l,k}^{(t)} = 0$ in this case. Then, the data rate of SC_k between VT_n and VR_l can be expressed as

$$R_{n,l,k}^{(t)} = \begin{cases} W \cdot \log_2\left(1 + \widetilde{\Gamma_{n,l,k}^{(t)}}\right), & R_{n,l,k}^{(t)} \geq R_{\min}, \\ 0, & \text{otherwise}. \end{cases} \quad (7)$$

Therefore, the total rate of the NOMA-enabled IoV system in time slot t can be expressed as

$$R^{(t)} = \sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{n\in\mathbb{N}_l} R_{n,l,k}^{(t)}, \quad (8)$$

where L is the sum of VR users in time slot t.

SIC technique in NOMA-enabled IoV system has been investigated in [11]. At the VR side, as the maximum number of VT users who are multiplexing the same subchannel increases, the difficulty of SIC technology increases dramatically. To avoid excessive SIC complexity for VR users, in this paper, we assume that each VT user delivers information to at most one VR user during each slot. In addition, it also reduces transmission errors.

3.2. Problem Formulation. In NOMA-enabled IoV system, data transmission rate and system power consumption are both important parameters to measure system performance. Our goal is to minimize the overall power consumption of all VT users while maintaining the system transmission rate, i.e., transmitting more bits per unit Joule. Therefore, we set the optimization objective as the ratio of the overall transmission rate to the total transmit power of VT users, i.e., EE, which can be expressed as

$$\mathrm{EE}^{(t)} = \frac{R^{(t)}}{P_{\mathrm{sum}}^{(t)} + P_c}, \quad (9)$$

where $P_{\text{sum}}^{(t)} = \sum_{k=1}^{K} \sum_{n=1}^{N} p_{n,k}^{(t)}$ denotes the sum transmitted power for all VT users in time slot t and P_c is additional circuit power consumption.

Thus, the optimization problem can be expressed mathematically as

$$\begin{aligned} &\max_{\left\{\alpha_{n,k}^{(t)}, p_{n,l,k}^{(t)}\right\}} \mathrm{EE}^{(t)} \\ &\text{s.t.} \quad \mathrm{C1}: \quad \sum_{k=1}^{K} \left(\alpha_{n,k}^{(t)} + \alpha_{n',k}^{(t)}\right) \leq 1 \\ &\left\{n, n'\right\} \in \left\{1 \leq n, n' \leq N \mid d_{n,n'}^{(t)} < d_{\max}\right\} \\ &\mathrm{C2}: \quad \sum_{n=1}^{N} \alpha_{n,k}^{(t)} \leq U_{\max}, \quad \forall k \in \mathcal{SC} \\ &\mathrm{C3}: \quad \alpha_{n,k}^{(t)} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \forall \mathrm{SC}_k \in \mathcal{SC} \\ &\mathrm{C4}: \left|\mathbb{L}_n^{(t)}\right| \equiv 1, \quad \forall n \in \mathcal{N} \\ &\mathrm{C5}: \quad 0 \leq \sum_{k=1}^{K} p_{n,k}^{(t)} \leq P_{\max}^{VT}, \quad \forall n \in \mathcal{N}. \end{aligned} \tag{10}$$

Constraint C1 indicates that two vehicles within the communication range cannot pass messages to each other, i.e., VT_n cannot pass messages to $\mathrm{VT}_{n'}$ within its communication range. This is because of the half-duplex nature that no two vehicles can receive a message at the same time as it is passed, according to [21]. To reduce the SIC complexity at the receiver side, we assume that each subchannel SC_k is multiplexed by at most $U_{\max}$ VT users and that each VT user delivers information to at most one VR user within its communication range during slot t, which are reflected in constraints C2, C3, and C4. Constraint C5 limits the threshold of transmit power for VT users.

4. DRL-Based Subchannel and Power Allocation Algorithms

The optimization problem in (10) is nonconvex and NP hard, which has a complex system with high computational dimensionality. The problem requires exponential levels of time complexity for direct computation of all possible subchannel selections and power allocations, which is difficult to implement in practice. Therefore, we use reinforcement learning methods to select the subchannel selection and power allocation strategies of maximizing EE. We first transform the resource allocation problem in NOMA-enabled IoV system into an MDP-based resource allocation problem and then solve the model using DRL methods.

4.1. Optimize Problem Conversion. In the proposed NOMA-enabled IoV system, the system state in each time slot $t+1$ depends only on the actions, including subchannel selection and power allocation, made by the VT users in time slot t. Therefore, we transform the developed model for maximizing EE into a resource allocation model based on MDP and then solve it through the DRL method. The state space **S**, action space **A**, and reward **R** of the MDP model are defined below, respectively.

4.1.1. State Space. The system state information can be described jointly by the system data transmission rate and the energy consumption. Thus, the system state space **S** includes the transmission rates between all VT users and the corresponding VR users, as well as the transmission power of all VT users, and this information is the basis for this resource allocation. Since we assume that each VT user transmits information to only one VR user, during time slot t, the state $s_t \in \mathbf{S}$ can be expressed as follows:

$$s_t = \left\{R_1^{(t)}, R_2^{(t)}, R_3^{(t)}, \cdots, R_N^{(t)}, p_1^{(t)}, p_2^{(t)}, p_3^{(t)}, \cdots, p_N^{(t)}\right\}. \tag{11}$$

4.1.2. Action Space. The action space **A** includes all possible subchannel choices for each VT user, $\alpha_{n,k}^{(t)}$, as well as the choice of transmit power, $p_{n,k}^{(t)}$. In time slot t, action a_t can be expressed as

$$\begin{aligned} a_t &= \left\{a_t^1, a_t^2\right\}, \\ a_t^1 &= \left\{\alpha_{1,1}^{(t)}, \cdots, \alpha_{n,k}^{(t)}, \cdots, \alpha_{N,K}^{(t)}\right\}, \\ a_t^2 &= \left\{p_{1,1}^{(t)}, \cdots, p_{n,k}^{(t)}, \cdots, p_{N,K}^{(t)}\right\}. \end{aligned} \tag{12}$$

4.1.3. Reward. We denote the reward for selecting the action a_t under state s_t as EE of the current system, which can be calculated by Equation (9). Specifically, for $r_t \in \mathbf{R}$, it can be expressed as

$$r_t = \mathrm{EE}^{(t)}. \tag{13}$$

The goal of reinforcement learning is to find the optimal policy π^* through multiple iterations to achieve the maximum long-term discounted reward

$$R_t = \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots\right] = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right], \tag{14}$$

where $\gamma \in [0, 1)$ is the discount factor. When γ is equal to 0, only the current reward has been considered, while the subsequent has been ignored. As γ increases, the system will focus more with long-term discount rewards.

The reward function can be set to satisfy the requirement of receiving a higher reward when the agent chooses to perform an action that makes the system EE larger and otherwise receives a lower reward or even receives zero reward. After several rounds of iterations, the agent will gradually choose the policy that can obtain higher rewards, i.e., a better resource allocation policy.

4.2. Event Trigger. The framework of the proposed DSPA algorithm is shown in Figure 2. During the process of interacting with the environment, the agent selects and executes an action a_t based on the environment's current state s_t,

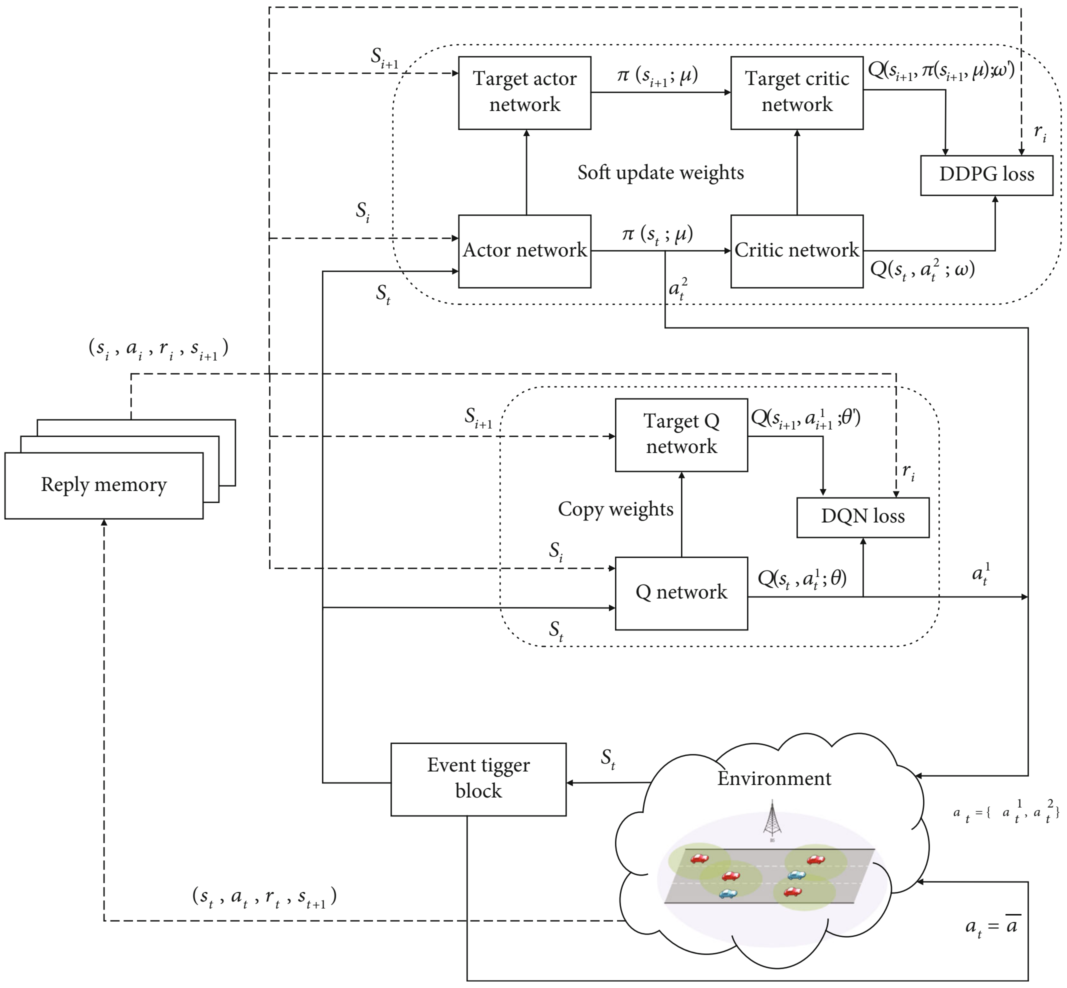


Figure 2: DSPA algorithm framework.

after which the state s_t becomes state s_{t+1}, and the agent gets a reward r_t given by the environment. Then, the agent executes a new action a_{t+1} according to a certain policy π based on the new state and the reward. After a long iterative process, the agent will get an optimal policy π^* that earns the most reward.

Policy π is a mapping of the state space $\mathbf{S}$ to the action space $\mathbf{A}$. Specifically, $\pi = \mathbf{S} \longrightarrow \mathbf{A}$. Considering the state-action value function of the action $Q : \mathbf{S}\times\mathbf{Q} \longrightarrow R$ that represents the expected reward for performing action a with policy π in state s, i.e.,

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\left\{\sum_{i=0}\gamma^i r_{t+i} \mid s_t=s, a_t=a\right\} \\ &= \mathbb{E}_\pi\{r_t+\gamma_{t+1}+\gamma^2 r_{t+2}+\cdots | s_t=s, a_t=a\} \\ &= \mathbb{E}_\pi\{r_t+\gamma Q^\pi(s_{t+1}+a_{t+1}) \mid s_t=s, a_t=a\} \\ &= \sum_{s'} P\left(s,a,s'\right)\left(R\left(s,a,s'\right)+\gamma Q^\pi\left(s',a'\right)\right). \end{aligned} \quad (15)$$

For the established MDP model, the ultimate goal is to find an optimal policy π^* that can be satisfied as $Q^{\pi^*} \geq Q^\pi$ for all policy π. The optimal action-value function can be expressed as

$$Q^*(s,a) = \sum_{s'} P\left(s,a,s'\right)\left(R\left(s,a,s'\right)+\gamma \max_{a'} Q^*\left(s',a'\right)\right). \quad (16)$$

Equation (16) is the Bellman equation, which indicates that when the agent makes an optimal decision, the obtained Q value must be the expected reward for the optimal action in that state.

For the MDP model, the schemes to obtain the optimal policy π^* mainly include model-based approaches and model-free approaches. Since a part of the prior knowledge, such as transfer probability, is unknown in the NOMA-enabled IoV system, it is necessary to use the model-free approach RL to obtain statistical information of the unknown model. DRL combines RL with deep neural networks (DNN) and solves high-dimensional state and action space problems by DNN, which is widely used in IoV systems.

However, solving the MDP model using the DRL method is still time costly, as it takes more time to update the neural network weight parameters, generate the actions, and calculate the rewards. Several methods have been proposed for reducing the computation time. In [23], the authors propose an event trigger module, which is a controller that updates the neural network parameters only when the system state deviates from a certain level. Such method can effectively reduce the computation time, so we introduce it into our DSPA algorithm.

In NOMA-enabled IoV systems, there may be two adjacent time slots in which the system states are similar or even identical, and then, the action selection corresponding to these two states should also be the same. So when the DNN outputs the action in the first time slot, the same action in the next time slot can be executed directly without the DNN. Referring to Lemma 1 in [24], we give a proof for this consideration.

Theorem 1. *For two consecutive states s_t and s_{t+1}, their corresponding optimal actions a_t and a_{t+1} should be the same when $s_t = s_{t+1}$.*

Proof. According to Equation (16), after obtaining the optimal state-action value function $Q^*(s, a)$ for all states, by using the greedy strategy, the optimal actions a_t and a_{t+1} corresponding to states s_t and s_{t+1} can be expressed as

$$\begin{aligned} a_t^* &= \arg \max_{a\in\mathbf{A}} Q^*(s_t, a), \\ a_{t+1}^* &= \arg \max_{a\in\mathbf{A}} Q^*(s_{t+1}, a), \end{aligned} \tag{17}$$

where $\mathbf{A}$ represents the action space of two actions. Assuming that $s_t = s_{t+1}$, we can obtain

$$a_t^* = \arg \max_{a\in\mathbf{A}} Q^*(s_t, a) = \arg \max_{a\in\mathbf{A}} Q^*(s_{t+1}, a) = a_{t+1}^*, \tag{18}$$

which proves our assumption. □

Based on the above assumption, we add the event trigger module into the DRL framework as a way to decide whether to output new actions by using the neural network. Specifically, the previous state $\bar{s}$ and the corresponding action $\bar{a}$ are stored in the event trigger. The new state s_t is firstly compared with $\bar{s}$; if the difference between the two is less than a certain threshold, $\bar{a}$ is directly output as the action of state s_t. Otherwise, the DNN outputs the action a according to state s_t, and $\bar{s}$ and $\bar{a}$ are replaced with s_t and a_t. Using the binary variable ζ as the event trigger decision, specifically,

$$\zeta = \begin{cases} 1, & \|s_t - \bar{s}\|^2 \geq \rho, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

where ρ is the threshold, $\zeta = 1$ means outputting action a_t through the neural network, and $\zeta = 0$ means obtaining action $\bar{a}$ stored in the event trigger.

4.3. DRL-Based Resource Allocation Framework. In the proposed DSPA algorithm, the subchannel selection action, i.e., a_t^1 in Equation (12), is obtained by the DQN method. Since the transmission power is a continuous interval, we use the DDPG method for power allocation, i.e., a_t^2 in Equation (12).

4.3.1. DQN-Based Subchannel Selection Method. In the DQN algorithm, the Q function is approximated by DNN and the Q value is approximated by the DNN weight parameter θ. The Q value is updated by minimizing the loss function to update the parameter θ; the loss function can be defined as

$$L(\theta) = \mathbb{E}\left[(\text{Target}Q - Q(s, a, \theta))^2\right], \tag{20}$$

where

$$\text{Target}Q = r(s, a) + \gamma \max_{a'} Q\left(s', a', \theta'\right). \tag{21}$$

According to Equations (20) and (21), the gradient descent method can be used to solve for the weight parameter θ. DQN uses the current network to evaluate the current value function and uses the target network to generate the target value in Equation (21). The combination of these two networks can decouple the current Q value and the target Q value to some extent, which in turn improves the stability of the algorithm.

The DQN algorithm further introduces an experience replay mechanism to solve the problem of high sample coupling. At each step, the data of the intelligent body interacting with the environment, i.e., the current state s, action a, reward r, and next state s', are stored in the experience pool. The data can later be drawn from the experience pool for training.

The introduction of the experience replay mechanism makes it easier to store the feedback data and allows training samples to be drawn by random sampling, reducing the high coupling between samples. Furthermore, this mechanism can also solve the problems of nonindependent correlation and nonstationary distribution among data in reinforcement learning, which reduces the convergence difficulty of the network model.

4.3.2. DDPG-Based Power Allocation Method. The DQN method is able to solve large-scale state space problems, but its limitation is that it can only solve discrete action space problems, so it is not feasible to use the DQN method to make choices in continuous power intervals. For this case, we use the DDPG method for power allocation. DDPG is a DRL method based on value function and policy gradient, which can effectively solve the problem of high-dimensional and continuous action space. The method generates a deterministic action directly through a DNN network named actor, i.e.,

$$\mu(s_t \mid \omega^\mu) \approx \mu^*(s_t), \tag{22}$$

where $\mu^*(s_t)$ is the optimal behavior policy and ω^μ is the parameter of actor network. The resulting actions are then evaluated by a DNN network called critic, with the aim of minimizing the loss function. The loss function is

$$L\left(\omega^Q\right) = \mathbb{E}\left[\left(Q\left(s_t, a_t \mid \omega^Q\right) - y_t\right)^2\right], \tag{23}$$

where

$$y_t = r_t + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1} \mid \omega^{\mu'}\right) \mid \omega^{Q'}\right). \tag{24}$$

Similar to DQN, two independent target networks, namely, the target actor network and the target critic network, are introduced to further improve the stability of learning. The parameters of the target network are related to the current network and updated in real time, with the update criterion

$$\begin{aligned} \omega^{Q'} &= \delta\omega^Q + (1-\delta)\omega^{Q'}, \\ \omega^{\mu'} &= \delta\omega^\mu + (1-\delta)\omega^{\mu'}, \end{aligned} \tag{25}$$

where $\delta \ll 1$ is used to limit the change rate of the target value and improve the stability of DNN training.

Based on the above theory, the DSPA algorithm in the NOMA-enabled IoV system is shown in the algorithm.

5. Simulation Experiments and Analysis

5.1. Simulation Environment. In this section, we conduct simulation experiments on the proposed resource allocation scheme and analyze the results. The simulation experiments are conducted on Windows 10 operating platform with Intel i5-8300H CPU, NVIDIA 1050Ti GPU, and 16 G memory size and based on Python 3.7 and use the TensorFlow 1.13 framework. All networks contain two hidden layers with 128 and 64 neurons, respectively. Following the 3GPP standard and existing studies, we set the parameters to meet the simulation requirements of the NOMA-enabled IoV system, as shown in Table 1.

5.2. Parameter Analysis

5.2.1. Learning Rate. In the DSPA algorithm, the learning rate is an extremely important hyperparameter. Generally speaking, the larger learning rate, the faster convergence speed, but will ignore the optimal solution due to premature convergence, and the convergence value is normally lower than the global optimal value. As the learning rate approaches zero, the speed of obtaining the optimal policy π^* decreases gradually and could not obtain the optimal solution quickly. This is because the learning rate controls the size of the optimization gradient step, too large learning rate will lead to too large gradient step, ignoring the optimal solution, while too small learning rate will lead to too small step, requiring more time to converge. Therefore, it is first necessary to choose a suitable learning rate.

We set the values of learning rate as 0.1, 0.01, and 0.001, respectively. The simulation results are shown in Figure 3. When the learning rate is 0.1, the algorithm obtains the maximum EE of 2.8 Mbit/Joule after about 400 iterations. The EE after convergence is not much different between learning rates 0.01 and 0.001, both of which are about 3.2 Mbit/Joule. However, the optimal value is obtained after 500 iterations with the learning rate of 0.01, while the learning rate of 0.001 requires 700 rounds of iterations. In order to take into account the convergence speed and quality, we set the learning rate to 0.01 in the following simulation.

5.2.2. Discount Factor. Figure 4 shows the impact of different discount factors on the convergence of the system EE. We set the values of the discount factor γ as 0.1, 0.5, and 0.9, respectively. As the number of iterations increase, the system EE gradually leveled off. The system EE for each of the three discount factors is maximized after about 500 iterations, when the EE is 3.0 Mbit/Joule, 3.1 Mbit/Joule, and 3.2 Mbit/-Joule, respectively. The comparison leads to the conclusion that the smaller γ, the more system focuses on the current reward, and the larger the γ, the more system focuses on the long-term reward. Our goal is to maximize the long-term discounted rewards of the system, so we choose $\gamma = 0.9$ for the following simulation.

5.2.3. Transmission Rate Thresholds. We compare the effect of different transmission rate thresholds $R_{\min}$ on the system EE, as shown in Figure 5. According to Equation (7), when the transmission rate $R_{n,l,k}^{(t)} < R_{\min}$, VR$_l$ cannot successfully decode the information from VT$_n$, and we set $R_{n,l,k}^{(t)} = 0$ in this case. That is, $P_{n,l,k}^{(t)} > 0$ but $R_{n,l,k}^{(t)} = 0$, which will seriously affect the system EE. We set $R_{\min}$ as 0 Mbps, 0.1 Mbps, 0.5 Mbps, and 1 Mbps, respectively. Simulation results show that the system EE is maximum when $R_{\min} = 0$. In this case, all messages are decoded successfully as valid messages. However, this setting is not reasonable considering the QoS demand of VT users. The increase of $R_{\min}$ indicates that the QoS demand of VT users becomes more strict, and more messages are discarded as invalid messages because they cannot meet the QoS requirement; the system EE gradually decreases as a result. In the following simulations, we choose $R_{\min} = 0.1$ Mbps because the QoS demand of most VT users can be satisfied.

5.3. Comparison Experiments

5.3.1. Comparison on SIC Technology. We compare the EE of the NOMA-enabled IoV system with SIC technology, the NOMA-enabled IoV system without SIC technology, and the OMA IoV system with different vehicles, as shown in Figure 6. It can be seen that when the system contains only 10 vehicles, whether to use SIC technology has less impact on the system EE, while OMA system has the lowest EE. This is because when there are fewer vehicles, the probability of two VT users occupying the same subchannel is lower and only a small amount of interference is generated at the receiving end. The increase of the total number of vehicles means that there are more VT users that need to transmit

```
1: Initialize the $Q$ network weight parameters $\theta$
2: Initialize the actor and critic network weight parameters $\omega^Q$ and $\omega^\mu$
3: Initialize the weight parameters of the target network $\theta' \longleftarrow \theta$, target actor network $\omega^{Q'} \longleftarrow \omega^Q$, and target critic network $\omega^\mu \longleftarrow \omega^{\mu'}$
4: Initialize replay memory $\mathscr{D}$ and event trigger block $\bar{s}, \bar{a}$
5: for episode = 1, M do
6:     Initialize random noise $\varrho_t$
7:     Initialize the state of the NOMA-enabled IoV system $s_1$
8:     for $t = 1, T$ do
9:         Calculate the difference between $\bar{s}$ and $s_t$ according to Equation (19)
10:        if $\zeta = 1$ then
11:            Select action $a_t^1$ according to the DQN method
12:            Select action $a_t^2$ according to the DDPG method
13:            Replace $\bar{s}$ and $\bar{a}$ in the event trigger with $s_t$ and $a_t = \{a_t^1, a_t^2\}$
14:        else
15:            Output the action $a_t = \bar{a}$
16:        end if
17:        Perform $a_t$, get reward $r_t$ and new state $s_{t+1}$
18:        Store sample $(s_t, a_t, r_t, s_{t+1})$ into replay memory $\mathscr{D}$
19:        Sampling samples $(s_i, a_i, r_i, s_{i+1})$ from replay memory $\mathscr{D}$
20:        Update the $Q$ network, actor network, and critic network weight parameters $\theta$, $\omega^Q$, and $\omega^\mu$
21:        Update the target network, target actor network, and target critic network weight parameters $\theta' \longleftarrow \theta$, $\omega^{Q'} \longleftarrow \omega^Q$, and $\omega^\mu \longleftarrow \omega^{\mu'}$
22:    end for
23: end for
```

Algorithm 1: DRL-based resource allocation algorithm.

Table 1: Parameter setting in simulation.

Parameter	Value
K	5
M	10 MHz
τ	1 ms
D	500 m
$d_{\max}$	50 m
$P_{\max}^{VT}$	23 dBm
$v_{\max}$	36 km/h
σ^2	-114 dBm
Selectable power levels	10
Pathloss model	LOS in WINNER +B1
Fast fading	Rayleigh fading

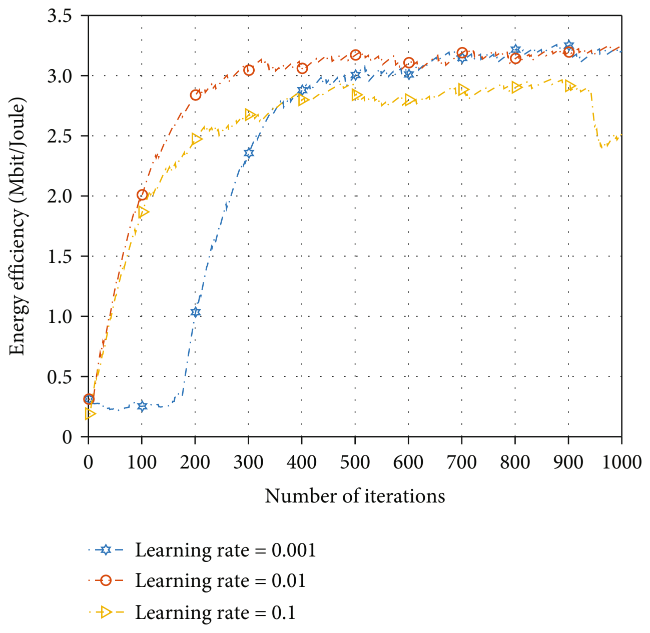


Figure 3: Impact of learning rate.

information; under the condition of a certain number of subchannels, the EE of all three approaches gradually decreases, the EE of the system with SIC technology is always the highest, and the EE of the system without SIC technology is gradually lower than the EE of the OMA approach. The reason is that NOMA actively introduces interference, the large number of VT users multiplexing the same subchannel, the stronger interference received of VR user, and not using SIC technology can lead to disastrous results.

5.3.2. Comparison on Event Trigger Block. Next, we analyze the event trigger block by comparing the impact of the event trigger block on the system EE. The threshold ρ of the event trigger module is set to 0.1, and the results are shown in Figure 7. In a variety of different situations, the event trigger block has little impact on the system EE.

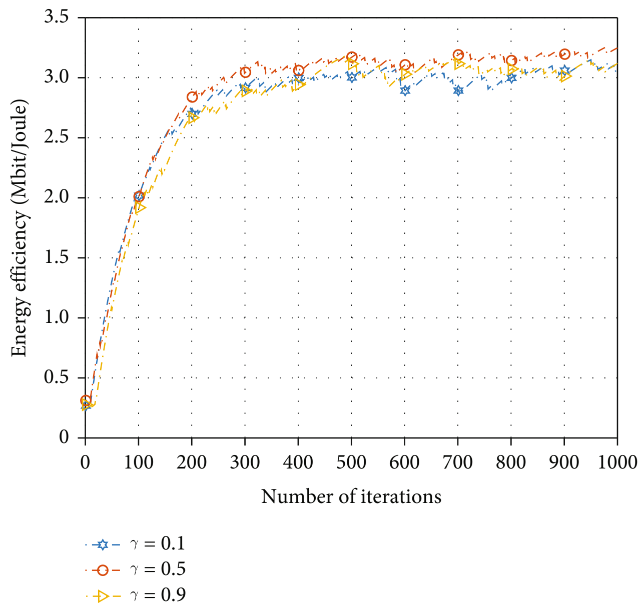


FIGURE 4: Impact of discount factor γ.

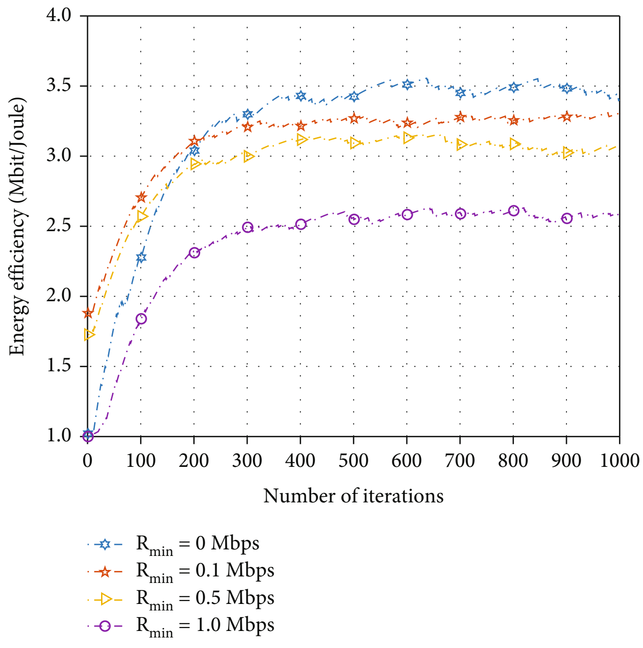


FIGURE 5: Impact of transmission rate threshold.

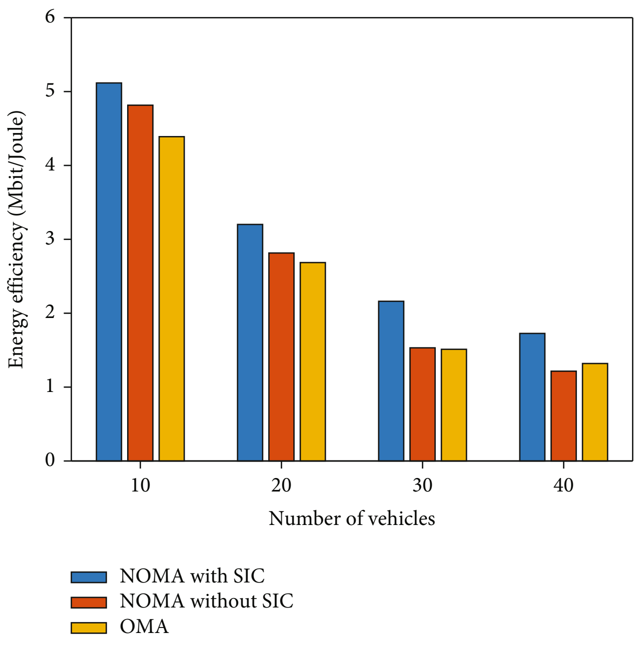


FIGURE 6: The comparison of different access methods.

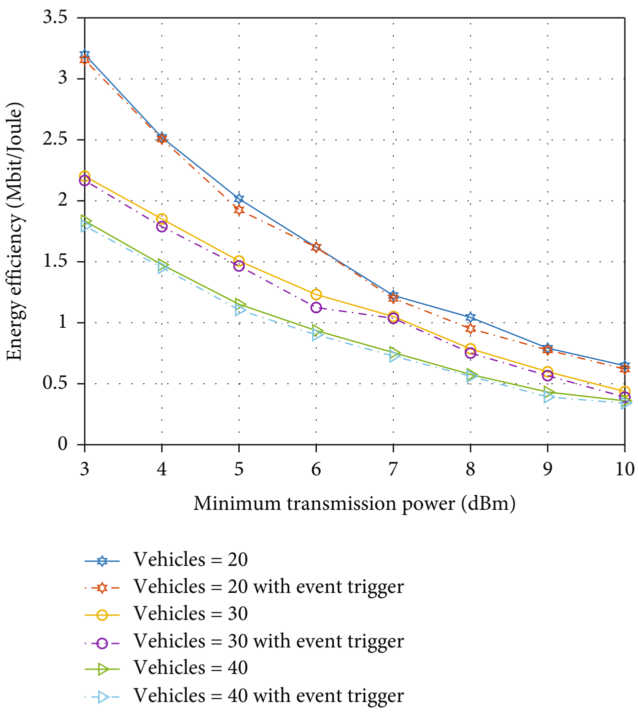


FIGURE 7: Impact of the event trigger.

Figure 8 reflects the average computation time for the three comparisons. As can be seen from the figure, the average computation time per execution increases as the number of vehicles increases, and the event trigger block effectively reduces the computation time. Such result shows that although the event trigger block costs extra time to compute the environment similarity, it can reduce some unnecessary neural network computations, which take more time.

We further compared the impact of event trigger thresholds ρ on the system EE, and the results are shown in Figure 9. It can be seen that when the threshold ρ is equal to 0.1, it only slightly decreases the system EE. Combining Figures 7–9, choosing an appropriate threshold ρ can reduce the computation time of the DSPA algorithm with a slight reduction in system performance.

5.3.3. Comparison with Other Algorithms. Finally, we compare the system EE of the DSPA, DQN, and random method under different numbers of vehicles, and the results are

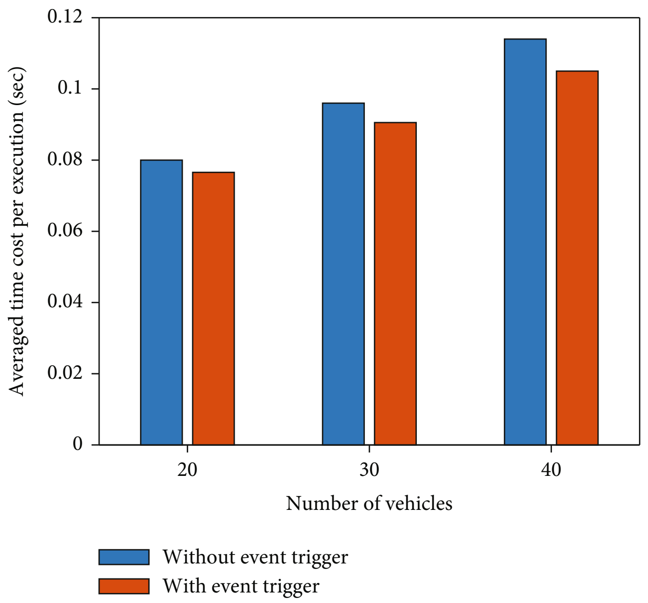


Figure 8: The comparison of averaged time cost.

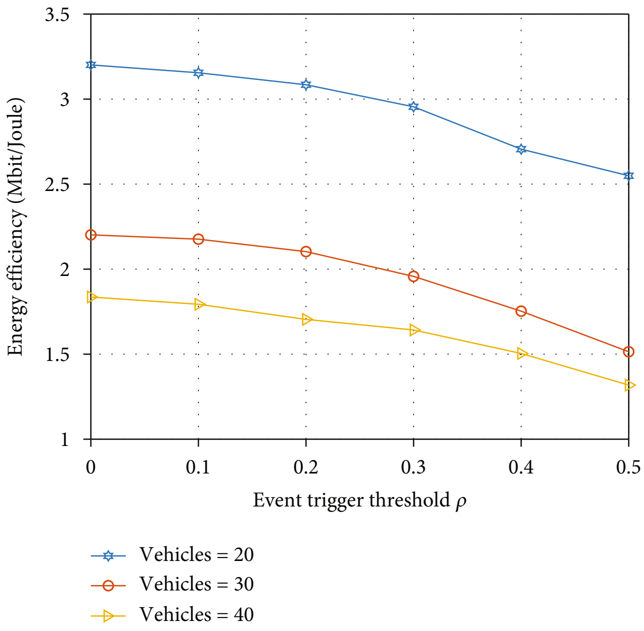


Figure 9: The comparison of different threshold ρ.

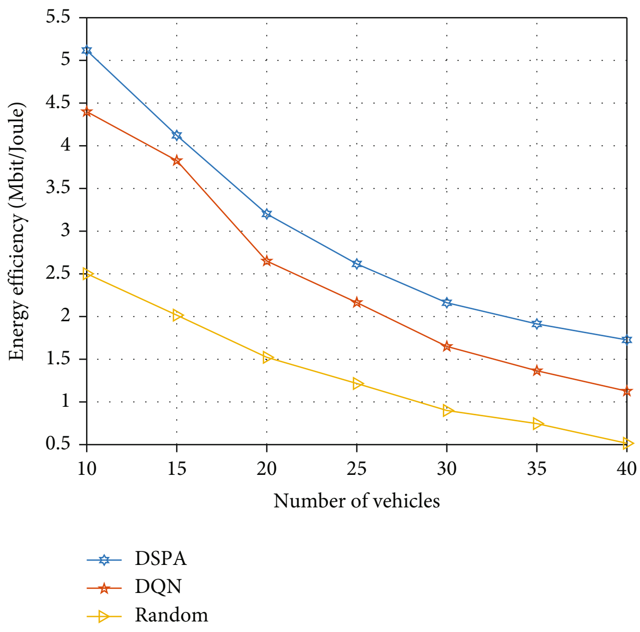


Figure 10: The comparison of different algorithms under different vehicle numbers.

shown in Figure 10. In the DQN method, we discrete the transmission power uniformly into 10 levels to meet the demand of DQN for discrete action space. The random algorithm indicates that the VT user randomly selects the channel and transmit power each time. As shown in Figure 10, we can see that the system EE decreases for all three algorithms. For both the DSPA algorithm and the DQN algorithm, the system EE decreases faster when the number of vehicles first increases and then gradually decreases. The reason is that, when the system interference is low, adding vehicles causes a significant change in system interference; with the gradual increase of vehicles, the change of system interference gradually flattens out. The system EE of the DQN algorithm is lower than that of our proposed DSPA framework because in the DSPA algorithm we use the DDPG method to select among continuous power intervals, while in the DQN algorithm we can only select among discrete 10 power levels. We believe that the performance of the DQN algorithm will be improved if the power selection levels in the DQN algorithm are increased. However, this would increase the action dimension of the DQN algorithm and take a lot of time. The system EE using the random algorithm is always the lowest due to the random selection of subchannels and transmission power at each step, which can produce catastrophic results.

6. Conclusion

In this paper, we study the NOMA-enabled resource allocation problem in IoV system. Firstly, we have maximized the system EE by allocating channel resources and power resources for VT users to reduce transmission power consumption on the basis of guaranteed system transmission rate. Secondly, we have transformed the resource allocation problem of maximizing EE into an MDP model. Finally, we designed a DSPA algorithm to obtain the subchannel selection and power allocation strategies for maximizing system EE and used the event trigger block to reduce the computation time. Simulation results show that the NOMA-enabled IoV system outperforms the OMA system, and the proposed resource allocation scheme can significantly improve the system EE compared to other schemes and reduce the computation time. In future work, we will study other NOMA-enabled resource allocation strategies and consider the introduction of mobile edge computing in IoV.

Data Availability

Data is available on request from the corresponding authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China (No. 61902029 and No. 61872044), the Excellent Talents Projects of Beijing (No. 9111923401), and the Scientific Research Project of Beijing Municipal Education Commission (No. KM202011232015).

References

[1] L. Qiao, "Mobile data traffic offloading through opportunistic vehicular communications," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 3093581, 12 pages, 2020.

[2] J. Huang, C. Zhang, and J. Zhang, "A multi-queue approach of energy efficient task scheduling for sensor hubs," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 242–247, 2020.

[3] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 3533–3546, 2020.

[4] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1050–1060, 2019.

[5] 3rd Generation Partnership Project: Technical Specification Group Radio Access Network, *Study LTE-Based V2X Services*, (Release 14), Standard 3GPP TR, 2016.

[6] T. T. T. Dao and P. N. Son, "Cancel-decode-encode processing on two-way cooperative NOMA schemes in realistic conditions," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 8828443, 15 pages, 2021.

[7] M. Bello, W. Yu, A. Chorti, and L. Musavian, "Performance analysis of NOMA uplink networks under statistical QoS delay constraints," in *ICC 2020-2020 IEEE international conference on communications (ICC)*, pp. 1–7, Dublin, Ireland, 2020.

[8] Z. Liu, G. Hou, Y. Yuan, K. Y. Chan, K. Ma, and X. Guan, "Robust resource allocation in two-tier NOMA heterogeneous networks toward 5G," *Computer Networks*, vol. 176, p. 107299, 2020.

[9] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1299–1306, 2018.

[10] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-to-Peer Networking and Applications*, vol. 13, no. 5, pp. 1776–1787, 2020.

[11] Z. Ding, R. Schober, and H. V. Poor, "Unveiling the importance of SIC in NOMA systems—part 1: state of the art and recent findings," *IEEE Communications Letters*, vol. 24, no. 11, pp. 2373–2377, 2020.

[12] X. Chen, X. Liu, Y. Chen, L. Jiao, and G. Min, "Deep Q-network based resource allocation for UAV-assisted ultra-dense networks," *Computer Networks*, vol. 196, article 108249, 2021.

[13] Z. Ma, X. Chen, T. Ma, and Y. Chen, "Deep deterministic policy gradient based resource allocation in Internet of vehicles," in *International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 295–306, Springer, Singapore, 2020.

[14] C. Guo, L. Liang, and G. Y. Li, "Resource allocation for vehicular communications with low latency and high reliability," *IEEE Transactions on Wireless Communications*, vol. 18, no. 8, pp. 3887–3902, 2019.

[15] Y. Chen, Y. Wang, M. Liu, J. Zhang, and L. Jiao, "Network slicing enabled resource management for service-oriented ultra-reliable and low-latency vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7847–7862, 2020.

[16] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2282–2292, 2019.

[17] H. Yang, K. Zhang, K. Zheng, and Y. Qian, "Joint frame design and resource allocation for ultra-reliable and low-latency vehicular networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3607–3622, 2020.

[18] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, 2019.

[19] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[20] H. Peng and X. S. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2416–2428, 2020.

[21] B. Di, L. Song, Y. Li, and G. Y. Li, "NOMA-based low-latency and high-reliable broadcast communications for 5G V2X services," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, Singapore, 2017.

[22] G. Liu, Z. Wang, J. Hu, Z. Ding, and P. Fan, "Cooperative NOMA broadcasting/multicasting for low-latency and high-reliability 5G cellular V2X communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7828–7838, 2019.

[23] X. Yang, H. He, and D. Liu, "Event-triggered optimal neuro-controller design with reinforcement learning for unknown nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 9, pp. 1866–1878, 2017.

[24] X. Wang, Y. Zhang, R. Shen, Y. Xu, and F. C. Zheng, "DRL-based energy-efficient resource allocation frameworks for uplink NOMA systems," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7279–7294, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5371330, 15 pages
https://doi.org/10.1155/2021/5371330

Research Article

Universal Method for Constructing Fault-Tolerant Optical Routers Using RRW

Meaad Fadhel, Lei Huang, and Huaxi Gu

State Key Laboratory of Integrated Service Network, Xidian University, Xi'an, China

Correspondence should be addressed to Huaxi Gu; hxgu@xidian.edu.cn

Received 7 May 2021; Accepted 26 July 2021; Published 18 August 2021

Academic Editor: Yong Zhang

High-speed data transmission enabled by photonic network-on-chip (PNoC) has been regarded as a significant technology to overcome the power and bandwidth constraints of electrical network-on-Chip (ENoC). This has given rise to an exciting new research area, which has piqued the public's attention. Current on-chip architectures cannot guarantee the reliability of PNoC, due to component failures or breakdowns occurring, mainly, in active components such as optical routers (ORs). When such faults manifest, the optical router will not function properly, and the whole network will ultimately collapse. Moreover, essential phenomena such as insertion loss, crosstalk noise, and optical signal-to-noise ratio (OSNR) must be considered to provide fault-tolerant PNoC architectures with low-power consumption. The main purpose of this manuscript is to improve the reliability of PNoCs without exposing the network to further blocking or contention by taking the effect of backup paths on signals sent over the default paths into consideration. Thus, we propose a universal method that can be applied to any optical router in order to increase the reliability by using a reliable ring waveguide (RRW) to provide backup paths for each transmitted signal within the same router, without the need to change the route of the signal within the network. Moreover, we proposed a simultaneous transmission probability analysis for optical routers to show the feasibility of this proposed method. This probability analyzes all the possible signals that can be transmitted at the same time within the default and the backup paths of the router. Our research work shows that the simultaneous transmission probability is improved by 10% to 46% compared to other fault-tolerant optical routers. Furthermore, the worst-case insertion loss of our scheme can be reduced by 46.34% compared to others. The worst-case crosstalk noise is also reduced by 24.55%, at least, for the default path and 15.7%, at least, for the backup path. Finally, in the network level, the OSNR is increased by an average of 68.5% for the default path and an average of 15.9% for the backup path, for different sizes of the network.

1. Introduction

The on-chip networking fabric, generally introduced as networks-on-chip (NoCs), has become a restricting factor in terms of efficiency and power dissipation as the movement toward many-core processors proceeds [1, 2]. This is mostly due to electrical interconnects' intrinsic technical shortcomings in scaling energy and latency at the same level as transistors. As a result, the photonic network-on-chip was proposed as a potential networking architecture for future multiprocessor systems [3–5]. Photonic network-on-chip improves the intercore connectivity performance even more than ENoCs by using the advantages of silicon photonics technology, such as the high bandwidth, low end-to-end (ETE) delay, less energy consumption, and less crosstalk [6–8].

Topologies and optical routers are the most significant component of the photonic network-on-chip architecture. Therefore, several network topologies have been documented in the literature [9, 10]. One of the main components of photonic interconnects is optical routers, which connect a local core to the neighbouring nodes and are critical components in the development of a variety of photonic interconnections. As a result, they define the communication's precision and effectiveness. Many optical routers have been also reported in [11–14]. However, none of these designs provide an alternative path to transmit data when faults occur.

Faults can occur in any circuit, and photonic circuitries are not an exception. Although photonic networks are resilient to radiation-induced transient faults [15], it is still vulnerable to thermal variation (TV), process variation (PV), and ageing [16–18]. Process variation is affected by deviations from the manufacturing standard, which can lead to system failure. Moreover, active materials, as well as items with a high thermal variation, usually age more quickly [19]. Faults may occur in the microring resonators (MRs), waveguides, routers, and other optical components. Photodetectors, for example, have a higher failure rate compared to passive components like waveguides [19]. Furthermore, since a single MR failure will cause the entire message to be misrouted or lost, PNoCs are particularly susceptible to single-point failures. These misrouted or lost messages, even those small ones used for cache coherence, may have a huge impact on the network performance. This puts the reliability of communications in photonic network-on-chips in jeopardy. Since the system in critical mission applications must operate correctly under all conditions, high reliability is needed. This led many researchers to investigate the reliability of optical routers intensively [20–23], since they are the major active components of the network. However, the designs reported in these papers neglect the effect of backup paths on signals sent over the default paths.

In this manuscript, we propose a universal heuristic scheme to construct N-port fault-tolerant optical routers for photonic networks-on-chip. This scheme can be implemented to any size of optical routers to ensure the reliability of the network. This implementation does not expose the original optical router to some contention or further blocking issues. In this method, we implement a reliable ring waveguide with a restricted number of MRs to any optical router in order to provide a backup path for any unreliable port-to-port communication. Furthermore, signals sent over the provided backup path have no effect on signals sent over the main path within the original router. The suggested scheme is then used to optimize some of the most well-known optical routers for different topologies, such as mesh and torus topologies. This helps to decrease the failure probability of the optical router. This manuscript also proposes a simultaneous transmission analysis method for optical routers, which will show the feasibility of our proposed solution. In addition, we have made the following additional contributions to our previous work [24]: (1) in this manuscript, the RRW-based scheme mentioned in our previous work [24] is further modified to be more practical for different types of optical routers. (2) The proposed scheme is implemented to more types of nonreliable optical routers, as well as provide a case study. (3) In our previous work [24], we did not consider the reliability analysis of the optical router. In this work, we provide a detailed mathematical analysis and compare the failure probability of different fault-tolerant optical routers. (4) This manuscript introduces a mathematical simultaneous transmission analysis for optical routers to demonstrate that the blocking occurs while using the backup path provided by the fault-tolerant optical router architecture. (5) Unlike the previous conference version [24], this work uses a special simulator called VPIphotonics Design Suite (VPI) to simulate the optical router and provide analytical results such as insertion loss. (6) The previous work [24] only considers the router level evaluation. However, we consider the router and network level evaluation in this manuscript. As a conclusion, our major contributions in this manuscript are summarized as follows:

(i) Proposing a universal scheme architecture to build fault-tolerant optical routers from the standard original ones by using a reliable ring waveguide and limited number of MRs to increase the path diversity and provide backup paths

(ii) Providing a traffic configuration in RRW-based optical routers, along with a case study

(iii) Analysing the reliability parameters of different well-known optical routers using our proposed scheme

(iv) Developing a simultaneous transmission probability to prove the feasibility of our fault-tolerant scheme

(v) Improving the OSNR of the network. Our proposed RRW-based optical routers can improve network-level OSNR by an average of 68.5% for the default path and an average of 15.9% for the backup path under different network sizes

The rest of this manuscript has been organized as follows. We summarized the existing solutions for reliability issues in Section 2. The main structure of the proposed architecture and the traffic configuration are provided in Section 3. Section 4 presents a case study and illustrates the communication mechanism within routers using this scheme. Section 5 evaluates the proposed method compared to some other reliable optical routers for different network architectures; moreover, it analyzes the insertion loss and crosstalk noise of several routers in the router and network level.

2. Related Work

If one or more of an optical router's components fail for any reason, the optical router loses its effectiveness. The PNoC reliability issues have been discussed in a few papers. Since the system's reliability is endangered by a variety of factors, thus each of these papers has focused their efforts on one or more of these issues.

Thermal variance has a significant impact on the resonant wavelength of an MR, putting the reliability of on-chip interconnection in jeopardy. Via temperature profiling, Li et al. [25] investigated the effects of thermal change on the efficiency of on-chip optical data transmission. They also showed the relationship between temperature fluctuations, power usage, and contact reliability in PNoC. The authors in [26] suggested the SAFT-PHENIC, a thermal-aware fault-resilient hybrid optoelectrical on-chip interconnection. This paper reports a mesh-based fault-aware routing algorithm that is aimed at reducing thermal variance around the chip by using a traffic-aware approach to spread the load and avoid using some individual nodes. To mitigate defects, the authors of [22] suggested a low-power thermal-resilient

ONoC (RONoC) and studied the thermal variation of on-chip power delivery.

Process variation is the second major problem, which can lead to photonic system failure or inaccurate data transmission. The process variation is affected by critical physical fabrication defects caused by lithography imperfections and etch nonuniformity in photonic components [27]. Via proper arrangement among MRs and wavelengths with a low power requirement, an Integer Linear Programming (ILP) problem called "MinTrim" attempted to address the dilemma of wavelength shifting of MRs due to PV [28].

In PNoC, several academic papers deal with system-level fault-tolerant techniques. Rerouting the optical signal has been studied by several scholars [20, 21]. One disadvantage of this solution is that it often reroutes traffic in the same way that it induces traffic in one place, while ignoring the problematic routers entirely. The authors in [17, 29] suggested a fault-tolerant 3D ONoC with a smaller number of redundant MRs in very important paths. Furthermore, in [5, 23], the authors created a highly stable OR system for PNoCs. They expanded the number of alternative restore routes by adding minimal hardware redundancy to their previous OR, ensuring that standard communications could still be maintained. They also reported that their FTRA-NR fault-tolerant node reuse algorithm would find the best restore route within each faulty OR. Although both of the designs in [23, 29] increase the number of MRs for reliability purposes, they cause extra contention and insertion loss, since they use more resources (e.g., MRs) to reroute faulty signals in many cases. Similarly, the authors in [30] proposed a framework to construct fault-tolerant optical routers by using redundant MRs to provide extra paths. However, this framework is applied manually and does not have a regular rule to fit all types of optical routers, as well as provide more blocking as the previous two proposed ORs.

In this manuscript, we present a universal method for improving the reliability of any optical router without exposing it to any contention or blocking issues. This approach connects any optical router to a reliable ring waveguide with a small number of MRs as a backup path for any faulty port-to-port transmission. Furthermore, signals sent over the backup path have no effect on signals sent over the default path.

3. Fault-Tolerant Optical Router Architecture

3.1. Photonic Switching Elements. Optical routers have some irreplaceable elements such as MRs, waveguides, optical terminators (OTs), and basic switching elements (BSEs). OTs are optical devices that avoid the reflection of light in the waveguide. Waveguides act as a medium allowing optical signals to be transmitted from one port to another. They are the equivalent of wires in ENoCs. MRs are used to modulate optical signals as well as switch them. The combination of two waveguides and one or two MRs can construct a basic switching element. Depending on the location of the MRs and waveguides, the BSE can be a crossing switching element (CSE) or a parallel switching element (PSE) as shown in Figure 1. If an MR is connecting two parallel waveguides, it is called a PSE. Contrarily, if an MR or two MRs are connecting two crossed waveguides, it is called a CSE. If the MR connecting these two waveguides is powered on, it will have an on-state resonance wavelength λ_{on} and will switch any optical signal passing through it such as in Figures 1(b) and 1(d). Otherwise, if it is powered off, it will have an OFF-state resonance wavelength λ_{off} and the optical signal will pass through the MR without being switched as shown in Figures 1(a) and 1(c).

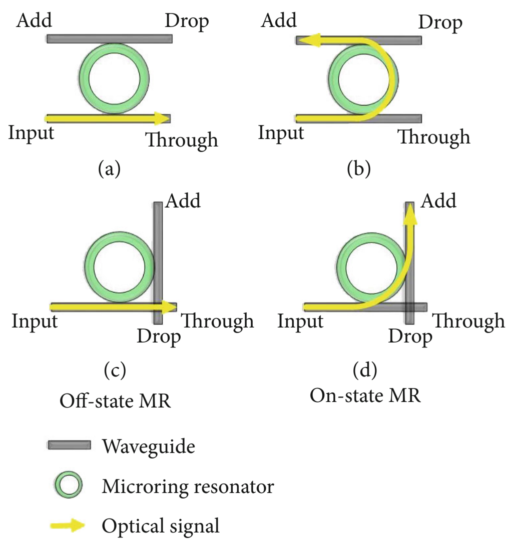


Figure 1: Basic switching elements: (a) off-state PSE, (b) on-state PSE, (c) off-state CSE, and (d) on-state CSE.

3.2. RRW Structure. The reliable ring waveguide is a universal scheme to construct fault-tolerant optical routers that guarantees a backup path for each faulty port-to-port communication within the same router. RRW scheme is constructed by implementing a single ring waveguide and some MRs to any optical router. If either of the original router's components malfunction for any reason, the ring waveguide serves as a backup path. The ring waveguide should be placed at the beginning and end of each input and output port, respectively, as seen in Figure 2(a). This avoids any possible contradictions with the original structure of the optical router while also simplifying the process. In addition, as seen in the figure, the MRs are located at each waveguide crossing created by the intersections of the ring waveguide and the input or output waveguides of the original optical router. Depending on the location of the input port and the output port regarding the port itself, the location of the MR is decided to be on the left side or the right side of the input/output waveguide of the port. Figures 2(a) and 2(b) show an example of both methods. When the input waveguide is located on the left side of the port and the output waveguide is on the right side of the port, one MR must be placed on the left side of the input waveguide of the port to provide an add point to the ring waveguide and another MR must be placed on the right side of the output waveguide of the port to create a drop point to the ring waveguide. Quite the opposite, if the input waveguide is located on the right side of the port and the output

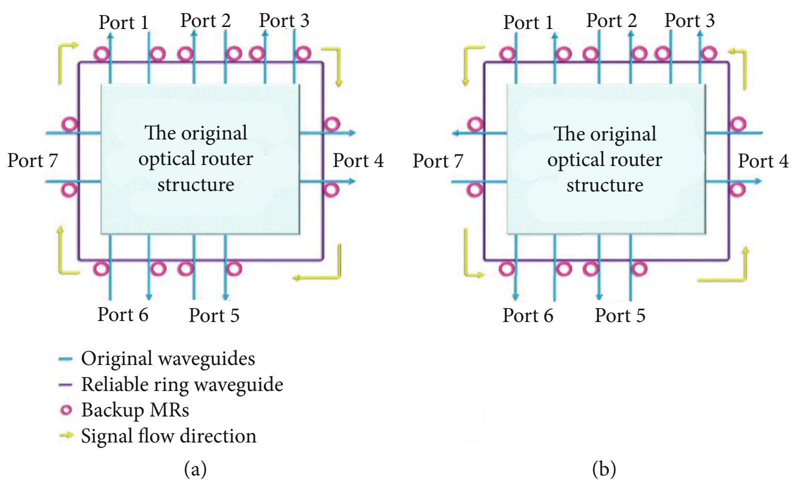


Figure 2: The reliable ring waveguide architecture: (a) clockwise RRW and (b) counterclockwise RRW.

waveguide is on the left side of the port, one MR must be located on the right side of the input waveguide of the port to work as an add point to the ring waveguide and another MR must be placed on the left side of the output waveguide of the port to act as a drop point to the ring waveguide. Drop points are where the MR downloads the signal inserted into another waveguide from the input waveguide, while add points are where the MR uploads the signal from another waveguide to the output waveguide [31].

It is worth mentioning that the location of the input and output waveguides regarding the port must be the same among the whole router in order to implement RRW properly. Moreover, depending on the location of the MR in regard to the input and output waveguides of each port, the data flow within the RRW is either clockwise or counterclockwise, but never both at the same time. When the MR is placed on the left side of the input waveguide and the right side of the output waveguide, the data flow will be clockwise. Otherwise, it will be counterclockwise as depicted in Figure 2(b).

Depending on the ports of the optical router, the total number of MRs in an $n \times n$ RRW-based optical router including the additional MRs is given by

$$\mathrm{MR}_{\mathrm{total}} = \mathrm{MR}_{\mathrm{org}} + 2n, \quad (1)$$

where $\mathrm{MR}_{\mathrm{org}}$ is the total number of MRs in the original optical router and n is the number of ports in the optical router. Furthermore, $2n$ is the number of MRs increased due to RRW scheme. Similarly, the total number of waveguides, waveguide crossings, waveguide bends, and optical terminators including the waveguides, waveguide crossings, and waveguide bends added by the ring waveguide is given by the following four equations, respectively:

$$W_{\mathrm{total}} = W_{\mathrm{org}} + 1, \quad (2)$$

$$C_{\mathrm{total}} = C_{\mathrm{org}} + 2n, \quad (3)$$

$$B_{\mathrm{total}} = B_{\mathrm{org}} + 4, \quad (4)$$

$$T_{\mathrm{total}} = T_{\mathrm{org}}, \quad (5)$$

where W_{org}, C_{org}, B_{org}, and T_{org} are the total number of waveguides, waveguide crossings, waveguide bends, and optical terminators utilized in the original optical router, respectively. Additionally, n is the number of ports in it, and $2n$ is the number of waveguide crossings increased after implementing the RRW scheme.

In case of component failure, the router will utilize the reliable ring waveguide to provide a backup path for data transmission, rather than using the default path. Furthermore, only a single signal is permitted to be transmitted alone the ring waveguide at a time. In other words, the reliable ring waveguide is only capable of transmitting one defective signal at a time. Lastly, the backup path's communications would never impact any other communications on the default path.

Given that the original optical router is strictly nonblocking, the only blocking that may happen would be either in the ring waveguide or at its intersections with the original router. However, our architecture ensures that it is implemented only at the beginning and end of each port, with an add point at the beginning of the input waveguide of a port and a drop point at the end of the output waveguide of the same port. As a result, it ensures that the add points are still ahead of the drop points to avoid overlapping [31].

This architecture offers an alternate path for the majority of faulty communication paths. In the router, each communication pair has two paths for data sharing. As a result, there is more route diversity for reliability purposes.

3.3. Traffic Configuration in RRW. In this subsection, we demonstrate how traffic is configured within RRW architecture. As previously mentioned, our architecture allows each router to function normally. Thus, we, here, illustrate that the signals flow within the RRW. Signals can be either propagated clockwise or counterclockwise, depending on the location of the MRs with regard to the input or output port. We here consider the case in which MRs are located at the left side of the input waveguide and the right side of the output waveguide; thus, the flow of signals should be clockwise as shown in Figure 2(a).

For each port-to-port communication within RRW, the MR configuration is given as follows:

$$I_i \longrightarrow O_j : R_{2i}, R_{2j-1}, \tag{6}$$

where I_i is the input of i^{th} port and O_j is the output of j^{th} port. Furthermore, R_{2i} is the MR located at the input of the i^{th} port, and R_{2j-1} is the MR located at the output of the j^{th} port.

Since for any $n \times n$ optical router, the output matrix is given by Equation (7) or Equation (8):

$$\textbf{OUTPUT} = \textbf{switch matrix} \cdot \textbf{INPUT}, \tag{7}$$

$$\begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ \vdots \\ O_n \end{bmatrix} = \textbf{switching matrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_n \end{bmatrix}. \tag{8}$$

According to Equation (8), the switching matrix of the backup bath in any $n \times n$ RRW-based optical router is presented as

$$\textbf{Switching matrix}\ (\textbf{RRW}) = \begin{bmatrix} 0 & R_2R_3 & R_2R_5 & \cdots & R_2R_{2n-1} \\ R_4R_1 & 0 & R_4R_5 & \cdots & R_4R_{2n-1} \\ R_6R_1 & R_6R_3 & 0 & \cdots & R_6R_{2n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{2n}R_1 & R_{2n}R_3 & R_{2n}R_5 & \cdots & 0 \end{bmatrix}. \tag{9}$$

Therefore, the matrix function of RRW is given by

$$\begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ \vdots \\ O_n \end{bmatrix} = \begin{bmatrix} 0 & R_2R_3 & R_2R_5 & \cdots & R_2R_{2n-1} \\ R_4R_1 & 0 & R_4R_5 & \cdots & R_4R_{2n-1} \\ R_6R_1 & R_6R_3 & 0 & \cdots & R_6R_{2n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{2n}R_1 & R_{2n}R_3 & R_{2n}R_5 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_n \end{bmatrix}, \tag{10}$$

where 0 denotes that there is no communication between these ports, since no U-turns are allowed. U-turns are the turns in which the signal is transmitted from the input of a port to the output of the same port. Moreover, R is the ON-state MRs along RRW to establish a backup path from any one particular port to any other particular port. In the absence of faults, the signal will be transmitted according to the original switching matrix of the router itself. The following section will further illustrate this switching mechanism and MR configuration of RRW.

4. Case Study

In this section, we implement RRW in a well-known nonreliable optical router. Figure 3(a) presents an example of a nonreliable 5×5 optical router, proposed in [31], whereas Figure 3(b) shows the same OR after implementing our proposed scheme. As shown, the ring waveguide does not affect the functionality or connections of the original optical router yet increases the path diversity. Meanwhile, it adds some insertion loss and crosstalk to the original router, which is a slight increment compared to other reliable optical routers, as will be further illustrated in the following sections.

Since the input waveguide is located on the left side of the port and the output waveguide is on the right side of the port, we here use the method in Figure 2(a) to implement RRW into this router. MRs should be located on the left side of the input port and on the right side of the output port. Thus, the flow of signals is going on clockwise.

Given that this router has 15 original MRs, 15 waveguide crossings, and 5 waveguides, thus, according to Equation (1), the total number of MRs in this router after implementing RRW will be $\text{MR}_{\text{total}} = 15 + 2 \times 5 = 25$. Similarly, using Equation (2) and Equation (3), the total number of waveguides used is $W_{\text{total}} = 5 + 1 = 6$ and the total number of waveguide crossings will be $C_{\text{total}} = 15 + 2 \times 5 = 25$, respectively.

According to the labelled MRs in Figure 3(b), the switching matrix function of RRW in this 5×5 optical router is given as

$$\begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \end{bmatrix} = \begin{bmatrix} 0 & R_2R_3 & R_2R_5 & R_2R_7 & w \\ w & 0 & R_4R_5 & R_4R_7 & R_4R_9 \\ R_6R_1 & w & 0 & R_6R_7 & R_6R_9 \\ R_8R_1 & R_8R_3 & w & 0 & R_8R_9 \\ R_{10}R_1 & R_{10}R_3 & R_{10}R_5 & w & 0 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix}, \tag{11}$$

where 0 means no U-turn connections can be established and w means that waveguides are provided to realize this communication and thus no need for reliable MRs to realize it.

Furthermore, Table 1 lists all the possible communication pairs and denotes the corresponding MRs to realize them. The table presents the default paths as well as the reliable paths using RRW. The default path (also known as the original path) is presented first, then the reliable path (also known as the backup path). In the table, "w" signifies a path that does not require the activation of any MR, whereas "-" means a port cannot send itself. The table shows that for each communication pairs, there must be two paths at least. One of them is used as a backup path if and only if a malfunctioned MR is detected. For example, if port 2 is requesting to communicate with port 5 as shown in Figure 4, R_{16} should be turned on in order to switch the light signal from the input waveguide of port 2 to the output waveguide of port 5. However, in the presence of faulty MRs, the optical router has to take the backup path, which is represented by the reliable

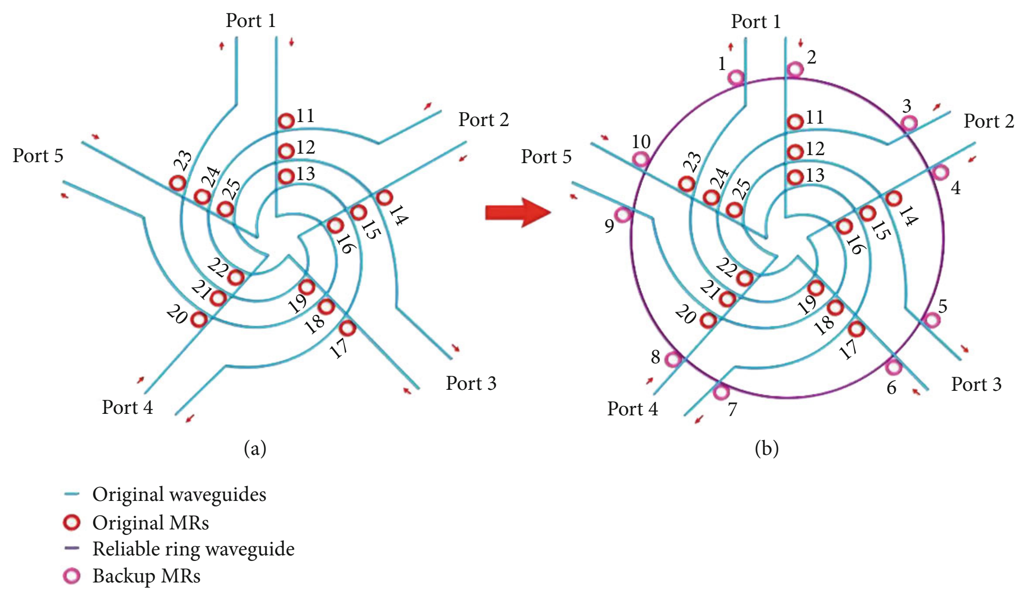


Figure 3: An example of implementing the reliable ring waveguide to the 5×5 optical router in [31].

Table 1: Resonator configuration in the 5×5 optical router proposed in [31] after implementing RRW.

From-to	Out 1	Out 2	Out 3	Out 4	Out 5
In 1	-	R_{11}; R_2, and R_3	R_{12}; R_2, and R_5	R_{13}; R_2, and R_7	w
In 2	w	-	R_{14}; R_4, and R_5	R_{15}; R_4, and R_7	R_{16}; R_4, and R_9
In 3	R_{19}; R_6, and R_1	w	-	R_{17}; R_6, and R_7	R_{18}; R_6, and R_9
In 4	R_{21}; R_8, and R_1	R_{22}; R_4, and R_3	w	-	R_{20}; R_8, and R_9
In 5	R_{23}; R_{10}, and R_1	R_{24}; R_{10}, and R_3	R_{25}; R_{10}, and R_5	w	-

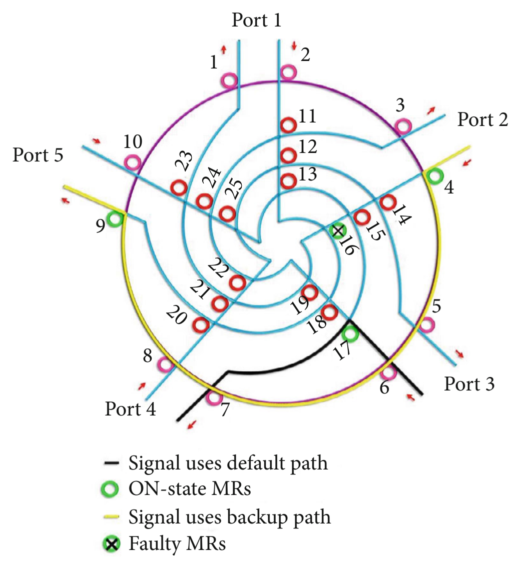


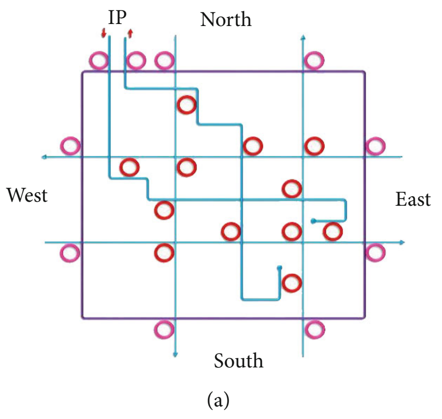
Figure 4: An example of communications in both the default and the backup paths in an RRW-based OR.

ring waveguide. Thus, two MRs should be switched on in order to transmit the light signal within the reliable ring waveguide. The exact MRs can be decided by Equation (6). Since in this case $i = 2$ and $j = 5$, the on-status MRs are R_4 and R_9. The light signal will be switched immediately using the first backup MR (i.e., R_4) and will be propagating along the reliable ring waveguide without interfering with the signals in the default paths, such as the signal transmitted from port 3 to port 4, until it will be coupled by R_9 to be ejected at the output of port 4. Therefore, RRW-based ORs are deemed to be nonblocking optical routers. Although in this case the backup path uses more resources and tend to have more insertion loss and crosstalk, this method can provide simultaneous transmission in the default path along with the backup path. In other words, this method does not disturb the normal flow of signals for reliability purposes. This will be further illustrated in the following section.

5. Performance Evaluation and Analysis

In this section, we evaluate the performance of RRW-based optical routers using the theoretical analysis and simulation evaluations using VPIphotonics Design Suite (VPI) [32].

To evaluate the performance of our architecture, we implement the reliable ring waveguide into several nonreliable optical routers proposed in [31, 33–38]; two of them are presented in Figure 5, and we compare them with 7×7 and 5×5 FTTDOR proposed in [29] and the NRFT optical router in [23] and some other fault-tolerant optical routers presented in [30].



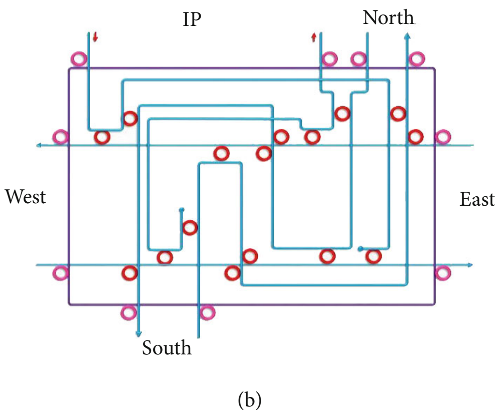


Figure 5: Some fault-tolerant optical routers based on RRW: (a) RRW-Crux and (b) RRW-Cygnus.

Table 2 compares the features of the proposed fault-tolerant scheme compared with some previously proposed fault-tolerant optical routers for different sizes. The table shows that RRW-based optical routers increase the reliability in the cost of increasing the number of MRs in some cases, especially when compared to 7×7 optical routers. However, RRW-based optical routers provide the least maximum number of ON-state MRs where a signal will pass through while traveling within the router in both the default and backup paths compared to other 7×7 optical routers. On the other hand, when compared to 5×5 optical routers, RRW-based optical routers have either the same number of additional backup MRs, such as in RRW-Crux and RRW-Cygnus, or less additional backup MRs, such as in RRW-crossbar. Similar to 7×7 optical routers, 5×5 RRW-based optical routers still provide the least maximum number of ON-state MRs where a signal passes through while being transmitted from one port to the other within the router.

5.1. Network Reliability Analysis. In this section, we analyze the reliability of the network using our design. The analysis is a modified version of the analysis presented in [23]. Moreover, this section shows the probability of our design functioning well, when the optical router is suffering from some faulty MRs. This can be evaluated using the following equation:

$$R_{\text{total}} = \prod_{i=1}^{N_{\text{w}}} \text{R}_{\text{w}}(i) \times \prod_{j=1}^{N_{\text{OR}}} R_{\text{OR}}(j). \tag{12}$$

In this analysis, multiple faults are introduced into some MRs of the optical router in a random pattern, to examine whether the optical signals are properly received by the desired output port of the router or not. R_{total} is the entire network reliability; N_{w} and N_{OR} are the number of optical waveguides and the number of the optical routers used within the network, respectively. Similarly, R_{w} and R_{OR} denote the reliability of the i^{th} optical waveguide and the j^{th} optical router in the network, respectively. Unlike active components such as MRs, passive components such as waveguides are constantly in normal status and not prone to failures. Thus, the reliability of waveguides is the same throughout the network and can be ignored and set to be 1. Therefore, we neglect the optical waveguide reliability and focus on the reliability of optical routers within the network.

One major reason for optical router failures is a broken MR. Thus, given the reliability p of an MR (events of faulty MRs are independent of each other), the reliability of the optical router is given by

$$R_{\text{OR}} = 1 - \sum_{m=0}^{R} \partial_m \binom{R}{m} (1-p)^m (p)^{R-m}. \tag{13}$$

The number of simultaneously faulty MRs is denoted by m, whereas R represents the total number of MRs within the optical router (i.e., the MRs in the original router design and the backup MR added by RRW). Furthermore, the probability of an OR failure caused by m faulty MRs is presented by ∂_m.

Since our scheme uses the same optical router to tolerate physical faults occurring in the router, such as faults caused by thermal variations, the failure probability of our RRW-based optical router in the presence of m faulty MRs is illustrated as follows:

$$\partial_m = \frac{\sum_{k=1}^{P} \binom{MR_k}{m}}{\binom{R}{m}} \times 100\,(\%). \tag{14}$$

Equation (14) shows that the failure probability of an optical router is given by dividing the number of cases in which an optical signal is misrouted from its proper direction when one or several faulty MRs are introduced, by the cumulative number of cases in which m MRs out of an optical router's entire MRs (i.e., R, which includes both the original and the backup MRs) are unreliable. In the equation, k refers to the k^{th} path among the P sets of possible paths within the optical router, and MR_k denotes the number of MRs along the path k.

In most cases, when $m = 0$ or $m = 1$, i.e., no faulty MRs or only 1 faulty MR, the optical router still can manage to function as normal. This means that $\partial_0 = 0$ and $\partial_1 = 0$, since no misrouting is happening in the OR. However, when the number of faulty MRs increases to 2, i.e., $m = 2$, most fault-tolerant routers will start facing some misrouting difficulties, and when there are more than two faulty MRs, this

Table 2: Design features of several fault-tolerant optical routers compared to RRW-based scheme.

Routers	Default MRs	Additional MRs	Total MRs	Max. No. ON-MRs (default)	Max. No. ON-MRs (backup)
7×7 RO-Uni	26	14	40	1	2
7×7 RO-Votex	24	14	38	1	2
7×7 FTTDOR	22	10	32	2	3
7×7 NRFT	20	8	28	4	5
5×5 RRW-crossbar	20	10	30	1	2
5×5 FT-crossbar	20	13	33	1	3
5×5 RRW-ODOR	12	10	22	1	2
5×5 FT-ODOR	12	10	22	1	3
5×5 RRW-Crux	12	10	22	1	2
5×5 FT-Crux	12	10	22	1	3
5×5 RRW-Cygnus	16	10	26	1	2
5×5 FT-Cygnus	16	10	26	1	3

phenomenon is referred to as a state–space explosion problem. Thus, we here present the failure probability of different optical routers when 2 faulty MRs occur in the router.

Depending on the position of the faulty MR, the failure probability can be increased or decreased drastically. As a result, we computed all the possible positions for the faulty MRs (according to Table 1) and presented the average results in Figure 6. For instance, for RRW-based Cygnus, there exist 26 MRs, including 16 original MRs and 10 redundant MRs. The number of scenarios in which there exist two malfunctioned MRs is given by $\binom{26}{2}$. Moreover, there exist sixteen cases in which the signal is deflected due to the presence of two malfunctioned MRs. Therefore, the probability of failure of this router is $16/\binom{26}{2} = 4.9\%$. Accordingly, Figure 6 shows that RRW-based optical routers enjoy less failure probability compared to the 5×5 FTTDOR, which has the worst failure probability of 5.79%. On the other hand, the failure probability of RRW-Cygnus, RRW-Crossbar, RRW-Crux, RRW-ODOR, RRW-OXY, and RRW-OR is 4.9%, 4.59%, 5.19%, 5.19%, 5.19%, and 5.19%, respectively.

5.2. Simultaneous Transmission Analysis. Most of the proposed fault-tolerant optical router designs have addressed the reliability issue of the OR at the blocking expense. In other words, these architectures provide backup paths for most or all possible communication pairs; however, these backup paths block other default communication pairs within the optical router, which means that in several architectures, simultaneous transmission is not supported. Thus, in this manuscript, we propose a simultaneous transmission probability analysis for optical routers.

The probability of simultaneous transmission within the router using the default path can be calculated by obtaining the blocking probability of each possible communication pairs as follows:

$$T_{\text{sim}} = 1 - \frac{\sum_{i=1}^{P}(B/n)}{P}, \tag{15}$$

where T_{sim} is the simultaneous transmission in the router; i denotes the i^{th} path among P, which is the set of possible paths within the optical router; and (B/n) refers to the average probability of blocked paths out of all the possible simultaneous paths in the router, which is set to be n at most, for every $n \times n$ optical router. Since most of these optical routers are nonblocking architectures in general, we will only consider, here, the case of one faulty MR in the router, to check the possible simultaneous signal transmission within the router. For example, for FT-Crux, there is an average of 1.8 possible blocked default paths and there are 16 sets of possible paths in the router; thus, $T_{\text{sim}} = 1 - (1.8/16) = 0.887$.

Figure 7 depicts the probability of simultaneous signal transmission using the default path of several fault-tolerant routers and compares them to our scheme. It is so clear that NRFT suffers the most among other optical routers; this can be explained by the fact that this router is not a nonblocking router in general. This means that signals transmitted simultaneously within the router using default paths are blocking each other. On the other hand, most of the rest of the routers are nonblocking routers originally. However, since they use the same waveguides to transmit signals, blocking occurs, unlike RRW-based optical routers, because RRW-based optical routers are using the ring waveguide instead of the original waveguides in the router, which avoids conflicts and guarantees that the simultaneous signal transmission within the default path stays in a nonblocking state at all times.

5.3. Simulation Analysis. The simulation in this section is curried out using VPIphotonics Design Suite (VPI) [31]. The simulation results can illustrate the effect of various optical router design parameters, such as waveguide size and MR radius. As a result, this platform can provide simulation results that are closer to fabrication [39].

Table 3 demonstrates the parameter setting interface of all the four modules used to build the optical router considered in the case study and shown in Figure 3. These four modules are straight waveguide, waveguide bends, waveguide crossings, and crossing switching elements (CSEs).

Our simulation analysis is divided into three parts. First, we have simulated the original router without implementing

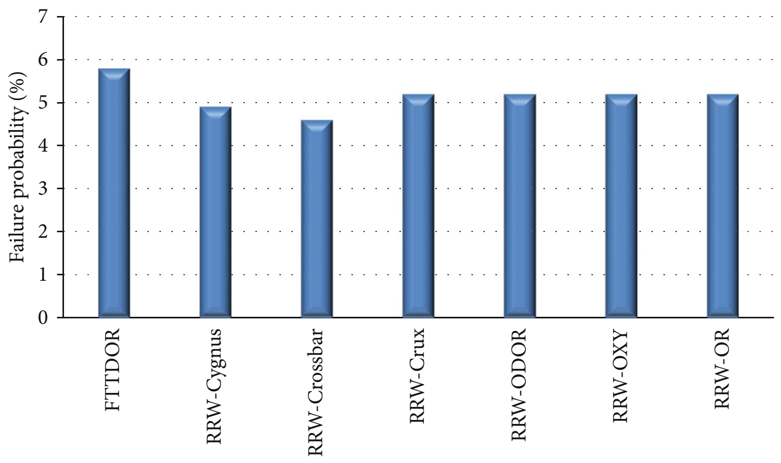


Figure 6: Failure probability of several fault-tolerant optical routers.

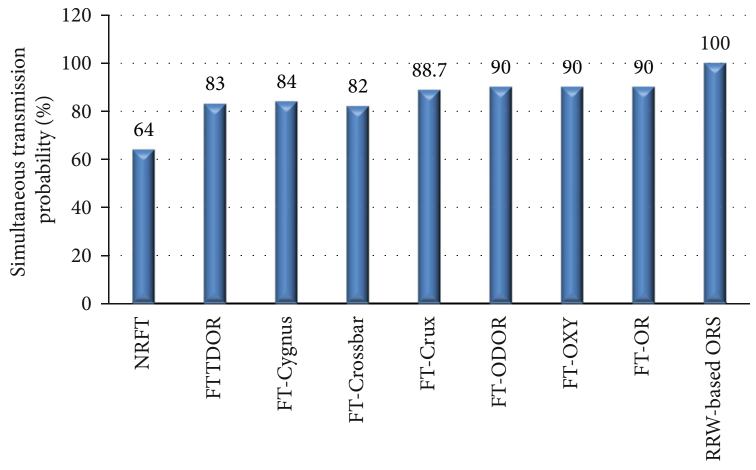


Figure 7: The probability of simultaneous signal transmission using the default path.

Table 3: The component parameters in the simulation environment.

Parameter	Value
Waveguide height	220 nm
Waveguide width	450 nm
Waveguide length	1 cm
Waveguide's refractive index	3.5
Surface roughness	10^{-9}
Radius of waveguide bends	5 μm
Radius of MRs	5 μm
Gap in CSEs	170 nm
Laser power	1 mW

the reliable ring waveguide scheme. As mentioned earlier and shown in Figure 3(a), the original optical router is made up of 15 MRs, 15 waveguide crossings, and 9 waveguide bends. According to the sequence of these components, we constructed the optical router using the corresponding module. In the simulation, we evaluate the insertion loss of each port-to-port communication. Figure 8 displays the worst-case insertion loss results, including power spectra for all of the five output ports. The worst-case IL occurs when the optical signal is traveling from the West port to the East port, since it passes through four OFF-state MRs, four waveguide crossings, one bend, and one ON-state MR. From the results shown in (Figure 8), the maximum insertion loss of the original optical router is -2.1 dB for optical signal at the frequency of 193.44 THz.

Second, we simulated the default path of the original router after implementing the reliable ring waveguide scheme. As depicted earlier in Figure 3(b), the RRW-based optical router has a total of 25 MRs, 25 waveguide crossings, and 13 waveguide bends. The RRW-based optical router is built up as shown in Figure 3. Figure 9 depicts the results of the worst-case insertion loss after evaluating all five communication pairs. The worst-case IL in RRW occurs when the optical signal is traveling from the West port to the East port as well, since it still passes through similar optical devices as the previously mentioned ones. The only difference is the 2 CSEs located at the beginning and the end of the default path. The results show that the maximum insertion loss of the RRW-based optical router when the optical signal is using the default path is -2.6 dB for optical signal at the frequency of 193.44 THz.

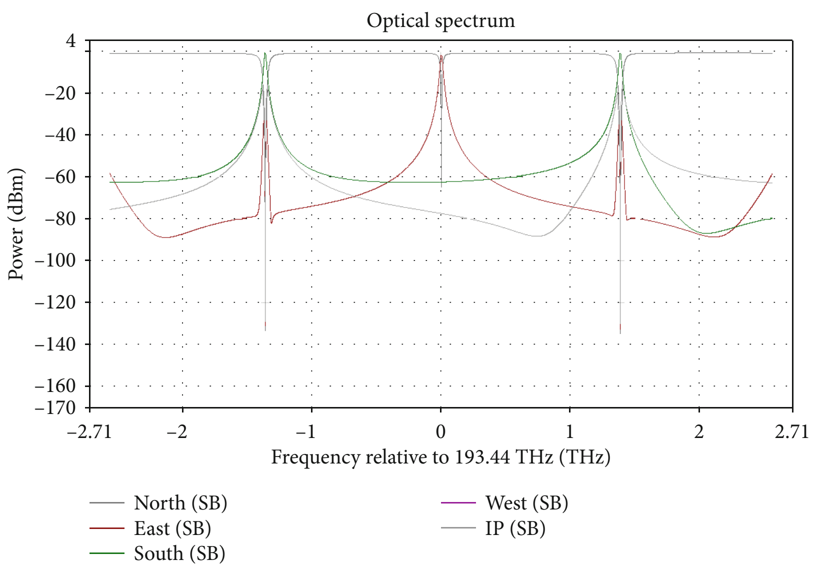


FIGURE 8: Power spectra of the optical router at all five output ports under the worst-case insertion loss.

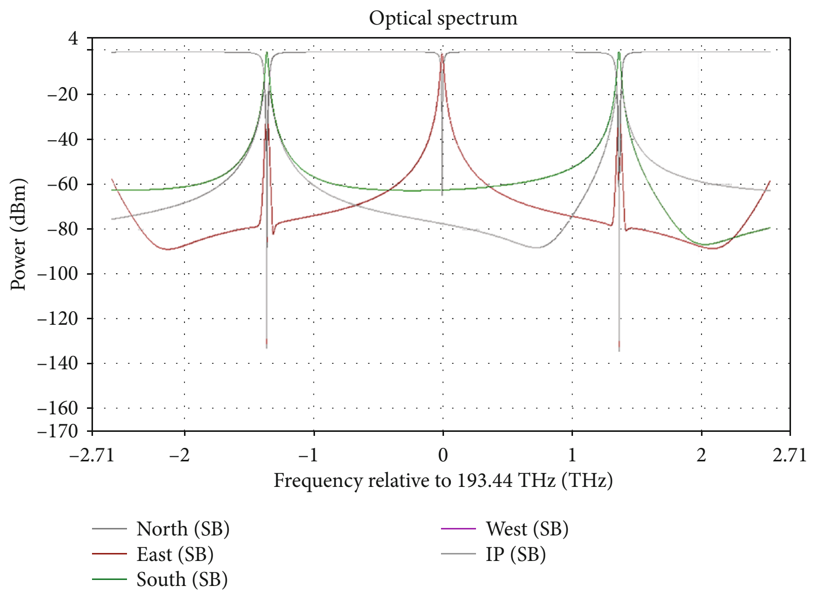


FIGURE 9: Power spectra of the RRW-based optical router using the default path, at all five output ports under the worst-case insertion loss.

Third, we run the simulation on the backup path of the RRW-based optical router. Similar to the simulation in the second part, the RRW-based optical router has a total of 25 MRs, 25 waveguide crossings, and 13 waveguide bends. However, in this simulation, the optical signal in the worst case will pass through four OFF-state MRs, four waveguide crossings, two bends, and two ON-state MRs. Figure 10 presents the simulation result of the worst-case insertion loss after evaluating all five communication pairs. This worst-case insertion loss happens while transmitting from West to East, too. The findings in the figure indicate that the maximum insertion loss of the RRW-based optical router when the optical signal is using the backup path is -3.6 dB for optical signal at the frequency of 193.44 THz.

Although the insertion loss increases using the backup path provided by RRW, the default path is still having an acceptable increment compared to the original router as shown in the previous figures. The figures show that the worst-case insertion loss of the default path has increased by -0.5 dB using RRW and the worst-case insertion loss using the backup path is -1.5 dB more than the default path in the original router.

Similarly, in the following two sections, we will further introduce detailed insertion loss and crosstalk noise results of more architectures in the router and the network levels.

5.4. Insertion Loss. We here evaluate more optical routers using RRW compared to others proposed in [23, 29, 30] in

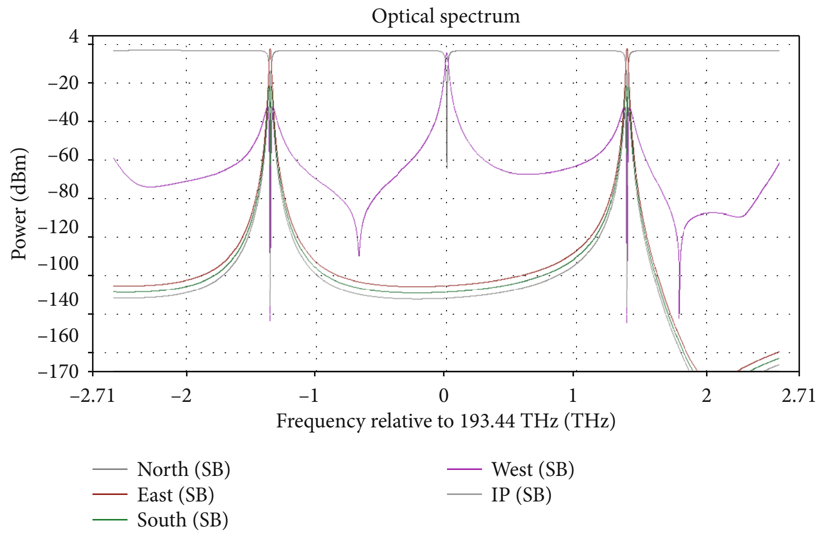


FIGURE 10: Power spectra of the RRW-based optical router using the backup path, at all five output ports under the worst-case insertion loss.

the same method as [40]. We evaluate the insertion loss (IL) of the other optical routers in Figure 5 and show the worst-case IL in both the default and the backup paths. Moreover, to evaluate the backup path of each router, we assume that only one MR fails at a time.

Let us, first, consider the optical routers used in 3D networks and compare them together since they have the same size. Moreover, since the NRFT and FTTDOR are based on XYZ routing algorithm, whereas the universal router in [31] and Votex in [35] are fully connected routers, we first optimize them by reducing the unused MRs of the unused communication pairs from the architecture, then implement RRW scheme. In fully connected optical routers, all input ports can communication with all output ports, whereas XYZ optical routers connect inputs with higher priority to outputs with the same priority or less priority. Thus, the routers are optimized by reducing the MRs used for unused communications, such as the communications from up to West/East/North/South.

Table 4 presents the maximum, average, and minimum port-to-port IL of all evaluated optical routers. From the table, the NRFT optical router utilizes two separated optical routers, which are a 6×6 OR and a 3×3 OR for intra- and interlayer interconnections, respectively. Therefore, the minimum port-to-port IL (-0.12 dB) would occur in the smaller router from the up port to the down port and vice versa. Similarly, the minimum port-to-port insertion loss in 7×7 FTTDOR is -0.115 dB, which occurs in the communication from the up port to the down port as well, since the up and down waveguides in FTTDOR are only connected to themselves and the IP core but not to others. Nonetheless, the minimum port-to-port insertion loss of the RRW-based optimized universal router (RO-Uni) is -0.53 dB for both the default and backup paths, and the minimum port-to-port IL of the RRW-based optimized Votex (RO-Votex) is -0.59 dB and -0.73 dB for the default path and the backup path, respectively. Although the first two mentioned routers could introduce the minimum insertion loss, routers based on our method enjoy the least average and worst-case port-to-port insertion losses. RO-Uni introduces the least average IL (-0.75 dB) in the default path and -1.101 dB IL for the backup path. On the other hand, the average ILs of RO-Votex are -0.87 dB and -1.127 dB for both the default and the backup paths, respectively. The average IL of 7×7 FTTDOR is low as well, with -0.76 dB and -0.98 dB for both the default and the backup paths, respectively. However, NRFT has the highest average IL for both the default and backup paths, with -0.93 dB and -1.48 dB, respectively. In terms of the worst-case insertion loss, RRW-based routers introduce the least IL, with -1.28 dB and -1.45 dB in RO-Votex and -0.955 dB and -1.36 dB in RO-Uni for the default and the backup paths, respectively. On the other hand, the worst-case ILs in 7×7 FTTDOR are -1.365 dB and -2.245 dB and in NRFT are -2.185 dB and -2.705 dB for the default and backup paths, respectively.

Now, we consider routers with the size of 5×5. Since the 5×5 Crux and 5×5 Cygnus are originally not fault-tolerant optical routers, thus we implement our RRW scheme and compare it with the fault-tolerant architectures designed based on them and proposed in [30]. Table 4 shows that optical routers based on RRW encounter a slightly more insertion loss in the minimum and the average insertion loss, which can be regarded as 6.6%. However, the 5×5 RRW-based optical routers encounter the least worst-case insertion loss in the back up path, with a 25% less insertion loss than the one encountered by FT-Crux and a 30.4% less insertion loss than FT-Cygnus.

Figures 11 and 12 depict the port-to-port insertion loss of FT-Crux, RRW-Crux, FT-Cygnus, and RRW-Cygnus for both the default and the backup paths. The numbers 0, 1, 2, 3, and 4 denote the ports from the injection/ejection port, North, East, South, to West, respectively. From both figures, we can notice that the maximum insertion loss of FT-Crux is mainly introduced when the signal is sent out of the injection port, whereas in FT-Cygnus, the maximum insertion loss is introduced by the North port.

Table 4: The insertion loss comparison of optical routers based on our method and other reliable optical routers for port-to-port communications.

Routers	Maximum IL (dB)		Minimum IL (dB)		Average IL (dB)	
	Default	Backup	Default	Backup	Default	Backup
7×7 RO-Uni	-0.955	-1.36	-0.53	-0.53	-0.75	-1.101
7×7 RO-Votex	-1.28	-1.45	-0.59	-0.73	-0.87	-1.127
7×7 FTTDOR	-1.365	-2.245	-0.115	-0.115	-0.76	-0.98
7×7 NRFT	-2.185	-2.705	-0.12	-0.12	-0.93	-1.48
5×5 RRW-crux	-0.77	-1.29	-0.23	-0.23	-0.56	-0.9
5×5 FT-crux	-0.71	-1.72	-0.15	-0.15	-0.51	-0.73
5×5 RRW-Cygnus	-0.86	-1.28	-0.28	-0.28	-0.64	-0.98
5×5 FT-Cygnus	-0.8	-1.84	-0.22	-0.22	-0.57	-0.94

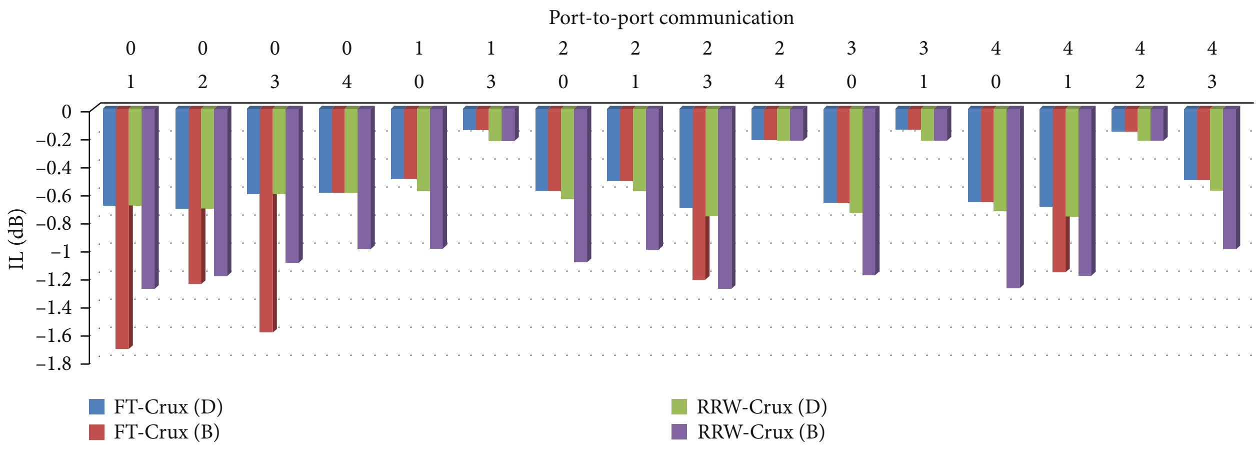


Figure 11: Port-to-port insertion loss of FT-Crux compared to RRW-Crux.

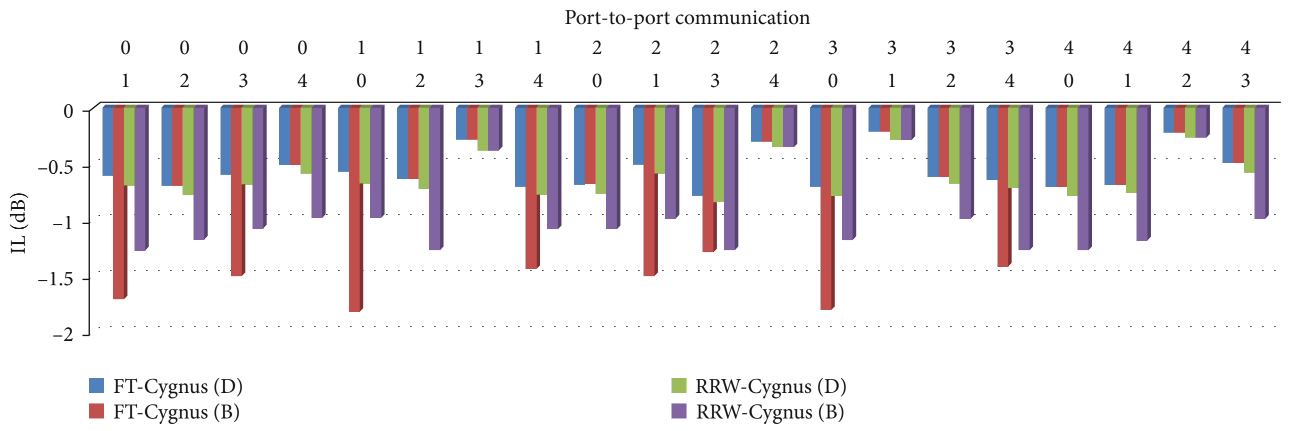


Figure 12: Port-to-port insertion loss of FT-Cygnus compared to RRW-Cygnus.

5.5. Crosstalk Noise. Here, we present the crosstalk comparisons of several optical routers; moreover, consider the worst-case OSNR of 2D mesh network using RRW and another reliable router. Similar to the previous section, the crosstalk of the longest path is obtained similar to the method in [40].

Figure 13 depicts the crosstalk noise of 7×7 RO-Uni default and backup paths and 7×7 FTTDOR default and backup paths for each port-to-port communication pair. The numbers 0, 1, 2, 3, 4, 5, and 6 are the injection/ejection port, East, down, South, West, up, and North, respectively. The results show that the RRW-based router reduces the worst-case crosstalk noise by 39.9% for the backup path and 24.55% for the default path. Furthermore, this figure reports that RO-Uni introduces the least crosstalk noise in several communication pairs.

On the other hand, Figure 14 presents the crosstalk noise of the 5×5 FT-Crux and 5×5 RRW-Crux for each communication. It is clear that the worst-case noise is introduced by

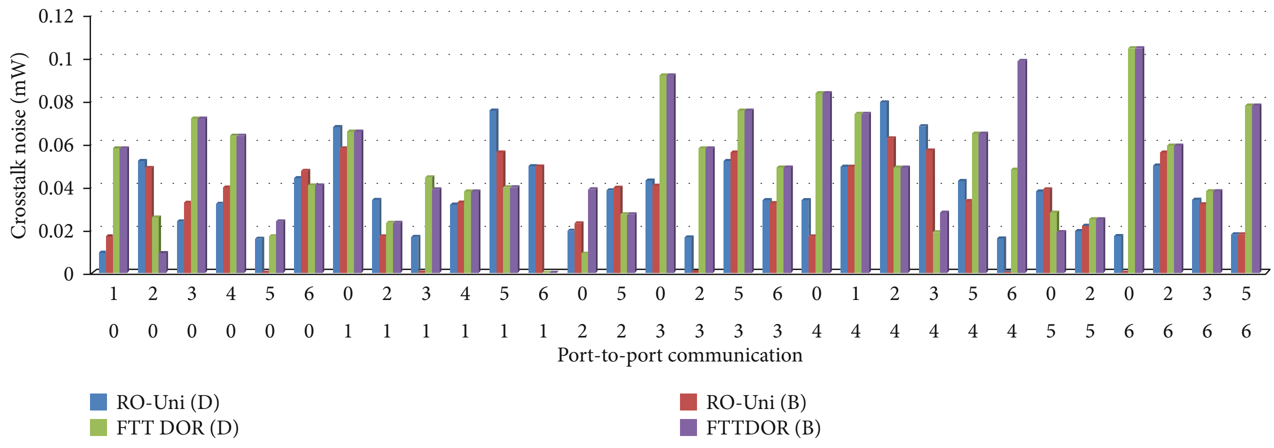


Figure 13: Port-to-port crosstalk noise of 7×7 optical routers compared to RRW.

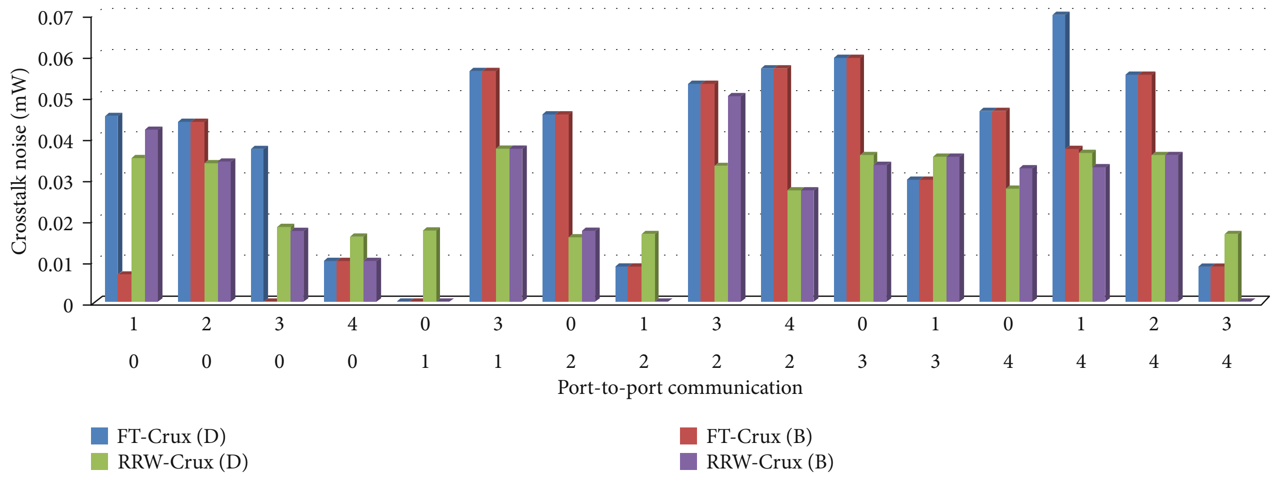


Figure 14: Port-to-port crosstalk noise of 5×5 FT-Crux compared to 5×5 RRW-Crux.

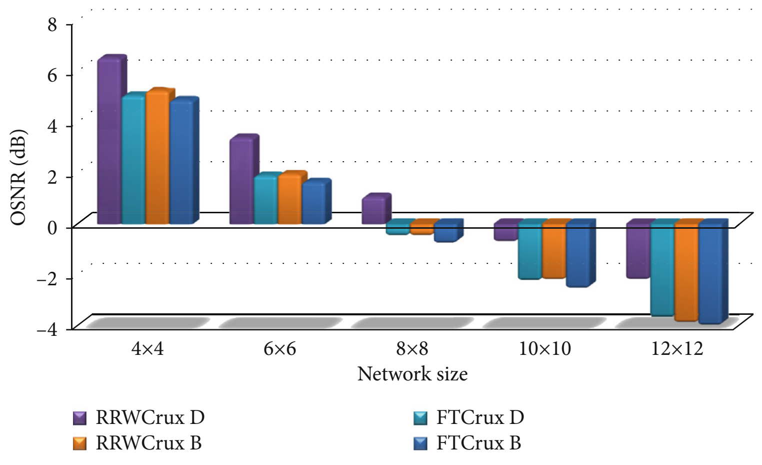


Figure 15: Worst-case OSNR of the FT-Crux and RRW-Crux optical routers for several network sizes.

FT-Crux while using the default path. The results show that the RRW-based router reduces the worst-case crosstalk noise by 46.7% for the default path and 15.7% for the backup path. Furthermore, RRW-Crux has the least crosstalk noise in many port-to-port communications.

The performance of different sizes of mesh network is evaluated using FT-Crux and RRW-Crux for several longest paths in the network and compared them to take out the worst-case crosstalk among all. Figure 15 illustrates the worst-case OSNR of FT-Crux and RRW-Crux for different sizes of the network. The figure shows that the OSNR of FT-Crux using both the default and the backup paths and RRW-Crux using backup path is decreasing drastically as the network gets bigger in size. RRW-Crux increases the OSNR by an average of 68.5% for the default path and an average of 15.9% for the backup path compared to FT-Crux default and backup paths, respectively, for the size of 4×4, 6×6, 8×8, 10×10, and 12×12 2D mesh.

6. Conclusions

Reliability of an optical router is a hot topic for researchers. It determines the efficiency and performance of the network. We proposed a universal method that is easily implemented to an optical router for reliability purposes. The method does not expose the optical router to further contention or blocking problems. In this method, we implement a reliable ring waveguide (RRW) with a specific number of MRs, which is $2n$, to any $n \times n$ optical router to provide an alternative path for any faulty communication. Another important feature of the proposed method is that signals transmitted using the alternative path do not affect the transmitted signals using the original path. Moreover, we proposed a simultaneous transmission analysis for optical routers to show the feasibility of our method.

The results show that the failure probability of RRW-based optical routers is at most 5.19% (which is the failure probability of RRW-ODOR) compared with FTTDOR which has a failure probability of 5.79%. The simultaneous transmission in RRW-based optical routers is improved by at least 10% compared to FT-OXY and at most 46% compared to NRFT. The chapter also provides a case study by implementing RRW scheme on one of the well-known optical routers. The simultaneous results using VPIphotonics Design Suite (VPI) is providing results that are closer to fabrication. It shows that the worst-case insertion loss of the default path is increased by -0.5 dB using RRW and the worst-case insertion-loss using the backup path is increased by -1.5 dB compared to the default path in the original router. Furthermore, the worst-case insertion loss of RRW-based optical routers can be reduced by 46.34% at most for 7×7 optical routers and 30.2% at most for 5×5 optical routers. RO-Uni reduces the worst-case crosstalk noise by 24.55% and 39.9% for the default and backup paths, respectively, compared to FTTDOR. RRW-Crux reduces the worst-case crosstalk noise by 46.7% for the default path and 15.7% for the backup path compared to FT-Crux. Finally, RRW-Crux increases the OSNR by an average of 68.5% for the default path and an average of 15.9% for the backup path compared to FT-Crux default and backup paths, respectively, for the size of 4×4, 6×6, 8×8, 10×10, and 12×12 2D mesh. In the future work, we aim to combine the architecture design with a routing algorithm to further improve the network reliability.

Data Availability

All data can be obtained from the author.

Disclosure

This paper is an extended version of a previously presented one in the International Symposium on Parallel Architectures Algorithms and Programming [24].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFE0202800, the National Natural Science Foundation of China under Grant Nos. 61634004 and 61934002, the Natural Science Foundation of Shaanxi Province for Distinguished Young Scholars under Grant No. 2020JC-26, and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing under Grant No. 2019A01. This work was supported by the Youth Innovation Team of Shaanxi Universities, as well.

References

[1] M. Baharloo, R. Aligholipour, M. Abdollahi, and A. Khonsari, "*ChangeSUB*: a power efficient multiple network-on-chip architecture," *Computers & Electrical Engineering*, vol. 83, article 106578, 2020.

[2] S. Borkar, "Role of interconnects in the future of computing," *Journal of Lightwave Technology*, vol. 31, no. 24, pp. 3927–3933, 2013.

[3] Q. Cai, W. Hou, C. Yu, P. Han, L. Zhang, and L. Guo, "Design and OPNET implementation of routing algorithm in 3D optical network on chip," in *2014 IEEE/CIC International Conference on Communications in China (ICCC)*, Shanghai, China, 2015.

[4] W. Hou, G. Lei, Q. Cai, and L. Zhu, "3D Torus ONoC: topology design, router modeling and adaptive routing algorithm (invited talk)," in *2014 13th International Conference on Optical Communications and Networks (ICOCN)*, Suzhou, China, 2014.

[5] P. Guo, W. Hou, L. Guo, Q. Cai, Y. Zong, and D. Huang, "Reliable routing in 3D optical network-on-chip based on fault node reuse," in *International Workshop on Reliable Networks Design & Modeling*, Munich, Germany, 2015.

[6] S. Werner, J. Navaridas, and M. Lujan, "Designing low-power, low-latency networks-on-chip by optimally combining electrical and optical links," in *IEEE International Symposium on High Performance Computer Architecture*, Austin, TX, USA, 2017.

[7] T. Barwicz, H. Byun, F. Gan et al., "Silicon photonics for compact, energy-efficient interconnects [invited]," *Journal of Optical Networking*, vol. 6, no. 1, pp. 63–73, 2007.

[8] M. Abdollahi and S. Mohammadi, "Insertion loss-aware application mapping onto the optical Cube-Connected Cycles architecture," *Computers & Electrical Engineering*, vol. 82, article 106559, 2020.

[9] W. Yang, Y. Chen, Z. Huang, H. Zhang, H. Gu, and C. Yu, "A survey of multicast communication in optical network-on-

chip (ONoC)," in *International Symposium on Parallel Architectures, Algorithms and Programming*, Singapore, 2020.

[10] L. Guo, W. Hou, and P. Guo, "Designs of 3D mesh and torus optical network-on-chips: topology, optical router and routing module," *China Communications*, vol. 14, no. 5, pp. 17–29, 2017.

[11] S. Sun, V. Narayana, I. Sarpkaya et al., "Hybrid photonic-plasmonic non-blocking broadband 5×5 router for optical networks," *IEEE Photonics Journal*, vol. 10, no. 2, pp. 1–12, 2018.

[12] X. Shi, N. Wu, F. Ge, G. Yan, Y. Xing, and X. Ma, "Srax: a low crosstalk and insertion loss 5×5 optical router for optical network-on-chip," in *IECON - 45th Annual Conference of the IEEE Industrial Electronics Society*, pp. 3102–3105, Lisbon, Portugal, 2019.

[13] M. R. Yahya, N. Wu, Z. Fang, F. Ge, and M. H. Shah, "A low insertion loss 5 × 5 optical router for mesh photonic network-on-chip topology," in *2019 IEEE conference on sustainable utilization and development in engineering and technologies (CSUDET)*, pp. 164–169, Penang, Malaysia, 2019.

[14] M. Fadhel, H. Gu, and W. Wei, "DORR: a DOR-based non-blocking optical router for 3D photonic network-on-chips," *EICE Transactions on Information & Systems*, vol. E104.D, no. 5, pp. 688–696, 2021.

[15] R. Kappeler, *Radiation testing of micro photonic components Stagiaire Project Report*, ESA/ESTEC. Ref. No. EWP 2263, 2004.

[16] Y. Ye, Z. Wang, P. Yang et al., "System-level modeling and analysis of thermal effects in WDM-based optical networks-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 11, pp. 1718–1731, 2014.

[17] M. C. Meyer, A. B. Ahmed, Y. Okuyama, and A. B. Abdallah, "FTTDOR: microring fault-resilient optical router for reliable optical network-on-chip systems," in *2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, Turin, Italy, 2015.

[18] I. Datta, D. Datta, and P. P. Pande, "Design methodology for optical interconnect topologies in NoCs with BER and transmit power constraints," *Journal of Lightwave Technology*, vol. 32, no. 1, pp. 163–175, 2013.

[19] H. U. Zhan-Shuo, F. Y. Hung, K. J. Chen, S. J. Chang, W. K. Hsieh, and T. Y. Liao, "Improvement in thermal degradation of ZnO photodetector by embedding silver oxide nanoparticles," *Functional Materials Letters*, vol. 6, no. 1, p. 1350001, 2013.

[20] P. Loh and W. J. Hsu, "Design of a viable fault-tolerant routing strategy for optical-based grids," in *International Symposium on Parallel and Distributed Processing and Applications*, Berlin, Heidelberg, 2003.

[21] X. Qi, Q. Feng, Y. Chen, D. Qiang, and D. Wenhua, "A fault tolerant bufferless optical interconnection network," in *Eigth IEEE/ACIS International Conference on Computer & Information Science*, Shanghai, China, 2009.

[22] M. Tinati, S. Koohi, and S. Hessabi, "Low-overhead thermally resilient optical network-on-chip architecture," *Nano Communication Networks*, vol. 20, pp. 31–47, 2019.

[23] P. Guo, W. Hou, L. Guo et al., "Fault-tolerant routing mechanism in 3D optical network-on-chip based on node reuse," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 547–564, 2020.

[24] M. Fadhel, L. Huang, and H. Gu, "RRW: a reliable ring waveguide-based optical router for photonic network-on-chips," in *International Symposium on Parallel Architectures, Algorithms and Programming*, Singapore, 2021.

[25] H. Li, H. Gu, Y. Yang, and Z. Zhu, "Impact of thermal effect on reliability in optical network-on-chip," *Optik - International Journal for Light and Electron Optics*, vol. 124, no. 20, pp. 4172–4176, 2013.

[26] M. Meyer, Y. Okuyama, and A. B. Abdallah, "SAFT-PHENIC: a thermal-aware microring fault-resilient photonic NoC," *The Journal of Supercomputing*, vol. 74, no. 9, pp. 4672–4695, 2018.

[27] X. Chen, M. Mohamed, Z. Li, L. Shang, and A. R. Mickelson, "Process variation in silicon photonic devices," *Applied Optics*, vol. 52, no. 31, pp. 7638–7647, 2013.

[28] Y. Xu, J. Yang, and R. Melhem, "Tolerating process variations in nanophotonic on-chip networks," in *39th Annual International Symposium on Computer Architecture (ISCA)*, Portland, OR, USA, 2012.

[29] M. Meyer, Y. Okuyama, and A. B. Abdallah, "Microring fault-resilient photonic network-on-chip for reliable high-performance many-core systems," *The Journal of Supercomputing*, vol. 73, no. 4, pp. 1567–1599, 2016.

[30] M. Abdollahi and S. Mohammadi, "Vulnerability assessment of fault-tolerant optical network-on-chips," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 140–159, 2020.

[31] R. Min, R. Ji, Q. Chen, L. Zhang, and L. Yang, "A universal method for constructing N-port nonblocking optical router for photonic networks-on-chip," *Journal of Lightwave Technology*, vol. 30, no. 23, pp. 3736–3741, 2012.

[32] https://www.vpiphotonics.com.

[33] H. Gu, K. H. Mo, J. Xu, and W. Zhang, "A low-power low-cost optical router for optical networks-on-chip in multiprocessor systems-on-chip," in *2009 IEEE Computer Society Annual Symposium on VlSI*, pp. 19–24, Tampa, FL, USA, 2009.

[34] Y. Xie, M. Nikdast, J. Xu et al., "Crosstalk noise and bit error rate analysis for optical network-on-chip," in *47th ACM/EDAC/IEEE Design Automation Conference*, pp. 657–660, Anaheim, CA, USA, 2010.

[35] K. Zhu, H. Gu, Y. Yang, W. Tan, and B. Zhang, "A 3D multi-layer optical network on chip based on mesh topology," *Photonic Network Communication*, vol. 32, no. 2, pp. 293–299, 2016.

[36] H. Gu, X. Jiang, and W. Zheng, "A novel optical mesh network-on-chip for gigascale systems-on-chip," in *IEEE Asia Pacific Conference on Circuits & Systems*, pp. 1728–1731, Macao, China, 2008.

[37] H. Gu, X. Jiang, and W. Zheng, "ODOR: a microresonator-based high-performance low-cost router for optical networks-on-chip," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/software Codesign & System Synthesis*, pp. 203–208, Atlanta, GA, USA, 2008.

[38] A. W. Poon, X. Luo, F. Xu, and H. Chen, "Cascaded microresonator-based matrix switch for silicon on-chip optical interconnection," *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1216–1238, 2009.

[39] J. Zhao, H. Li, H. Gu, and X. Diao, "Model-based platform for design space exploration in single-microring-based silicon photonic interconnects on chip," *Optics Communications*, vol. 453, article 124375, 2019.

[40] Y. Xie, M. Nikdast, J. Xu et al., "Formal worst-case analysis of crosstalk noise in mesh-based optical networks-on-chip," *IEEE Trans VLSI Syst*, vol. 21, no. 10, pp. 1823–1836, 2013.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 3833644, 16 pages
https://doi.org/10.1155/2021/3833644

Research Article

Towards an Elastic Fog-Computing Framework for IoT Big Data Analytics Applications

Linh Manh Pham,[1,2,3] **Truong-Thang Nguyen**,[2] **and Tien-Quang Hoang**[4]

[1]*Graduate University of Science and Technology, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet, Cau Giay, Hanoi, Vietnam*
[2]*Institute of Information Technology, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet, Cau Giay, Hanoi, Vietnam*
[3]*VNU University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam*
[4]*Hanoi Pedagogical University 2, 32 Nguyen Van Linh, Xuan Hoa, Vinh Phuc, Vietnam*

Correspondence should be addressed to Linh Manh Pham; linhmp@vnu.edu.vn

Received 3 May 2021; Revised 8 June 2021; Accepted 28 July 2021; Published 15 August 2021

Academic Editor: Yong Zhang

IoT applications have been being moved to the cloud during the last decade in order to reduce operating costs and provide more scalable services to users. However, IoT latency-sensitive big data streaming systems (e.g., smart home application) is not suitable with the cloud and needs another model to fit in. Fog computing, aiming at bringing computation, communication, and storage resources from "cloud to ground" closest to smart end-devices, seems to be a complementary appropriate proposal for such type of application. Although there are various research efforts and solutions for deploying and conducting elasticity of IoT big data analytics applications on the cloud, similar work on fog computing is not many. This article firstly introduces AutoFog, a fog-computing framework, which provides holistic deployment and an elasticity solution for fog-based IoT big data analytics applications including a novel mechanism for elasticity provision. Secondly, the article also points out requirements that a framework of IoT big data analytics application on fog environment should support. Finally, through a realistic smart home use case, extensive experiments were conducted to validate typical aspects of our proposed framework.

1. Introduction

1.1. Deployment of IoT Big Data Analytics Applications in the Context of Fog Computing. The last decade has seen the emerging of cloud computing as a trendy technology and business model bringing to incremental value for cloud stakeholders. At the side of service consumers, this value comes from saving both deployment time and investment cost on IT infrastructure, offloading it to cloud service providers. Taking advantage of virtualization technology, hardware resources at data centers are shared by infrastructure service providers and sold to customers with a reasonable price. Platform or application services built on top of virtual resources now are delivered to the hands of cloud consumers with invoices billed by fine-grained time or resource unit like in the water or electric power industries.

However, Internet of Things (IoT) applications are not appropriate to be converted completely into services of the cloud computing model. For example, latency-sensitive IoT applications such as big data analytics (BDA for short) systems require prompt responses to outliers which need to be within several milliseconds or even microseconds in some special emergency cases. Moreover, these BDA systems can create petabytes of data that are not practical to stream back and forth between cloud data centers and end-devices. With this kind of application, it is better to keep some of their components staying at centralized clouds and move some of them down to the edge close to end-devices. The components at the edge should take care of operations requiring fast responses or reducing a huge amount of data which may consume much bandwidth if transferred over a wide area network. On the contrary, the components on the cloud is often responsible for compute-intensive long-running tasks on permanent data storage.

The fog-computing model, aiming at bringing computation, communication, and storage resources from "cloud to ground" closest to end-devices, seems to be an appropriately complementary proposal for such type of application. This model encourages developers to divide their applications into more fine-grained components deployed in the nodes throughout from the cloud, fog, to end-device strata as shown in Figure 1. The position of a component depends on specific tasks that this component has to be in charge of and whether its tasks are latency-sensitive or not.

Elasticity is a characteristic of cloud according to NIST [1]. Many cloud applications provide elasticity features to accommodate scalability and adapt to changes in demand. Unlike in the cloud, implementing elasticity for IoT applications in the fog is not an easy task, especially for BDA ones. An elasticity-supported platform for this kind of application must provide the ability to modelize all heterogeneous software and hardware components participating in these applications and distribute them throughout strata from the cloud, fog, to end-devices. The components of these modelized applications should be migrated horizontally between fog nodes or vertically between cloud and fog strata where there is an increase or decrease in the density of the end-devices. These components should be also duplicated where workloads from the end-devices increase. To avoid vendor lock-in issues, those components also should be moved flexibly among IoT service providers when needed. Application programming interfaces need to be provided so that intercommunication among components can transparently cross boundaries created by different communication protocols.

In short, we are lacking a holistic fog framework which allows IoT BDA service providers to deploy and conduct elasticity on their applications to adapt fluctuation of big data workload coming from various IoT end-devices.

1.2. Our Contributions. Before talking about our contributions, we discuss dedicated features that a fog elasticity framework should support in addition to the features inherited from the cloud.

First, a distributed application needs to be modelized before its automatic deployment. In cloud computing, a standard modeling domain-specific language such as TOSCA [2] or Camel [3] is enough to abstract both software and hardware components of application services. However, in cloud computing, it is often that only a small set of concepts need to be described such as "cloud provider," "virtual machine," or "hosting server." These sets of terms need to be extended to fulfill the demand of describing a large of new concepts of fog computing coming from widely heterogeneous components such as gateway, set-top box, base station, physical machine, and cloudlet.

Secondly, mobility is a conspicuous characteristic of end-devices in fog computing. The end-devices can move or are moved physically from this geographical area to another one such as vehicles and smartphones or eliminated at one place and replaced logically at another position such as short-lived wireless sensors. Along with these movements, some components located at the upper strata of a fog application such as fog or cloud nodes should be moved or migrated correspondingly. An elasticity controller for fog-based applications needs to provide modules to take the end-device mobility and its migrated workload into consideration.

In the third place, software or hardware components in fog do not always stay with the same provider. They can be distributed horizontally between fog providers or vertically across both fog and cloud providers. Moreover, one component can be found in the fog at this moment but can be migrated to the cloud at another time when some input conditions vary. Therefore, the cloud/fog federation to break the vendor lock-in issue is another concern for developers of fog elasticity tools.

Finally, fog computing's ecosystem is dominated by millions of chatty embedded devices with thousands kind of communication protocols. On the other hand, complex applications have many components which need to be interconnected to enhance automation and autonomy. These interconnections can be either fog-fog, fog-cloud, cloud-cloud, fog-devices, or cloud-devices. Providing mechanisms for interconnection between the components is one of the critical missions of a modern fog elasticity platform.

Supporting such extended requirements needs a holistic framework that can catch the required aspects of configuration to deliver a highly automated system for managing any IoT BDA application on the fog. In this paper, we present AutoFog, which supports the transformation of complex applications to fog-based ones as well as supports their large-scale automatic deployment and scaling. The transformation is smooth and less time-consuming, thanks to the reuse and extension of an existing domain-specific language (DSL) [4] and off-the-shelf components (COTS). Besides the extension of the DSL, another contribution is the introduction of a mechanism for automatic deployment and elasticity provisioning which automatically implements all the predefined component instances as well as monitors and conducts all the elasticity strategies in fog environment. The last one is a runtime system ensuring that the deployed fog application is properly globally configured while scaling the application model such as adding, removing, or migrating component instances including fog/cloud nodes. We also have implemented a prototype for the framework and conducted extensive experiments to evaluate AutoFog in deploying and scaling a real-world IoT BDA application on typical aspects that a fog-computing elasticity framework should resolve.

The rest of this article is organized as follows. Section 2 describes a real-world complex IoT BDA use case as a fog-based application that we use throughout this paper. Section 3 discusses important modules of our proposed framework. Section 4 presents our mechanism for dynamic deployment and elasticity provision of IoT applications. Section 5 reports some extensive experiments performed on a prototype of AutoFog serving as a proof of concept of our work. Finally, section 6 presents related work, and section 7 concludes the paper.

2. Use Case

In fog computing, components of an application are not only divided into tiers but also distributed to the three strata:

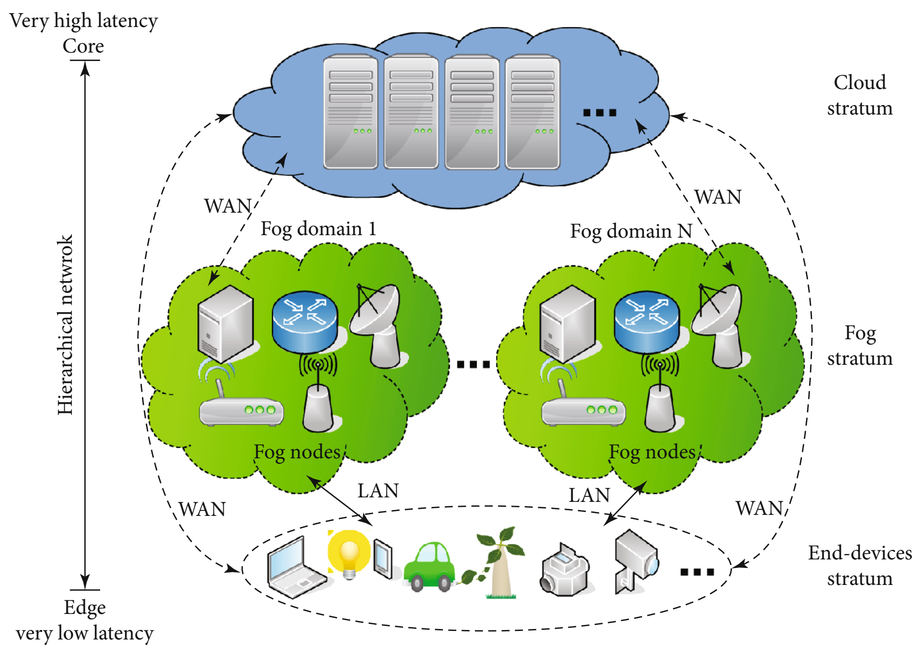


Figure 1: Three strata of fog-computing environment.

cloud, fog, and end-devices. We consider in this section a use case of a complex IoT BDA application which is divided into strata as shown in Figure 2. The components of the application can appear at positions marked by package icons. It is an energy management application for smart homes. In the application model, the smart home center manages and monitors the power consumption of multiple houses in a district.

There may have thousands of end-devices that consume electricity serving for human regular activities in the houses. Some mobile end-devices can be moved between the houses such as smartphones, laptops, and vehicles. This movement can cause temporarily peak demand in some discrete houses. In each household of a house, many IoT smart plugs are implemented to measure the energy consumption of end-devices. A smart plug is installed between the electrical smart device and the wall power outlet. A range of sensors is equipped in a smart plug to measure various values of associated power consumption. These raw data from thousands of smart plugs are sent to local agencies of the smart home center located closest to the corresponding houses for preprocessing or abnormal detection. Fog nodes in clusters, cloudlets, or private clouds are implemented in the local agencies to perform these operations. If an anomaly such as an unusual peak load or an outage is detected at this step, corresponding reactions are triggered from the application components distributed to the fog nodes. These reactions can be sending a simple notification to the administrator or adding more electric power from renewable energy sources to fulfill a peaking consumption demand. To ensure that these operations are fast and timely, the connection from the local agencies to the houses must be ensured by high-speed local transmission lines.

The preprocessed data are sent to more compute-intensive nodes in the cloud for further analysis to generate more valuable information such as energy consumption prediction during a period. The processed data and generated information can be stored permanently in cloud storage for long-running batch-processing tasks which may need to be conducted later. The IoT BDA application for energy management in smart homes is the real-world example used throughout this article.

3. AutoFog Architecture

As mentioned, fog computing adds an intermediate stratum between cloud and end-devices resulting in participation of more heterogeneous and fine-grained resources. In general, AutoFog architecture detailing in Figure 3 is composed of modules distributed into 4 layers: design, orchestration, elasticity, and infrastructure.

3.1. Design Layer. The design layer allows users to abstract and generalize complex distributed applications into application models using concepts defined by the framework. At heart of this layer is AutoFog DSL, a domain-specific language evolving from [4], which supports the description of hardware and software components of the application model and its fine-grained resources arranged hierarchically. In this language, the abstraction of a software component is called a *software type*. Similar software types can be generated from a software type template. These templates are stored in a software type catalog of the design layer. A software type is instantiated into *software instance* which is the running version of this type. Software instance inherits all the parameters and default values of its software type.

AutoFog DSL also proposes terms of *container type* and *container instance*. A container type represents a physical or virtual hardware component/device hosting software instances at runtime. It is worth mentioning that a small

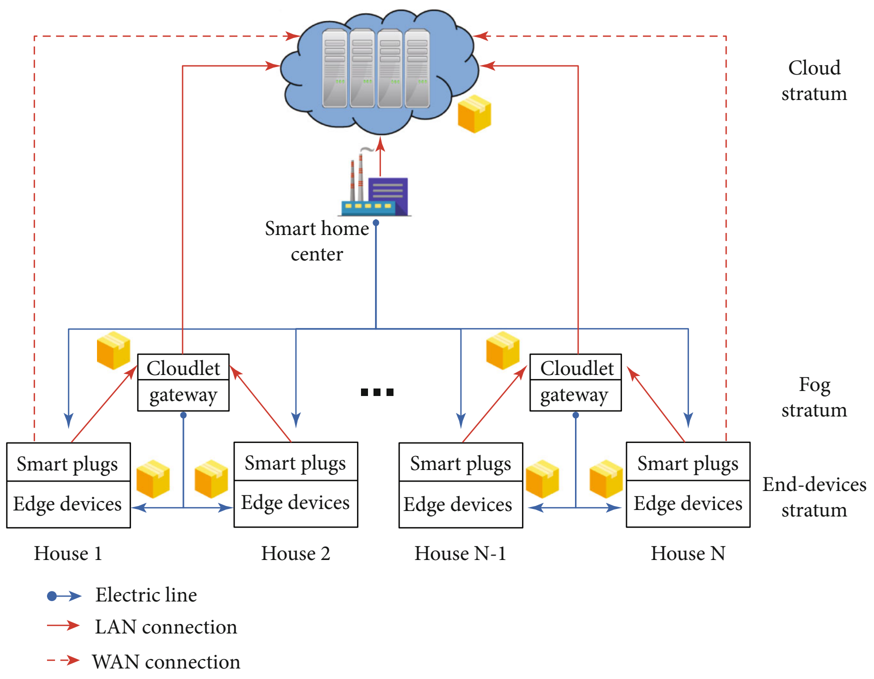


Figure 2: The IoT BDA use case for an energy management application.

end-device or a huge cloud data center can also be represented by a container type. Like software type templates, container type templates also can be stored in a container type catalog. Container instance is a container type in running state. The software instances instantiated from the same type can be distributed into multiple container instances of different container types. Moreover, software instances of different software types may collocate on the same container instance. Each software instance contains a reference to the container instance on which it is running.

Relations between software types (i.e., horizontal relationship) can be defined in the application model by series of "exported" and "imported" configuration variables. While an exported variable is a structure that a component exposes to remote components using it, an imported variable is a structure containing configuration information required by a component to initialize a service. Receiving the imported structure to boot is mandatory or not depending on specific software types.

Another kind of relation supported in AutoFog DSL is the parent-child relationship between a software type and its containers (i.e., vertical relationship). This relation in cloud applications is usually a simple map between the software components and their hosts (e.g., virtual machines). With fog-computing applications, it is often that a software component is deployed inside multiple levels of software and hardware containers. For example, a Tomcat war file is contained inside a Tomcat server, the Tomcat server, in turn, is packed in a Docker container, and a Docker container is hosted in a virtual machine of a cloudlet or a physical machine of a fog cluster. AutoFog DSL supports such a complex description to fulfill the gap when defining very heterogeneous resources of fog-computing applications.

In AutoFog, an application is a collection containing descriptions of container types, container instances, software types, software instances, and vertical/horizontal relations. An excerpt of the IoT BDA application model under AutoFog DSL is depicted in Figures 4 and 5. As shown in the figures, the means used for installation and configuration of the software instances in the container must be defined such as Bash, Chef, or Puppet. Corresponding to the selected technology, some scripting files defining operations on how to install and configure the software on the container may need to be provided along with the application model.

3.2. Orchestration Layer. Since a completed model of the fog-based application is sent to the orchestration layer, the model is parsed, and the life cycle of the application is managed and ensured by *Application Manager* (AM). It also checks the application's current state (not deployed, deploying, deployed and stopped, starting, deployed and started, stopping, etc.). Through this module, the running application can be updated by adding/removing types and instances to/from current application model. Application Manager has a global view of the application, all software components, and the links between them, but it does not intervene in the physical deployment of software components and containers. A copy of the current global view is sent to *Placement Manager* (PM) to compute a placement plan when the application is initialized or updated. Placement Manager supports various kinds of solvers such as constraint programming-based solvers, heuristics-based solvers, learning automata-based allocator, and metasolvers. The users need to select one of the supported solvers depending on what is more important to them, accuracy or performance. Another module in this layer is *Monitoring Manager* which is used to monitor states of container

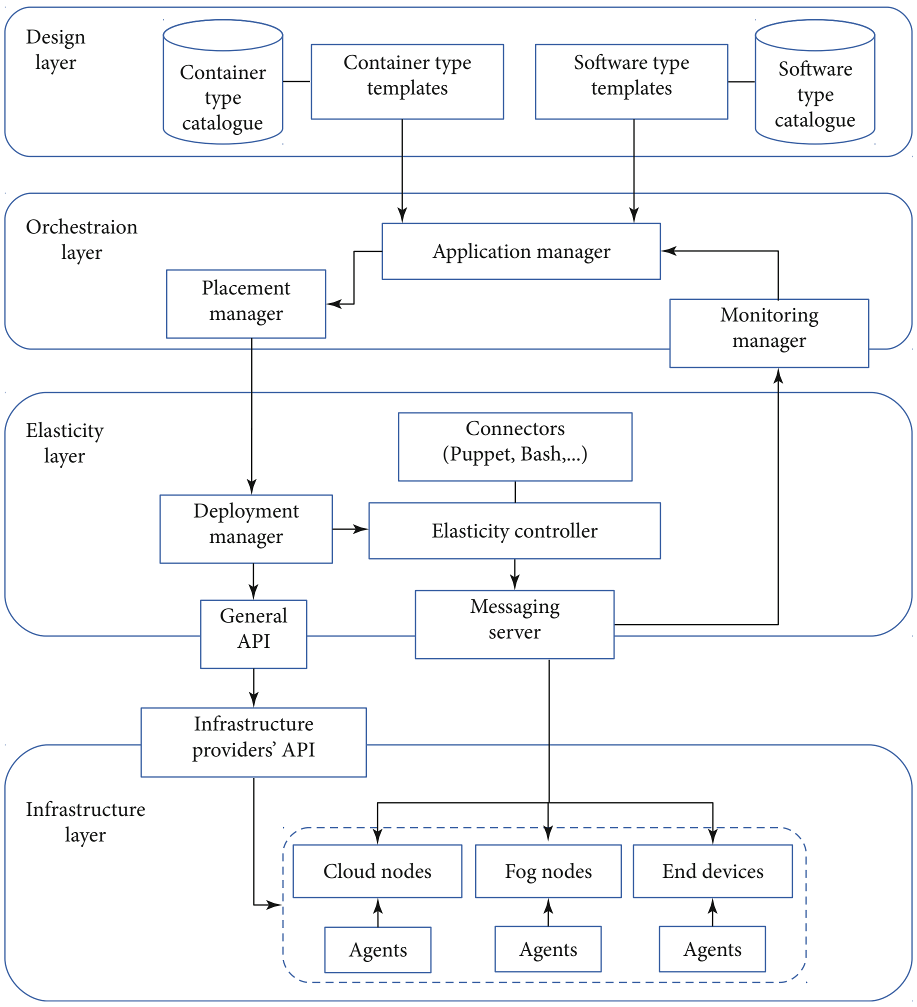


Figure 3: AutoFog architecture and interactions.

instances using heartbeats and notify the administrator that the container went down.

3.3. Elasticity Layer. The core components of this layer are *Deployment Manager* (DM) and *Elasticity Controller* (EC). They are modules coordinating the physical deployment of the application across containers. They must ensure all software instances are deployed with the correct configuration in hierarchical container instances. The DM instantiates containers through the provider's API, and the EC manages the deployment and scaling of the software types on the containers. Another mission of the DM is ensuring the federation between infrastructure providers. To do this, heterogeneous infrastructures from these providers must be abstracted. In the cloud, the DM must ensure that the software instances hosted in different containers belonging to various cloud providers work as in the same provider. To avoid vendor lock-in issues, some access lists may need to be added according to each provider's policy. In fog, this federation needs to be enforced not only among cloud providers but also between cloud and fog providers. Therefore, AutoFog provides a flexibly plug-in mechanism to add and abstract different cloud/fog providers. It provides a general AutoFog API with critical infrastructure primitives including the creation and deletion of software or container instances as well as minimal information about their states. These general requests will be translated and sent to specific cloud/fog providers' APIs, thanks to their corresponding infrastructure plug-ins. Thus AutoFog is completely independent of any fog/cloud infrastructure.

Right after a software instance is switched to running state, the EC maintains an *admin* topic on a *Messaging Server* to keep track of all components. The EC, therefore, plays an important role since it constitutes the entrance for both the initial configuration and the upcoming scale(s). Briefly, this module has the responsibility of elasticity control of the components it manages.

3.4. Infrastructure Layer. Installation and configuration of software at the Infrastructure layer are done by *AutoFog Agents*. An agent is a lightweight software installed in advance inside a container instance to manage the installation and elasticity of software instances of this container. Therefore, each agent only knows about the local components of its hosted container. In general, it is responsible for carrying out communication on

```
val iot_bda: application = { name="IOT_BDA",

  softwareTypeList= SOME
   (* Storm Cluster for Event Stream Processing *)
    [{name="Storm-Cluster",
      varList= SOME [{name="ip",value=""},
                     {name="port",value=""}],
      exportedStructList= NONE,
      importedStructList= SOME [{name="OpenHAB",
                                 channel= NONE,
                                 varName=["OpenHAB.ip","OpenHAB.port"],
                                 requiredToBoot=true}],
      hostingType="Cloudlet-Node", configurator="bash", children="Nimbus, Storm-Supervisor"},

    (* OpenHAB: Home Automation Bus *)
     {name="OpenHAB",
      varList= SOME [{name="ip",value=""},
                     {name="bindingChoice",value=""}],
      exportedStructList= SOME [{name="OpenHAB",
                                 channel= NONE,
                                 varName=["ip","bindingChoice"],
                                 requiredToBoot=true}],
      importedStructList= SOME [{name="SmartPlug",
                                 channel= NONE,
                                 varName=["SmartPlug.ip","SmartPlug.fogDomain"],
                                 requiredToBoot=true}],
      hostingType="Cloudlet-Node", configurator="puppet", children= NONE},

    (* Cassandra: Cloud Storage *)
     {name="Cassandra",
      varList= SOME [{name="ip",value=""},
                     {name="portCQL",value=""}],
      exportedStructList= SOME [{name="Cassandra",
                                 channel= NONE,
                                 varName=["ip","portCQL"],
                                 requiredToBoot=true}],
      importedStructList= SOME [{name="Storm-Supervisor",
                                 channel= NONE,
                                 varName=["Storm-Supervisor.ip","Storm-Supervisor.port"],
                                 requiredToBoot=true}],
      hostingType="VM-EC2", configurator="chef", children= NONE},
...

  containerTypeList= SOME
    (* EC2 VM - A Cloud Node *)
     [{name="VM-EC2",
      configurator= CLOUD_IAAS, vmTypeName="m4.xlarge", children="Cassandra, Storm-Cluster"},

    (* CLOUDLET NODE - A Fog Node *)
     {name="Cloudlet-Node",
      configurator= FOG_IAAS, fogTypeName="docker", children="OpenHAB, Storm-Cluster"}]
...
```

Figure 4: Types of IoT BDA application described by AutoFog DSL.

behalf of its container. A software instance is configured by the agent using the configuration connector specified in its corresponding software type. Additionally, the agents publish variables a software instance exports (i.e., exported vars) and variables this instance imports (i.e., imported vars) to corresponding topics in the messaging server. The agents communicate with each other and with the remaining AutoFog modules, thanks to communication channels in the messaging server.

A messaging protocol has also been implemented based on exchanging asynchronous messages and publishing/subscribing message topics which allows the upper layers to dynamically add/remove containers as well as software components to a running application. Using messaging services to exchange messages promotes interoperability between application components. All communication protocols used by fog hardware components need to be abstracted and converted to uniform messages supported by the message server. Currently, AutoFog implements RabbitMQ as its unique messaging service. Supports for other messaging brokers or services can be added to AutoFog as new plug-ins. In the following section, we describe how AutoFog manages its applications at runtime.

4. Dynamic Deployment and Elasticity Provision

In this section, we describe how AutoFog can be used to install and manage the IoT BDA application mentioned in

```
   softwareInstanceList= SOME                                                              1
    (* Nimbus instance *)                                                                  2
      [{name="storm-nimbus-1", softwareTypeName="Nimbus",                                  3
      varList= SOME [{name="ip", value=""},                                                4
                     {name="port", value="6627"}],                                         5
      exportedStructList= SOME [{name="Storm-Nimbus",                                      6
                                channel= NONE,                                             7
                                varName=["ip","port"],                                     8
                                requiredToBoot=true}],                                     9
      importedStructList= SOME [{name="Storm-Supervisor",                                  10
                                channel= NONE,                                             11
                                varName=["Storm-Supervisor.ip","Storm-Supervisor.port"],   12
                                requiredToBoot=true}],                                     13
      configurator= INHERITED, hostingInstanceName= NONE, state= NONE},                    14
                                                                                           15
     (* Cassandra instance *)                                                              16
      {name="cassandra-1", softwareTypeName="Cassandra",                                   17
      varList= SOME [{name="ip", value=""},                                                18
                     {name="portCQL", value="9042"}],                                      19
      exportedStructList= INHERITED,                                                       20
      importedStructList= INHERITED,                                                       21
      configurator= INHERITED, hostingInstanceName= NONE, state= NONE},                    22
                                                                                           23
    (* OpenHAB instance *)                                                                 24
      {name="openhab-1", softwareTypeName="OpenHAB",                                       25
      varList= SOME [{name="ip", value=""},                                                26
                     {name="bindingChoice", value="SmartPlugBinding"}],                    27
      exportedStructList= INHERITED,                                                       28
      importedStructList= INHERITED,                                                       29
      configurator= INHERITED, hostingInstanceName= NONE, state= NONE}],                   30
...                                                                                        31
                                                                                           32
  containerInstanceList= AUTO []                                                           33
                                                                                           34
  autoProtection= SOME                                                                     35
    [{defaultRules= {                                                                      36
      rule="iptables -A INPUT -i lo -j ACCEPT",                                            37
      rule="iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT",       38
      rule="iptables -A INPUT -p icmp -j ACCEPT",                                          39
      rule="iptables -A INPUT -p tcp --dport ssh -j ACCEPT",                               40
      rule="iptables -P FORWARD DROP",                                                     41
      rule="iptables -P INPUT DROP"}}],                                                    42
                                                                                           43
  autoRepair= NONE []                                                                      44
...                                                                                        45
```

Figure 5: Instances of IoT BDA application described by AutoFog DSL.

section 2. We depict in Figure 6 various steps required to deploy the IoT BDA application and conduct elasticity on the Fog using AutoFog. Below are details of these steps:

(1) The model of the fog application is sent to AutoFog to be deployed. Figures 4 and 5 represent the IoT BDA application model under AutoFog DSL. Figure 4 shows the different software and container types of the application. As depicted, we describe some software and container types such as Storm cluster [5] (Figure 4, lines 5-13), Cassandra [6] (Figure 4, lines 30-41), and OpenHAB [7] (Figure 4, lines 16-27). The model first is parsed by the AM module to generate a provider-independent model (PIM). The PIM then goes through the constraint-problem solvers of the PM module to generate a provider-specific model (PSM) describing which containers belonging to which infrastructure of platform providers will host the software components

(2) The PSM is sent to the DM module at the beginning of this step. The DM, through the local General API, contacts the infrastructure/platform API to ask for the instantiation of container instances. For example, in the case of the IoT BDA application, we initially deploy one Nimbus instance (Figure 5, lines 3-14), one Cassandra instance (Figure 5, lines 17-22), and one OpenHAB instance (Figure 5, lines 25-30). These instances are deployed on container types named “Cloudlet-node” (Docker [8]) on the fog or “VM-EC2” on the cloud. The users can either specify explicitly the name of a hosting container instance for a specific software instance or leave this task for the solvers. At the end of the step, the infrastructure/platform providers instantiate the container instances

(3) Each container instance has a message queue in the messaging server. The DM asks the EC to include software instance definition and corresponding scripting

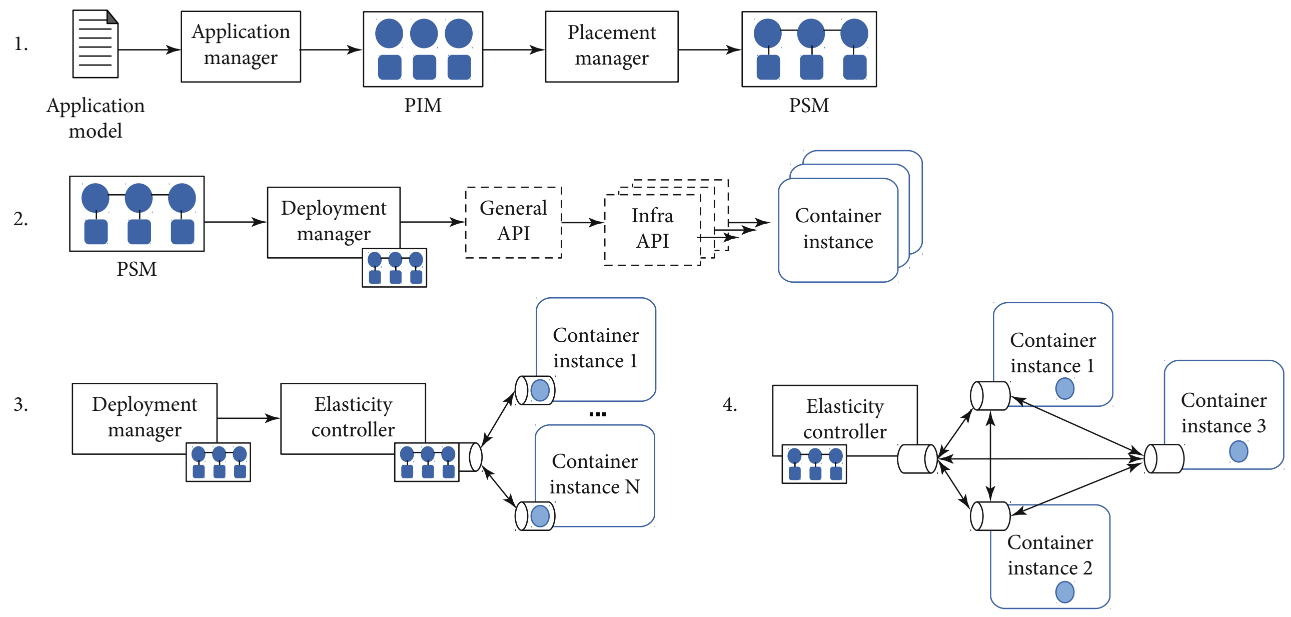


Figure 6: Steps of the AutoFog mechanism of dynamic deployment and elasticity provision.

files onto the message queues including the EC queue. When containers are started up and running, they receive information of instances that they are in charge of from the messaging server and start to install them

(4) All software and container instances of the application take part in the application-wide configuration after being installed. The imported and exported variables are exchanged, and when an event triggers elasticity actions, they are scaled out/in automatically. This ensures the correctness of the elasticity mechanism.

From now on, container instances are autonomous and independent from other modules of AutoFog. They can exchange information with each other, thanks to corresponding topics in the messaging server. This decentralized approach allows the application to work even when errors occur on the EC.

5. Evaluation

We implemented a prototype of AutoFog framework and conducted several experiments validating its functionality in terms of auto deployment and elasticity. The Smart Home BDA application in section 2 is chosen as the deployment's target of the framework.

5.1. Experiment Setup. The modules of AutoFog and software components of the smart home BDA application were developed to be packaged easily into Docker containers. These containers are orchestrated by the Kubernetes cluster [9] implemented in our homegrown infrastructure at the VNU University of Engineering and Technology (VNU-UET). Each Kubernetes pod is configured to contain only one container. The experimental implementation is depicted in Figure 7. In our implementation, fog nodes are ensured by Kubernetes and Cloud nodes are provisioned by OpenStack [10]. Because Kubernetes can provide both elasticity function and fog node, it works at both elasticity and infrastructure layers of AutoFog architecture. We have also created different Docker images which are all embedded an AutoFog agent beforehand:

(i) AutoFog image contains main modules of AutoFog such as AM, PM, DM, and EC. This image is used for AutoFog nodes at both Fog and Cloud strata

(ii) Storage image contains an instance of Cassandra, a NoSQL distributed database management system. It supports handling large amounts of data across many nodes with a highly available service. Its data model allows incremental modifications of rows. This image is used to instantiate the storage nodes where permanent data of the smart home BDA application are stored at the cloud stratum

(iii) Storm Master/worker images represent for two types of Storm nodes: Nimbus master node and Supervisor worker node at fog strata. The master node manages cluster of Storm Supervisor nodes where Storm topology is submitted to execute. Storm topology is composed of Spouts who pump data to the topology and Bolts who consume and process the data in parallel from Spouts

(iv) OpenHAB image includes an OpenHAB message binding which gathers measurements from smart plugs and forwards them to the Storm cluster. OpenHAB nodes created from this image working as edge gateways locate at the border between end-device and fog strata

(v) Message server image contains a RabbitMQ server to asynchronously handle message queuing telemetry

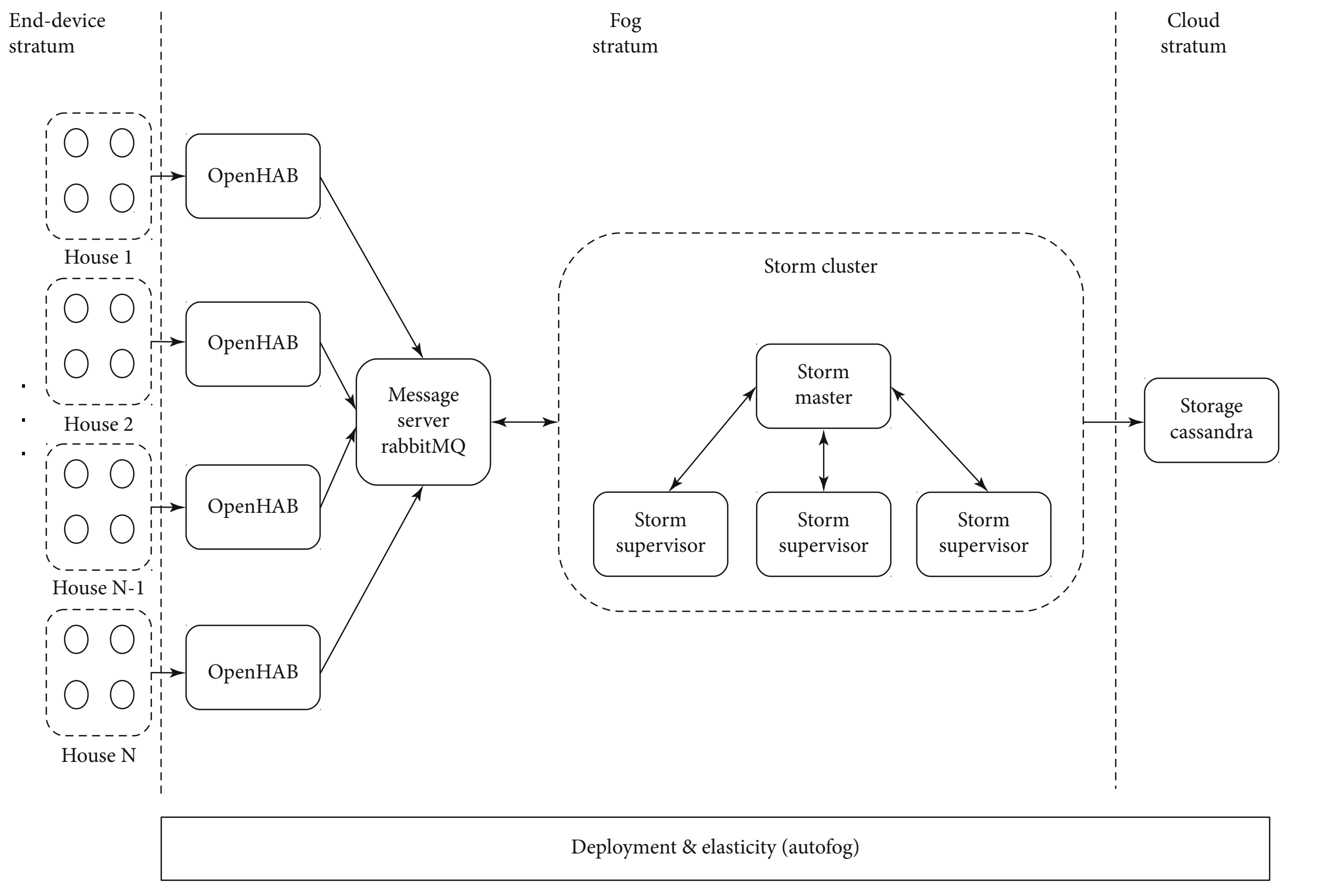


Figure 7: The experimental implementation of the smart home BDA application.

transport (MQTT) messages back and forth in the system. MQTT is a lightweight communication protocol broadly used for IoT applications [11]. This image is used for message server nodes at both fog and cloud strata.

In our experiments, the input data are synthesized from a practical data source provided by DEBS grand challenge 2014 [12]. The dataset contains over 4055 million of measurements for 2125 smart plugs deployed in multiple houses in Germany. The full data cover a period of one month in September 2013. We have developed an end-device image including a program which regenerates measurements retrieved from the DEBS dataset.

All the practical deployment times are calculated over 20 different runs to get the mean. A new container is needed for each software instance. The time in Table 1 covers the instantiation of the container, the initial configuration of the software until they reach states from which they can be started. For example, Storm Master (Nimbus) is the one whose instances take the longest time to deploy with ≈248 seconds on average.

The initial deployment contains one Cassandra storage node at the cloud stratum, one Storm Master node and one to two Storm Supervisor nodes working at fog stratum, two OpenHAB nodes working as edge gateways, and a maximum of 40 end-device nodes. Modules of AutoFog and its message server are grouped into one node called AutoFog node. The selection of these software components is just one of many specific combinations of IoT BDA applications. Other combinations can also be used in the experiments without losing the generality and validity of the AutoFog architecture.

Table 1: Deployment time of five smart home BDA's instances over 20 runs.

Software types	Mean	99th percentile
Storm Master	248.05 ± 6.4	221.16 ± 9.4
Storm Supervisor	112.02 ± 16.1	125.53 ± 12.3
OpenHAB	145.43 ± 8.9	154.21 ± 9.7
RabbitMQ	112.71 ± 14.8	118.04 ± 11.2
Cassandra	103.88 ± 11.6	98.09 ± 8.5

5.2. Storm Topology. Storm is one of components of the smart home BDA application. Thus, Storm can be described by AutoFog DSL at the design layer and deployed and managed by submodules of both orchestration and elasticity layers. Storm topology to process the DEBS IoT data is shown in Figure 8. The topology is composed of 5 components as follows.

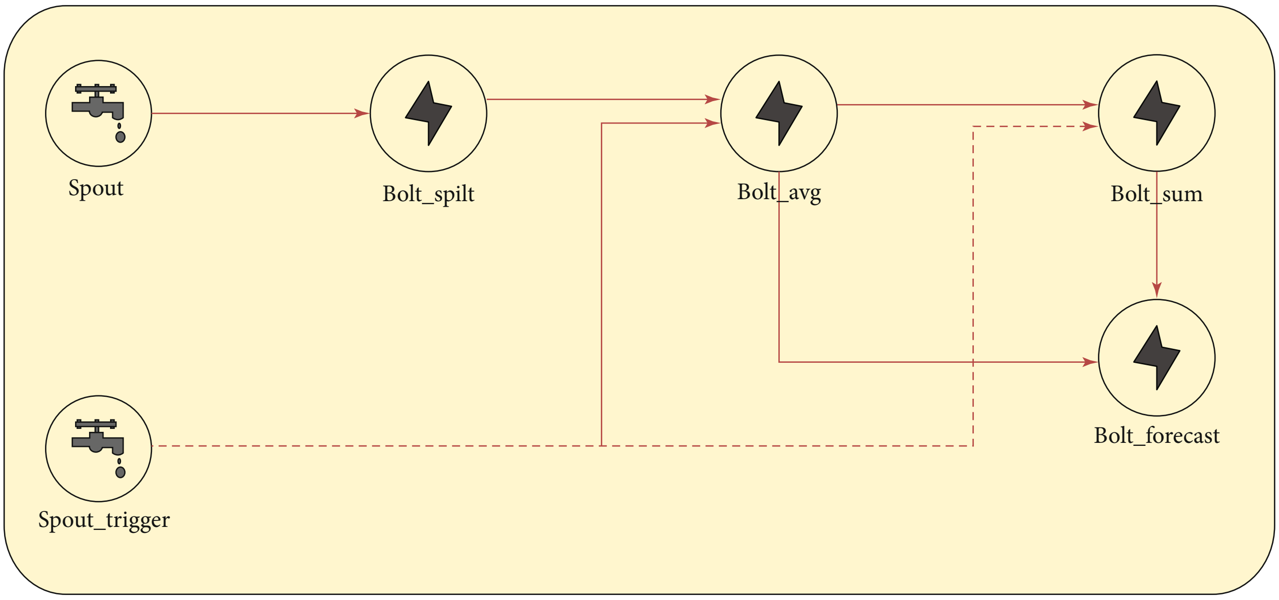


Figure 8: Storm topology of the smart home BDA application.

(i) Spout_data: it has the function to create MQTT clients that connect to the message broker, subscribe to predetermined topics, receive data from the broker, separate the data into meaningful fields, and then send them to the back Bolts for processing

(ii) Bolt_split: it has the function to read data sent from Spout_data, read the timestamp field, divide the data into time slices with predetermined window sizes according to the system's needs (1 minute, 5 minutes, to 120 minutes), and then send it to Bolt_avg for further processing

(iii) Bolt_avg: its function is to receive data from Bolt_split to calculate the average amount of electricity that the device uses in time slice with predetermined window sizes according to the needs of the system (1 minute, 5 minutes, to 120 minutes). The data after calculating will be saved to RAM memory and then sent to Bolt_sum for further processing. In addition, Bolt_avg stores the average data of the energy consumed by each device in the local database. The data stored on the database will be released to save memory to ensure the long-term operation of the system

(iv) Bolt_sum: it has the function to add the total energy used of all the equipment in the house to calculate the total amount of electricity consumed by that house in the time slice with predetermined window sizes according to the needs of the system. Similar to Bolt_avg, Bolt_sum stores the average data of the energy consumed by each house in a local database. The data stored on the database will also be released to save memory to ensure the long-term operation of the system

(v) Bolt_forecast: it has the function that uses data from previous time slice to predict energy usage value of next two time slices and then save it to database.

To vary IoT workload to the Storm topology, in the Bolt_forecast, we implement three prediction models making short-term electric load forecast of smart IoT devices. The utilization of these models causes differences in the amount of input tuples used in Storm's Bolts, especially in Bolt_avg, Bolt_sum, and Bolt_forecast. In the first model, time is divided into time slices, and the load average of any future time slice is predicted based on the average electric load of the previous ith time slices having the same timeframe of all preceding days. Assuming we predict the average load of the second time slice $P(ts_{i+2})$ from the current slide ts_i, the formula used is

$$P(ts_{i+2}) = \frac{\text{avgLoad}(ts_i) + \text{median}\left(\text{avgLoad}(ts_j)\right)}{2}. \quad (1)$$

In formula (1): $\text{avgLoad}(ts_i)$ is the average load of current time slice ts_i, $\text{avgLoad}(ts_j)$ is the average load of time slices ts_j —time slices of previous days have the same timeframe as slice ts_i—and $\text{median}(\text{avgLoad}(ts_j))$ is the median of all previous time slices ts_j.

With the second model, $\text{avgLoad}(ts_j)$ is the average load of all previous time slices in the same day up to the current time slice. For the third one, $\text{avgLoad}(ts_j)$ is average load of time slices of previous weeks having the same timeframe as slice ts_i on the same date.

5.3. Result. We publish messages from the smart home dataset to the Storm topology of the smart home IoT application. In Bolt_forecast, we change the prediction models, and results measured on Storm Nimbus Master node representing many experimental runs are shown in Figure 9. The time on the x-axis is the execution time for experimental system to publish all messages from the DEBS data files and process these messages through Bolts of the Storm topology. The

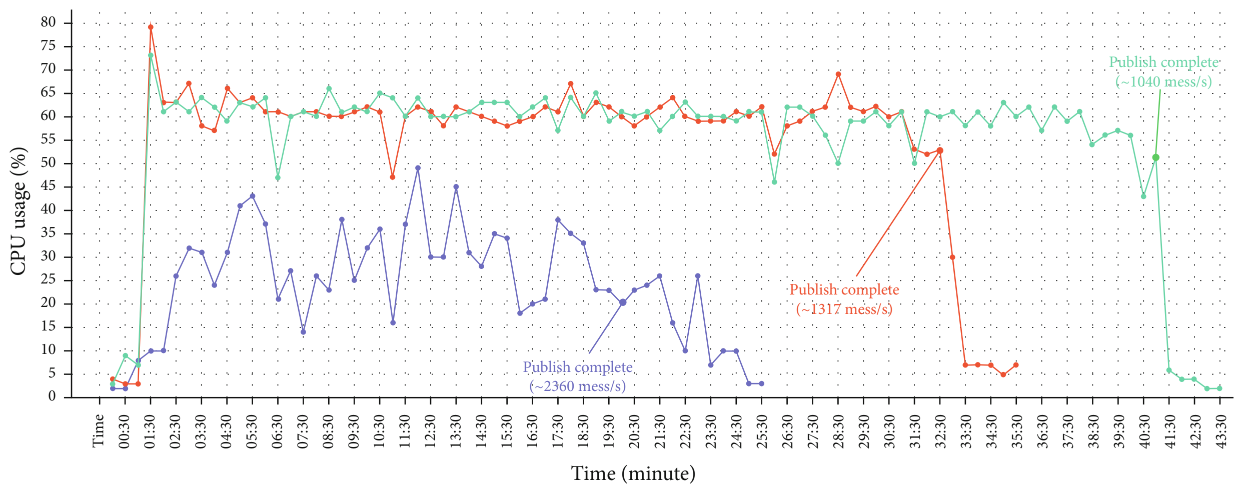


FIGURE 9: CPU usage over time of processing smart home dataset with three prediction models. The green, red, and blue lines represent the first, second, and third prediction models.

green, red, and blue lines represent the first, second, and third prediction models, respectively. The execution time of the blue one lasts about 20 minutes with the average CPU usage is quite different from time to time, averaging approximately 30%. The average throughput at the end of data processing is about 2306 messages per second. As for the green and red ones, the CPU usage is roughly the same, maintaining at 65-70%, and at a time spike can be as high as 75-80%. The time to publish and process data with the first model is about 41 minutes, greater than the second one which is about 32.5 minutes. The average throughput at the end of data processing of the first and second models is about 1040 and 1317 messages per second, respectively. The main reason for the difference in execution time is that the prediction models use different amounts of historical data leading to various computation times in the Bolts. The predicting results of all three prediction models are depicted in Figure 10.

We see in Figure 9 that both the first and second models exhibit the average CPU usages higher than 75%. Therefore, elasticity strategies can be applied to reduce the average CPU usage and at the same time shorten the execution time. To perform elasticity, two techniques can be implemented on Kubernetes: Vertical Pod Autoscaler and Horizontal Pod Autoscaler.

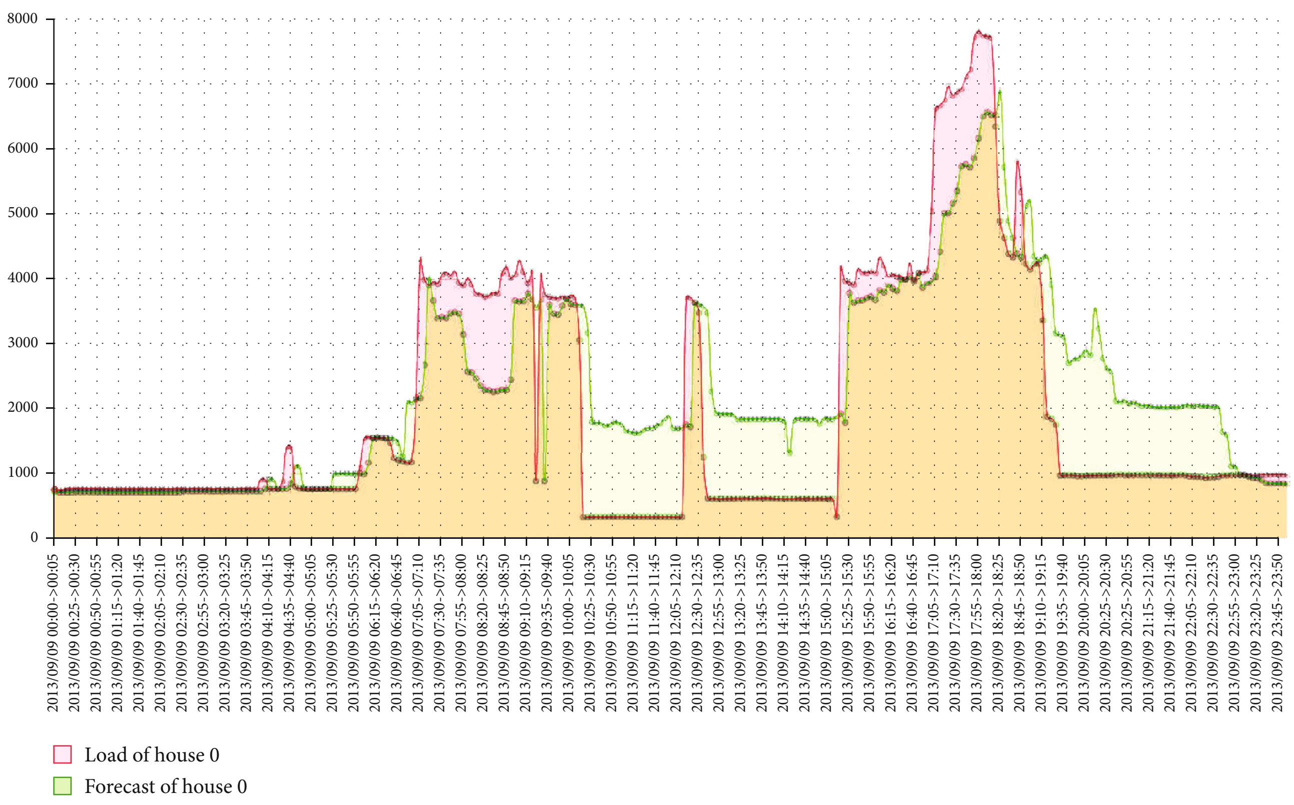
Horizontal Pod Autoscaler (HPA) is a technique to automatically increase or decrease the number of Kubernetes pods by collecting and evaluating CPU usage metrics from the Kubernetes Metrics Server. The number of pods will be in the range min and max which are set when generating HPA. The HPA is implemented as a Kubernetes API resource and as a controller. Every 15 seconds, the controller periodically checks and adjusts the number of pods so that the observed average CPU usage matches the value specified by the user. HPA calculates the number of pods based on a formula where Ceil() is the rounding function:

$$\#\text{RequiredPods} = \text{Ceil}\left(\#\text{CurrentPods} \times \frac{\text{PresentValue}}{\text{ExpectedValue}}\right). \tag{2}$$

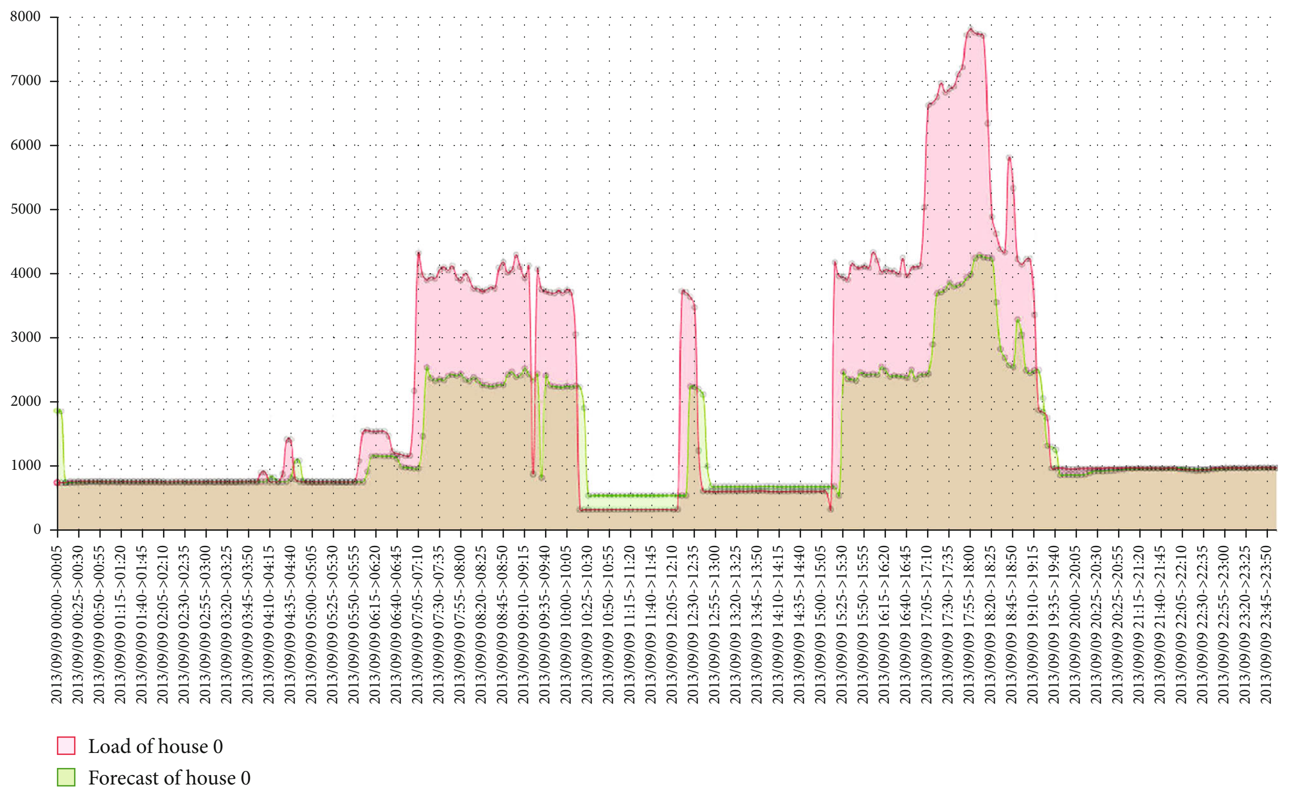
Vertical Pod Autoscaler (VPA) is a technique that automatically increases and decreases resources such as CPU and memory for pods depending on the needs of the pods. Technically, VPA does not dynamically change resources for existing pods; instead, it checks the managed pods to see if the resources are set correctly and, if incorrectly, removes them so that the controller can create other pods with updated configurations.

5.3.1. Horizontal Elasticity. Without loss of generality, we conduct HPA with the first model only. A HPA object for the deployment and the pod "StormWorker" are built with the following limits: When the average CPU usage greater than 75% will trigger an auto scale up increasing number of StormWorker pods, it will do a scale down to decrease the number of pods. When the used RAM memory over 3 GB (75%) will perform auto scale up, it will do a scale down. The minimum and maximum numbers of pods are 1 and 5, respectively. Elasticity results are shown in Figure 11. The horizontal scaling mechanism responds very quickly and works quite smoothly to changes in pod's CPU resource usage even when resource usage spikes during very small amount of time. This mechanism does not cause downtime of the Storm workers during use. We see that the average CPU usage and execution time are reduced to 40% and 36.5 minutes in the case of using elasticity comparing to 75% and 41 minutes in the case of not using elasticity.

5.3.2. Vertical Elasticity. By default, for stability, the VPA will not perform an automatic update of pod resources if the number of pod copies is less than 2. So two pods are created with each pod configuration as following: the minimum resource is 0.5 CPU core and 1 GB RAM; the maximum resource is 1 CPU core and 2 GB RAM. Next, a VPA object is created for pods with *updateMode=Auto*. After the pods are created, the IoT load is injected to the two pods with input messages from the DEBS dataset. The obtained results are depicted in Figure 12. Each line with a specific color is a representation of a pod containing the running container of

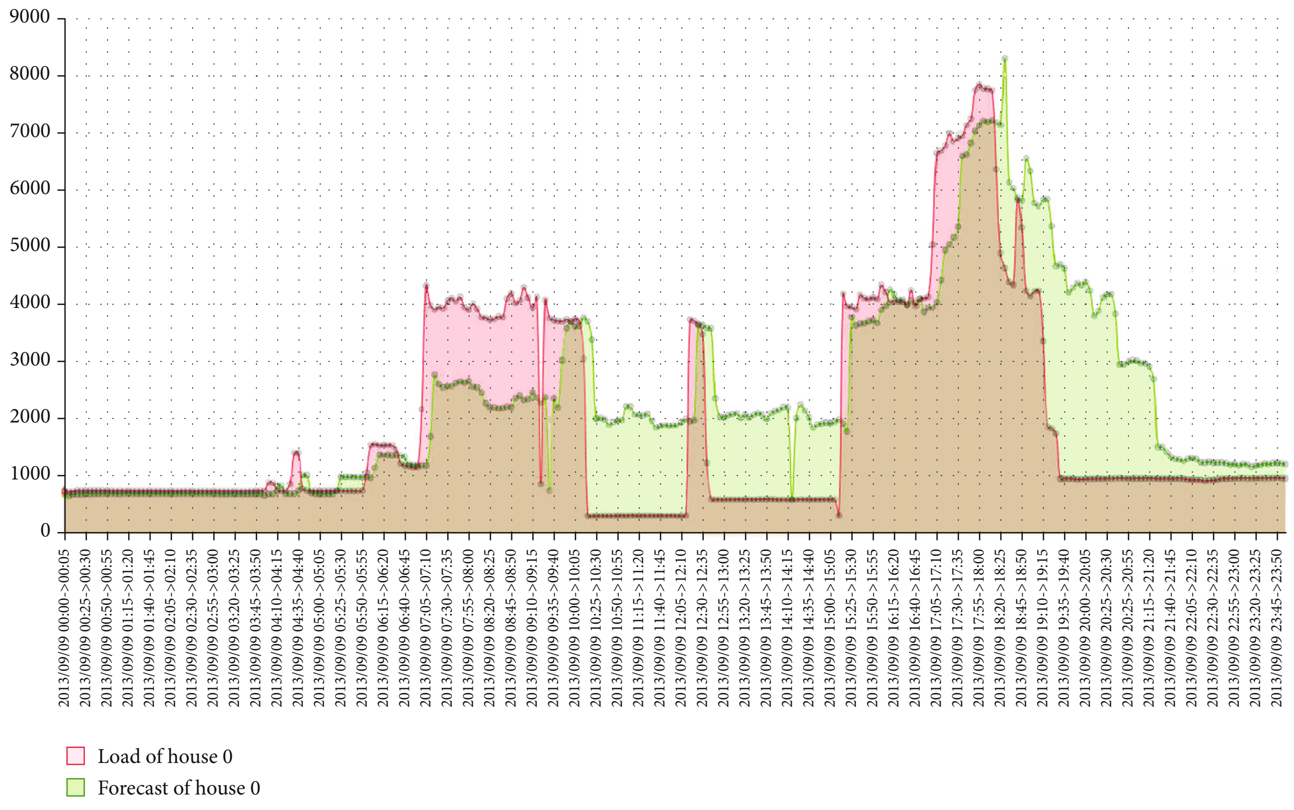


(a)



(b)

FIGURE 10: Continued.



(c)

Figure 10: Results of all three prediction models. (a)The first model. (b)The second model. (c)The third model.

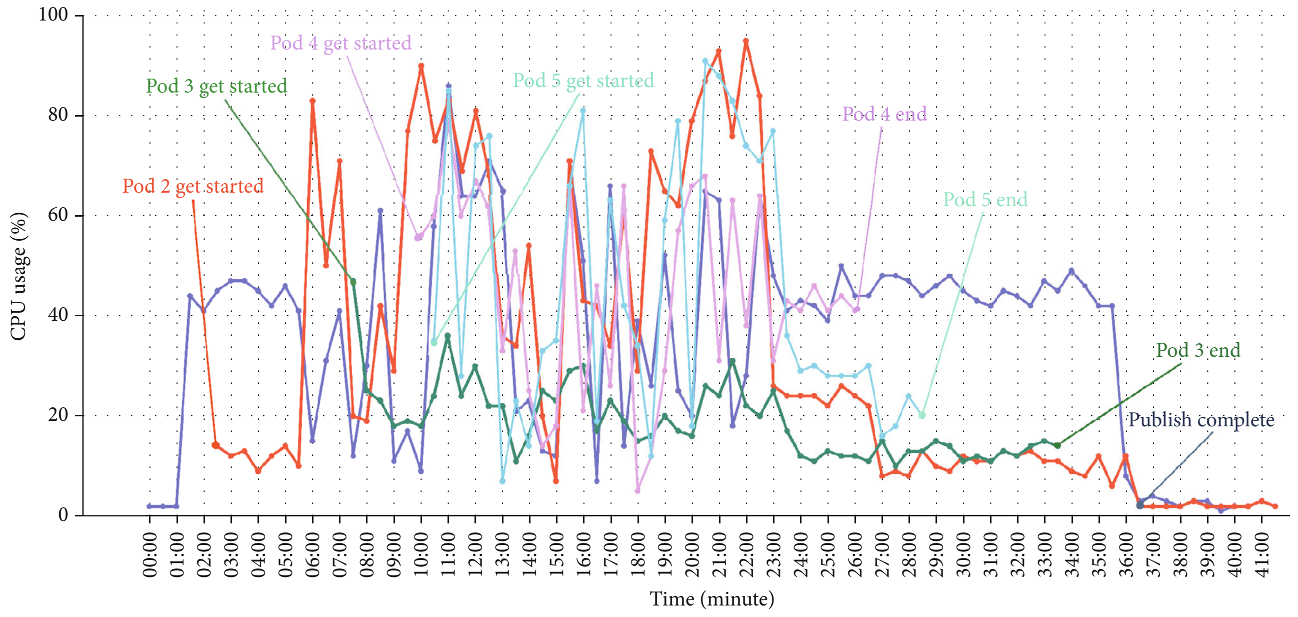


Figure 11: CPU resources are automatically scaled horizontally; 5 lines represent 5 StormWorker pods.

Storm worker. After vertical elasticity actions, two pods (orange, blue) with minimum configuration are replaced by two new pods with maximum configuration (green, red), respectively. It is obvious that the average CPU usage is reduced significantly.

At the moment, Kubernetes did not have a mechanism for updating resources directly on a running pod; replicating pod is the only way. Therefore, VPA can probably bring unexpected downtime for the Storm worker. When VPA creates new pods and causes downtime in about 30 seconds,

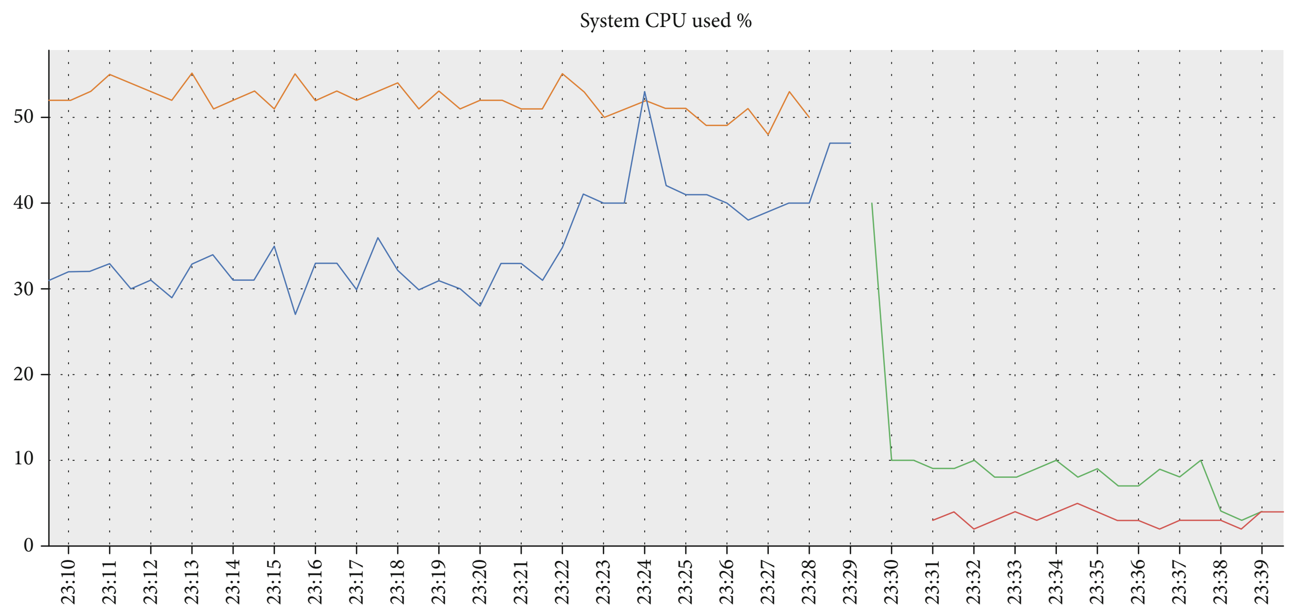


FIGURE 12: Real-time average CPU usage when using VPA.

there is a certain amount of messages lost during that time that depends on the speed of publishing. After that, the succeeding connections could be functioning normal.

6. Related Work

6.1. General Frameworks for Fog-Based IoT BDA Applications. For almost a decade since the introduction of fog computing, many frameworks have been being proposed to support fog-based IoT BDA applications. Almost all frameworks own one [13–18] or multiple fog orchestrators (FO) [19–21] operating at the orchestration layer. With the former, FO must have holistic view of fog resources and connect to all fog nodes in the framework. Multiple FO can resolve the scalability issue of the single one but might incur some overhead from communication between these FO.

Chen et al. propose a FA2ST (fog-as-a-service technology) fog framework supporting any kind of IoT application [14]. On-demand discovery of fog service is provided to figure out if a connected fog node's resource is currently available when an IoT request comes. In another research, an IoV-fog infrastructure is defined to provide supports to overworked RSUs of UAVs [16]. Such a RSU can trigger a deployment of UAV, and data is migrated to this UAV to decrease response latency and increase the IoV computation. Storm, a stream processing platform, is extended by Cardellini et al. to enable a distributed IoT resource scheduler which is latency aware [13]. Fog nodes in this extension have knowledge of resource availability of each other and thus ensures QoS of IoT service distribution. Donassolo et al. propose FITOR, a Fog-IoT ORchestrator which monitors the fog infrastructure and keeps track of every fog resources anytime [15]. It helps to deploy the IoT data to fog nodes automatically. Foggy framework introduced by Yigitoglu et al. allows the deployment of IoT task requests to an appropriate fog node having available resources and satisfying several QoS requirements such as priority, latency, and privacy [18]. In the same vein, Foggy FOC uses MQTT protocol to monitor all fog resources [17]. To increase future deployment, it has a mechanism to store historical IoT workloads and requirements.

To increase security and reliability, Fogbus [20], a scalable fog framework, partitions fog nodes into various roles including computing, gateway, repository, and broker nodes. A defective fog node can be restored by repository nodes and taken over by other fog nodes. A blockchain solution is applied to validate dependability of IoT data sources. Fog nodes are clustered into colonies in research of Skarlat et al. [19]. In their fog architecture, FO of each colony keeps all fog available resource information. IoT requests firstly are allocated to fog nodes in a colony. If the colony does not have enough resources, the FO will find another colony to fulfill the tasks through transferring the requests using REST API. It also can propagate the requests to the cloud stratum if appropriate. Data migration between fog nodes and RSUs in an IoV-fog application is considered by Zhang et al. [21]. Multiple fog nodes in a region are grouped into a cluster and managed by a coordinator (FO). If a vehicle moves to a new region, an IoT module may be handed over to another fog cluster to avoid interrupted IoT processing.

Although these frameworks are aimed at satisfying deploying and provisioning fog resources using one or multiple FO, they do not take elasticity feature into account as in our research.

6.2. Elasticity Frameworks for Fog-Based IoT BDA Applications. Although a large number of frameworks are proposed for fog-based IoT BDA applications, not many studies consider elasticity for this kind of application. Mobile fog [22] proposes a scaling mechanism where overloaded workloads are resolved by fog nodes created dynamically. It also properly distributes

IoT data to these new fog nodes. Moreover, its API data migration is suitable for ambulant IoT devices like smart phones, cameras, and vehicles. To enable elasticity for IoT data stream processing applications using container, Wu et al. modify Kubernetes HPA to adapt at runtime the deployment of containerized BDA applications to the estimated load arrival rate [23]. In a similar way, Netto et al. scale Docker containers in Kubernetes using a state machine approach [24]. Adaptive AI services run on IoT gateways and fostered on the cloud are enabled by Elastic-IoT-Fog (EiF), a flexible fog-computing framework [25]. EiF virtualizes an IoT service layer platform and orchestrates various fog nodes. The feasibility of elasticity feature in EiF is depicted via an example of intelligent traffic flow management and monitoring, in which network slicing units and respective resource elasticity are dynamic provisioned. Zanni et al. present and report the evaluation of a system consisting of virtual services in a combined fog, cloud, and IoT environments with various device settings [26]. By using geometric monitoring, the paper proposes an original solution to dynamically scale and provision the resources for the fog-computing layer. Elasticity is expressed in aspect of moving and redeploying more mobile components to the fog nodes closest to the targeted end-devices. Wang et al. design a three-tier edge computing system architecture to dynamically route data to proper edge servers and elastically adjust their computing capacity for the real-time urban surveillance applications [27]. Moreover, the paper also introduces schemes of workload balance and resource redistribution in emergency situations. The EU ELASTIC project is aimed at developing a software architecture for extreme-scale BDA in fog-computing ecosystems [28]. With the architecture, ELASTIC supports elasticity across the fog compute strata while fulfilling communication, real-time, energy, and secure properties.

The above-mentioned studies and solutions bring elasticity feature for the resources of IoT BDA applications on fog-computing environment but do not mention the automatic deployment of these applications based on the description of given software/hardware components and deployment plans as the function provided by AutoFog.

7. Conclusion

We have presented AutoFog, a framework with a four-layer architecture, which supports transformation of IoT BDA applications to elastic fog-based ones and automatic deployment of these applications on fog environment. A mechanism of elasticity provision is integrated into the framework to enable adaptation to changes of workload from IoT smart devices. The transformation is more smooth and less time-consuming through the reuse and extension of an existing domain-specific language and off-the-shelf components. The validating experiments with the practical smart home use case were conducted with Kubernetes for fog nodes and OpenStack for cloud nodes. The results show that the implementation of AutoFog framework accompanied by our proposed elasticity mechanism is more flexible and faster when there was fluctuations in managed resources.

Data Availability

The CSV data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research is funded by Graduate University of Science and Technology under grant number GUST.STS.ĐT2019-TT02.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing (draft)," *NIST Special Publication*, vol. 800, no. 2011, p. 145, 2011.

[2] A. Brogi, J. Soldani, and P. W. Wang, "TOSCA in a nutshell: promises and perspectives," in *In Service-Oriented and Cloud Computing*, M. Villari, W. Zimmermann, and K.-K. Lau, Eds., pp. 171–186, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2014.

[3] A. Sintef, "Cloud application modelling and execution language (CAMEL) and the PaaSageWorkflow," *In European Conference on Service-Oriented and Cloud Computing*, vol. 567, pp. , 2015437–439, 2015.

[4] L. M. Pham and T. Pham, "Autonomic fine-grained migration and replication of component-based applications across multi-clouds," *In 2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, pp. , 20155–10, 2015.

[5] A. Storm, https://storm.apache.org/releases/2.2.0/index.html. Accessed:2021-04-19.

[6] A. Cassandra, https://cassandra.apache.org/doc/latest/ .Accessed:2021-04-19.

[7] OpenHAB, https://www.openhab.org/docs/. Accessed:2021-04-19.

[8] Docker container, https://docs.docker.com/.Accessed:2021-04-19.

[9] Kubernetes website, https://kubernetes.io/.Accessed:2021-04-19.

[10] OpenStack, "Open source cloud computing infrastructure," https://www.openstack.org/.Accessed:2021-04-19.

[11] MQTT, https://mqtt.org/mqtt-specification/.Accessed:2021-04-19.

[12] DEBS, "Grand challenge: smart homes," 2014, https://debs.org/grand-challenges/2014/.Accessed:2021-04-19.

[13] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, "On QoS-aware scheduling of data stream applications over fog computing infrastructures," *2015 IEEE Symposium on Computers and Communication (ISCC)*, pp. 271–276, 2015.

[14] N. Chen, Y. Yang, T. Zhang, M. Zhou, X. Luo, and J. K. Zao, "Fog as a service technology," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 95–101, 2018.

[15] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos, "Fog based framework for IoT service provisioning," *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC).*, pp. 1–6, 2019.

[16] N. Madan, A. W. Malik, A. U. Rahman, and S. D. Ravana, "On-demand resource provisioning for vehicular networks using flying fog," *Vehicular Communications*, vol. 25, no. 2020, p. 100252, 2020.

[17] D. Soni and A. Makwana, "A survey on MQTT: a protocol of internet of things(IOT)," *In International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT)*, 2017.

[18] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, "Foggy: a framework for continuous automated IoT application deployment in fog computing," *In 2017 IEEE International Conference on AI Mobile Services (AIMS)*, pp. , 201738–45, 2017.

[19] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for IoT services in the fog," *In 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. , 201632–39, 2016.

[20] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: a blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, no. 2019, pp. 22–36, 2019.

[21] W. Zhang, Z. Zhang, and H. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[22] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the Internet of Things. In Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing (Hong Kong, China) (MCC'13)," *Association for Computing Machinery*, 2013, pp. 15–20, New York, NY, USA, 2013.

[23] Y. Wu, R. Rao, P. Hong, and J. Ma, "FAS: a flow aware scaling mechanism for stream processing platform service based on LMS," *In Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences (Wuhan, China) (ICMSS'17). Association for Computing Machinery*, 2017, pp. 280–284, New York, NY, USA, 2017.

[24] H. V. Netto, A. F. Luiz, M. Correia, L. de Oliveira Rech, and C. P. Oliveira, "Koordinator: a service approach for replicating Docker containers in Kubernetes," *In 2018 IEEE Symposium on Computers and Communications (ISCC). 00058–00063*, 2018.

[25] J. An, W. Li, F. L. Gall et al., "EiF: toward an elastic IoT fog framework for AI services," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 28–33, 2019.

[26] A. Zanni, F. Stefan, U. Jennehag, and P. Bellavista, "Elastic provisioning of Internet of Things services using fog computing: an experience report," *2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. , 201817–22, 2018.

[27] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing. In Proceedings of the Workshop on Smart Internet of Things (San Jose, California) (SmartIoT'17)," , New York, NY, USA, Association for Computing Machinery, 2017.

[28] EU ELASTIC website, https://elastic-project.eu/about/objectives.Accessed:2021-04-19.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8693978, 11 pages
https://doi.org/10.1155/2021/8693978

Research Article

Equation Chapter 1 Section 1 Differentially Private High-Dimensional Binary Data Publication via Adaptive Bayesian Network

Sun Lan, Jinxin Hong, Junya Chen, Jianping Cai, and Yilei Wang

College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China

Correspondence should be addressed to Yilei Wang; yilei@fzu.edu.cn

Received 17 April 2021; Accepted 21 June 2021; Published 17 July 2021

Academic Editor: Yong Zhang

When using differential privacy to publish high-dimensional data, the huge dimensionality leads to greater noise. Especially for high-dimensional binary data, it is easy to be covered by excessive noise. Most existing methods cannot address real high-dimensional data problems appropriately because they suffer from high time complexity. Therefore, in response to the problems above, we propose the differential privacy adaptive Bayesian network algorithm PrivABN to publish high-dimensional binary data. This algorithm uses a new greedy algorithm to accelerate the construction of Bayesian networks, which reduces the time complexity of the GreedyBayes algorithm from $O(nkC_{m+1}^{k+2})$ to $O(nm^4)$. In addition, it uses an adaptive algorithm to adjust the structure and uses a differential privacy Exponential mechanism to preserve the privacy, so as to generate a high-quality protected Bayesian network. Moreover, we use the Bayesian network to calculate the conditional distribution with noise and generate a synthetic dataset for publication. This synthetic dataset satisfies ε-differential privacy. Lastly, we carry out experiments against three real-life high-dimensional binary datasets to evaluate the functional performance.

1. Introduction

Various data are continuously collected and stored in different information systems with the continuous development of information technology. In the actual application process, people often encounter various data, such as medical, market trade, and travel track. These data usually have hundreds or thousands of attribute dimensions, and some are even higher. If these data are released, it may cause the leakage of sensitive personal information because high-dimensional data usually contain numerous personal privacy information. Therefore, this requires consideration of some measures to protect these information. However, protecting data privacy while ensuring data availability is a very challenging problem. The main reason is that the publishing space formed will grow exponentially as the attribute dimension increases.

The traditional privacy protection methods are mainly k-anonymity [1], (α, k)-anonymity [2], t-closeness [3], l-diversity [4], etc. However, all of these methods require special attack assumptions and knowledge background as support. It cannot be applied to general scenarios. Nevertheless, the data processed by techniques, such as differential privacy [5] and random disturbance [6, 7], do not need to make a series of conditional assumptions for the attacker and can be applied to various problem scenarios universally. Therefore, in recent years, such related technologies, especially differential privacy, have received increasing attention. Differential privacy is a typical data perturbation technique that perturbs information by adding noise that satisfies a specific distribution into the data. The disturbing data still retains the original statistical characteristics, but the attacker cannot reconstruct the real original data.

Many high-dimensional data publishing methods based on differential privacy are available, but these methods can only solve the problem to a certain extent, some problems still exist: first, these methods usually deal with the dimensional disasters caused by high-dimensional data by converting them into low-dimensional data forcibly. This will cause serious information loss.

Second, the time complexity of these methods is generally too high. Although it can handle data of any dimension, in theory, it can only handle low dimension in the actual operation process. Because it takes such a long time, it cannot satisfy the need for higher-dimensional data.

Third, although most existing methods can handle high-dimensional binary data, it is easy for these methods to add too much noise, which leads to the data being completely covered, thus affecting the accuracy of publishing.

To address the challenges above, we propose the PrivABN algorithm, which is a high-dimensional binary data publishing method. Our main contributions are presented as follows:

(1) Instead of directly adding noise to the data, we use the Bayesian network method to avoid the impact of the dimensional disaster. In this way, the increase of global sensitivity with attribute dimension can be avoided, and the dimensional disaster can be solved effectively

(2) To reduce the total time complexity of the algorithm and enable it to process real high-dimensional data, a construction algorithm ABN is proposed by using a greedy algorithm, adaptive algorithm, and differential privacy index mechanism

(3) We propose a synthetic data generation algorithm SDG by using the characteristics of binary data and the topological order of the Bayesian network. This algorithm can reduce the magnitude of added noise and prevent excessive noise from covering the actual value

2. Materials and Methods

The materials and methods section should contain sufficient detail so that all procedures can be repeated. It may be divided into headed subsections if several methods are described.

3. Related Work

So far, many differential privacy publishing methods for high-dimensional data are available. Aiming at the privacy protection of high-dimensional binary data, Qardaji et al. [8] proposed the PriView method. This method assumes that all attributes are independent of each other and, then, answers user queries by constructing a set of low-dimensional noisy views, thereby reducing the impact of dimensional disasters. Zhang et al. [9, 10] proposed the PrivBayes method, which was directed at the issue of high-dimensional data privacy release. This method assumes that all attributes have a certain correlation and, then, constructed a Bayesian network between the attributes of the dataset by a greedy algorithm. Next, the Bayesian network is used to calculate the noisy joint distribution among attributes, and this joint distribution was utilized to generate a synthetic dataset for release. Based on the PrivBayes method, a series of derivative methods, such as weighted PrivBayes [11], Jtree [12], PrivHD [13], and PrivMN [14], has been proposed one after another. Wang et al. [11] set up a method for calculating attribute weights. They think that the importance of different attributes is different, and they will choose these important attributes first when building a Bayesian network. Chen et al. [12] used sparse vector sampling techniques to explore the relationships among attributes. Then, these relationships are used to build a Markov network (a special Bayesian network). Based on the Markov network, the joint tree algorithm is used to accelerate the solution of joint distribution, and the differential privacy protection is realized by adding Laplace noise to the joint distribution. Subsequently, Zhang et al. [13] introduced high-pass filtering technology based on the Jtree method to accelerate the construction of the Markov network. Finally, the PrivMN method, which is also based on the Markov network, is proposed by Wei et al. [14] to solve the joint distribution among attributes. The difference is that the approximate reasoning method is used in the calculation of the joint distribution.

The above analysis suggests that most of the existing methods consider how to construct the Bayesian or Markov network better to obtain a higher-precision joint distribution, and reducing publishing errors. However, these methods have high time complexity, which makes it impossible to process real high-dimensional data in practical applications. Moreover, the constructed Bayesian network still cannot reflect the true distribution well because of the degree's limitation. Therefore, this study proposes the PrivABN algorithm to solve the problems above.

4. Theorems and Definition

4.1. Differential Privacy

Definition 1 (ε-differential privacy). Let $D, D' \in \chi^d$ be two neighboring datasets, i.e., D and D' differ in only one record. Giving a randomized mechanism A, if A satisfies ε-differential privacy, the following is true:

$$\Pr\left[A(D) = O\right] \leq \exp(\varepsilon) \times \Pr\left[A\left(D'\right) = O\right], \tag{1}$$

where ε is the privacy budget and the smaller the privacy budget is, the higher the degree of privacy protection will be. $\Pr\left[A(D) = O\right]$ and $\Pr\left[A(D') = O\right]$ represent the probability that the algorithm A outputs as on the data set D and D', respectively.

Generally, Laplace [15] and Exponential mechanism [16] can realize differential privacy. Both of these mechanisms disturb the value or selection of the original data by generating noise. The magnitude of the generated noise is related to the global sensitivity of the query function.

Definition 2 (global sensitivity (see [15])). Let $f : D \longrightarrow \mathbb{R}^n$ be the query function. The global sensitivity is defined as

$$\Delta f = \max_{D,D'} \left\| f(D) - f\left(D'\right) \right\|_p, \tag{2}$$

where D and D' are two neighboring datasets and $\|.\|_p$ is the p-norm, which is a more commonly used 1-norm. Generally, the greater the global sensitivity is, the greater the noise generated by the mechanism and the impact on the algorithm results will be.

Theorem 1 (Laplace mechanism (see [15])). *Let $f : D \longrightarrow \mathbb{R}^n$ be the query function. Giving a randomized mechanism A, if A satisfies ε-differential privacy, the following is true:*

$$A(D) = f(D) + Lap(\Delta f / \varepsilon), \tag{3}$$

where $Lap(\Delta f/\varepsilon)$ is the noise variable that satisfies the Laplace distribution and where $Lap \sim Laplace(0, \Delta f/\varepsilon)$. Equation (3) shows that the larger the privacy budget ε or the smaller the global sensitivity Δf is, the smaller the noise generated will be.

Theorem 2 (Exponential mechanism (see [16])). *Let the score function $u(x)$ denote the score of x. Giving a random algorithm A, if A satisfies ε-differential privacy, the following is true:*

$$A(D) = \left\{ O_i \mid \Pr\left[A(D) = O_i\right] \in \exp\left(\frac{\varepsilon u(O_i)}{2\Delta u}\right)\right\}, \tag{4}$$

where Δu is the global sensitivity of the score function $u(x)$. This formula means that for each output result O_i of algorithm A, a probability of $\exp\left(\varepsilon \cdot u(O_i)/(2\Delta u)\right)$ *being selected is likely to exist. The higher the score of O_i is, the greater the probability of being selected will be.*

In addition, in designing and proving to meet the differential privacy algorithm, an important combination of differential privacy needs to be used.

Property 1 (sequential composition (see [17])). Giving a dataset D and a set of differential privacy algorithms $A_1(D), A_2(D), \cdots, A_m(D)$ and the algorithm $A_i(D)$ satisfies ε_i-differential privacy. Moreover, the random processes of any two algorithms are independent of each other. Then, the combination of these algorithms A satisfies $\sum_{i=1}^m \varepsilon_i$-differential privacy.

4.2. Bayesian Network. Bayesian network is a probabilistic graph model, mainly used to explore the relationship between a group of objects. Usually, a directed acyclic graph is used to represent the Bayesian network. The nodes in the graph represent objects, and the edges represent relationships.

In general, giving a set of attributes set $S = \{S_1, S_2, \cdots, S_m\}$, its joint distribution can be expressed as

$$\Pr[S] = \Pr[S_1] \cdot \cdots \cdot \Pr[S_m \mid S_1, \cdots, S_{m-1}]. \tag{5}$$

Through the Bayesian network constructed by the attribute set, its joint distribution can be approximated as

$$\Pr[S] \approx \Pr_{\mathcal{N}}[S] = \prod_{i=1}^{m} \Pr[S_i \mid \Pi_i], \tag{6}$$

where Π_i is the parent node set of node S_i. If the constructed Bayesian network can represent the relationship between attributes well, then $\Pr_{\mathcal{N}}[S] \longrightarrow \Pr[S]$.

Therefore, how to build a better Bayesian network is important.

4.3. Conditional Entropy. Conditional entropy can be used to measure the degree of interdependence among attributes. The larger the value, the higher the degree of dependence between attributes.

Definition 3 (conditional entropy). Giving two discrete random variables $X \in \{x_1, x_2, \cdots, x_n\}$ and $Y \in \{y_1, y_2, \cdots, y_m\}$, the conditional entropy between them is

$$I(X, Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} \Pr\left[x_i, y_j\right] \mathrm{lb} \frac{\Pr\left[x_i, y_j\right]}{\Pr[x_i]\Pr\left[y_j\right]}, \tag{7}$$

where $\Pr[x_i, y_j]$ is the joint distribution probability value of $X = x_i$ and $Y = y_j$.

Equation (7) shows that when $I(X, Y) \longrightarrow 0$, there is $\Pr[x_i, y_j] \longrightarrow \Pr[x_i]\Pr[y_j]$, that is, variables X and Y are close to independent of each other.

5. The PrivABN Algorithm

In Table 1, the meanings of the commonly used symbols in this section are provided, and the other symbols are explained when used.

5.1. Differential Privacy Bayesian Network Algorithm. Zhang et al. [9] proposed a conditional entropy-based degree Bayesian network construction algorithm GreedyBayes in PrivBayes. The main idea of this algorithm is to select a pair of the largest conditional entropy to join the current Bayesian network each time.

The GreedyBayes algorithm is a common algorithm used to construct Bayesian networks, and its implementation is shown in Algorithm 1.

Considering that Zhang et al. [9, 10] did not provide the time complexity formula of the algorithm in the article, this study demonstrates the time complexity of the GreedyBayes algorithm.

Table 1: Table of notations.

Notation	Description
D	Original data set
$\tilde{D}$	Synthetic data set
S	D's attribute set
n	Number of records in D
m	Number of attributes of D
$\mathcal{N}$	Bayesian network

Theorem 3. *The time complexity of the GreedyBayes algorithm is* $O(nkC_{m+1}^{k+2})$.

Proof. The time consumption of the GreedyBayes algorithm is mainly concentrated in the for loop of Step 3. The for loop is executed a total of $m-1$ times, and each time $|S \setminus V| \cdot C_V^k$ pairs of (S_i, Π_i) are generated. Therefore, a total of

$$\begin{aligned}(m-1)\cdot C_1^1+\cdots+(m-d)\cdot C_k^k+\cdots+C_{m-1}^k \\ = \frac{(2m-k-2)(k-1)}{2} + C_{k+1}^{k+1} + C_{k+2}^{k+1}+\cdots+C_m^{k+1}, \\ = \frac{(2m-k-2)(k-1)}{2} + C_{m+1}^{k+2}.\end{aligned} \tag{8}$$

(S_i, Π_i) pairs will be generated in the whole process. In the worst case, when$k+2=(m+1)/2$, any $k \in \mathrm{N}_+$ holds for

$$\begin{aligned}\frac{(2m-k-2)(k-1)}{2} = \frac{(3k+4)(k-1)}{2} < C_{m+1}^{k+2} \\ = \frac{(2\cdot(k+2))!}{((k+2)!)^2} \\ = \frac{(k+3)\cdot(k+4)\cdots(2k+3)\cdot(2k+4)}{1\cdot 2\cdots(k+1)\cdot(k+2)}.\end{aligned} \tag{9}$$

For each (S_i, Π_i) pair, the conditional entropy size needs to be calculated, and each calculation takes $O(nk)$ time. Therefore, the total time complexity of the algorithm is $O(nkC_{m+1}^{k+2})$.

Evidently, due to the influence of time complexity, the GreedyBayes algorithm can only be applied when the number of attributes m is small or the maximum degree k of the Bayesian network is small.

To process real high-dimensional data, a more complex Bayesian network is constructed. This paper proposes a simple and efficient Bayesian network construction algorithm ABN. This algorithm only needs the time complexity of $O(nm^4)$ to construct a complete Bayesian network.

In order to more intuitively illustrate the advantages of the ABN algorithm in time performance, we take a dataset containing 50 attributes as an example. When the maximum degree k of the Bayesian network is 5, 10, 15, 20, 25, the number of (S_i, Π_i) pairs enumerated by the GreedyBayes algorithm and ABN algorithm are shown in Table 2.

It can be seen that, assuming, the computer can calculate the mutual information of 10,000 (S_i, Π_i) pairs per second. When the maximum degree $k=10$, it already takes 184 days of uninterrupted processing to complete. When $k=25$, even the computer cannot solve it, because it requires a total of 728 years and 11 days of uninterrupted processing to complete the task. However, the ABN algorithm can find a relatively good (S_i, Π_i) pair no matter how much k value is; it only takes 250 times. This is very valuable in practical application.

The specific implementation of the ABN algorithm is shown in Algorithm 2.

In addition to solving the problem of the GreedyBayes's execution efficiency, the ABN algorithm also introduces an Exponential mechanism of differential privacy to disturb the Bayesian network construction process to solve privacy leakage caused by the Bayesian network.

It can be seen from the algorithm flow that the ABN algorithm removes the limitation of the maximum degree of the Bayesian network. And adopt the way of adaptively selecting the number of degree of each node in the network, which makes the network structure become more complex and diverse. The resulting network contains more information, and the synthetic data generated from the network is more likely to match the real data.

The main difference between the ABN algorithm and the GreedyBayes algorithm is that the former uses a greedy algorithm GParentSet with time complexity of $O(nm^2)$ to solve the optimal parent attribute set under each in-degree of the current attribute S_i. Compared with the ABN algorithm, the GreedyBayes algorithm traverses the values of all parent attribute sets through brute force enumeration every time it searches for the optimal parent attribute set. Actually, in this process, a lot of repeated and useless calculations are performed. Therefore, the ABN algorithm adopts the memorization, and its characteristics are suitable for the optimal substructure. On the premise of ensuring the best solution, it can greatly reduce the repetitive and useless calculation process and improve the construction speed of the Bayesian networks.

The specific implementation of the GParentSet algorithm is shown in Algorithm 3.

In this algorithm, $dp[S, i]$ is a set of attributes, which indicates that the current attribute S has selected i attributes as the best choice of parent attributes. In fact, when the GParentSet algorithm is executed in the new round, $dp[S, i]$ has recorded the optimal parent set selection in this state in the previous round. Therefore, for this round, we only need to care about the selection of the newly added parent attribute fa in the previous round.

In the ABN algorithm, in Step 5, each candidate attribute S_i and its optimal parent attribute set $\{\Pi_{i,1},\cdots,\Pi_{i,i-1}\}$ under various in-degree values are all added to the set Ω. In Step 6, it is selected through the differential privacy Exponential mechanism. This process does not limit the maximum in-degree of the Bayesian network. Among all the optional degrees, the degree with the greater amount of

```
Input: data set $D$, attribute set $S$, maximum in-degree $k$
Output: synthetic data set $\tilde{D}$
1: Initialize $\mathcal{N} = \varnothing$, $V = \varnothing$;
2: Randomly select an attribute from the attribute set $S$ as $S_1$, add $(S_1, \varnothing)$ to$\mathcal{N}$, and add $S_1$ to the vertex set $V$;
3: for $i = 2$ to $m$ do
4:        Initialize collection $\Omega = \varnothing$;
5:        For each $S_i \in S/V$ and $\Pi_i \in \binom{V}{k}$, add $(S_i, \Pi_i)$ to $\Omega$;
6:        Select $(S_i, \Pi_i)$ with the largest mutual information from $\Omega$ and join $\mathcal{N}$, add $S_i$ to $V$;
7: end for
8: return $\mathcal{N}$.
```

Algorithm 1: GreedyBayes Algorithm.

Table 2: Eumeration times of Greedybayes algorithm and ABN algorithm.

k	C_{m+1}^{k+2}	m^2
5	$\approx 1.16 \times 10^8$	=250
10	$\approx 1.59 \times 10^{11}$	=250
15	$\approx 1.48 \times 10^{13}$	=250
20	$\approx 1.56 \times 10^{14}$	=250
25	$\approx 2.30 \times 10^{14}$	=250

conditional entropy is easier to be selected. We do not need to control the structure of the resulting Bayesian network. Its construction is an adaptive process toward greater conditional entropy. This method is more flexible and reasonable than the traditional method of constructing Bayesian networks that requires a fixed network's maximum in-degree. Considering that the maximum in-degree of the Bayesian network constructed by this method is not fixed, it will adjust adaptively according to different datasets. Moreover, it will adjust in the direction of making the conditional entropy of the entire Bayesian network larger.

The ABN algorithm uses the differential privacy Exponential mechanism in Step 6. Its idea is to select the (S, Π) pairs currently added to the Bayesian network based on probability. We use conditional entropy I as the scoring function of the Exponential mechanism $u(S, \Pi) = I(S, \Pi)$. The greater the conditional entropy of the (S, Π) air, the higher the scoring function value and the greater the probability of being selected.

Zhang et al. [10] proved in the article that the global sensitivity of conditional entropy on binary data is

$$\Delta I = \frac{1}{n}\mathrm{lb} n + \frac{n-1}{n}\mathrm{lb}\frac{n}{n-1}. \tag{10}$$

Although they further gave a scoring function F with lower global sensitivity in the article, its global sensitivity is $\Delta F = 1/n$. However, the time complexity required to calculate the scoring function is $O(n2^m)$, which can only be applied when the attribute dimension m is small. Therefore, we do not adopt this method and still uses conditional entropy as the scoring function.

Step 6 will be executed $m-1$ times, each time the Exponential mechanism is used to select a (S, Π) pair from Ω to join the Bayesian network. From Property 1, we know that each choice needs to consume a part of the privacy budget. Here, we use the average division method to allocate the privacy budget, that is, ε_1 is divided equally into $m-1$ shares. Then, combined with the Exponential mechanism, we can obtain the expression of $\mathrm{Pc}(S, \Pi)$ as

$$\mathrm{Pc}(S, \Pi) = \frac{\exp\left(\varepsilon_1 u(S, \Pi)/(2(m-1)\cdot \Delta u)\right)}{\sum_{(S_i, \Pi_i)\in\Omega} \exp\left(\varepsilon_1 u(S_i, \Pi_i)/(2(m-1)\cdot \Delta u)\right)}. \tag{11}$$

Finally, proving that the ABN algorithm satisfies ε_1-differential privacy. The final output of the algorithm is a Bayesian network $\mathcal{N}$. In Step 6, the Exponential mechanism is used to select the currently added attribute node and its parent attribute set node. This operation disturbs the construction of the Bayesian network. According to Theorem 2, this step satisfies ε_1-differential privacy. Moreover, the entire ABN algorithm satisfies ε_1-differential privacy because no other operations involve the use of the original dataset D.

5.2. Differential Privacy Synthetic Data Release. The Bayesian network can simplify the calculation of the joint distribution between attributes to a large extent, and the better the Bayesian network is, the closer the joint distribution is to the true value. However, if the Bayesian network is directly used to calculate the joint distribution between attributes, it may still cause privacy leakage. Therefore, we need to perturb the calculated joint distribution further to achieve the purpose of protecting privacy.

Zhang et al. [9] used the NoisyConditionals algorithm to realize the secure calculation of Bayesian networks. This algorithm adds Laplace noise into the joint distribution $\Pr[S, \Pi]$ to obtain the joint distribution $\Pr^*[S, \Pi]$ with noise. Although this algorithm can ensure that the obtained joint distribution meets the differential privacy protection, its joint distribution may become very sparse, and the original probability value will generally be small when the

```
Input: data set $D$, attribute set $S$, privacy budget $\varepsilon_1$
Output: Bayesian network $\mathcal{N}$
1: Initialize $\mathcal{N} = \varnothing$, $V = \varnothing$;
2: Randomly select an attribute from the attribute set $S$ as $S_1$, add $(S_1, \varnothing)$ to $\mathcal{N}$, and $S_1$ to $V$, $fa = S_1$;
3: for $i = 2$ to $m$ do
4:     Initialize collection $\Omega = \varnothing$;
5:     For each, find $\Pi_i = \{\Pi_{i,1},\cdots,\Pi_{i,i-1}\} \longleftarrow$ GParentSet$(S_i, fa, i)$ and then add $(S_i, \Pi_{i,1}), \cdots, (S_i, \Pi_{i,i-1})$ to $\Omega$;
6:     Use the Exponential mechanism with privacy budget $\varepsilon_1/(m-1)$ to select a $(S_i, \Pi_{i,j})$ from $\Omega$ to add to $\mathcal{N}$ with probability
       Pc$(S, \Pi)$, and add $S_i$ to $V$, $fa = S_i$;
7: end for
8: return $\mathcal{N}$.
```

Algorithm 2: ABN Algorithm.

```
Input: current attribute $S$, new parent attribute $fa$, maximum in-degree $k$
Output: optimal parent attribute set $\Pi = \{\Pi_1,\cdots,\Pi_k\}$
1: Initialize $\Pi = \varnothing$;
2: for $i = 1$ to $k$ do
3:     $\Pi' = dp[S, i-1] \cup fa$;
4:     $\Pi'' = dp[S, i]$;
5:     $dp[S, i] = \begin{cases} \Pi' & I(S,\Pi') \geq I(S,\Pi'') \\ \Pi'' & I(S,\Pi') < I(S,\Pi'') \end{cases}$;
6:     Add $dp[S, i]$ to $\Pi$;
7: end for
8: return $\Pi$.
```

Algorithm 3: GParentSet Algorithm.

attribute dimension increases. At this time, if Laplace noise is directly added into these smaller probability values, then it may cause the problem that the noise completely covers the true value, and seriously affecting the accuracy of the release.

Therefore, we do not directly use a joint distribution like the NoisyConditionals algorithm. Instead, we use conditional distribution to generate synthetic data because of the characteristics of binary data with only 0 and 1. The main reason is that when the distribution is a conditional distribution, we have the following equation:

$$\Pr\,[S_i = 0 \mid \Pi_i] + \Pr\,[S_i = 1 \mid \Pi_i] = 1. \tag{12}$$

According to the equation, we can infer that at least one relatively large conditional probability value exists in the conditional distributions $\Pr\,[S_i = 0 \mid \Pi_i]$ and $\Pr\,[S_i = 1 \mid \Pi_i]$. In this way, if noise is directly added into the conditional probability, then at least one larger conditional probability will not be covered by the noise. Even if the conditional probability is completely covered by noise, it has minimal effect on the accuracy of the final release. The reason is that if such a conditional probability exists, then its probability value must be very small or negligible relative to another probability value. Therefore, even after normalization, they can still reflect the original distribution law.

Nevertheless, synthetic data can still be generated. The traditional method is to generate it directly through joint distribution, but it still has the same problems as data sparseness. So this study uses conditional distribution to generate synthetic data. To this end, we design a synthetic data generation algorithm SDG. The main idea of the SDG algorithm is to use the conditional distribution of the node and its parent node to generate synthetic data according to the topological order of the Bayesian network.

The specific implementation of the SDG algorithm is shown in Algorithm 4.

The synthetic data $\tilde{D}$ generated by the SDG algorithm can make the attacker unable to infer a specific record in the original dataset D; thus, it protects the personal privacy information in the data from being leaked.

Finally, proving that the SDG algorithm satisfies ε_2-differential privacy. The algorithm adds Laplace noise Lap$(2m/(n \cdot \varepsilon_2))$ to each conditional distribution $\Pr\,[S_i \mid \Pi_i]$ in Step 5 and obtains the conditional distribution $\Pr^*[S_i \mid \Pi_i]$ with noise. Then, these conditional distributions are used to generate synthetic data for release. According to Theorem 2, this process satisfies ε_2-differential privacy. Moreover, because no other subsequent steps involve the use of the original data set D, the entire SDG algorithm satisfies ε_2-differential privacy.

5.3. Differential Privacy Adaptive Bayesian Network Algorithm. The PrivABN algorithm can be divided into two independent steps:

(1) Through the ABN algorithm, construct a Bayesian network $\mathcal{N}$ which satisfies differential privacy for

```
Input: data set $D$, Bayesian network $\mathcal{N}$, privacy budget $\varepsilon_2$
Output: synthetic data set $\tilde{D}$
1: Initialize $P^* = \varnothing$;
2: for $i = 1$ to $m$ do
3:     Calculate the joint distribution Pr $[S_i, \Pi_i]$ and the marginal distribution Pr $[\Pi_i]$;
4:     Calculate conditional distribution Pr $[S_i \mid \Pi_i] =$ Pr $[S_i, \Pi_i]/$Pr $[\Pi_i]$;
5:     Add Laplace noise Lap$(2m/n \cdot \varepsilon_2)$ to Pr $[S_i \mid \Pi_i]$ to get Pr$^*[S_i \mid \Pi_i]$;
6:     Reset the negative value in Pr$^*[S_i \mid \Pi_i]$ to 0, normalize other values, and then add$P^*$;
7: end for
8: Traverse the node $S_i$ according to the topological order of the Bayesian network $\mathcal{N}$;
9: Obtain the noisy conditional distribution Pr$^*[S_i \mid \Pi_i]$ from $P^*$, and update the value of each record attribute $S_i$ in $\tilde{D}$ according to the conditional distribution;
10: return $\tilde{D}$.
```

Algorithm 4: SDG Algorithm.

the original dataset D, and extract the noise conditional distribution P^* from the Bayesian network

(2) Through the SDG algorithm, generate a synthetic dataset $\tilde{D}$ for release, according to the topological order of the Bayesian network $\mathcal{N}$ and the conditional distribution P^* with noise

The specific implementation process of the PrivABN algorithm is shown in Algorithm 5.

In the PrivABN algorithm, the subalgorithms ABN and SDG involve the use of the original dataset D. According to property 1, in order for the PrivABN algorithm to satisfy ε-differential privacy, privacy budgets must be allocated for these two subalgorithms first. According to the analysis in the previous chapters of this paper, the privacy budget ε_1 and ε_2 will, respectively, correspond to the two noise distributions exp $(\varepsilon_1 u(S, \Pi)/(2(m-1)\cdot \Delta u))$ and Lap$(2m/(n\cdot\varepsilon_2))$. The first noise distribution aims to make the value of $\varepsilon_1/(2(m-1)\cdot\Delta u)$ (with the nondependent $u(S,\Pi)$) as large as possible. The second noise distribution aims to make the value of $2m/(n\cdot\varepsilon_2)$ as small as possible. By simplifying the two formulas above, we can obtain

$$\begin{aligned}
&\max\left(\frac{\varepsilon_1}{2(m-1)\cdot\Delta u}\right)\\
&\quad=\max\left(\varepsilon_1\cdot\frac{1}{2(m-1)\cdot((1/n)\mathrm{lb}n+(n-1/n)\mathrm{lb}(n/n-1))}\right)\\
&\quad\approx\max\left(\varepsilon_1\cdot\frac{n}{2(m-1)}\right),
\end{aligned}$$

$$\min\left(\frac{2m}{n\cdot\varepsilon_2}\right)=\min\left(\frac{1}{\varepsilon_2}\cdot\frac{2m}{n}\right)=\max\left(\varepsilon_2\cdot\frac{n}{2m}\right). \tag{13}$$

From the simplified results, the proportions of the two privacy budgets that need to be allocated are roughly the same. Therefore, we adopt an even distribution strategy to allocate the privacy budget ε.

Finally, proving that the PrivABN algorithm satisfies ε-differential privacy. It can be seen that the ABN algorithm and SDG algorithm satisfy ε_1-differential privacy and ε_2-differential privacy. Apart from the above two algorithms, the PrivABN algorithm has no other place that involves the use of the original data set D. Therefore, according to property 1, the PrivABN algorithm satisfies $(\varepsilon_1+\varepsilon_2)$-differential privacy.

6. Experiences

6.1. Experiences Environment. The experimental platform is a 4-core Intel i5-6300HQ CPU (2.3GHz), 8GB memory, Windows 10 operating system, and the compilation environment is Dev-C++5.11. Our experiments use the C++ programming language to implement all the methods, among which the implementation of the Bayesian network refers to the relevant code of the paper experiment by Zhang et al. [10].

6.2. Datasets. Our experiments use three real-world datasets, NLTCS, ACS, and Retail. NLTCS [18] is an American long-term care survey record that includes the daily life and medical conditions of 21,574 elderly disabled persons. ACS [19] is the global census data released by IPUMS-USA, which records 47,461 pieces of personal information. Retail [20] is 88,162 shopping records in the US retail market. Each record contains items purchased by it, a total of 16,469 categories of goods, from which we have retained the top 50 best-selling goods. The specific information of these three data sets is shown in Table 3.

6.3. Evaluation. For each set of experiments, we will compare the $L1$ error (mean error) between the generated synthetic dataset $\tilde{D}$ and the original dataset D. Moreover, the $L2$ error is caused by the same number of α-way queries on these two datasets.

Definition 4 ($L1$ error). The $L1$ error between the original dataset D and the synthetic dataset $\tilde{D}$ is

$$L_1(D,\tilde{D})=\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left|D_i^{(j)}-\tilde{D}_i^{(j)}\right|}{N}, \tag{14}$$

```
Input: dataset $D$, attribute set $S$, privacy budget $\varepsilon$
Output: synthetic data set $\tilde{D}$
1: $\varepsilon_1 = \varepsilon/2, \varepsilon_2 = \varepsilon/2$;
2: $\mathcal{N} \longleftarrow \mathrm{ABN}(D, S, \varepsilon_1)$;
3: $\tilde{D} \longleftarrow \mathrm{SDG}(D, \mathcal{N}, \varepsilon_2)$;
4: return $\tilde{D}$.
```

Algorithm 5: PrivABN Algorithm.

Table 3: Description of datasets.

Dataset	Type	Cardinality	Dimensionality	Domain size
NLTCS	Binary	21 574	16	2^{16}
ACS	Binary	47 461	23	2^{23}
Retail	Binary	88 162	50	2^{50}

where $D_i^{(j)}$ is the value of the jth row of the ith experiment using the original dataset and $\tilde{D}_i^{(j)}$ is the value of the jth row of the ith experiment using the synthetic dataset. N is the number of experiments, and M is the number of rows of the dataset.

Definition 5 ($L2$ error). Let the α-way edge table generate by the original dataset D through the ith query is $T_i(D)$, and the α-way edge table generate by the synthetic dataset $\tilde{D}$ is $T_i(\tilde{D})$; the $L2$ error between them is

$$L_2\left(D, \tilde{D}\right) = \frac{\sum_{i=1}^{N} \sqrt{\sum_{j=1}^{M} \left(T_i^{(j)}(D) - T_i^{(j)}\left(\tilde{D}\right)\right)^2}}{N}, \quad (15)$$

where $T_i^{(j)}$ is the value of the jth row of the α-way edge table generated by the ith query, N is the number of queries, and M is the number of rows of the edge table.

6.4. Result Analysis. The first part of the experiments is to analyze the availability of the PrivABN method. To know the size of the noise generated by the PrivABN method in a noise-free environment, we set up the NoPrivABN method without differential privacy protection for comparison. We will conduct 100 random repeated experiments on NLTCS and ACS. We set the privacy budget ε to 0.001, 0.005, 0.01, 0.05, 0.1, and 1.0. The experimental results will be verified by 200 random α-way queries, the values of which are 3, 7, and 12. We will test their performance on $L1$ error and $L2$ error. The experimental results are shown in Figure 1.

Figure 1 shows that the PrivABN algorithm only needs a very small privacy budget. When $\varepsilon = 0.05$, it is very close to the effect of NoPrivABN, which shows that the PrivABN algorithm has high availability.

Moreover, as the privacy budget ε increases, the error generated by PrivABN gradually approaches the error of NoPrivABN, which is in line with the differential privacy law. This finding further verified the credibility of the experiment.

Another finding is that the PrivABN algorithm without differential privacy protection will produce certain errors. The reason is that the algorithm itself generates a synthetic dataset based on the Bayesian network and conditional distribution. The process of constructing the Bayesian network itself may produce certain errors, and the process of generating synthetic data through conditional distribution is probabilistic, thereby resulting in the production of certain errors. Therefore, taking the error of NoPrivABN as the lower bound is in line with the experimental standard.

Finally, the performance of PrivABN on different α-way queries under the same dataset is observed. Their overall trend of change and their turning points are also the same. Their error sizes under different privacy budgets do not differ greatly, and they are basically the same within a certain error range. Therefore, we can consider that the stability of PrivABN in the face of different conditional parameters is a manifestation of the PrivABN's remarkable robustness.

According to the $L1$ errors and $L2$ errors between the generated synthetic dataset and the original dataset, PrivABN only needs a very small privacy budget to achieve the effect of NoPrivABN. To know this threshold more accurately, we further subdivide the value of the privacy budget. We set the privacy budget to increase from 0.05 to 0.5 in increments of 0.025 and from 0.5 to 1.0 in increments of 0.1. Then, experiments with 3-way query under the NLTCS and ACS datasets, respectively (from the previous experimental conclusions, it can be known that PrivABN has better robustness, and its error under different α-way queries is not much different, so only one query needs to be compared). The experimental results are shown in Figure 2.

Figure 2 shows that when the privacy budget is 0.4 (NLTCS dataset) and 0.225 (ACS dataset), the $L2$ error is lower than the error bar of 0.01. This finding shows that the PrivABN method only needs to consume a very small privacy budget to achieve good privacy protection. Therefore, we can definitely believe that privabn has high availability.

From another point of view, the PrivABN algorithm only needs to give a small amount of privacy budget. It can achieve a good privacy protection effect and can greatly reduce the error of differential privacy protection. This is the reason for its high availability.

In the second part of the experiments, the performance of PrivABN on the real high-dimensional dataset Retail is analyzed. To reflect the pros and cons of the results better, the experiments will be compared with these three methods, namely, PriView [8], PrivBayes [9], and Jtree [12]. We will conduct 100 repeated random experiments on the Retail dataset and set the privacy budget ε to 0.1 and 1.0. The experimental results will be verified by 200 random α-way queries, where α values correspond to 4, 6, and 8. We will test their performance on $L2$ errors. The experimental results are shown in Figure 3.

Figure 3 shows that under different privacy budgets, the PrivABN method performs significantly better than the three other methods on the Retail dataset. Further, when the privacy budget is small ($\varepsilon = 0.1$), the PrivABN method performs

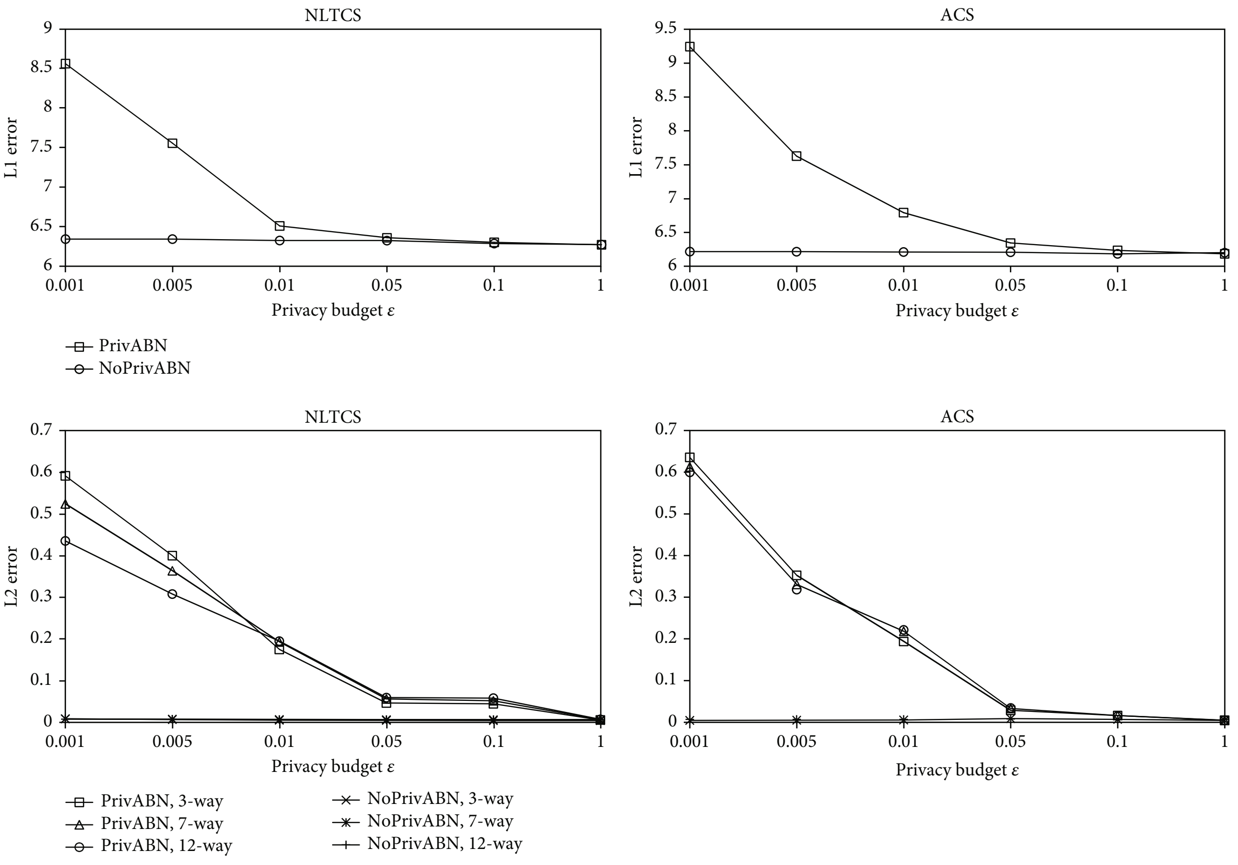


Figure 1: Error analysis with or without privacy protection—1.

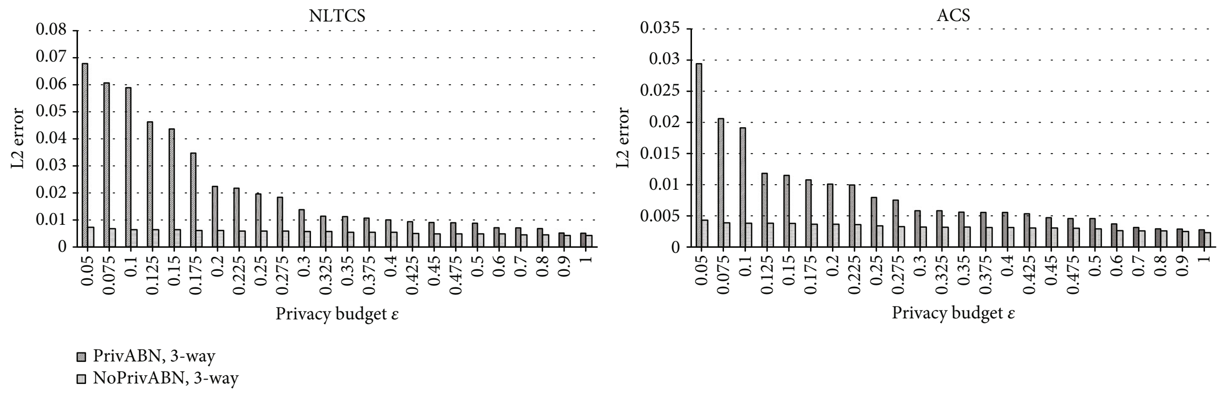


Figure 2: Error analysis with or without privacy protection—2.

significantly better than the other three methods. When the privacy budget is large ($\varepsilon = 1.0$), PrivABN is still superior to the other three methods, but it is not different from the JTree method. This is in line with the law of differential privacy, because as the privacy budget continues to increase, the degree of privacy protection of the algorithm will continue to decline, until the implementation result is consistent with that of the algorithm without differential privacy protection. Therefore, it can be seen that the accuracy of the probabilistic graph model itself built by the Privabn method is better than that of the model built by the JTree method. Compared with the model built by the PrivBayes method, the accuracy is much better. This further verifies that the improvement strategies proposed in this paper are effective and have achieved good results.

Moreover, with the increase in α-way query dimension, the variation range of $L2$ error of the PrivABN method is significantly smaller than those of the three other methods. Therefore, we can further infer that the PrivABN method has higher availability and better robustness.

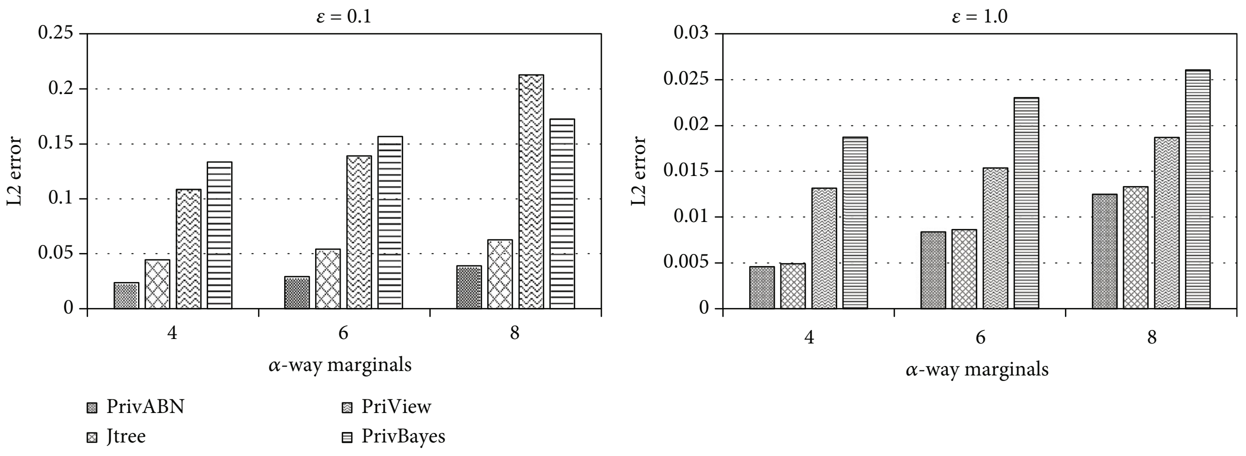


FIGURE 3: Error analysis of α-way query in different methods on Retail dataset.

7. Conclusion

Private releasing of high-dimensional data has been a research hotspot and a challenge in the field of differential privacy. This study proposes an efficient and low-noise differential privacy publishing method called PrivABN for high-dimensional binary data. The method uses the ABN algorithm to construct the Bayesian network over the dataset quickly and adaptively while using the differential privacy Exponential mechanism to protect the privacy of the Bayesian network during the construction. Subsequently, the SDG algorithm uses the differential privacy Laplace mechanism to initially extract the noisy conditional distribution from the Bayesian network and then uses these conditional distributions and the Bayesian network topology to generate synthetic data for release. By performing experiments on three real data sets, we demonstrate that PrivABN deserves higher usability and robustness than existing methods.

The main focus for our future work will be continually on the differential privacy publication of high-dimensional data. We will investigate the differential privacy publication of high-dimensional nonbinary data and explore the issue of differential privacy publication in a streaming high-dimensional data environment.

Data Availability

NLTCS is an American long-term care survey record that includes the daily life and medical conditions of 21,574 elderly disabled persons. ACS is the global census data released by IPUMS-USA, which records 47,461 pieces of personal information. Retail is 88,162 shopping records in the US retail market.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[2] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang, "(α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 754–759, Philadelphia, USA, 2006.

[3] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd International Conference on Data Engineering*, pp. 106–115, Istanbul, Turkey, 2007.

[4] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "*L*-diversity: privacy beyond *k*-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[5] C. Dwork, "Differential privacy," 2006.

[6] K. Muralidhar and R. Sarathy, "Security of random data perturbation methods," *ACM Transactions on Database Systems*, vol. 24, no. 4, pp. 487–493, 1999.

[7] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, Florida, USA, 2003.

[8] W. Qardaji, W. Yang, and N. Li, "Priview: practical differentially private release of marginal contingency tables," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1435–1446, UT, USA, 2014.

[9] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "*PrivBayes: private data release via bayesian networks*," 2014.

[10] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: private data release via bayesian networks," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 4, pp. 1–41, 2017.

[11] W. Liang, W. Weiping, and M. Dan, "Privacy preserving data publishing via weighted bayesian networks," *Journal of Computer Research and Development*, vol. 53, no. 10, article 2343, 2016.

[12] R. Chen, Q. Xiao, Y. Zhang, and J. Xu, "Differentially private high-dimensional data publication via sampling-based inference," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 129–138, Sydney, NSW, Australia, 2015.

[13] Z. Xiaojian, C. Li, J. Kaizhong, and M. Xiaofeng, "Private high-dimensional data publication with junction tree," *Journal of Computer Research and Development*, vol. 55, no. 12, article 2794, 2018.

[14] F. Wei, W. Zhang, Y. Chen, and J. Zhao, "Differentially private high-dimensional data publication via markov network," in *International Conference on Security and Privacy in Communication Systems*, pp. 133–148, Singapore, 2018.

[15] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*, pp. 265–284, New York, NY, USA, 2006.

[16] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *2012 IEEE 28th International Conference on Data Engineering*, pp. 20–31, New York, USA, 2012.

[17] F. D. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30, Providence, Rhode Island, USA, 2009.

[18] K. G. Manton, *National Long-Term Care Survey: 1982, 1984, 1989, 1994, 1999, and 2004*, Inter-university Consortium for Political and Social Research, 2010.

[19] S. Ruggles, K. Genadek, R. Goeken, J. Grover, and M. Sobek, *Integrated public use microdata series: version 6.0 [dataset]*, vol. 23, Minneapolis: University of Minnesota, 2015.

[20] B. Tomhttp://fimi.uantwerpen.be/data/.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5518927, 9 pages
https://doi.org/10.1155/2021/5518927

Research Article

UAV Task Allocation Based on Clone Selection Algorithm

Xiaopan Zhang[1] and Xingjun Chen[2]

[1]*Resources and Environmental Engineering School of Wuhan University of Technology, Wuhan 430070, China*
[2]*Operational Software and Simulation Institution of Dalian Naval Academy, Dalian 116018, China*

Correspondence should be addressed to Xingjun Chen; cxj_dna@yeah.net

Received 26 February 2021; Accepted 20 June 2021; Published 5 July 2021

Academic Editor: Yong Zhang

With the continuous development of computer and network technology, the large-scale and clustered operations of drones have gradually become a reality. How to realize the reasonable allocation of UAV cluster combat tasks and realize the intelligent optimization control of UAV cluster is one of the most challenging difficulties in UAV cluster combat. Solving the task allocation problem and finding the optimal solution have been proven to be an NP-hard problem. This paper proposes a CSA-based approach to simultaneously optimize four objectives in multi-UAV task allocation, i.e., maximizing the number of successfully allocated tasks, maximizing the benefits of executing tasks, minimizing resource costs, and minimizing time costs. Experimental results show that, compared with the genetic algorithm, the proposed method has better performance on solving the UAV task allocation problem with multiple objectives.

1. Introduction

With the rapid development of Internet of Things and 5G communication technologies, UAV systems are increasingly used in the military field and UAV operations have become an important part of modern military operations. Single-UAV combat often lacks support, guarantee, and target cover, and it usually requires an overall force to complete combat missions. Task allocation is one of the most important problems that need to be solved in multi-UAV operations, and it directly affects the efficiency and profitability of operations. Finding the optimal solution of the task allocation problem has been proven to be NP-hard, and the solving difficulty increases exponentially with the scale of UAV cluster and tasks. Furthermore, task allocation is often a multiobjective optimization problem, which makes the model more complicated, e.g., simultaneously maximizing the number of tasks to be performed, maximizing the benefits of performing tasks, minimizing time costs, and minimizing resource consumption.

Many methods for solving the task allocation problem have been proposed, which can be roughly divided into four categories: graph theory [1], integer linear programming [2], state space search [3], and Artificial Intelligence (AI) methods such as genetic algorithm, particle swarm algorithm, simulated annealing, and ant colony algorithm [4–8]. Most methods in the first three categories are complete search algorithms. These algorithms can get the optimal solution, but they require a lot of computing resources and time cost, and it is impractical to apply them to a large-scale problem. AI methods cannot guarantee an optimal solution, but they usually can obtain a local-optimal solution [9] within a reasonable period of time. The above algorithms often optimize a certain goal, such as task revenue [10] or time/resource cost [11].

Artificial immune system (AIS) is an emerging research direction of computational intelligence. A clone selection algorithm (CSA) is proposed based on related immune principles [12–14]. This algorithm is widely used in function optimization [15] (e.g., multimodal optimization and continuous function optimization), pattern recognition [16, 17] (e.g., binary character and face recognition), and scheduling problems [18]. Compared with those complete search algorithms, CSA has some advantages and is convenient for practicality and engineering. At the same time, CSA can be used to solve multiobjective problems. Comparing CSA with the GA [19–21], the main difference is the way the population evolves. In the GA, the population evolves through crossover and mutation, and in the CSA, cell reproduction is asexual, with each offspring produced by one cell being an exact copy of its parents, and mutation and selection are made through these offspring. This paper proposes to use CSA to optimize four objectives in UAV task allocation, i.e., maximizing the number of successfully assigned tasks, maximizing the benefits of executing tasks, minimizing resource cost, and minimizing time cost, and comprehensively considers the time constraints, resource constraints, and functional constraints in real-world scenarios. Comparing with the brute-force search algorithm and genetic algorithm, experimental results show that the proposed method could achieve better performance on solving high-dimensional multiobjective task allocation problems.

The remaining sections of this paper are organized as follows: Section 2 introduces the description of task allocation problems; Section 3 present the details of the proposed method; Section 4 presents the experimental results and our analysis; the proposed work is summarized in Section 5.

2. Problem Description

Task allocation involves many objects and related attributes. This section will give a formal representation of the elements, goals, and constraints involved in task allocation [22, 23] to facilitate later expression.

2.1. Task Allocation

(1) UAV: as the code of military operations, UAV is expressed as $U = \{u_1, u_2, \cdots, u_m\}$, where u_i represents UAV $i\,(i = 1, \cdots, m)$ and m is the number of UAVs. The initial position of the UAV is expressed as $\mathrm{Pu} = \{\mathrm{pu}_1, \mathrm{pu}_2, \cdots, \mathrm{pu}_m\}$, where pu_i represents the initial position of the UAV i. The ammunition and fuel carried by each UAV are expressed as $\mathrm{ResU} = \{\mathrm{resu}_1, \cdots, \mathrm{resu}_m\}$, where $\mathrm{res}u_i$ represents the number of resources carried by the UAV i.

(2) Task: as a combat task and an indivisible unit, the task is expressed as $T = \{t_1, t_2, \cdots, t_n\}$, where t_j represents the task $j\,(j = 1, \cdots, n)$ and n is the number of tasks. The initial position of the task is expressed as $\mathrm{Pt} = \{\mathrm{pt}_1, \mathrm{pt}_2, \cdots, \mathrm{pt}_n\}$, where pt_j represents the initial position of task j. The execution of each task requires certain resources to be consumed, which is expressed as $\mathrm{ResT} = \{\mathrm{rest}_1, \cdots, \mathrm{rest}_n\}$, where rest_j represents the resources consumed by task j. Performing each task will obtain a different task revenue expressed as $\mathrm{Reward} = \{\mathrm{reward}_1, \cdots, \mathrm{reward}_n\}$, where reward_j represents the revenue of executing task j. The validity period of each task is limited by time. Each task has the earliest start execution time early_j and the latest start execution time late_j; the executable range of each task is expressed as $\mathrm{TR} = \{[\mathrm{early}_1, \mathrm{late}_1], \cdots, [\mathrm{early}_n, \mathrm{late}_n]\}$, where $[\mathrm{early}_j, \mathrm{late}_j]$ is the executable range of task j. It tasks different time to execute different tasks; the time consumed to execute the task is expressed as $\mathrm{Time} = \{\mathrm{time}_i, \cdots, \mathrm{time}_n\}$, where time_j represents the time consumed to execute task j.

(3) Execution sequence: a task is executed by only one drone. The execution sequence of the UAV is represented as $\mathrm{Su} = \{\mathrm{su}_1, \cdots, \mathrm{su}_m\}$. Among them, su_i represents the execution sequence of UAV i, where su_i is composed of corresponding tasks, specifically represented as $\mathrm{su}_i = \{t_x t_y t_q t_z t_d\}$, where $0 < x, y, q, z, d \le n$. $|\mathrm{su}_i|$ is the number of tasks executed by the UAV i, and they are, respectively, task x, task y, task q, task z, and task d. su_{ij} is the jth task in the execution sequence of the UAV i. According to the position of the task in the execution sequence, the corresponding task can be expressed as $t_x = \mathrm{su}_{i1}$, $t_y = \mathrm{su}_{i2}$, $t_q = \mathrm{su}_{i3}$, $t_z = \mathrm{su}_{i4}$, and $t_d = \mathrm{su}_{i5}$.

(4) Task allocation: UAV and task are the two subjects of allocation. Since a task can only be performed by one UAV, the allocation relationship can be expressed as $A = \{a_1, \cdots, a_n\}$, where a_j represents the assignment relationship of the task j. If task j is not assigned to a drone, then $a_j = \mathrm{NULL}$, and if assigned, it means the inequality $a_j \neq NULL$. At the same time, the assignment to the relevant UAV can be obtained one step further, expressed as $a_j = u_i$; it means that task j is assigned to UAV i for execution.

2.2. Optimization Objective.

With the basic description of the above basic elements, we further derive the definition of objectives to be optimized in UAV task allocation problems.

(1) Maximize the number of successfully assigned tasks:

$$\mathrm{target}_1 = \max_{\forall A} \sum_{j=1}^{n} \{a_j \neq \mathrm{NULL}\}, \tag{1}$$

where $a_j \neq \mathrm{NULL}$ is true; the return value is 1; otherwise, the value is 0.

(2) Maximize the benefits of performing tasks:

$$\mathrm{target}_2 = \max_{\forall A} \sum_{j=1}^{n} \{a_j \neq \mathrm{NULL}\} \times \mathrm{reward}_j. \tag{2}$$

(3) Minimize resource cost:

$$\mathrm{target}_3 = \min_{\forall Su} \sum_{i=1}^{m} (\mathrm{arrive\_cost}(\mathrm{pu}_i, \mathrm{su}_{i1})) + \sum_{j=2}^{|\mathrm{su}_i|} \left(\mathrm{arrive\_cost}\left(\mathrm{su}_{i(j-1)}, \mathrm{su}_{ij}\right)\right) + \sum_{j=1}^{|\mathrm{su}_i|} \mathrm{res}_{\mathrm{su}_{ij}}, \tag{3}$$

where $\text{arrive\_cost}(\text{pu}_i, \text{su}_{i1})$ represents the resource consumption cost required to execute the first task in the task sequence from agent i to agent.

(4) Minimize time consumption:

$$target_2 = \min_{\forall Su} \max \sum_{i=1}^{m} \text{time\_cost}(\text{su}_i, |\text{su}_i|), \tag{4}$$

where $\text{time\_cost}(\text{su}_i, |\text{su}_i|)$ represents the time; it tasks for the UAV r_i to execute the tasks in the s_i sequence.

2.3. Constraints. In real-world scenarios, there are often some constraints in task allocation problems. In this paper, we mainly consider the following common constraints [16]:

(1) Time constraint: a UAV can only successfully execute the task when it starts within the executable time range of the task. For the execution sequence Su_i of the UAV u_i, the constraint is as follows:

For $j = 1$,

$$\text{time\_constraint}_{i1}^{T} = \text{arrive}_{\text{time}(\text{pu}_i,1)} = \text{travel}_{\text{time}(\text{pu}_i,\text{su}_{i1})} \le \text{late}_{\text{su}_{i1}}. \tag{5}$$

For $j = 2, \cdots, \mid \text{Su}_i \mid$,

$$\begin{aligned}\text{time\_constraint}_{ij}^{T} &= \text{arrive}_{\text{time}(u_i,j)} \\ &= \text{time}\left(\text{su}_{ij}\right) + \text{travel}_{\text{time}\left(\text{pu}_i,\text{su}_{ij}\right)} \le \text{late}_{\text{su}_{i1}}.\end{aligned} \tag{6}$$

(2) Resource constraints: UAV tasks are limited by its own resources. For the execution sequence Su_i of UAV u_i, the time constraints are as follows:

For $j = 1, \cdots, \mid \text{Su}_i \mid$,

$$\begin{aligned}\text{assets\_constraint}_{ij}^{R} &= \text{resource}_{\text{cost}(\text{pu}_i,j)} \\ &= \text{travel}_{\text{csot}(\text{pu}_i,\text{su}_{i1})} + \sum_{k=2}^{j} \text{rest}_{\text{su}_{ik}} \le \text{resu}_i.\end{aligned} \tag{7}$$

(3) Functional constraints: in real scenarios, different types of UAVs have different functions and therefore perform different tasks. This constraint is explained from the perspective of drones and tasks:

(i) From the perspective of agents, for $i = 1, \cdots, m$,

$$\text{function\_constraint}_{i}^{U} = \left\{t_x t_y \cdots t_z t_d\right\}, \tag{8}$$

The constraint indicates that agent i can only execute task x, task y, task z, and task d due to functional limitations, where $0 < x, y, q, z, d \le n$.

(ii) From the perspective of tasks, for $j = 1, \cdots, n$,

$$\text{function\_constraint}_{j}^{T} = \left\{u_e, \cdots, u_f\right\}. \tag{9}$$

This shows that the constraint indicates that task j can only be executed by UAV e, UAV f, etc. due to functional limitations, where $0 < e, f \le m$.

3. The Proposed Method

The CSA algorithm is a kind of the artificial immune system, which mainly contains ideas such as clone selection, receptor editing, and antibody circulation supplement mechanism and selects mature antibody cells through affinity, and uses a limited gene library to identify endlessly changing antigens. The CSA algorithm simulates immune mechanisms such as clone selection and amplification of antibodies, high-frequency mutations, and receptor editing during the immune response process of the immune system, so that it has strong self-learning, self-organization, and adaptive capabilities. Optimization-related fields are widely used. In this paper, the distribution plan of the UAV is mainly coded into antibody cells, and the final Pareto solution set is obtained through continuous iteration of antibodies [24, 25] (described in detail in Section 3.5).

3.1. Basic Framework. In this section, the basic program flow chart of the algorithm will be given, as shown in Figure 1.

It can be seen that the CSA model is relatively simple and convenient for coding operation. The main steps are described as follows:

(1) First, randomly initialize the UAV task allocation solution, organize the execution sequence of each UAV, and evaluate the affinity function of each antibody, and set the number of antibodies to$N$

(2) Judging the number of iterations, when it reaches a certain number (maxGen is the maximum number), the algorithm ends the output distribution plan

(3) Perform cloning operations on the current population, which involves the number of clones and the proportion of clones selected according to their affinity

(4) The cloned antibody is mutated according to the affinity ratio of the cloned individual

(5) Select$N$from the original population and the cloned population 2$N$for the next iteration

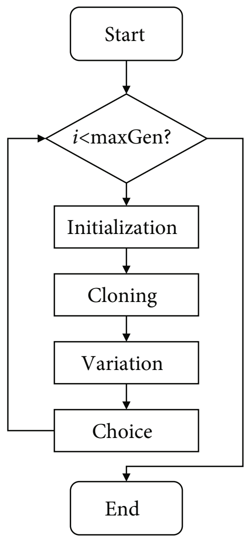


Figure 1: Flow chart of CSA [12].

(6) Repeat steps (2)–(5). This is a simple program description process of the algorithm, and each process will be described in detail below

3.2. Encoding of Antibody. An important part of using evolutionary algorithms is to encode real-world problems into antibodies, which are feasible solutions to allocation problems. For the CSA algorithm, the concept of antibody is very important, and the code of the antibody also determines the actual optimization effect of the algorithm. These solutions are composed of the execution sequence of each UAV. These components are called "genes," and their practical meaning is the actual distribution execution sequence of the UAV. Here is a specific description of the encoding rule [22]. For example, Table 1 shows an example of 3 drones and antibody codes composed of 10 tasks.

The following information can be obtained from the coding example in the figure above. From the functional constraints of the UAV, $\text{function\_constraint}_1^U = \{1, 7, 2, 4\}$, $\text{function\_constraint}_2^U = \{3, 6, 4, 5\}$, and $\text{function\_constrain t}_3^U = \{8, 9, 2,10,1\}$. From the perspective of the functional constraints of the task, $\text{function\_constraint}_2^T = \{1, 3\}$, and $\text{function\_constraint}_3^T = \{2\}$. This kind of coding is directly based on the functional constraints of the drone, reducing a lot of useless calculations and judgments, which is conducive to improving efficiency. At the same time, it can also get $\text{su}_1 \subseteq \{1, 7, 2, 4\}$, $\text{su}_2 \subseteq \{3, 6, 4, 5\}$, and $\text{su}_3 \subseteq \{8, 9, 2,10,1\}$. We know the actual execution sequence that the UAV finally allocates must be a subset of the UAV's functional constraints. At the same time, there are some shortcomings in this coding. As far as t_2 is concerned, due to its simultaneous existence in the functional constraints of u_1 and u_3 (in simple terms, there is a many-to-many relationship between the UAV and the task), it is difficult for the algorithm to choose who performs the more excellent, simple processing is carried out in the encoding here, and it is stated that the UAV with lower encoding will be executed first; that is to say, if u_1 is not limited in resources and time, t_2 can be executed first, and then t_2 in the function constraint of u_3 fails; if the u_1 constraint is not satisfied, u_3 starts to judge the conditions for executing t_2.

Table 1: Example of antibody coding.

UAV 1	UAV 2	UAV 3
1 7 2 4	3 6 4 5	8 9 2 10 1

3.3. Cloning, Mutation, and Selection. The original CSA algorithm selects cloned antibodies according to the degree of affinity. Most antibodies with higher affinity are cloned for mutation in order to produce better individuals; a small number of antibodies with poor affinity are cloned to prevent mutation. The algorithm enters the local optimum to improve the quality of the solution. Such a strategy is in line with the realistic model and has a certain optimization effect; however, because the problem is a multiobjective optimization, each solution may evolve into a Pareto solution. Therefore, under this problem model, we assume that each antibody in each original population will be cloned according to its affinity to change the following individuals.

The mutation operation is performed in the cloned individual. The main operation of mutation is to randomly change part of the gene in the antibody. The algorithm mutates according to the degree of affinity. It is assumed that individuals with higher affinity have higher quality solutions, individuals with higher clone affinity have a lower mutation rate, and individuals with lower affinity have a higher mutation rate. This is because the quality of individual solutions with high affinity has been further optimized. Less variation is to maintain the quality of its own solutions and to explore around the solution at the same time; while individuals with low affinity undergo a lot of variation to let it explore a larger solution space. This also means that individuals with less affinity change fewer gene segments, while individuals with greater affinity change more. The detailed change parameters will be given in the experimental part.

3.4. Affinity Function. The four objectives in Section 2 are normalized, and the corresponding constraints are added to the calculation of the affinity function. When evaluating each individual x, the following affinity function can be used:

(1) The value of the first objective can be calculated by the following formula:

$$\text{affinity}_1(x) = 1.0 - \frac{\sum_{i=1}^{m} e1(x_i)}{n}, \tag{10}$$

where $x_i = i_1, \cdots, i_b$ and $e1(x_i)$ are calculated by

$$e1(x_i) = \sum_{w=1}^{b} h(i_w), \tag{11}$$

where

$$h(i_w) = \begin{cases} & \text{if } v_{i_w} \text{ has been allocated,} \\ 0, & \text{or arrival\_time}\left(x_i^E + i_w, \left|x_i^E\right| + 1\right) > \text{late}_{i_w}, \text{ or resource\_cost}\left(x_i^E + i_w, \left|x_i^E\right| + 1\right) > \text{res}_i, \\ 1, & \text{otherwise,} \end{cases} \quad (12)$$

and it judges whether v_{i_w} can be added to the current execution sequence of u_i, x_i^E (it is the initial empty set) is the current execution sequence of a_i, and $x_i^E + i_w$ means adding i_w to x_i^E

(2) The value of the second objective can be calculated by the following formula:

$$\text{affinity}_2(x) = 1.0 - \frac{\sum_{i=1}^{m} e2(x_i)}{\sum_{i=1}^{n} \text{reward}_i}, \quad (13)$$

where $e2(x_i) = \sum_{w=1}^{b} h(i_w) \times \text{reward}_{i_w}$

(3) The value of the third objective can be calculated by the following formula:

$$\text{affinity}_3(x) = \sum_{i=1}^{m} \frac{e3(x_i)}{m \times \text{max\_}r}, \quad (14)$$

where max_r represents the maximum amount of resources carried by the drone, $e3(x_i) = \sum_{w=1}^{k} h(i_w) \times (\text{travel\_cost}(p_{\text{last}}, p_{vi_w}) + \text{rescost}_{i_w})$, and p_{last} represents the location of the last task assigned to u_i (when $w = 1$, p_{last} represents the location of u_i)

(4) The value of the forth objective can be calculated by the following formula:

$$\text{affinity}_4(x) = \max_{i=1}^{m} \frac{e4(x_i)}{m \times \text{max\_}t}, \quad (15)$$

among them, max_t is the longest time to complete and $e4(x_i) = \sum_{w=1}^{k} h(i_w) \times (\text{travel\_time}(p_{\text{last}}, p_{vi_w}) + \text{timecost}_{i_w})$

It is worth noting that not all tasks encoded in antibodies can be successfully assigned to drones, and the order in which drones perform tasks is implicit in the antibody. Through above transformation, we can input the individual into the four functions in formulas (10)–(15) to facilitate the evaluation of the individual.

3.5. Multiobjective Optimal Solution Set. Since in the case of multiobjective, each individual has multiple attributes (each task is identified as an attribute in the coding here); the comparison between two individuals cannot simply use the size relationship. Therefore, this section will briefly introduce the basis of multiobjective optimization.

3.5.1. Domination Relationship. In the domination relationship between different individuals, let x and y be two different individuals in the multiobjective optimization population, if the following two conditions are met:

(1) For all subgoals (referred to as four goals in this article), x is no worse than y, that is, $f_k(x) \leq f_k(y)(k = 1, 2, \cdots, r)$

(2) There is at least a certain subgoal that makes x better than y. It is expressed as $\exists l \in \{1, 2, \cdots, r\}$, which satisfies $f_l(x) < f_l(y)$

At this time, x is called nondominated and y is dominated, where x dominates y. It can be symbolized as $x \prec y$. If x and y do not meet the above conditions, it proves that there is no dominant relationship between the two.

3.5.2. Pareto Solution Set. It can be obtained from the above dominance relationship that all individuals in a population can be sorted by the definition of dominance relationship, but because some solution sets may not have dominance relationships, these solution sets are in the same position. Through these characteristics, a solution set can be obtained. Each individual z in this solution set satisfies

(1) Individuals in the population are dominated by it, expressed as $z \prec p, p \in A$

(2) Other individuals cannot dominate it; that is, there is no dominance relationship between the two, expressed as $z \nprec q, q \nprec z, q \in B$

In (1) and (2), $A \cap B = \varnothing$, and $A \cup B = U$.

At the end of the experiment, we will select the Pareto-based multiobjective optimal solution set as the task allocation plan, in which the subobjectives compared between individuals can refer to the affinity function.

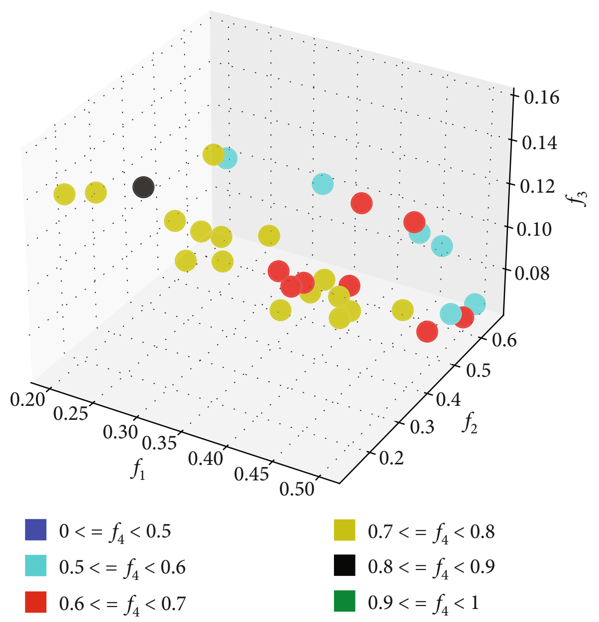
Table 2: Parameter settings.

Parameter name	Value
Hypermutation ratio	[0.2, 0.5, 0.8, 1]
Population size	200
Number of iterations	1000
Number of clones	1
Recombination ratio	[0, 0.5]
Optimization number	4

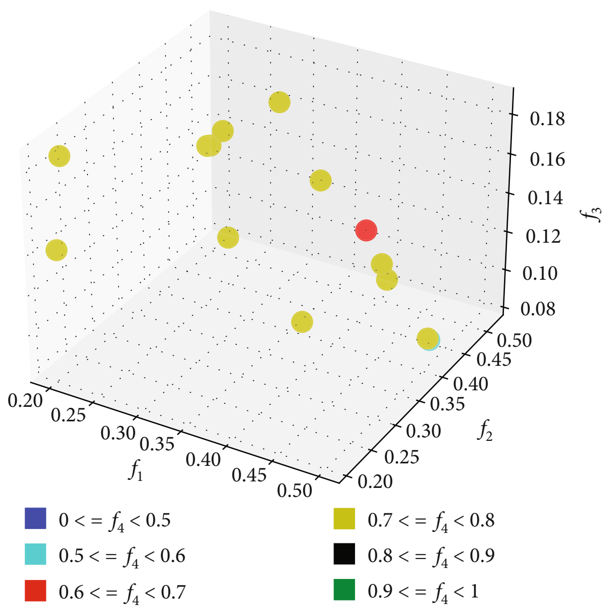
4. Experimental Results

This section will demonstrate the effectiveness of the method proposed in this article through experimental comparison and conclusion analysis.

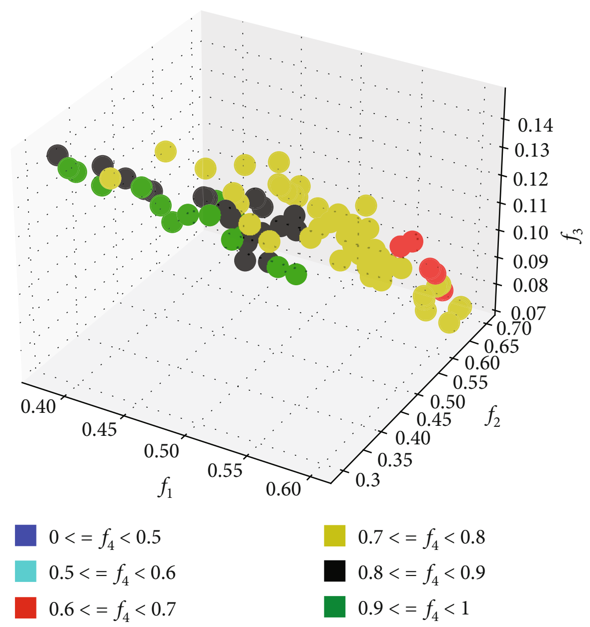
4.1. Experimental Environment. The experimental environment of this article is as follows: Window 10 operating system 64-bit professional edition, Intel i7-7600U CPU, clocked at 2.80 GHz, memory 8 G, and the programming environment is Visual Studio 2010.



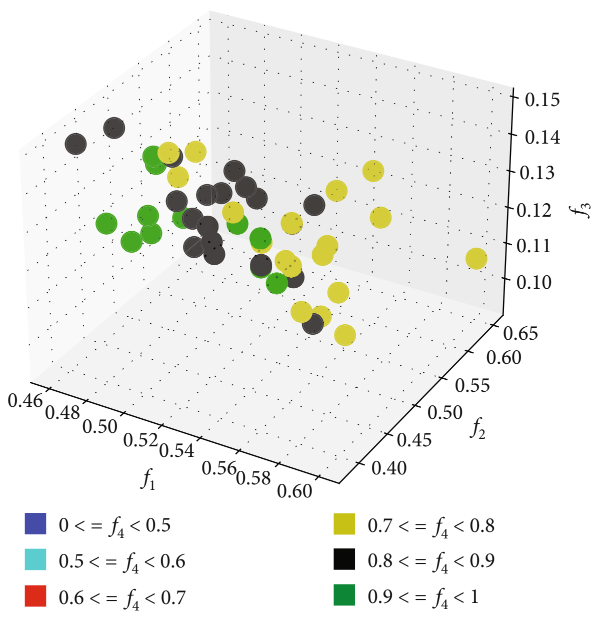
(a) CSA algorithm ($m = 5$, $n = 10$)



(b) GA algorithm ($m = 5$, $n = 10$)

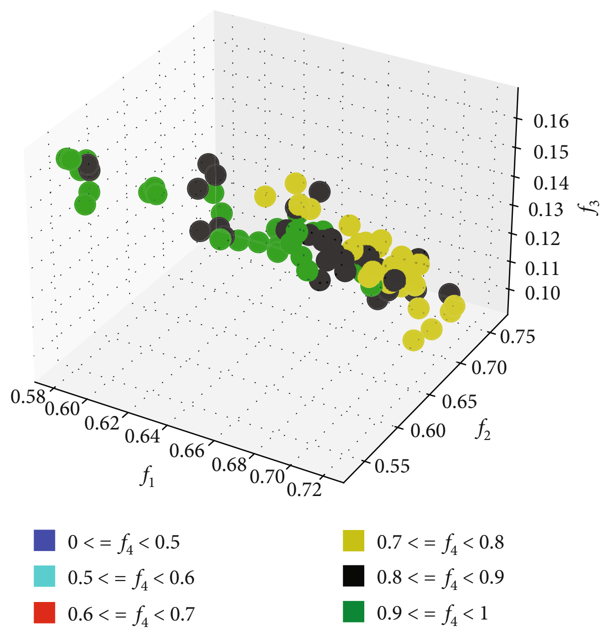


(c) CSA algorithm ($m = 20$, $n = 50$)

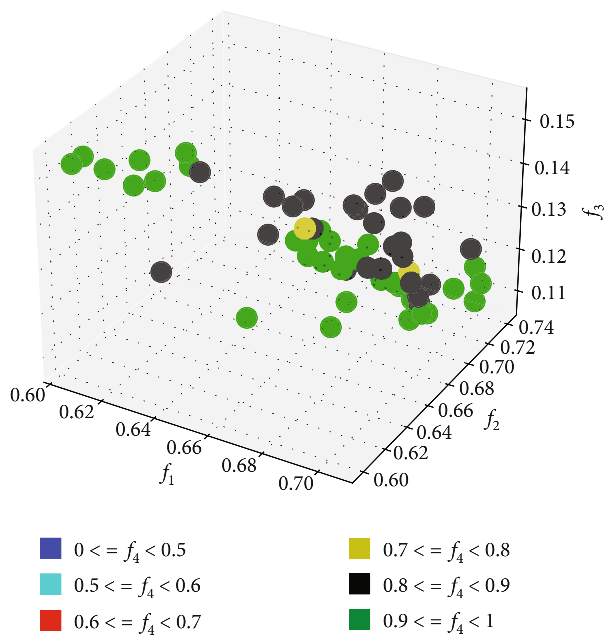


(d) GA algorithm (m=20, n=50)

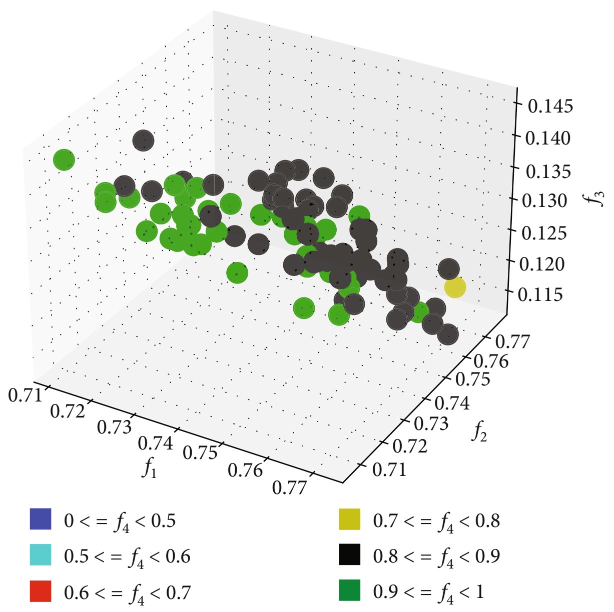
Figure 2: Continued.



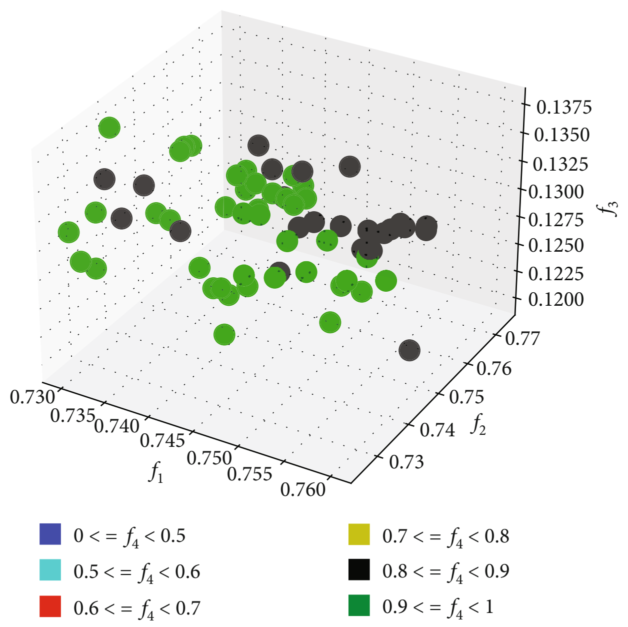
(e) CSA algorithm ($m = 50$, $n = 200$)



(f) GA algorithm ($m = 50$, $n = 200$)



(g) CSA algorithm ($m = 200$, $n = 1000$)



(h) GA algorithm ($m = 200$, $n = 1000$)

Figure 2: distribution of Pareto solution sets for different scales of problems.

First, randomly generate relevant agent and task data. In order to facilitate comparative experiments, randomly generate test data in the preparation phase. In order to simulate the experiment, supposing the initial test position of each UAV is $\mathrm{pu}_i = (x, y)(i = 1 \cdots m)$, x and y are random integers between [0,100]; the earliest possible execution time early_j is a random number between [0,100], and the latest execution time late_j is a random number between $[\mathrm{early}_j,150)$; the time required to execute the task time_j is random number between [1,10); the resources consumed to execute the task rest_j is a random number between [0,100); the revenue of the task reward_j is a random number between [0,1) [22]. For functional constraints, this article assumes that each task must have one or more drones that can perform it. After a certain number of iterations, the overall UAV and task change. We make corresponding changes to the antibody through the change information. In order to facilitate the experiment, it is assumed that the time cost and resource cost of transmission

Table 3: Comparison results of different algorithms.

Problem scale	GA (No. of nondominated solutions)	CSA (number of nondominated solutions)	No. of solutions (GA dominates CSA)	No. of solutions (CSA dominates GA)
5_10	14	31	0	1
20_50	43	87	0	5
50_200	57	88	0	3
200_1000	59	93	0	6

Table 4: Optimization objective results for different ratios of tasks to UAVs.

Problem scale	Number of nondominated solutions	Min f_1	Min f_2	Min f_3	Min f_4
50_100	21	0.2500	0.2402	0.0802	0.6804
50_150	16	0.4733	0.4227	0.0988	0.7448
50_200	13	0.5750	0.5479	0.1026	0.7358
50_250	11	0.6680	0.6200	0.1111	0.7001

have a linear relationship with distance. The experimental parameters of the main CSA algorithm are shown in Table 2.

4.2. Comparison of Results of Different Algorithms. In this paper, the CSA algorithm and the GA algorithm are consistent with the experimental test data. The algorithm parameter environment and parameters are as shown in the subsection. Four sets of task allocation test data of different scales will be carried out to compare the final Pareto set. Figure 2 shows the two algorithms $(m = 5, n = 10)$, $(m = 20, n = 50)$, $(m = 50, n = 200)$, and $(m = 200, n = 1000)$, respectively. The distribution of Pareto solution set. Table 3 compares the dominance of the Pareto solution set.

It can be observed from Figure 2 that in the comparative experiments of different scales, the number of nondominated solutions of the CSA algorithm is more, and the distribution area of the solutions is wider, which also means that the diversity is better, from the analysis in Table 3. It can be obtained that the solution of the CSA algorithm is generally better than that of the GA algorithm. This trend becomes more obvious with the increase of scale. The main reason is that the mutation strategy of the CSA algorithm is better than the mutation strategy of the GA; the CSA variation dynamically changes individuals with the degree of affinity, while the GA algorithm only performs a small amount of mutation on the basis of crossover and variation range is often small, resulting in a small exploration pace, and the dynamic change of the variation range is beneficial to improve the efficiency of the solution. The experimental results just proved this point.

4.3. Results on Different Ratios of Tasks to UAVs. In this section, the results of CSA algorithm on different ratios of tasks to UAVs are presented, i.e., $n = 100$, 150, 200, and 250, when $m = 50$. Since there are many Pareto solutions and it is impossible to compare all the solutions, there is only one of the similar data (i.e., if the difference between two data is within 0.05 in f_1, f_2, f_3, f_4). Table 4 compares the experimental results of different ratios.

According to Section 3.4, the smaller the target value of f_1, f_2, f_3, f_4, the better the optimization effect. In Table 4, it can be seen that in the comparative experiments of different ratios, when the number of UAVs remains the same, as the task scale increases rapidly, the difficulty of solving the problem is further increased. It will lead to a decrease in the number of nondominated solutions and a decrease in quality (the values of f_1, f_2, f_3, f_4 continue to increase), which is in line objective facts. Combining this, in order to maximize the number of tasks successfully assigned, maximize the benefits of tasks execution, minimize the resource costs, and minimize the time costs, the ratio of drones and tasks must be balanced as much as possible. Increase the number of drones as much as possible to optimize the allocation plan, when considering the constraints of many conditions.

5. Conclusion

In order to solve the problem of UAV combat task allocation, this paper proposes a clone selection algorithm based on the artificial immune system; this algorithm simultaneously optimizes four objectives, namely, maximizing the number of tasks successfully assigned, maximizing the benefits of tasks execution, minimizing the resource costs, and minimizing the time costs. And the effectiveness of method is proved by experimental comparison. In the later stage, the comparison algorithm model will be improved mainly by the high-dimensional multiobjective optimization strategy of genetic algorithm (for example, niche strategy and reference point based on hyperplane), and the gene recombination of clone selection algorithm will be added to optimize the CSA algorithm to further improve the quality of solutions.

Data Availability

All the data can be generated according to the steps described in our paper, and readers can also ask for the data by contacting cxj_dna@yeah.net.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the Natural Science Foundation of China (No. 71701208).

References

[1] V. Chaudhary and J. K. Aggarwal, “A generalized scheme for mapping parallel algorithms,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 3, pp. 328–346, 1993.

[2] W. W. Chu, L. J. Holloway, Min-Tsung Lan, and K. Efe, “Task allocation in distributed data processing,” *IEEE computer*, vol. 13, no. 11, pp. 57–69, 1980.

[3] M. Kafil and I. Ahmad, “Optimal task assignment in heterogeneous distributed computing systems,” *IEEE Concurrency*, vol. 6, no. 3, pp. 42–50, 1998.

[4] V. Lesser, C. L. Ortiz Jr., and M. Tambe, *Distributed Sensor Networks: A Multiagent Perspective*, vol. 9, Springer Science & Business Media, 2012.

[5] A. Chapman, R. A. Micillo, R. Kota, and N. Jennings, *Decentralised Dynamic Task Allocation: A Practical Game–Theoretic Approach*, The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09), Hungary, 2009.

[6] S. Fitzpatrick and L. Meertens, *Distributed Sensor Networks. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series)*, vol. 9, Springer, Boston, MA, 2003.

[7] H. Yedidsion, R. Zivan, and A. Farinelli, “Applying max-sum to teams of mobile sensing agents,” *Engineering Applications of Artificial Intelligence*, vol. 71, pp. 87–99, 2018.

[8] C. Wei, Z. Ji, and B. Cai, “Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2530–2537, 2020.

[9] W. Lee, N. Vaughan, and D. Kim, “Task allocation into a foraging task with a series of subtasks in swarm robotic system,” *IEEE Access*, vol. 8, pp. 107549–107561, 2020.

[10] A. Salman, I. Ahmad, and S. Al-Madani, “Particle swarm optimization for task assignment problem,” *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[11] G. Yang and V. Kapila, “A dynamic-programming-styled algorithm for time-optimal multi-agent task assignment,” *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol. 2, 2001, pp. 1959–1964, Orlando, FL, USA, 2001.

[12] S. Valluru, *Implementation of a Clonal Selection Algorithm*, 2014, Technical Report.

[13] W. Tian and J. Liu, “An improved immune clone selection algorithm for multi robot task scheduling,” in *2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009)*, pp. 128–131, Zhangjiajie, China, 2009.

[14] H. Dai, Y. Yang, and C. Li, “Hybrid crossover based clonal selection algorithm and its applications,” in *In international conference on intelligent data engineering and automated learning*, H. Yin, Y. Gao, B. Li, D. Zhang, M. Yang, Y. Li, F. Klawonn, and A. J. Tallón-Ballesteros, Eds., pp. 468–475, Springer, Cham, 2016.

[15] M. Y. Zhao, K. E. Tang, G. Lu et al., “A novel clonal selection algorithm and its application,” in *2008 International Conference on Apperceiving Computing and Intelligence Analysis*, pp. 385–388, Chengdu, China, 2008.

[16] J. H. Li, H. W. Gao, and S. A. Wang, “A novel clone selection algorithm with reconfigurable search space ability and its application,” in *2008 Fourth International Conference on Natural Computation*, vol. 6, pp. 612–616, Jinan, China, 2008.

[17] J. A. White and S. M. Garrett, “Improved pattern recognition with artificial clonal selection?,” in *Artificial Immune Systems. ICARIS 2003*, J. Timmis, P. J. Bentley, and E. Hart, Eds., vol. 2787 of Lecture Notes in Computer Science, pp. 181–193, Springer, Berlin, Heidelberg, 2003.

[18] X. Pan, F. Liu, and L. Jiao, “A dynamic clonal selection algorithm for project optimization scheduling,” in *Simulated Evolution and Learning. SEAL 2006*, T. D. Wang, Ed., vol. 4247 of Lecture Notes in Computer Science, pp. 821–828, Springer, Berlin, Heidelberg, 2006.

[19] K. Huang, Y. Dong, D. Wang, and S. Wang, “Application of improved simulated annealing genetic algorithm in task assignment of swarm of drones,” in *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pp. 266–271, Xi'an, China, 2020.

[20] Y. Jia, “Research on UAV task assignment method based on parental genetic algorithm,” in *Advances in Swarm Intelligence. ICSI 2019*, Y. Tan, Y. Shi, and B. Niu, Eds., vol. 11655 of Lecture Notes in Computer Science, pp. 439–446, Springer, Cham, 2019.

[21] Z. Zha, C. Li, J. Xiao et al., “An improved adaptive clone genetic algorithm for task allocation optimization in ITWSNs,” *Journal of Sensors*, vol. 2021, Article ID 5582646, 12 pages, 2021.

[22] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, “Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm,” *IEEE Access*, vol. 8, pp. 19306–19318, 2020.

[23] J. Zhou, X. Zhao, D. Zhao, and Z. Lin, “Task allocation in multi-agent systems using many-objective evolutionary algorithm NSGA-III,” in *Machine Learning and Intelligent Communications. MLICOM 2019*, X. Zhai, B. Chen, and K. Zhu, Eds., vol. 294 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 677–692, Springer, Cham, 2019.

[24] J. Teich, “Pareto-front exploration with uncertain objectives,” in *Evolutionary Multi-Criterion Optimization. EMO 2001*, E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, Eds., vol. 1993 of Lecture Notes in Computer Science, pp. 314–328, Springer, Berlin, Heidelberg, 2001.

[25] D. A. Van Veldhuizen and G. B. Lamont, “Evolutionary computation and convergence to a Pareto front,” in *Late breaking papers at the genetic programming 1998 conference*, pp. 221–228, Madison, Wisconsin, 1998.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9970942, 14 pages
https://doi.org/10.1155/2021/9970942

Research Article

Design and Performance Analysis for Edge Intelligence-Based F-PMIPv6 Mobility Support for Smart Manufacturing

Donghyun Kim, ByungJun Park, Junhyung Moon, Jaeen Lee, and Jongpil Jeong

Department of Smart Factory Convergence and Physical Science Research Institute, Sungkyunkwan University, Suwon 16419, Republic of Korea

Correspondence should be addressed to Jongpil Jeong; jpjeong@skku.edu

Received 14 March 2021; Accepted 31 May 2021; Published 24 June 2021

Academic Editor: Yong Zhang

In this paper, we propose a new mobility management network, i-FP, to be used in the smart factory that continues to develop in the Fourth Industrial Revolution. i-FP was created to solve the current local mobility management problem of legacy frameworks. MN (mobile node) refers to a mobile device in a manufacturing environment that includes workers, production facilities, and AGV. To allow mobile nodes (MNs) to move from one domain to another, i-FP uses three network entities: LFA (Local Factory Anchor), FAG (Factory Access Gateway), and MN, as an extended concept of PMIPv6. Among the three network entities in i-FP, LFA and FAG can act as edge intelligence devices to reduce the handover latency of the MNs. i-FP also uses IP header-swapping mechanisms to prevent traffic overhead and enhance network throughput. We evaluate new framework i-FP, PMIPv6, and HMIPv6, which are legacy protocols of local mobility management, in various ways and evaluate three schemes. We confirm that i-FP works better than do the other network methods used in the smart factory.

1. Introduction

Globally, countries are rapidly changing in the Fourth Industrial Revolution. The governments of major countries are striving to become leaders of the Fourth Industrial Revolution by means of differentiated policy support. In particular, manufacturing-based companies are focusing on changes to upgrade their general factories to smart factories. Various research attempts are made on existing manufacturing processes, such as Cyber Physics System (CPS) [1, 2], robotics, 3D printing, edge computing, and cyber-security technologies [3]. Because these key technologies are applied across all manufacturing areas, innovations are emerging that dramatically increase the competitiveness of manufacturing.

Edge computing has become an integral part of the smart factory that emerged with the development of cloud computing. Edge computing became edge intelligence, where research was conducted on how AI (Artificial Intelligence) delivers data analysis [4]. Accordingly, data gathering uses wired and wireless methods for data analysis, and it is a recent trend that it is designed wirelessly to allow flexible movement of workers, mobile shelves, and production facilities in the smart factory [5, 6]. Thus, in the manufacturing industry, wireless network connectivity continues to be a challenge. Wireless and mobile communication network technologies play a major role in creating diverse environments in manufacturing industries. With the growing importance for new wireless networks for the smart factory, new technologies have been developed, leading to the emergence of a variety of hierarchical mobility frameworks.

In wireless network frameworks, the mobility of users is typically divided into intradomain and interdomain movements. These mobilities correspond to the global mobility protocol [7] and the local mobility [8, 9] of the wireless network. Whereas the global mobility protocols maintain the connectivity to the factory beyond the scope of the domain as a user's movement, local mobility protocols operate through a distribution of the restricted region within the domain.

When a network user, who accesses and receives from a mobile network with IPv6 [10–12], accesses another network, the network transmits traffic from the original domain using the global mobility protocol to manage the network that first access from the outside. It uses a local mobility protocol delivering traffic within a domain in succession, and enabling users can successfully communicate. Global and local mobility enables users to leverage seamless and flexible communication.

Global mobility protocols used to manage user mobility include F-PMIPv6 (Fast Proxy Mobile) [13, 14], HMIPv6 [15], TeleMIP [16], and HIP [17]. This paper focuses on the processes of global mobility and local mobility. Among the global mobility protocols, PMIPv6 [18, 19] and HMIPv6 [20] use traffic to locate target addresses in the network and to access top-level gateways. This is applied to the smart factory [21–24], which is divided into connecting the factory with the factory and connecting the mobile with the mobile. However, when applying the global mobility protocol to communications between adjacent mobile devices, the traffic is less efficient. We propose a new mobile network protocol to improve the problem of protocols in smart factories. The protocol to be used in the smart factory is i-FP, an acronym for "Intelligent Factory PMIPv6."

i-FP has three main objectives in total. (i) It provides routing services optimized for traffic in the domain. The routing optimization (RO) service enables rapid retrieval of communication paths between peers to deliver optimized traffic in the domain. (ii) It utilizes wireless links to reduce network traffic overhead. Efficient traffic management by wireless links can improve the performance of wireless networks, as the existing limited bandwidth overloads wireless applications. (iii) We introduce i-FP to reduce the cost to reduce the domain topology. We apply i-FP to the smart factory to improve the performance of existing applied local mobility protocols when connecting to the web. It adds IP header swap technology to prevent traffic overhead in the network. To evaluate the performance of i-FP, we compare the newly proposed framework with other frameworks related to the local mobility protocol. Based on the result, we prove that i-FP is an effective technique in the local domain of the smart factory.

In short, the key contributions of the study are provided as follows.

(i) Propose a new effective architecture, including gateways with edge intelligence capabilities within a smart factory environment

(ii) The proposed i-FP incorporates the architecture of HMIPv6 and PMIPv6, while leveraging IP swapping mechanisms to effectively reduce tunneling costs and latency

(iii) It is possible to effectively respond to mobility processing by performing what mobile nodes need to do directly on the gateway

(iv) Since protocol data is only transmitted over wired networks, i-FP does not generate signaling costs on wireless links, minimizing data loss and data costs on wireless links

(v) Finally, interactive connection of the factory cloud and the factory anchor within domains allows flexible management of the mobile nodes in the integrated system

In this paper, Section 2 introduces associated research, Section 3 discusses the architecture and procedures of the proposed framework, Section 4 evaluates the performance of the proposed technique, and Section 5 provides a conclusion based on performance evaluation results.

2. Related Work

In this section, we review existing research work related to our research to help readers understand the importance of mobile communication management and the latest technology trends in the smart factory. Convergence of IT and OT technologies is discussed in Smart Manufacturing Overview. In Edge Intelligence, we describe the edge computing that is the key technology applied to i-FP. Finally, we explain the FPMIPv6 and HMIPv6 which were used for mobility support in smart manufacturing.

2.1. Smart Manufacturing Overview. During the Fourth Industrial Revolution, the smart factory is becoming a very important element. General factories, consisting of production facilities, control systems, and factory management systems, are automated methods of the Third Industrial Revolution. With the advent of the Fourth Industrial Revolution, general factories are becoming smarter. Automation has been made intelligent by means of the fusion of IT and OT technologies [25], and intensive research is being conducted on data generation and analysis in terms of production facilities and production management by adding sensors and IoT technologies to production facilities [26, 27].

As illustrated in Figure 1, hyperconnections, digital integration, automation, integration, and data are keywords in smart factory fields. The collected big data is analyzed by AI technology or applied to machine learning. Robots cooperate with workers to add productivity, and IoT is used as a data collector. Based on technologies, the smart factory has been developed into an intelligent system that can achieve strategic goals, such as productivity, quality, and customer service in the entire production process, including design, development, manufacturing, distribution, and logistics [28]. To this end, various information exchanges by means of communication between devices and between devices and humans are essential for a smart factory, since the factory has centralized control over production facilities so far.

In order to establish a flexible production system of a smart factory, an organic response of production facilities is essential. Modular production equipment, as well as IoT technologies, can be changed according to the production process, so that production can be customized to meet consumer demands [29]. Development of mobile communication is essential for data communication and data analysis of this modular equipment.

2.2. Edge Intelligence. Edge intelligence (EI) is a concept that defines the communication, computing, and storage capabilities of a particular infrastructure that are closest to local unit users on a distributed network. The term "edge intelligence"

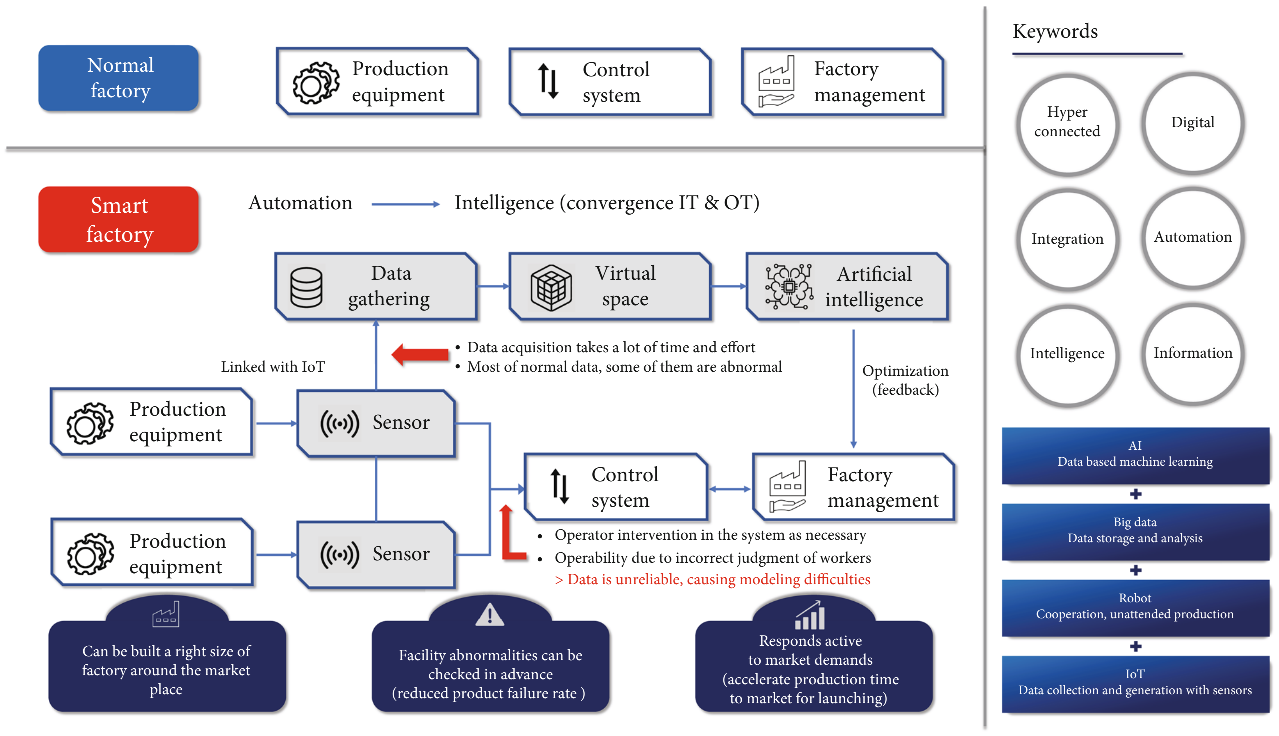

Normal factory
Production equipment
Control system
Factory management
Smart factory
Automation
Intelligence (convergence IT & OT)
Data gathering
Virtual space
Artificial intelligence
Linked with IoT
• Data acquisition takes a lot of time and effort
• Most of normal data, some of them are abnormal
Optimization (feedback)
Production equipment
Sensor
Control system
Factory management
Production equipment
Sensor
• Operator intervention in the system as necessary
• Operability due to incorrect judgment of workers
> Data is unreliable, causing modeling difficulties
Can be built a right size of factory around the market place
Facility abnormalities can be checked in advance (reduced product failure rate)
Responds active to market demands (accelerate production time to market for launching)
Keywords
Hyper connected
Digital
Integration
Automation
Intelligence
Information
AI Data based machine learning
Big data Data storage and analysis
Robot Cooperation, unattended production
IoT Data collection and generation with sensors

FIGURE 1: i-FP Architecture of Smart Factory.

was represented by Zhou et al.'s [4] research. "Edge" is a physical location representation in which data is generated and processed. In other words, an edge means that there is a control device and computing device. Recent research has shown that devices equipped with small computers are changing to IoT and IIoT. These devices operate with AI capabilities. Data processing devices in edge fields are used for intelligent collection of data, analysis, aggregation, and application. Edge intelligence uses edge computing [30] to support these edge devices in analysis performed on AI and ML models. Intelligent edges break traditional client-server models that are typically "double or thin" for clients, and servers are those that have the ability to process, analyze, and protect data.

Edge intelligence has three main entities: connectivity, computation, and control. This edge intelligence allows manufacturing systems to connect to each other over a wireless or wired network. This computing approach interconnects a specific range of workers, managers, smart facilities, robots, sensors, and AGVs and also applies to a wide range of connections, such as smart factory [31].

Fully connected edge intelligence systems can collect, manage, analyze, and archive large amounts of data through interdistributed computing. This local unit of computing can be combined with the cloud system to improve or replace computing power. Edge computing is performed mostly by edge servers (or IIoT gateways) that are part of IT equipment [32]. These servers can be half racks, two blades, or industrial embedded PCs. It is applicable to certain business services that require control of calculated insights from these local units and can also be extended to the cloud. Intelligent edges can perform control mechanisms for these local units and devices on the edges.

Cloud computing is a powerful solution if you want an Internet connection, but its use is very limited if it requires real-time data processing and communicating restrictions exist. Edge intelligence lets you keep some or all data close to where it was created, rather than sending it to a remote data center or cloud server for processing. Therefore, we want to intelligently manage communication to mobile devices moving within the smart factory by leveraging edge-computable entities [33].

2.3. Mobility Support for Smart Manufacturing. The FPMIPv6 is the "Fast Proxy Mobile IPv6" implementation of FMIPv6 in PMIPv6 environments, enabling high-speed handover. The mobile node (MN) detects mobile signals and preemptive Mobile Access Gateway (pMAG) for transport and prepares the MN's handover via HI (Handover Initiation) and HAck (Handover-Acknowledgement) messages. In the preparation phase, a bidirectional tunnel is formed between the pMAG and the new Mobility Access Gateway (nMAG). During the time when communication with MN is lost, data from LMA is buffered from pMAG to nMAG. When MN is connected to nMAG, packet data buffered by nMAG is sent to MN to prevent packet data loss due to separation during handover to facilitate communication. The mode of FPMIPv6 is divided into two: prediction and response.

In predictive handovers, the full handover begins by the MN detecting the need for handovers and delivering a handover indication message to the pMAG on its own. In reactive handover, on the other handovers, MN detects the need for handovers and performs a network reentry process directly into the target network, and handovers are initiated by

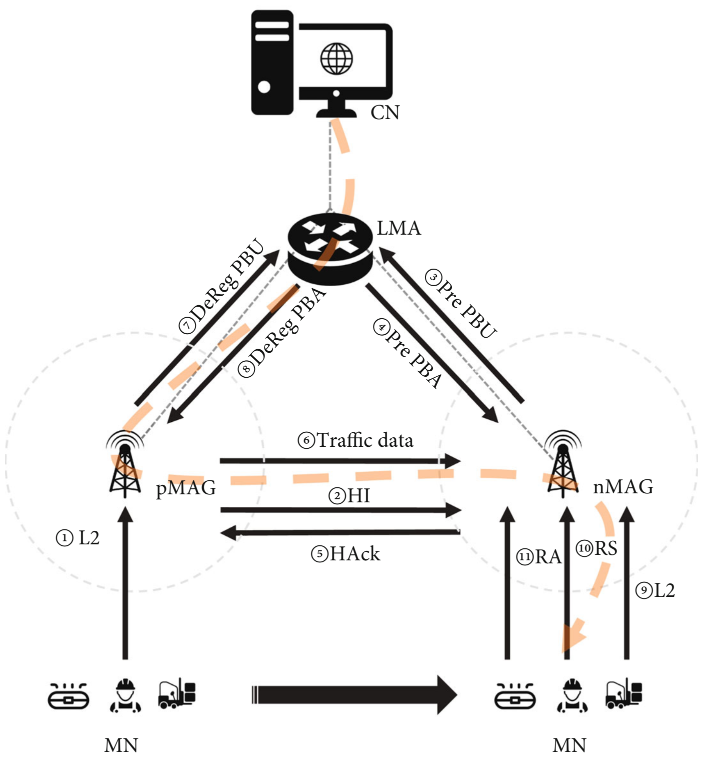


Figure 2: Handover process of FPMIPv6.

nMAG before pMAG recognizes the need for handovers. In a handover scenario, the MN basically uses a previable access network (pAN).

It is assumed that the situation is communicating with the pMAG. Before the MN moves, it sends a side message about moving to the pAN, including the information nAN MN-ID. Upon receipt of the message, the pAN sends HI messages, including MN-ID and nAN-ID, to the pMAG to inform the MN movement.

The pMAG transmits HI messages (including information from MN and LMA) to the nMAG. Upon receiving the HI message, the nMAG sends a HAck (Handover-Acknowledgement) to the pMAG. Upon receiving HAck messages, the pMAG forms a two-way tunnel with the nMAG. When a two-way tunnel is formed, the pMAG transmits packets stored in the MN to the nMAG and is stored in the nMAG buffer.

After the handover (L2), when the MN is connected to the nMAG, the MN sends packets from the nMAG and the nMAG binds the MN by sending a Proxy Binding Update (PBU) message to the LMA. After LMA receives a PBU message, it registers the status information of the MN in the Binding Cache Entry (BCE) and sends a Proxy Binding Acknowledgement (PBA) message to the nMAG. The binding process of MN is completed after the LMA has received a full PBA message over the nMAG. This content is schematized in Figure 2.

PMIPv6 is an IPv6-based mobility-enabled protocol for mobile nodes. The signal and routing state settings of the mobility node are performed by entities in the network. The main entities of PMIPv6 are MAG (Mobility Access Gateway) and LMA (Local Mobility Anchor). The role of LMA is responsible for the reachability of topology anchor points on mobile nodes and mobile node home network prefixes (HNP). The LMA manages information about MNs and has the right to manage topology anchor points for home network prefix information to be assigned to MN. MAG is the access link to which the mobile node is connected, and on behalf of the mobile node, it performs mobility management of the MN through the LMA and bidirectional passageways. MAG detects MN's entry and exit within the network and recommends binding registration tasks for LMA. The technique for supporting the mobility of MN in the network using the PMIPv6 protocol is shown in Figure 3.

Hierarchical Mobile IPv6 (HMIPv6) is a method proposed by the IETF (Internet Engineering Task Force) as a way to reduce the handover delay that occurs when a mobile node moves in MIPv6. HMIPv6 is a protocol that reduces signaling caused by handover of a mobile node by locally managing the movement of the mobile node and reduces the delay and signaling overhead caused by HMIPv6 during binding update. HMIPv6 requires a binding update to HA (home agent) and CN (Corresponding Node) when MN moves to another subnet.

If the MN is far from HA or CN, the binding update procedure causes unnecessary delay and signaling overhead. The access network is hierarchically structured in HMIPv6 to solve this problem. HMIPv6 can reduce signaling costs caused by user mobility and scalability in the growing network in managing local mobility and has separated global mobility management and local mobility. Global mobility is still managed by HMIPv6, but local mobility within the local domain is managed from the MAP, which is a local mobility management agent. Therefore, since movement in the MAP region is unnecessary in HA and CN, the delay and signaling overhead in HMIPv6 needed to maintain or manage information about it can be greatly reduced. The contents are shown in Figure 4.

However, there are many protocols, such as CIP [34], HAWAII [35], HMIPv6, Tele MIP, RDMA [36], and PMIPv6. For example, CIP and HAWAII force a strict tree structure in the domain topology. The hierarchical structure is based on a mobility agent, and all routers must be involved in mobility signaling. Therefore, CIP and HAWAII are expensive to implement, because all routers in the domain need to be upgraded. Other protocols, such as HMIPv6, TeleMIP, RDMA, and PMIPv6, do not require the participation of all routers in mobility signaling. Instead, a mobility agent as a topology anchor point and an access router as an external agent are introduced. By means of the cooperation of the mobility agent and the access router, the above protocols can deliver traffic to the moving users in the local domain. Even though these protocols do not require much functionality within the domain topology, they suffer from severe routing problems with intradomain traffic. If a user attempts to send a packet to a peer communicating in the same domain, the packet is first delivered to the domain gateway router and forwarded to the peer with which the user is communicating.

Multimedia applications such as online games are popular nowadays, and triangular routing paths cause additional transmission delays and waste bandwidth resources. HMIPv6, RDMA, and TeleMIP share similar delivery procedures and mobility signaling. These protocols use the domain's special address for global mobility and binding of network-specific addresses for domain forwarding. HMIPv6, RDMA, TeleMIP, and PMIPv6 are not network-enabled mobility, and they use only one address for domain routing and binding. Network support schemes can help PMIPv6 to reduce signal cost.

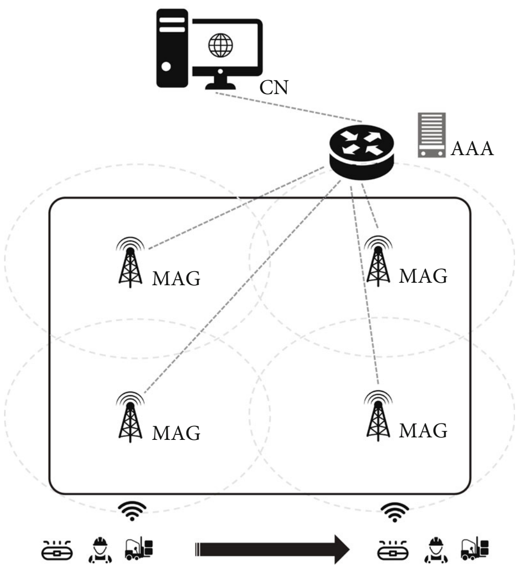


Figure 3: Handover Process of the PMIPv6.

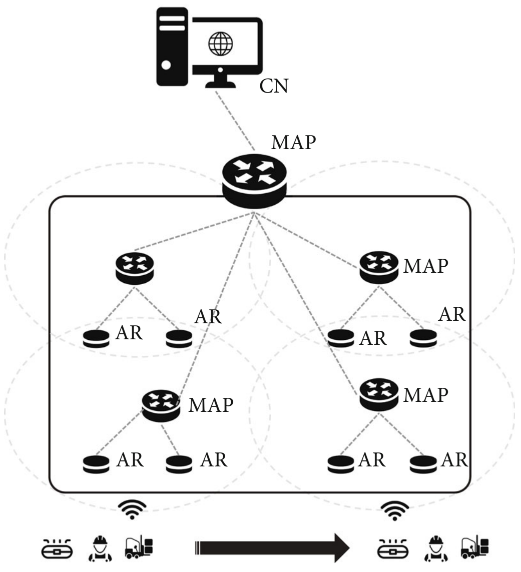


Figure 4: Concept of HMIPv6.

3. Edge Intelligence-Based Hierarchical Mobility Support for Smart Manufacturing

3.1. System Architecture. i-FP has four main entities, including the features of HMIPv6 and PMIPv6 to work better. i-FP contains four entities to enhance the network environment and performance of smart manufacturing using LFA, FAG, MN, and cloud. MN refers to mobile devices that include workers, production facilities, and AGV in the manufacturing environment, and LFA and FAG include the functions of edge intelligence. We propose the i-FP of a smart factory containing these major entities. The overall architecture can be found in Figure 5.

Various messages related to i-FP mobility support are used in the mobility management protocol. When analyzing mobility models, the following message sizes should be considered: MAG and LMA used in FPMIPv6 are used as the basis for FAG and LFA in i-FP. The first entity is LFA. The LFA performs the same role as the proxy home agent (HA) for MN. When MN is moved to the local domain, FAG receives traffic on behalf of MN and sends traffic to the link where MN is located. FAGs with edge computing function receive traffic on behalf of MN when MN is moved to the local domain and send traffic to MN's location link.

To make this possible, i-FP uses two types of addresses, RCoA (Regional Care of Address) and LCoA (Link Care of Address), which are the same way as HMIPv6 manages the MN. RCoA is the address obtained from MN when MN first enters the local domain. The address obtained from MN, RCoA, serves as the ID card for MN in the local domain. The MN also displays the location via the RCoA and updates HA or peers in communicating. If the MN moves within the local domain, the RCoA remains fixed. Therefore, it is not necessary for MN to send once binding update messages to HA or peers during communicating that does not deviate from the local domain. However, for this reason, RCoA can identify the domain in which MN is located but is not

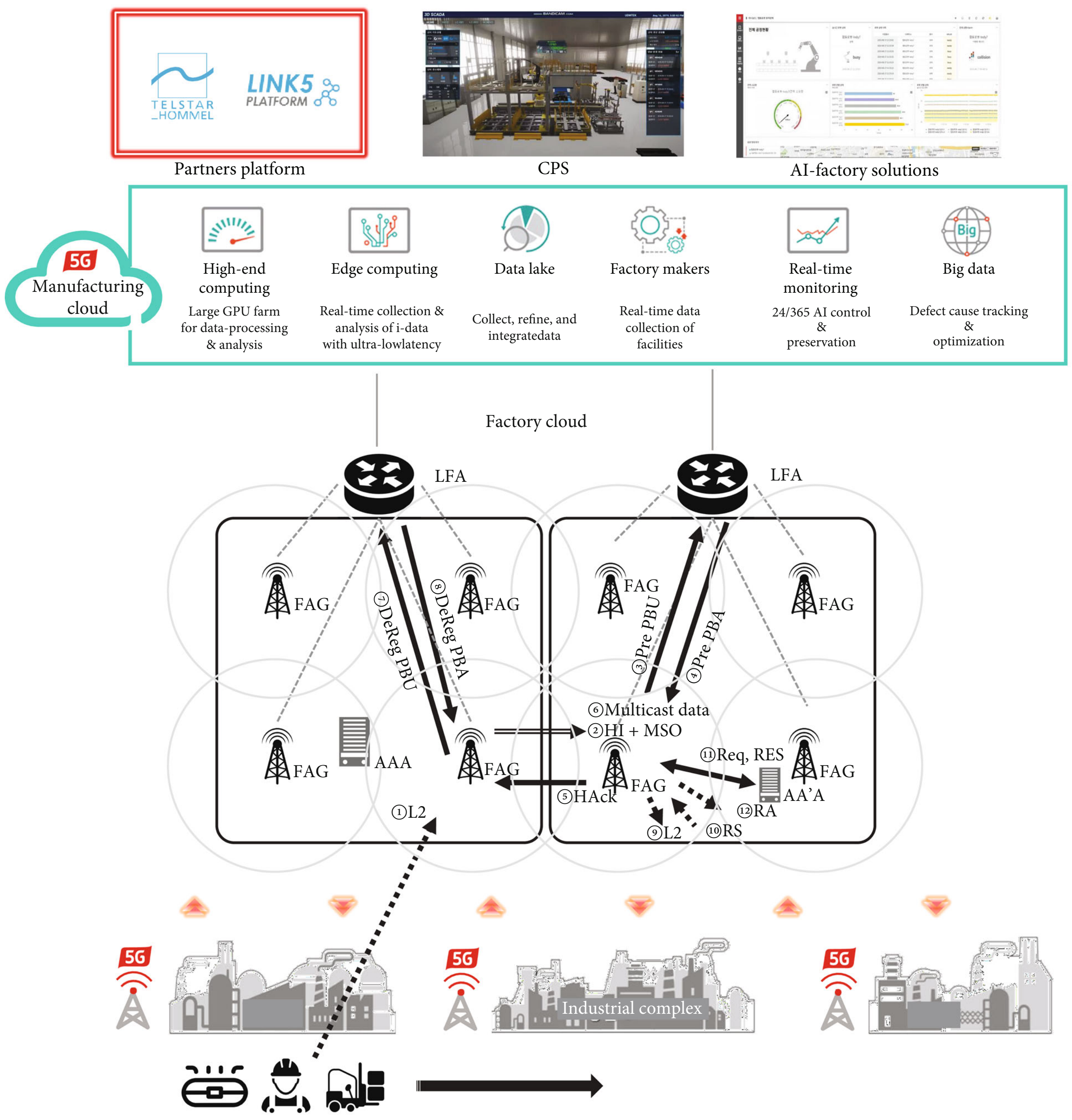


Figure 5: Concept of the smart factory.

affected by updates in intradomain movements. This means that it is not known whether the RCoA is connected to the FAG. Therefore, i-FP uses the address LCoA to obtain a more detailed location. LCoA is the same address as MN's location and is updated whenever MN is repositioned so that it always has accurate location information.

The LFA manages the RCoA and LCoA for MN. When the LFA receives traffic on the direction of MN, the LFA changes the target of the packet from RCoA to LCoA. The reason for using LCoA is that MN can quickly and accurately identify the stationary gateway in the domain. When the target of the packet changes to LCoA, the LFA sends the updated packet to the FAG with the MN connected. Because LCoA contains accurate location information for MN, MN can successfully receive packets. The binding of the RCoA and LCoA in LFA allows the i-FP to accurately and quickly transfer traffic from the MN to the domain. However, LFA cannot manage the local mobility of the domain.

Therefore, i-FP uses FAG, a second entity, to manage the transfer of MN. FAG is the access router (AR) of the local domain and is responsible for the i-FP's wireless network. FAG interconnects various APs that provide wireless links to the network. When the MN is connected to the FAG network, the FAG sends a Request Registration (RR) message to the MN to the LFA. If the MN is an authorized user, the FAG grants MN the authority to allow access. For example, the FAG specifies a new LCoA for MN and forwards MN's traffic to the corresponding radio link. If an MN attempts to send a packet to another MN located in the i-FP domain, the FAG updates the packet's destination address and ensures that the packet is routed to the peer of the MN over the optimal path. Packet delivery procedures are described in detail in the following sections.

The third entity in i-FP is MN. MN is a roaming wireless device that includes workers, production facilities, and AGV included in the factory local domain. HMIPv6 and PMIPv6 use IP tunneling techniques in the local domain, while i-FP uses IP swapping mechanisms. IP switching agents are located between the data layer and the network layer on all MNs in the i-FP domain. This agent is used to process IP headers for MN traffic.

The IP swapping agent changes the packet target LCoA to RCoA when MN receives the packet and then sends the packet to the network layer. When a packet is sent to a network layer, the IP swapping agent simultaneously updates the source address of the packet to the LCoA of the MN. In this way, MN can maintain connectivity even if the domain is moved to a different network. Detailed operating procedures for IP swapping mechanisms can be found in "TMSP: Terminal Mobility Support Protocol" [37]. i-FP uses IP swapping technology, so there is no need for additional IP headers as there are no tunneling cost and latency.

i-FP uses the same LMA and MAG and MN as PMIPv6 to manage mobility. Although i-FP uses more IP addresses than PIMPv6, IPv6 is not a big problem because it has enough IP addresses. While HMIPv6 generates protocol signal costs on wired and wireless links, PMIPv6 and i-FP do not generate signal costs on wireless links, and protocol data is transmitted only on wired networks. Therefore, PMIPv6 and i-FP do not generate wireless bandwidth overhead on handover.

The last entity, factory cloud, is associated with the LFA in each domain. MN's mobility information is stored within the integrated system, cloud. A two-way bridge is formed between factory cloud and LFA. A formed bridge provides a flexible connection of information between each domain, which groups LFAs located in each domain. Nonprocessable computational processing with FAG and edge computing applied to the LFA within the local domain is sent to factory cloud via the LFA. The transmitted data is processed and analyzed through high-end computing in cloud. This facilitates the handling of handovers of i-FPs within the local domain.

3.2. Operation Procedure. Adding the concept of cloud to the smart factory, we configure the system structure in Figure 5. We have configured systems that connect to the cloud to construct cloud-based edge computing and have preconditions for applications that correspond to the configured network to be provided. Cloud servers are configured using OpenStack, and edge computing is located in local units to form cloud-based edge computing. IoT data is also stored on cloud storage in real time. When configuring gateways and servers between real-time storage, you configure nodes at the end of the application to act as controllers. Finally, IoT detection data is stored through the gateway and analyzed by the server. The application layer to which storage and analytics data are applied configures servers by node for real-time processing.

3.3. Registration Procedure. It acquires RCoA when the MN first accesses the network before performing the registration procedure on the i-FP network. When MN acquires RCoA, DHCP or Stateless Configuration (SC) is used. The following RCoA is registered with the global mobility agent. RCoA does not change while MN is in the domain. Thus, a global mobility protocol is maintained within the domain. Figure 5 shows the registration procedure since MN obtained RCoA.

As the MN proceeds with the handover, the newly connected FAG sends a Router Advertisement (RA) message to the MN. RA messages serve as keys for MN to acquire LCoA in the new FAG. The FAG then binds the RCoA and LCoA obtained from the MN within the local domain and sends the information to the LFA with the binding information to the Local Binding Update (LBU) message. If LFA allows LBU messages, the binding entry is set between RCoA and LCoA. This information is used for the domain and domain communication. The LFA sends a Local Binding Acknowledgement (LBA) message to the FAG. As soon as the FAG receives the LBA, the registration process is completed and the MN is able to transmit packets from a new location. i-FP is divided into traffic between domains and traffic within the domain. If MNs are in different local domains, their traffic is between domains. Otherwise, the traffic is within the domain. Traffic handover in i-FP is described as follows.

3.4. Intradomain Handover. The traffic delivery within a domain involves address management at the FAG. When MN1 transmits a packet to MN2, the IP swapping agent (ISM) of MN1 updates the original source to the LCoA and sends the packet to the FAG1. The FAG1, which is the first hop on the transmission path, updates the destination of the packet to the LCoA of the MN2. Because it is a packet transmission within the domain, it does not pass through the LFA and sends the packet directly to the FAG2 by referring to the LCoA. The FAG2 that receives the packet through the LCoA recognizes that the destination of the packet is the MN that is connected to it, so the FAG2 changes the address of the packet source into the RCoA that is the original address. Last, if the packet is sent to the MN2, the IP swapping agent of the MN2 changes the destination of the packet to the RCoA. Then, the packet is transmitted to the MN2. If the MN2 processes the transmitted packet and then again transmits the response message to the MN1, the IP swapping mechanism of the wireless link changes the address of the MN2 to the LCoA and transmits the packet to the FAG2.

Because the FAG2 is a packet movement in the domain, the MAG2 changes the destination address to the LCoA and transmits the packet to the FAG1. Because the FAG1 to which the packet is delivered recognizes that the destination indicates the MN that manages itself, it changes the destination source address to the RCoA that is the original address and delivers the packet to the MN1. Before the MN1 receives the packet, the IP swapping mechanism of the wireless link changes the destination address, which is the address of the MN1, to the RCoA that is the original address and delivers

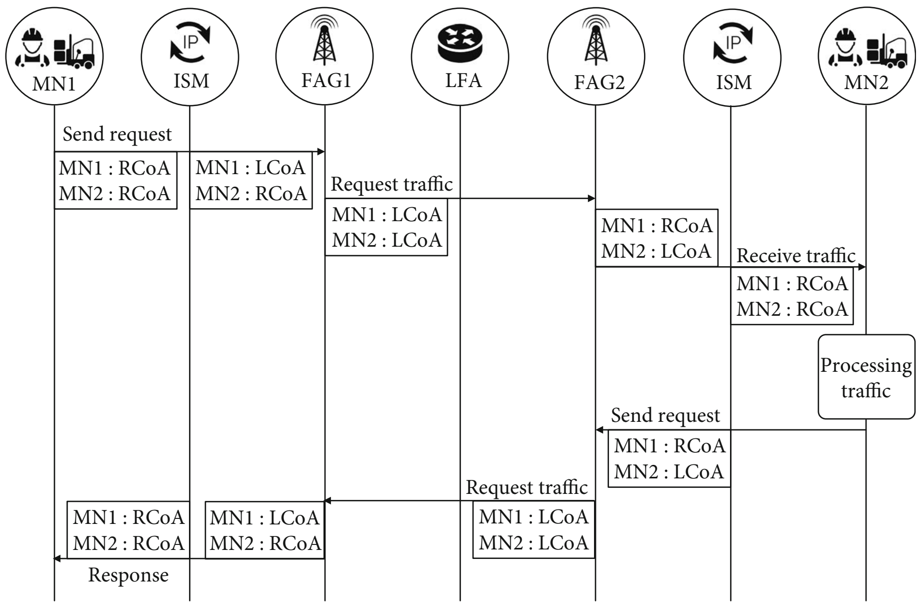


Figure 6: Flow of i-FP within the domain.

the packet. Because of this procedure, if the mobile node, such as MN1 or MN2, knows only the original address, it can exchange the packet without other functional requirements with the packet address management mechanism in the domain. This procedure is shown in Figure 6.

The handover of the MN in the domain sends an L2 signal, which means that the MN is about to do a handover to the pFAG to which it is first connected. The pFAG recognizes this signal sent to nFAG, which is expected to hand over the HI message to prepare the handover with the LCoA of the MN. The nFAG that delivers the MI message generates nLCoA, which will be newly assigned, and puts the LCoA and nLCoA in the message, which demands to be preregistered to the LFA, and transmits it. If the LFA receives the pre-BU message, it carries out the user certification to the AAA server, so that it maps the certified users with the RCoA and nLCoA information and preregisters them temporarily. Then, the LFA sends the pre-BA message to the nFAG to complete the registration. The nFAG that receives the message transmits the HAck message to pFAG to show that the handover is ready.

The pFAG that receives the HAck transmits the traffic that is transmitted to it to the nFAG and carries out the buffering. The pFAG puts the DeReg BU messages in the LCoA and sends them to the LMA in order to cancel the registration of the MN immediately; then, the LFA identifies it and updates the LCoA to nLCoA, so that it formally registers the MN, which is preregistered to the nFAG. As the response to it, the DeReg BU message is delivered to the pFAG, which transmits the response message to the L2 message to the MN. The MN transmits the RS message to the nFAG to demand the access. Because the nFAG registers the MN in advance, it can immediately put the nLCoA, which is the new LCoA address, in the RA and send it and then transmits the traffic that has just been buffered. The handover in the domain is completed by this procedure, and the MN can communicate with the nFAG. This procedure is shown in Figure 7.

3.5. Intradomain Handover. The traffic movement between other domains, which is different from the traffic movement in a domain, operates as follows. If the original MN sends the CN by means of the wireless network in order to request data, the wireless link operates the IP swapping mechanism before the packet arrives and changes the source address of the packet to the RCoA, which can be recognized even by the external domain. The packet with the changed address arrives at the FAG, which sends the packet as is to the LFA that is the local domain gateway. Because the destination of the packet is the CN that is an external domain, the LFA again converts the source address of the packet to the RCoA and sends it the CN, which processes it and then sends the response message for it to the RCoA.

Next, the CN receives the packet from the LFA and converts it to the LCoA, which is topologically identical, in order to send it to the internal domain. Then, the CN transmits the packet to the MAG that has the corresponding MN. The FAG delivers the packet to the MN and converts the address, which has already become the LCoA for the IP swapping mechanism which is antecedently operated, to the original RCoA and transmits it to the MN. By means of the IP converting mechanism in the domain, in the i-FP, the MN is involved in the IP change, or while it does not have to know it, the MN can exchange the packet with external domains. This is shown in Figure 8.

The handover between domains sends the L2 message to the pFAG to which it is originally connected, which means that the connection is about to be discontinued. The pFAG that receives the message transmits the RCoA of the MN with the HI message to the nFAG of the domain that expects the connection. The nFAG, which identifies the message, transmits the pre-BU message with the RCoA and the nLCoA, which is the newly assigned address, in order to preregister the MN to the nLFA to which it belongs. The nLFA receives the pre-BU message and carries out the user certification at the AAA server. If the message passes the certification, the

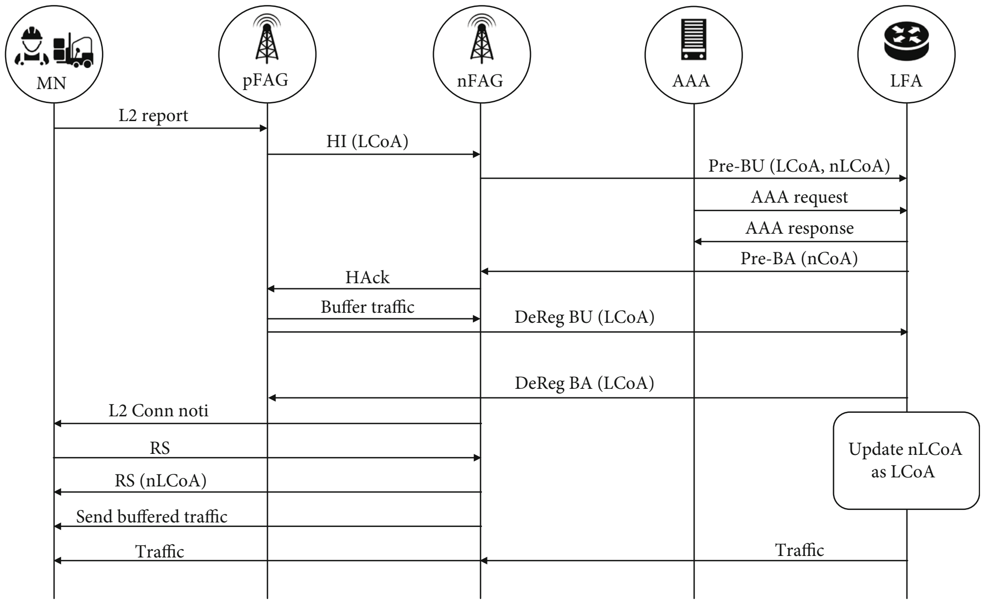


Figure 7: Flow of i-FP within the domain.

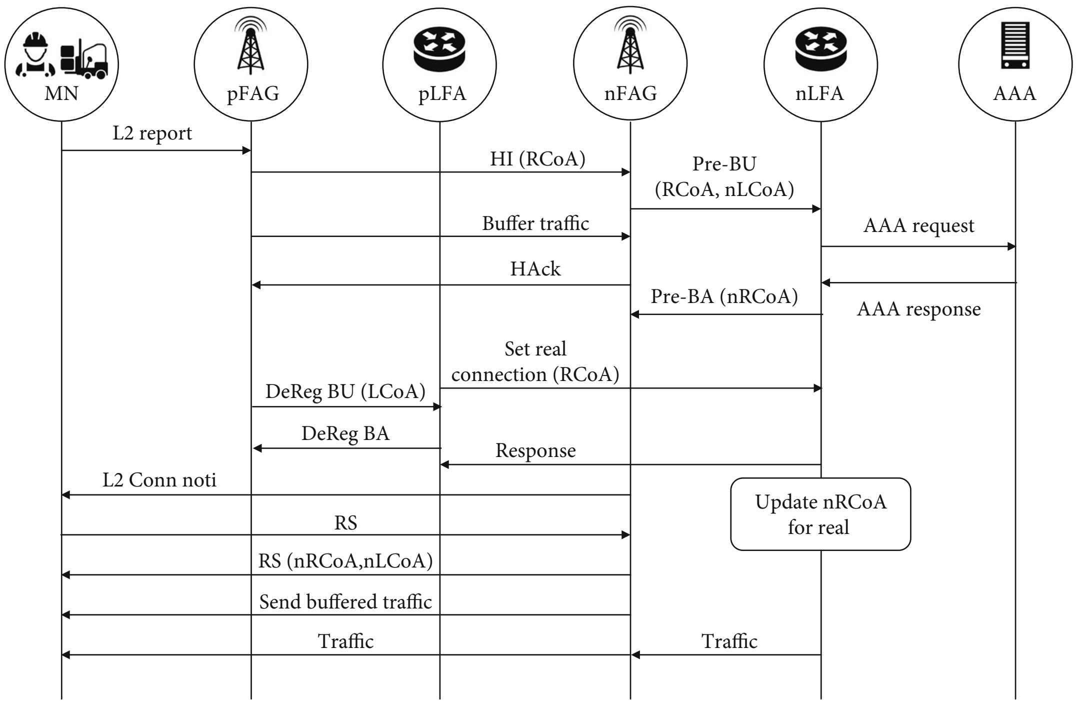


Figure 8: Handover of i-FP in the interdomain.

nLFA generates the nRCoA, saves it with the nLCoA, and lets it be preregistered. The LFA, as the response to it, transmits the pre-BA message with the nRCoA to the nFAG.

The nFAG that receives it then transmits the HAck message, which means that the handover is completely ready, to the pFAG. The pFAG that receives the message transmits the traffic that is delivered to it, and the nFAG buffers the transmitted traffic. After that, the pFAG, in order to cancel the connection of the MN, puts the LCoA in the DeReg-BU message and transmits it. The pLFA tells the nLFA to formally register the preaccessed MN, along with the RCoA. Then, the nLFA identifies the nRCoA by using the mapped RCoA and nRCoA, registers the nRCoA formally, and then sends the response message to the pLFA. The pLFA that receives the response message sends the DeReg BA message to the pFAG and informs the cancellation of the registration.

The nFAG transmits the response message to the L2 message, in which the MN transmits for the first time and then transmits the RS message to request the access to the nFAG. The nFAG receives the message, puts the newly assigned address of nRCoA and nLCoA in the RA message, and transmits them to the MN. After that, the nFAG transmits the

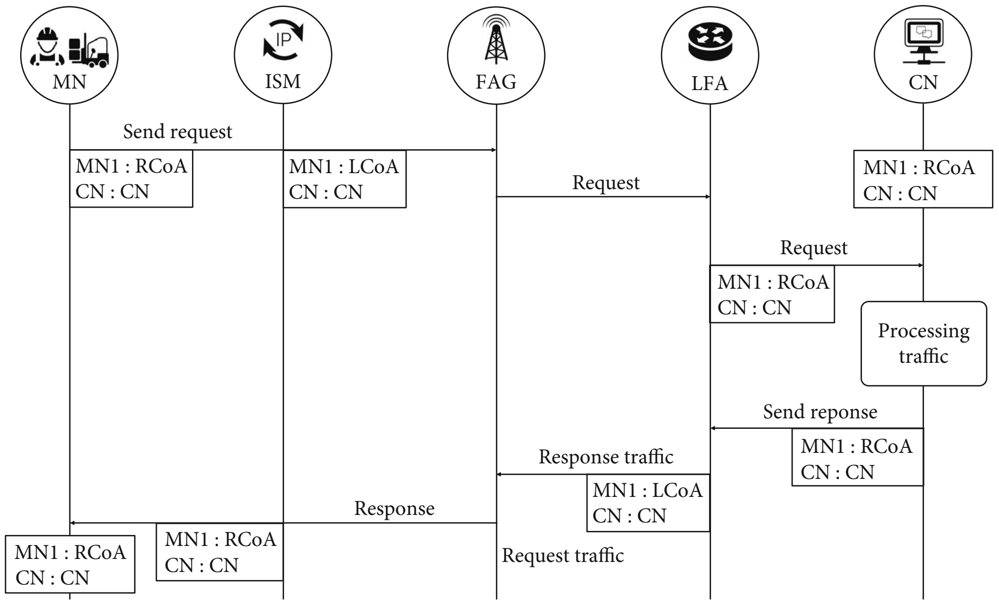


Figure 9: Traffic movement path between domains of i-FP.

buffered traffic data to the MN. The handover between domains is completed in these processes, and the fast connection and communication of the MN and the nFAG become possible. This is shown in Figure 9.

4. Performance Analysis

In this section, we conducted performance evaluations with mathematical modeling of the newly proposed i-FP, HMIPv6, and PMIPv6. This analysis objective is to minimize the costs arising from the network. Network costs are defined by message size and hop distance (bandwidth aspect). Each modeling was analyzed under the same conditions and ignored router processing costs. Table 1 shows parameters for mobile protocols used in performance analysis.

4.1. The Number of Routing Hops. We measure the number of traffic routing hops in the first performance analysis. The primary goal of i-FP is to provide the best routing route for intradomain traffic. This analysis compares the number of routing hops for intradomain traffic of the three protocols. At the same time, we compare and evaluate the transmission delay of intradomain traffic. The GR (Global Router) for the domain is located in the factory cloud (FC).

The number of routing hops of the three protocols is similarly set for domain internal traffic. When a packet is sent from CN, the FC in the domain receives the packet list. FC then forwards the packet to the new MN-connected AR. The AR sends packets over the radio link to the MN. Therefore, the number of routing hops can be expressed as 5, in which H_{X-Y} means the number of routing hops for node X and node Y. HMIPv6 and PMIPv6 require packets to be forwarded by FC. The FC encapsulates the packet and forwards it to the MN's current location.

Therefore, traffic inside the HMIPv6 and PMIPv6 domains causes triangular routing problems. However, the number of routing hops in the domain in i-FP is different from the traditional method. If an i-FP MN tries to forward a packet to another MN, the packet arrives at AR1 and forwards the traffic to AR2, where AR2 is located. Finally, AR2 forwards the packet with MN2. i-FP has fewer routing hops than HMIPv6 and PMIPv6 because packets are forwarded on the shortest path. Equations (1)–(3) represent the number of interdomain routing hops for HMIPv6, PMIPv6, and i-FP, and (4)–(6) represent the number of routing hops for the domain. FAGs for PMIPv6 can be configured as bridges for mobile nodes if MN1 and MN2 are on the same FAG network. The local routing optimization mechanism reduces forwarding delays.

$$H_{\text{Inter}}^{\text{HMIPv6}} = H_{\text{CN-FC}} + H_{\text{FC-AR}} + H_{\text{AR-MN}}, \tag{1}$$

$$H_{\text{Inter}}^{\text{P}_{\text{MIP}}\text{v6}} = H_{\text{CN-FC}} + H_{\text{FC-AR}} + H_{\text{AR-MN}}, \tag{2}$$

$$H_{\text{Inter}}^{\text{i-FP}} = H_{\text{CN-FC}} + H_{\text{FC-AR}} + H_{\text{AR-MN}}, \tag{3}$$

$$H_{\text{Intra}}^{\text{HMIPv6}} = H_{\text{MN1-AR1}} + H_{\text{AR1-FC1}} + H_{\text{FC1-AR2}} + H_{\text{AR2-MN2}}, \tag{4}$$

$$H_{\text{Intra}}^{\text{PMIPv6}} = H_{\text{MN1-AR1}} + H_{\text{AR1-FC1}} + H_{\text{FC1-AR2}} + H_{\text{AR2-MN2}}, \tag{5}$$

$$H_{\text{Intra}}^{\text{i-FP}} = H_{\text{MN1-AR1}} + H_{\text{AR1-FC1}} + H_{\text{AR2-MN2}}. \tag{6}$$

4.2. The Number of Routing Hops. The protocol signal cost is incurred in updating location information as MN moves, and usage is proportional to the amount of packets. Signal costs include RS (Router Solicitation) messages, BU (Binding Update) messages, and BA (Binding Acknowledgement) messages. The cost of the protocol signal, which is the cost of the handover procedure, is expressed as C_s. C_s is expressed as (7). P is the probability of one handover per t unit time.

Table 1: Parameter values for performance analysis.

Variable name	Value	Variable name	Value
$H_{\text{CN-FC}}$	2	$H_{AR1\text{-AR2}}$	1
$H_{\text{FC-AR}}$	1	i-FP$_{\text{BU}}$	96
$H_{\text{AR-MN}}$	1	i-FP$_{\text{BA}}$	96
$H_{\text{MN1-AR1}}$	1	i-FP$_{\text{RouterSol}}$	44
$H_{\text{AR1-FC1}}$	1	i-FP$_{\text{RouterAdv}}$	68
$H_{\text{FC1-AR2}}$	1	i-FP$_{\text{REU}}$	142
$H_{\text{AR2-MN2}}$	1	HMIPv6$_{\text{RBU}}$	80
HMIPv6$_{\text{RBA}}$	60	PMIPv6$_{\text{PBA}}$	88
HMIPv6$_{\text{RouterSol}}$	44	PMIPv6$_{\text{RouterSol}}$	44
HMIPv6$_{\text{RouterAdv}}$	68	PMIPv6$_{\text{RouterAdv}}$	68
PMIPv6$_{\text{PBU}}$	88	$T_{\text{MAG-LMA}}$	100
D_{L2}	100	$W_{\text{MAG-LMA}}$	300
A	10	$T_{\text{MN-LFA}}$	200
$T_{\text{MN-MAP}}$	100	L1$_{P\text{\_Header}}$	100
$W_{\text{MN-MAP}}$	300	$H_{\text{MAP-MN}}$	2
$H_{\text{LMA-MAG}}$	1	U	10000
R	1000		

H_{mn} is expressed as $H_{\text{mn}} = 1 - p$ with the probability of a handover to the MN. s is the total size of the protocol packets used for the handover procedure. m is the number of mobile nodes that exist in a domain per hour. The cost of HMIPv6, PMIPv6, and i-FP signals can be calculated by (8)–(10).

$$C_S = \sum_{n=1}^{\infty} n^* p^n * (1-p) * m * \frac{s}{t}, \tag{7}$$

$$C_S^{\text{HMIPv6}} = \sum_{n=1}^{\infty} n^* p^{n*} (1-p)^* m^* \frac{\text{RBU} + \text{RBA} + \text{RS} + \text{RA}}{t}, \tag{8}$$

$$C_S^{\text{PMIPv6}} = \sum_{n=1}^{\infty} n^* p^{n*} (1-p) * m * \frac{\text{PBU} + \text{PBA} + \text{RS} + \text{RA}}{t}, \tag{9}$$

$$C_S^{\text{i-FP}} = \sum_{n=1}^{\infty} n^* p^{n*} (1-p)^* m * \frac{\text{RS} + \text{RA}}{t}. \tag{10}$$

4.3. Handover Delay. If an MN handover is transmitted from one network to another, the MN may not be able to receive traffic. When the MN handover is moving through the network, MN's information is transmitted and MN cannot receive traffic. The time during which traffic is not received is called a handover delay. There are usually three possible causes of handovers. First, the MN's previously connected communication is broken when the MN moves to a different network. The MN should then be connected to a different radio link than before. Thus, D_{L2} represents the handover delay that occurs during the L2 link switching phase. MN gets

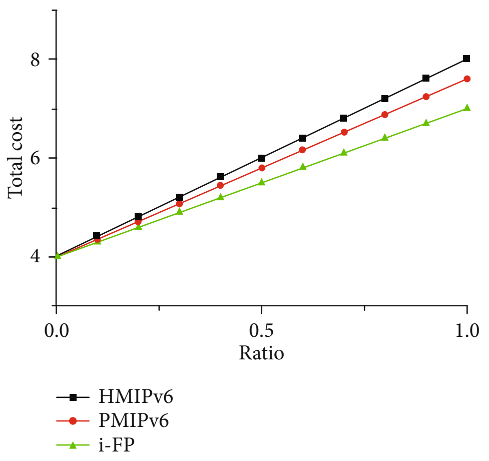


Figure 10: Average number of routing hops.

a new IP address after being linked to a new network. D_{IP} indicates the time it takes for MN to acquire a new IP address. The time generated during the IP acquisition phase has a significant impact on the handover delay. There is a prestudied mechanism to solve these problems [38–41]. Specifying an optimized IP address allows the MN to obtain an IP address without the need for Duplicate Address Detection (DAD). i-FP is designed to leverage these mechanisms to allow MN to establish optimized IP addresses in new networks. Finally, MN gets a new IP address and sends an LU message. D_{LU} is the time spent completing a location information update. D_{LU} is primarily affected by the physical distance between MN and the agent. Thus, D_{LU} is managed by a local mobility protocol that uses proxy HA in the domain. The handover delay is based on these variables and is expressed as expression (11).

We compared and evaluated the performance of PMIPv6, HMIPv6, and i-FP on the same network with D_{L2}. The three protocols have different D_{IP}s. PMIPv6 does not change the IP address of MN in the new network, so the value of D_{IP} is zero. However, the values of D_{IP} for i-FP and HMIPv6 are larger than 0 because MN must set a new address. In both protocols, MN takes time to automatically set MN's address according to RA messages when it moves to a new network. RA messages are sent by the AP at every interval. $D_{\text{IP}}^{\text{HMIPv6}}$ has a random value. If the time is the same mobile time, then the average value of $D_{\text{IP}}^{\text{HMIPv6}}$ becomes $A/2$. $D_{\text{IP}}^{\text{i-FP}}$ has the same value as $D_{\text{IP}}^{\text{HMIPv6}}$. The three protocols D_{LU} are also different. HMIPv6 changes the location message between MN and MAP, creating a tunnel after the handover procedure. The tunnel generation time $W_{\text{MN-MAP}}$ between MN and MAP is the same as the one-way transmission time $T_{\text{MN-MAP}}$ of MN and MAP. Therefore, in HMIPv6, D_{LU} is equal to $2 * T_{\text{MN-MAP}} + W_{\text{MN-MAP}}$. In PMIPv6, D_{LU} is used to change update messages between MAG and LMA. In PMIPv6, a tunnel is formed between MAG and LMA in PMIPv6, just as a tunnel is created between MN and MAP in HMIPv6. The time required to create a tunnel between MAG and LMA is $W_{\text{FAG-LMA}}$. The update message transfer time from MAG to LMA is $T_{\text{FAG-LFA}}$. In PMIPv6, D_{LU} is

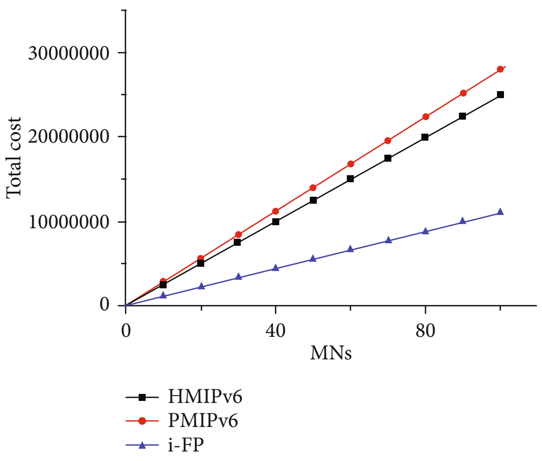


Figure 11: Traffic signaling cost.

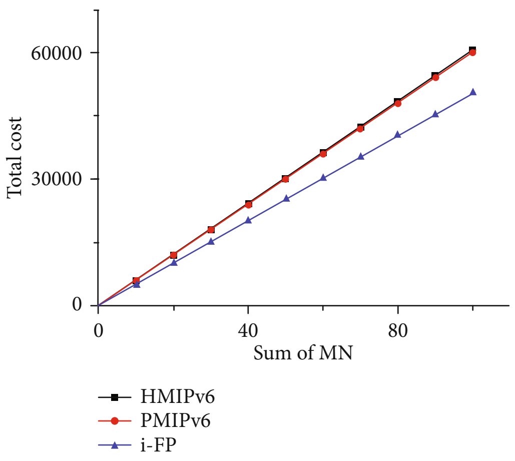


Figure 12: Handover delay.

defined as $2 * T_{\text{FAG-LFA}} + W_{\text{FAG-LFA}}$. For i-FP, no tunnel creation between MN and LFA is required. Thus, $D_{\text{IP}}^{\text{i-FP}}$ is represented by $2 * T_{\text{MN-LFA}}$ which is the MN and LFA bidirectional communication time. The handover latency of the three protocols is expressed in expressions (12)–(14), respectively. A is the interval between adjacent response messages.

$$D_{\text{HO}} = D_{\text{L2}} + D_{\text{IP}} + D_{\text{LU}}, \tag{11}$$

$$D_{\text{HO}}^{\text{HMIPv6}} = D_{\text{L2}} + \frac{A}{2} + 2 * T_{\text{MN-MAP}} + W_{\text{MN-MAP}}, \tag{12}$$

$$D_{\text{HO}}^{\text{PMIPv6}} = D_{\text{L2}} + 0 + 2 * T_{\text{FAG-LFA}} + W_{\text{FAG-LFA}}, \tag{13}$$

$$D_{\text{HO}}^{\text{i-FP}} = D_{\text{L2}} + \frac{A}{2} + 2 * T_{\text{MN-LFA}}. \tag{14}$$

4.4. Traffic Overhead. Finally, we compare the results by measuring the traffic overhead of HMIPv6, PMIPV6, and i-FP. HMIPv6 and PMIPv6 send traffic by IP tunneling technology. Tunneling headers cause overhead of user data in the network. C_{overhead} is the value of traffic overhead. $C_{\text{overhead}} = L_{\text{IPHeader}} * H$ denotes the traffic overhead of the three protocols. The length of the IP tunnel is defined as L_{IPHeader}. H is the number of hops the packet traverses in the local domain. The data rate is R bps and the packet size of user data is U; the overhead cost of HMIPv6, PMIPv6, and i-FP can be expressed as

$$C_{\text{overhead}}^{\text{HMIPv6}} = L_{\text{IPHeader}} * H_{\text{MAP-MN}} * \frac{R}{U}, \tag{15}$$

$$C_{\text{overhead}}^{\text{PMIPv6}} = L_{\text{IPHeader}} * H_{\text{LFA-FAG}} * \frac{R}{U}, \tag{16}$$

$$C_{\text{overhead}}^{\text{i-FP}} = 0. \tag{17}$$

4.5. Numerical Results. We evaluated the difference in performance between HMIPv6, PMIPv6, and i-FP with various conditions and obtained numerical results for routing hops, traffic signaling cost, handover delay, and traffic overhead. We analyze the numerical results of each evaluation method in the order mentioned. i-FP has the fewest routing hops, and the average number of routing hops of PMIPv6 is smaller than that of HMIPv6. δ represents the ratio of the intradomain traffic F_{intra} divided by the sum of the intradomain traffic F_{intra} and the interdomain traffic F_{inter}, which means $\delta = F_{\text{intra}}/(F_{\text{inter}} + F_{\text{intra}})$. In Figure 10, we can see that the average number of routing hops of i-FP is lower than the other two protocols.

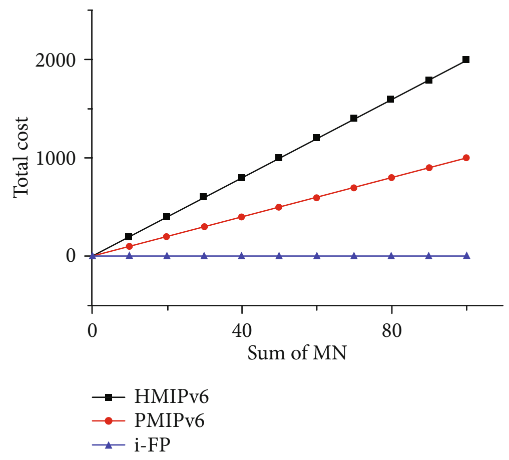
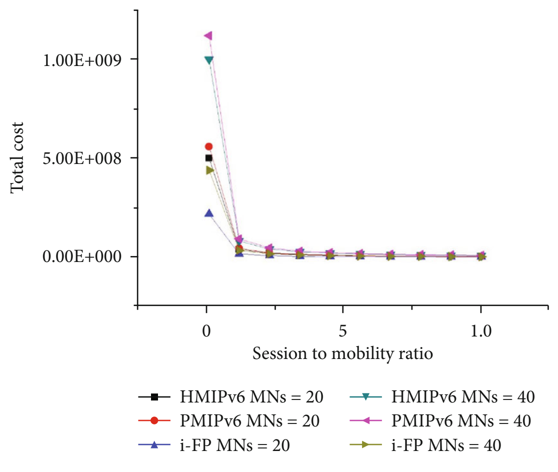


Figure 13: Traffic overhead.

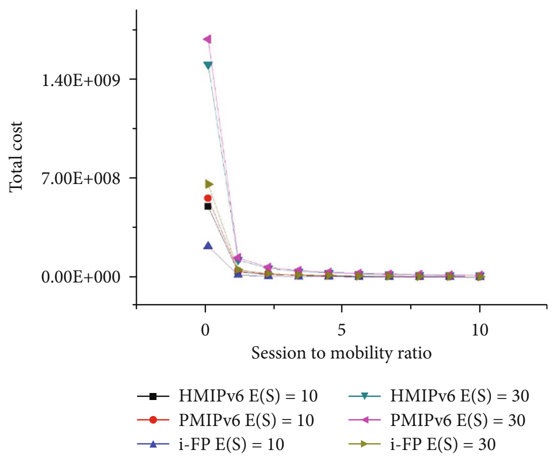
Figure 11 shows the total signal cost as MN increases. As a number of MNs initiated by handover increase, all three protocols increase the signal cost. However, i-FP increases to a slower slope than PMIPv6 and HMIPv6. Therefore, i-FP is more cost-effective because of its lower signal cost than PMIPv6 and HMIPv6.

The performance of the handover delay for each protocol is evaluated according to the total cost of the MN. The larger the handover delay, the greater the packet loss in the handover procedure. The total cost until packet reception is complete is used to measure the handover delay of the three protocols. In Figure 12, we can see that the latency of PMIPv6 and HMIPv6 is more than doubled as MN increases compared to the incidence cost of i-FP.

To measure the total cost of traffic overhead for each technique, we evaluated a number of MNs performing handover as variables. From the traffic overhead evaluation, we see



(a) A type



(b) B type

Figure 14: Total cost by SMR.

that i-FP is maintained regardless of the number of MNs performing handovers, because it uses the IP swapping mechanism without IP tunneling, and HMIPv6 and PMIPv6 perform IP tunneling. Therefore, it can be confirmed that the overhead increases as a number of MNs performing handover increase. Since a number of MNs increase compared to those of HMIPv6, PMIPv6 increases rapidly. The contents are shown in Figure 13.

Session-to-mobility ratio (SMR) was utilized to evaluate the total cost of the network. SMR is calculated by dividing the session arrival rate by the head-off rate. We classified SMR into two types, increasing the accuracy of the evaluation. If the value of SMR is large, the session activity is higher than the hand off speed. It is shown that i-FP has had little impact on the increase in SMR compared to other networks and has the lowest total cost. This is shown in Figure 14.

5. Conclusion

We proposed a new i-FP based on the features of HMIPv6 and PMIPv6 networks. We also address the cost issues arising from network architectures for MNs used in smart factory environments. Based on the proposed i-FP, HMIPv6, and PMIPv6 modeling, we analyzed and evaluated the network cost minimization. Comparing the cost and traffic overhead of packet data transmission, we demonstrate that i-FP, which appears to be the lowest in local units, is the enhanced technique. Therefore, we confirm that i-FP is the most suitable mobile network protocol framework for application in smart factory environments due to low data loss and low latency. Furthermore, through an optimization system of cross-domain handover via edge computing, the cost compared to existing techniques is also relatively low, which increases satisfaction. This can be the basis for judging that i-FP is the best solution for local mobile network environments in smart factory environments. Adding to the analysis of the cloud environment, the following work envisions additional research to build an integrated system to analyze performance compared to existing technologies.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request (jpjeong@skku.edu).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Industrial Cluster Program funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) and the Korea Industrial Complex Corporation (Project Number SKN19ED). And also, this research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Creative Consilience program (IITP-2020-0-01821) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

[1] E. A. Lee, “Cyber physical systems: design challenges,” in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 363–369, Orlando, FL, USA, 2008.

[2] J. Villalba-Diez and X. Zheng, “Quantum strategic organizational design: alignment in industry 4.0 complex networked cyber-physical lean management systems,” *Sensors*, vol. 20, no. 20, p. 5856, 2020.

[3] S. Praptodiyono, T. Firmansyah, M. Alaydrus, M. I. Santoso, A. Osman, and R. Abdullah, “Mobile IPv6 vertical handover specifications, threats, and mitigation methods: a survey,” *Security and Communication Networks*, vol. 2020, 18 pages, 2020.

[4] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge intelligence: paving the last mile of artificial intelligence with edge computing,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[5] A. Rahman, J. Jin, A. L. Cricenti, A. Rahman, and A. Kulkarni, "Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2500–2511, 2018.

[6] A. Radziwon, A. Bilberg, M. Bogers, and E. S. Madsen, "The smart factory: exploring adaptive and flexible manufacturing solutions," *Procedia Engineering*, vol. 69, pp. 1184–1190, 2014.

[7] D. Saha, A. Mukherjee, I. S. Misra, and M. Chakraborty, "Mobility support in IP: a survey of related protocols," *IEEE Network*, vol. 18, no. 6, pp. 34–40, 2004.

[8] J. Kempf, "Problem statement for network-based localized mobility management (NETLMM)," Tech. rep., RFC 4830, 2007.

[9] J. Kempf, "Goals for network-based localized mobility management (NETLMM)," Tech. rep., RFC 4831, 2007.

[10] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," 1998.

[11] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in IPv6," 2004.

[12] A. S. Ahmed, R. Hassan, and N. E. Othman, "IPv6 neighbor discovery protocol specifications, threats and countermeasures: a survey," *IEEE Access*, vol. 5, pp. 18187–18210, 2017.

[13] D. K. Oh and S. W. Min, "A fast handover scheme of multicast traffics m PMIPv6," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 36, no. 3B, pp. 208–213, 2011.

[14] H. Yokota, K. Chowdhury, R. Koodli, B. Patil, and F. Xia, "Fast handovers for proxy mobile IPv6," *RFC 5949*, 2010.

[15] C. Castelluccia, "HMIPv6," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 4, no. 1, pp. 48–59, 2000.

[16] S. Das, A. Misra, and P. Agrawal, "TeleMIP: telecommunications-enhanced mobile IP architecture for fast intradomain mobility," *IEEE Personal Communications*, vol. 7, no. 4, pp. 50–58, 2000.

[17] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, *Host Identity Protocol*, Tech. rep., RFC 5201, 2008.

[18] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, *Proxy Mobile ipv6*, 2008.

[19] J. Lei and X. Fu, "Evaluating the benefits of introducing PMIPv6 for localized mobility management," in *2008 International Wireless Communications and Mobile Computing Conference*, pp. 74–80, Crete, Greece, 2008.

[20] H. Soliman, C. Castelluccia, K. Elmalki, and L. Bellier, "Hierarchical mobile IPv6 mobility management (HMIPv6)," *RFC 4140*, p. 5380, 2008.

[21] J. Kim and J. Jeong, "Design and performance analysis of an industrial IoT-based mobility management for smart manufacturing," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0471–0476, Vancouver, BC, Canada, 2019.

[22] S. H. La, J. Jeong, J. Koo, and U. M. Kim, "On intelligent hierarchical F-PMIPv6 based mobility support for industrial mobile networks," *Procedia Computer Science*, vol. 155, pp. 169–176, 2019.

[23] D. G. Park, J. Lee, J. W. Oh, and J. Jeong, "A novel SDN-based cross handoff scheme in industrial mobile networks," *Procedia Computer Science*, vol. 155, pp. 642–647, 2019.

[24] J. A. Kim, D. G. Park, and J. Jeong, "Design and performance evaluation of cost-effective function-distributed mobility management scheme for software-defined smart factory networking," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 6, pp. 2291–2307, 2020.

[25] I. Heritage, "Protecting industry 4.0: challenges and solutions as IT, OT and IP converge," *Network Security*, vol. 2019, no. 10, pp. 6–9, 2019.

[26] M. Gohar, S. Anwar, M. Ali, J. G. Choi, H. Alquhayz, and S. J. Koh, "Partial bicasting with buffering for proxy mobile IPv6 mobility management in CoAP-based IoT networks," *Electronics*, vol. 9, no. 4, p. 598, 2020.

[27] A. Hussain, S. Nazir, F. Khan et al., "A resource efficient hybrid proxy mobile IPv6 extension for next generation IoT networks," *IEEE Internet of Things Journal*, 2021.

[28] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

[29] M. M. Mabkhot, A. M. Al-Ahmari, B. Salah, and H. Alkhalefah, "Requirements of the smart factory system: a survey and perspective," *Machines*, vol. 6, no. 2, p. 23, 2018.

[30] Y. Yuan, L. Qian, G. Jia, L. Yu, Z. Yu, and Q. Zhao, "Efficient computation offloading for service workflow of mobile applications in mobile edge computing," *Mobile Information Systems*, vol. 2021, p. 11, 2021.

[31] F. Sufyan and A. Banerjee, "Computation offloading for distributed mobile edge computing network: a multiobjective approach," *IEEE Access*, vol. 8, pp. 149915–149930, 2020.

[32] H. Wu, Y. Yan, D. Sun, H. Wu, and P. Liu, "Multi buffers multi objects optimal matching scheme for edge devices in IIoT," *IEEE Internet of Things Journal*, p. 1, 2021.

[33] Lanner America, Intelligent Edgehttps://www.lanner-america.com/knowledgebase/intelligent-edge/.

[34] A. G. Valk'o, "Cellular IP: a new approach to Internet host mobility," *ACM SIGCOMM computer communication review*, vol. 29, no. 1, pp. 50–65, 1999.

[35] R. Ramjee, T. La Porta, S. Thuel, K. Varadhan, and S. Y. Wang, "HAWAII: a domain-based approach for supporting mobility in wide-area wireless networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 396–410, 2002.

[36] A. Romanow, J. Mogul, T. Talpey, and S. Bailey, "Remote direct memory access (RDMA) over IP problem statement," *RFC 4290*, 2005.

[37] T. M. Lim, C. K. Yeo, F. B. S. Lee, and Q. V. Le, "TMSP: terminal mobility support protocol," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 849–863, 2008.

[38] S. Thomson, T. Narten, and T. Jinmei, "IPv6 stateless address autoconfiguration," *RFC 4862*, 2004.

[39] Y. Gvvon, J. Kempf, and A. Yegin, "Scalability and robustness analysis of mobile IPv6, fast mobile IPv6, hierarchical mobile IPv6, and hybrid IPv6 mobility protocols using a large-scale simulation," in *2004 IEEE International Conference on Communications*, vol. 7, pp. 4087–4091, Paris, France, 2004.

[40] X. Perez-Costa, M. Torrent-Moreno, and H. Hartenstein, "A performance comparison of mobile IPv6, hierarchical mobile IPv6, fast handovers for mobile IPv6 and their combination," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 4, pp. 5–19, 2003.

[41] G. Kim, "Low latency cross layer handover scheme in proxy mobile IPv6 domain," in *International Conference on Next Generation Wired/Wireless Networking*, pp. 110–121, Berlin, Heidelberg, 2008.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5598173, 6 pages
https://doi.org/10.1155/2021/5598173

Research Article

On Solving the Decycling Problem in a Torus Network

Antoine Bossard

Graduate School of Science, Kanagawa University, Tsuchiya 2946, Hiratsuka, Kanagawa, Japan 259-1293

Correspondence should be addressed to Antoine Bossard; abossard@kanagawa-u.ac.jp

Received 26 February 2021; Accepted 30 March 2021; Published 12 April 2021

Academic Editor: Yong Zhang

Modern supercomputers are massively parallel systems: they embody thousands of computing nodes and sometimes several millions. The torus topology has proven very popular for the interconnect of these high-performance systems. Notably, this network topology is employed by the supercomputer ranked number one in the world as of November 2020, the supercomputer Fugaku. Given the high number of compute nodes in such systems, efficient parallel processing is critical to maximise the computing performance. It is well known that cycles harm the parallel processing capacity of systems: for instance, deadlocks and starvations are two notorious issues of parallel computing that are directly linked to the presence of cycles. Hence, network decycling is an important issue, and it has been extensively discussed in the literature. We describe in this paper a decycling algorithm for the 3-dimensional k-ary torus topology and compare it with established results, both theoretically and experimentally. (This paper is a revised version of Antoine Bossard (2020)).

1. Introduction

The supercomputers of the 21st century are massively parallel systems: they embody thousands of compute nodes. Some recent devices even include several millions of nodes (e.g., 10,649,600 nodes for the Sunway TaihuLight as of November 2020's TOP500 list [1]). The interconnection of these compute nodes is thus a critical issue so as to maximise the parallel processing performance and thus the machine performance overall. Thanks to its advantageous topological properties, such as regularity, the torus network topology has proven very popular for the interconnect of modern supercomputers. For example, the supercomputer ranked number one in the world in the November 2020 TOP500 ranking, the supercomputer Fugaku built by Fujitsu and RIKEN, employs the torus topology to connect its nodes (Tofu interconnect D [2]). The IBM Blue Gene/L and Blue Gene/P, Cray Titan (Gemini interconnect [3]), and Fujitsu SORA-MA (Tofu interconnect 2 [4]) are other examples of supercomputers based on the torus topology.

It is well known that parallel processing is harmed by the presence of cycles: they are at the source of the deadlock, livelock, and starvation notorious resource allocation issues [5]. Notably, because it has important implications for parallel processing, the decycling problem, also known as the minimum feedback vertex set problem, has been extensively addressed in the literature. Karp has shown that finding a decycling set of minimum size (i.e., an optimal decycling set) in any graph is NP-complete [6]. For instance, Fomin et al. have described an algorithm that solves this problem in any graph in $O(1.7548^n)$ time [7]. Furthermore, polynomial solutions have been described for several particular classes of graphs such as 3-regular graphs [8], convex bipartite graphs [9], permutation graphs [10], and hypercube-based networks [11, 12]. Among others, the size of an optimal decycling set (i.e., the decycling number) in the case of cubes and grids has been discussed in [13, 14] and for hypercubes in [15]. We describe in this paper a polynomial time decycling algorithm for a 3-dimensional k-ary torus network. One should note that while the case of a grid as mentioned above seems close, or at least related, to the case of the torus which we investigate hereinafter, the wrap-around edges of the torus invalidate the grid decycling approach (refer to the next section for additional details).

The rest of this paper is organised as follows. Notations, definitions, and previous results are recalled in Section 2. The decycling algorithm is presented in Section 3, including the proof of its correctness and complexity analysis.

Theoretical and empirical evaluations are conducted in Section 4. Finally, concluding remarks are given in Section 5.

2. Preliminaries

We recall in this section several notations, definitions, and previously established results. The set of the vertices of a graph G is denoted by $V(G)$, and the set of its edges by $E(G)$. A path in a graph G is a subgraph of G that is an alternating sequence of distinct vertices and edges. Such a vertex–edge sequence but whose two terminal vertices are the same vertex is called a cycle. The length of a path or cycle is its number of edges. A graph that contains no cycle is said to be acyclic and is isomorphic to a tree.

Definition 1. An n-dimensional k-ary torus, denoted by $T(n, k)$, with $n \geq 1$ and $k \geq 1$, consists in the k^n vertices induced by the set $\{0, 1, \cdots, k-1\}^n$. Two vertices $u = (u_0, u_1, \cdots, u_{n-1})$ and $v = (v_0, v_1, \cdots, v_{n-1})$ of a $T(n, k)$ are adjacent if and only if there exists j $(0 \leq j < n)$ such that $\forall i$ $(0 \leq i < n, i \neq j)$ $u_i = v_i$ and $u_j = v_j \pm 1 (\text{mod } k)$.

A torus $T(n, k)$ is thus a regular graph of degree $2n$, of diameter $n\lfloor k/2 \rfloor$ and has nk^n edges. An essential torus property is next recalled.

Property 2. For a dimension δ $(0 \leq \delta < n)$, a $T(n, k)$ consists in k subtori $T^{i,\delta}(n-1, k)$ $(0 \leq i < k)$. Each subtorus $T^{i,\delta}(n-1, k)$ is induced by the k^{n-1} vertices $(u_0, u_1, \cdots, u_{\delta-1}, i, u_{\delta+1}, \cdots, u_{n-1})$ of $T(n, k)$ with $u_j (0 \leq j < n, j \neq \delta)$ the vertex coordinate for the dimension j and i the vertex coordinate for the dimension δ.

A torus $T(2, 3)$ is shown in Figure 1(a) and its recursive structure is illustrated in Figure 1(b).

Next, previously established results are recalled.

Beineke and Vandell have established a lower bound on the size of a decycling set for any graph [16]. This result is recalled in Theorem 3 below.

Theorem 3 (see [16]). *Given a graph $G = (V, E)$ of maximum degree Δ, any decycling set S of G satisfies the following relation:*

$$|S| \geq \left\lceil \frac{|E| - |V| + 1}{\Delta - 1} \right\rceil. \quad (1)$$

We were unaware until very recently (after the publication of [17]) that Pike and Zou have shown how to calculate a decycling set of minimum size for a 2-dimensional torus in [18]. The corresponding result is recalled in Theorem 4 below.

Theorem 4 (see [18]). *In a $T(2, k)$ with $k \geq 3$, a decycling set S^k of minimum size with*

$$\left|S^k\right| = \begin{cases} \dfrac{3k}{2} = 6, & k = 4, \\ \left\lceil \dfrac{k^2 + 2}{3} \right\rceil, & \textit{otherwise}, \end{cases} \quad (2)$$

can be found in $O(k^2)$ time.

3. The Case of a $T(3, k)$

We describe in this section the details of our approach to decycle a 3-dimensional k-ary torus $T(3, k)$.

3.1. Algorithm Description. We give below a constructive proof in the form of a decycling algorithm whose input is an arity k $(k \geq 1)$ and which outputs a decycling set S^k.

The main idea is to consider one dimension δ to reduce $T(3, k)$ into k 2-dimensional subtori as per Property 2 and to alternate for each such subtorus the optimal decycling of a $T(2, k)$ (i.e., Theorem 4) and two other decycling methods of a $T(2, k)$ which induce a graph with no edge.

The case $k = 1$ is trivial: $T(3, 1)$ consists of one unique vertex and is thus acyclic; $S^1 = \varnothing$ is thus a decycling set for this trivial graph. A $T(3, 2)$ is isomorphic to a 3-dimensional hypercube; $S^2 = \{(0, 0, 0), (1, 1, 0), (1, 1, 1)\}$ is thus an optimal decycling set by Theorem 3. Hence, we can now assume that $k \geq 3$.

Step 1. We distinguish the two cases k even and k odd.

Case k even:

Define two decycling sets S_1^k, S_2^k in a two-dimensional k-ary torus $T(2, k)$ as follows:

$$S_1^k = \{(i, j) \in V(T(2, k)) | 0 \leq i, j \leq k - 1, i + j \equiv 1 (\text{mod } 2)\},$$
$$S_2^k = \{(i, j) \in V(T(2, k)) | 0 \leq i, j \leq k - 1, i + j \equiv 0 (\text{mod } 2)\}. \quad (3)$$

In other words, the set S_1^k is induced by the vertices of $T(2, k)$ that are taken in one particular "quincunx" manner, and the set S_2^k by the vertices of $T(2, k)$ that are taken in the other "quincunx" manner. Precisely, we have $S_2^k = V(T(2, k)) \setminus S_1^k$ and $S_1^k = V(T(2, k)) \setminus S_2^k$.

The sets S_1^k and S_2^k when $k = 4$ are illustrated in Figures 2(a) and 2(b), respectively; they consist in the red vertices.

Case k odd:

Define two decycling sets S_1^k, S_2^k in a two-dimensional k-ary torus $T(2, k)$ as follows:

$$S_1^k = \{(i, j) \in V(T(2, k)) | 0 \leq i, j \leq k - 1, i + j \equiv 1 (\text{mod } 2)\} \cup \{(i, k-1) | 0 \leq i \leq k - 1\} \cup \{(k-1, i) | 0 \leq i \leq k - 1\},$$

$$S_2^k = \{(i, j) \in V(T(2, k)) | 0 \leq i, j \leq k - 1, i + j \equiv 0 (\text{mod } 2)\} \cup \{(i, k-1) | 0 \leq i \leq k - 1\} \cup \{(k-1, i) | 0 \leq i \leq k - 1\}. \quad (4)$$

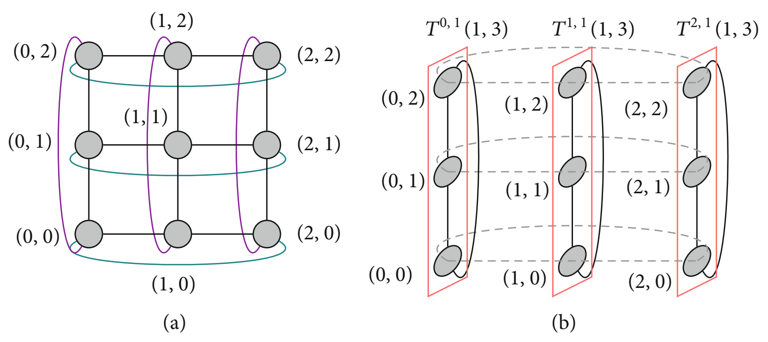


FIGURE 1: (a) A torus $T(2,3)$. (b) A torus has a recursive structure: a $T(2,3)$ consists in three subtori $T(1,3)$.

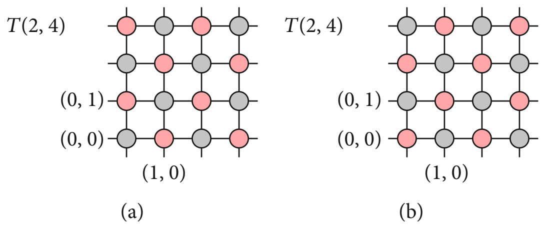


FIGURE 2: The sets S_1^k (a) and S_2^k (b) as defined in a $T(2,k)$ when $k=4$; they consist in the red vertices.

In other words, the set S_1^k is induced by the vertices of $T(2,k)$ that are taken in one particular "quincunx" manner and also includes the vertices of the top row and those of the right row. The sets S_1^k and S_2^k when $k=5$ are illustrated in Figures 3(a) and 3(b), respectively; they consist in the red vertices.

Step 2. Let $\delta=0$ and consider the k subtori $T^{i,\delta}(2,k)$ $(0\le i\le k-1)$ as per Property 2. Define S_0^k the optimal decycling set of $T(2,k)$ as induced by Theorem 4. We distinguish the three cases that are induced by the value of $k \bmod 3$.

Case $k\equiv 0 (\mathrm{mod}\ 3)$:

Decycle each subtorus $T^{i,\delta}(2,k)(0\le i\le k-1)$ with the vertex set $S^k_{i \bmod 3}$ (see Figure 4). In this figure, where there can be edges on the third dimension between two subtori, sample edges are shown.

Case $k\equiv 1 (\mathrm{mod}\ 3)$:

Decycle each subtorus $T^{i,\delta}(2,k)(0\le i\le k-2)$ with the vertex set $S^k_{i \bmod 3}$ and $T^{k-1,\delta}(2,k)$ with the vertex set S_2^k (see Figure 5). Again in this figure, where there can be edges on the third dimension between two subtori, sample edges are shown.

Case $k\equiv 2 (\mathrm{mod}\ 3)$:

Decycle each subtorus $T^{i,\delta}(2,k)(0\le i\le k-2)$ with the vertex set $S^k_{i \bmod 3}$ and $T^{k-1,\delta}(2,k)$ with the vertex set $V(T^{k-1,\delta}(2,k))$ (see Figure 6). Again in this figure, where there can be edges on the third dimension between two subtori, sample edges are shown.

3.2. Correctness and Complexities

3.2. Correctness and Complexities. In this section, we prove the correctness of the proposed algorithm and establish its complexities.

Theorem 5. *In a 3-dimensional k-ary torus $T(3,k)$ $(k\ge 1)$, a decycling set S^k of 0 vertex when $k=1$, 3 vertices when $k=2$, and in the other cases with*

$$\left|S^k\right| = \begin{cases} k\dfrac{\lceil (k^2+2)/3\rceil + k^2}{3}, & k\equiv 0 \,(\mathrm{mod}\ 3),\\ 30, & k\equiv 1\,(\mathrm{mod}\ 3), k=4,\\ (k-1)\dfrac{\lceil (k^2+2)/3\rceil + k^2}{3} + \dfrac{k^2}{2}, & k\equiv 1\,(\mathrm{mod}\ 3), k>4\\ (k+1)\dfrac{\lceil (k^2+2)/3\rceil + k^2}{3}, & k\equiv 2\,(\mathrm{mod}\ 3), \end{cases} \tag{5}$$

when k is even, and with

$$\left|S^k\right| = \begin{cases} k\dfrac{\lceil (k^2+2)/3\rceil + k^2+2k-1}{3}, & k\equiv 0 (\mathrm{mod}\ 3),\\ (k-1)\dfrac{\lceil (k^2+2)/3\rceil + k^2+2k-1}{3} + \dfrac{k^2-1}{2}+k, & k\equiv 1 (\mathrm{mod}\ 3),\\ (k-2)\dfrac{\lceil (k^2+2)/3\rceil + k^2+2k-1}{3} + \dfrac{3k^2-1}{2}+k, & k\equiv 2 (\mathrm{mod}\ 3), \end{cases} \tag{6}$$

when k is odd can be found in optimal $O(k^3)$ time.

Proof. The cases induced by $k\le 2$ are trivial; they have already been shown at the beginning of Section 3.1. So, we can assume $k\ge 3$.

By definition, the subgraph of $T(2,k)$ induced by the vertices of the set $V(T(2,k))\setminus S_1^k$ has no edge. And similarly, the

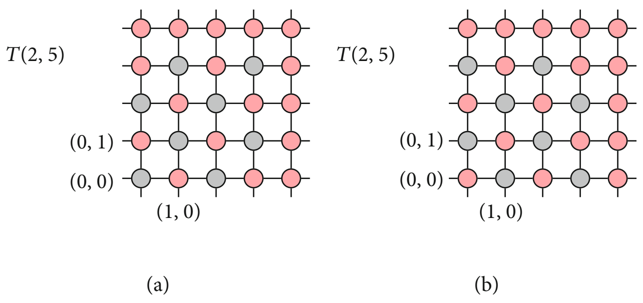


FIGURE 3: The sets S_1^k (a) and S_2^k (b) as defined in a $T(2,k)$ when $k = 5$; they consist in the red vertices.

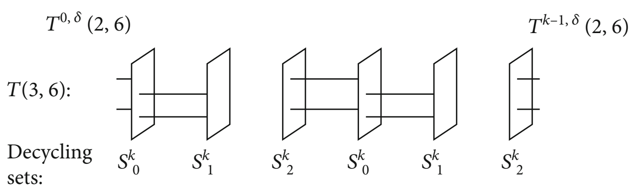


FIGURE 4: An illustration of the case $k \equiv 0 (\mathrm{mod}\ 3)$ with $k = 6$. The selected decycling set for each of the $T^{i,\delta}(2,k)$ $(0 \le i \le k-1)$ subtori is given below it. Where there can be edges on the third dimension between two subtori, sample edges are shown.

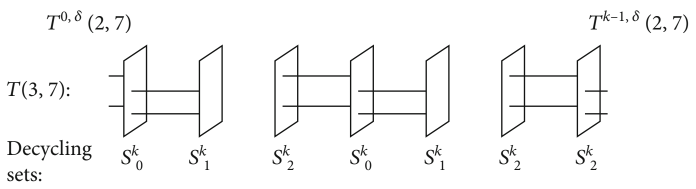


FIGURE 5: An illustration of the case $k \equiv 1 (\mathrm{mod}\ 3)$ with $k = 7$. The selected decycling set for each of the $T^{i,\delta}(2,k)$ $(0 \le i \le k-1)$ subtori is given below it. Where there can be edges on the third dimension between two subtori, sample edges are shown.

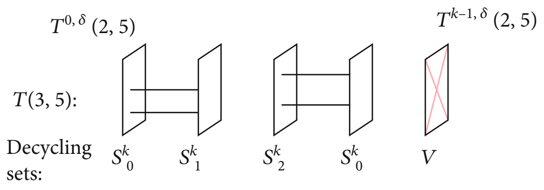


FIGURE 6: An illustration of the case $k \equiv 2 (\mathrm{mod}\ 3)$ with $k = 5$. The selected decycling set for each of the $T^{i,\delta}(2,k)$ $(0 \le i \le k-1)$ subtori is given below it. Where there can be edges on the third dimension between two subtori, sample edges are shown.

subgraph of $T(2,k)$ induced by the vertices of the set $V(T(2,k)) \setminus S_2^k$ has no edge either.

Hence, by definition of the algorithm, each subtorus $T^{i,\delta}(2,k)$ $(0 \le i \le k-1)$ is acyclic. Moreover, the only edges on the third dimension are at a vertex of a graph induced by a subtorus decycled with S_0^k. Consider two such graphs induced by a subtorus decycled with S_0^k, say the graphs that correspond to $T^{i,\delta}(2,k)$ and $T^{j,\delta}(2,k)$ where $i < j$ and $j - i$ minimal. In the case $k \equiv 0 (\mathrm{mod}\ 3)$, we have $j - i = 3$ and the greatest such j is $k - 3$. And since for every edge on the third dimension the vertex that is not in a graph induced by a subtorus decycled with S_0^k is inside a graph induced by a subtorus that has no edge, the resulting graph is acyclic. In the case $k \equiv 1 (\mathrm{mod}\ 3)$, we have $j - i = 3$ and the greatest such j is $k - 4$. So, again and for the same reason, the resulting graph is acyclic. In the case $k \equiv 2 (\mathrm{mod}\ 3)$, we have $j - i = 3$ and the greatest such j is $k - 2$, so there could be a path on the third dimension between a vertex of $T^{k-2,\delta}(2,k)$ (i.e., the rightmost graph induced by S_0^k) and of $T^{0,\delta}(2,k)$ (which is also induced by S_0^k). However, all the vertices of $T^{k-1,\delta}(2,k)$ are removed, so there is no such path and the resulting graph is thus acyclic.

The set S_1^k and the set S_2^k each have $k^2/2$ vertices when k is even and $(k-1)^2/2 + (2k-1) = (k^2-1)/2 + k$ when k is odd. They can thus each be calculated in $O(k^2)$ time. By Theorem 4, the set S_0^k has $3k/2$ vertices when $k = 4$ and $\lceil (k^2+2)/3 \rceil$ vertices otherwise, and this set can be calculated in $O(k^2)$ time.

Hence, in a $T(3,k)$, a decycling set S^k with $|S^k| = k(|S_0^k| + |S_1^k| + |S_2^k|)/3$ when $k \equiv 0 (\mathrm{mod}\ 3)$, $|S^k| = (k-1)(|S_0^k| + |S_1^k| + |S_2^k|)/3 + |S_2^k|$ when $k \equiv 1 (\mathrm{mod}\ 3)$, and $|S^k| = (k-2)(|S_0^k| + |S_1^k| + |S_2^k|)/3 + |S_0^k| + k^2$ when $k \equiv 2 (\mathrm{mod}\ 3)$ can be found in $O(k^3)$ time. By further distinguishing the two cases k even and k odd, we obtain the expected set sizes.

From Theorem 3, we have in a $T(3,k)$ that $|S| \geq (3k^3 - k^3 + 1)/5 = (2k^3 + 1)/5$. Therefore, $O(k^3)$ is optimal time.

4. Discussion

In this section, we discuss the obtained theoretical results and compare them with experimental data.

4.1. Comparison with the Lower Bound. We investigate in this section how close the size of the generated decycling set is to the lower bound given by Theorem 3. Figure 7 shows the values obtained from Theorem 3 and Theorem 5. Let us recall that the result of Theorem 3 is a lower bound on the size of a decycling set, and not necessarily the size of an optimal decycling set. So, the difference plotted in Figure 7 is given for reference, and it shows that the size of the obtained decycling set is promising, possibly optimal in some cases, given that it is rather close, and sometimes equal, to the lower bound of Theorem 3.

One can also notice that the size of the decycling set generated by the proposed algorithm is never smaller than the lower bound of Theorem 3. If that were the case, this would indicate a hole in the proposed algorithm.

4.2. Comparison with the Results of a Computer Experiment. We have implemented a stochastic decycling algorithm in order to compare the obtained theoretical results with those obtained experimentally. As recalled in introduction, this is an NP-complete problem; hence, the graph decycling implementation we use only approximates the size of an optimal decycling set. This implementation follows the method described in [19]. The stochastic implementation was run 1,000 times.

The values obtained from Theorem 5 and the computer experiment are shown in Table 1. The minimum size and the average size of the decycling set generated by the stochastic implementation are given. The values from Theorem 3—a lower bound on the size of a decycling set, and not necessarily the size of an optimal decycling set—are also given for reference.

From these results, it can be noticed that the proposed algorithm beats the stochastic implementation on each of all its 1,000 runs at $k \equiv 2 (\mathrm{mod}\ 4)$ and is equal or very nearly equal to it at $k \equiv 0 (\mathrm{mod}\ 4)$. In other words, when k is even, our proposal induces a smaller or nearly equal decycling set than the best decycling set found after 1,000 runs. And when k is odd, as k increases, the size difference between our proposal and the stochastic implementation continuously decreases, and at $k = 9$, our proposal beats the stochastic implementation when considering the average value of its 1,000 runs.

And, of course, the complexity of the stochastic implementation is always prohibitive [19], especially compared to the worst-case time complexity of the proposal ($O(k^3)$, see Theorem 5). These are very positive results which quantitatively show the significance of the proposal. For reference, in the case $k = 10$, the stochastic implementation took more than 2.5 hours to complete the 1,000 runs on a midrange computer (Intel Core i5-1035G7 CPU, 8 GB RAM).

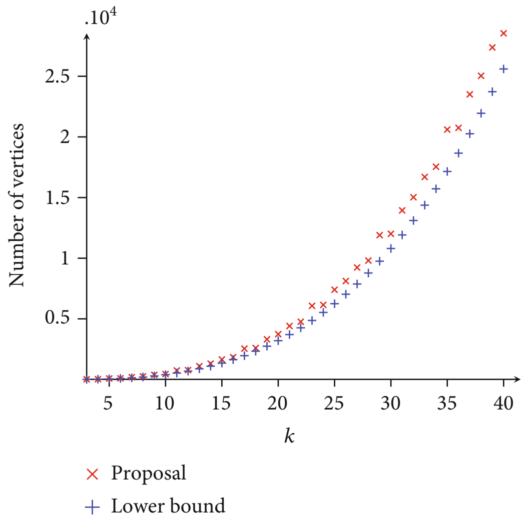


FIGURE 7: Investigating how close the size of the decycling set generated is to the lower bound of Theorem 3.

TABLE 1: Comparison of the size of the decycling set obtained with the proposed method in a $T(3,k)$ with that of the experimentally generated decycling sets. The theoretical lower bound is also given for reference.

Arity k	Proposal (Theorem 5)	Experiment (minimum size)	Experiment (average size)	Lower bound (Theorem 3)
3	18	14	14.71	11
4	30	30	33.85	26
5	85	60	65.86	51
6	98	106	113.85	87
7	189	171	180.46	138
8	258	257	269.15	205
9	378	371	383.21	292
10	452	505	526.01	401

Finally, it can also be noticed that the size of the decycling set generated by the proposed algorithm in a $T(3,k)$ is $O(k^n)$. Besides, by Theorem 4, the size of an optimal decycling set in a $T(2,k)$ is also $O(k^n)$. This is yet another positive indicator of the performance of our proposal.

5. Concluding Remarks

The torus topology is nowadays ubiquitous in supercomputing. It is the network topology of choice for the interconnect of massively parallel systems: it is for instance employed by the supercomputer ranked number one in the world as of November 2020, the supercomputer Fugaku. Besides, it is common knowledge that cycles in the network of compute nodes harm parallel processing, and this is one reason why the decycling problem—NP-complete—has been extensively addressed in the literature. We have described in this paper a decycling algorithm for a torus $T(3,k)$. Thanks to the recursive property of the torus topology, this proposal can be used to decycle parts (i.e., subtori) of a torus of higher dimension, which as explained will consequently facilitate parallel processing. Precisely, we have given a constructive proof of a decycling set S^k for a torus $T(3, k)$ where S^k has $k(|S_0^k| + 2|S_1^k|)/3$ vertices when $k \equiv 0 (\text{mod } 3)$, $(k-1)(|S_0^k| + 2|S_1^k|)/3 + |S_1^k|$ vertices when $k \equiv 1 (\text{mod } 3)$, $(k-2)(|S_0^k| + 2|S_1^k|)/3 + |S_0^k| + k^2$ vertices when $k \equiv 2 (\text{mod } 3)$ with S_0^k an optimal decycling set in $T(2, k)$, $|S_1^k| = k^2/2$ when k even, and $|S_1^k| = (k^2 - 1)/2 + k$ when k odd and can be obtained in $O(k^3)$ optimal time. We have formally evaluated the proposed algorithm and conducted evaluation experiments to compare it to conventional approaches. The obtained results have quantitatively shown the significance of the proposal.

Regarding future works, refining the proposed decycling algorithm so that the generated decycling set includes a smaller number of vertices is a first meaningful objective. Then, it will be very interesting to investigate, for instance, as explained above by using the recursive property of the torus topology, how to rely on the obtained results to produce nontrivial decycling sets for tori of higher dimensions.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author is sincerely grateful towards Professor David Pike (Memorial University of Newfoundland, NL, Canada) for communicating the existence of [18] after he noticed the publication of [17]. This research was partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under grant no. 19K11887.

References

[1] TOP500, *TOP500 list - November 2020, June 2020*February 2021, https://www.top500.org/lists/top500/list/2020/11/.

[2] Y. Ajima, T. Kawashima, T. Okamoto et al., “The Tofu interconnect D,” in *Proceedings of the IEEE International Conference on Cluster Computing*, pp. 646–654, Belfast, United Kingdom, September 2018.

[3] R. Alverson, D. Roweth, and L. Kaplan, “The Gemini system interconnect,” in *Proceedings of the 18th IEEE Symposium on High Performance Interconnects*, pp. 83–87, Mountain View, CA, USA, August 2010.

[4] Y. Ajima, T. Inoue, S. Hiramoto et al., “Tofu interconnect 2: system-on-chip integration of high-performance interconnect,” in *Proceedings of the 29th International Supercomputing Conference*, pp. 498–507, Leipzig, Germany, June 2014.

[5] P. Festa, P. M. Pardalos, and M. G. C. Resende, *Feedback Set Problems*, Springer US, Boston, MA, USA, 1999.

[6] R. M. Karp, “Reducibility among combinatorial problems,” in *Proceedings of the Symposium on Complexity of Computer Computations*, pp. 85–103, Yorktown Heights, NY, USA, March 1972.

[7] F. V. Fomin, S. Gaspers, and A. V. Pyatkin, “Finding a minimum feedback vertex set in time O(1.7548^n),” in *Proceedings of the International Workshop on Parameterized and Exact Computation*, vol. 4169, pp. 184–191, Zurich, Switzerland, September 2006.

[8] D. Li and Y. Liu, “A polynomial algorithm for finding the minimum feedback vertex set of a 3-regular simple graph,” *Acta Mathematica Scientia*, vol. 19, no. 4, pp. 375–381, 1999.

[9] Y. D. Liang and M.-S. Chang, “Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs,” *Acta Informatica*, vol. 34, no. 5, pp. 337–346, 1997.

[10] Y. Daniel Liang, “On the feedback vertex set problem in permutation graphs,” *Information Processing Letters*, vol. 52, no. 3, pp. 123–129, 1994.

[11] A. Bossard, “On the decycling problem in hierarchical hypercubes,” *Journal of Interconnection Networks*, vol. 14, no. 2, pp. 1350006.1–1350006.13, 2013.

[12] A. Bossard, “The decycling problem in hierarchical cubic networks,” *The Journal of Supercomputing*, vol. 69, no. 1, pp. 293–305, 2014.

[13] S. Bau, L. Beineke, Z. Liu, G.-M. Du, and R. Vandell, “Decycling cubes and grids,” *Utilitas Mathematica*, vol. 59, 2001.

[14] S. Bau and L. W. Beineke, “The decycling number of graphs,” *The Australasian Journal of Combinatorics*, vol. 25, pp. 285–298, 2002.

[15] D. A. Pike, “Decycling hypercubes,” *Graphs and Combinatorics*, vol. 19, no. 4, pp. 547–550, 2003.

[16] L. W. Beineke and R. C. Vandell, “Decycling graphs,” *Journal of Graph Theory*, vol. 25, no. 1, pp. 59–77, 1997.

[17] A. Bossard, “On the decycling problem in a torus,” in *Proceedings of the 11th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, vol. 1362, pp. 12–21, Shenzhen, China, December 2020.

[18] D. A. Pike and Y. Zou, “Decycling Cartesian products of two cycles,” *SIAM Journal on Discrete Mathematics*, vol. 19, no. 3, pp. 651–663, 2005.

[19] A. Bossard and K. Kaneko, “Approximating optimal decycling sets in HHC and HCN for small dimensions,” in *Proceedings of the 30th International Conference on Computers and Their Applications (CATA)*, pp. 145–150, Honolulu, HI, USA, March 2015.