［**Original Paper**］

Development of a Server Management System Incorporating a Peer-to-Peer Method for Constructing a High-availability Server System

Mitsuyoshi KITAMURA†, Yoshihisa UDAGAWA†, Hitoshi NAKAGOME‡, and Youta SHIMIZU‡

†Faculty of Engineering, Tokyo Polytechnic University
‡Graduate School of Engineering, Tokyo Polytechnic University

Abstract

In data download, a peer-to-peer (P2P) network system is superior to a server-client system with respect to the fault tolerance of the server. The present study proposes a server management system that incorporates the P2P method. The proposed system does not depend on the number of target servers for management, and uses a virtual server instead of a spare real server to recover the function of the troubled server. The following methods are introduced in order to construct the proposed system: (1) a dispersion defense method of a management program in consideration of the P2P method, (2) a server management priority to prevent a malfunction in the recovery processing, and (3) a dynamic management extension method to deal with problems that occur simultaneously on multiple servers. Furthermore, an experimental server system using the proposed method is constructed, and its architecture is described herein. Experiments to reproduce several types of trouble in the server system were conducted, and the recovery times of functions for the target server and a problem target server were measured and analyzed. The proposed system was found to be able to recover the server function even when several problems are generated simultaneously on a number of target servers.

1. Introduction

The spread of high-performance communication terminals and the high speed and broadband characteristics of access networks, such as fiber to the home (FTTH) and long-term evolution (LTE) networks, are rapidly advancing with the development of a highly information-oriented society. In addition, various Internet services are offered for the improvement of user services and have become indispensable in our daily lives. Therefore, in order to offer safer service, the role of a server system offering services has become increasingly important [1], and management, optimization, and construction of a server system capable of realizing stable operation have become necessary [2]. In the present study, we define a server in a server system that offers services to clients as a target server and a server that monitors and controls the target server as a management server. A specific management server manages multiple servers because general server management systems are constructed using server-client methods [3]. Thus, the load for management on the server increases in proportion to the number of target servers, and the problem of not being able to manage target servers occurs if the management server is abnormal. As such, the availability and reliability for the management server are crucial, and the server is constructed as a redundant system. Therefore, the complexity of the control and the increase in operative cost must be considered.

Various studies have been conducted on network systems that enable communication to servers even when a failure occurs. The design of a fault-tolerant network and its routing protocol [4], the development of high-availability network RRect for servers in a data center [5], a distributed virtual network architecture for grouping virtual machines into clusters of various service types [6], and the proposal of a network design method combining server arrangement and link

[Original Paper] Received 7 March 2017, revised 25 August 2017, accepted 26 October 2017.

protection in order to realize robustness network systems [7] have been reported.

In addition, a number of important studies related to failure countermeasures in data centers have been conducted. In these studies, an integrated method that combines the prediction of failure occurrence and failure location, enabling automated or operator actions [8]; integrated design considering both cloud service problems, such as a single link failure or service failure on the data center network, and placement of the data center network [9]; consideration of hardware trouble and service abnormality in relation to the security of a server and the network resources of a virtual data center [10]; a high-availability virtual infrastructure management framework considering the rate of problems encountered by a data center device [11]; and network coding to instantly start a hot spare node at the time of virtual machine trouble [12], were investigated.

The server system is the fundamental unit of a system providing service in a data center. Various ideas related to high availability in that system have been reported. A failover cluster system and a load balancer cluster system, which are commonly used to provide server system redundancy, can deal with server failures. However, there is a problem in that the cost becomes high because these cluster systems are basically applied to servers providing the same service. In the management system adopting the server-client method, demonstrating that a virtual server can be used for the recovery of server functions on a real server system and the realization of low cost and power savings by a multiple server backup system (MSBS) that can back up the function of several real servers by means of virtual servers [13], and the construction of a power-saving, high-availability server system by alternately operating a power-saving server system and a high-availability server system, which are able to operate independently [14], have been achieved.

Unlike the server-client method, a management system that does not require a specific management server has been reported. In a high-availability distributed database system used by multiple servers, a TCP ring to form connections that circulate all processes in one direction is adopted in order to immediately respond to process failures, and, regarding server monitoring, alive monitoring is performed randomly based on the target list [15]. In a highly available distributed cluster, the monitoring ring is configured taking into account the load difference due to monitoring at each server. As measures to address multiple failures, the detected fault information is gathered once to a specific server. Thereafter, the server informs all servers in the cluster [16]. Systems introducing a Peer-to-Peer (P2P) system have also been reported. Regarding a P2P-grid system that combines a P2P network system and a calculation grid system, a checkpoint and restart mechanism is adopted for the failure detection and recovery processing. The server performing the task periodically records the checkpoint in that system, and, if the server fails, the spare server takes over its role with reference to the record of the checkpoint [17]. In a system composed of a virtual server group providing services and a storage server group providing the data, an agent is executed on each server, and the P2P system, in which agents cooperate with each other, is operated. Multiple servers monitor target servers, and, regarding fault judgment, information is aggregated on a specific server and the server analyzes information. If the host server running virtual servers fails, the service is continued by moving the running virtual server to a spare host [18].

However, the management system incorporating the server-client method has the above-mentioned problem. Moreover, in a management system that does not require a specific management server, uneven use of network bandwidth caused by random selection of target servers; an increase in cost according to system size in order to prepare a spare server equivalent to a running server as measures at the time of failure; and, since fault information is aggregated in a specific server and a fault is judged, simultaneous faults (including the server making such judgments), must be considered.

Therefore, the present study proposes a server management system incorporating a P2P method with high fault tolerance in a server in the network system. Since the proposed system adopts the feature of the P2P system, the server providing services to clients has functions of both management and target servers. The proposed system is applicable to servers that provide various services and does not depend on the number of target servers for management. Each management server performs processing independently, from the monitoring of the target server to countermeasures at the time of failure. In addition, in order to recover the function of a troubled server, a virtual server is used instead of a spare real server. Since the virtual server is started on the target server providing services to clients, we adopt the virtual server

startup method considering the load state of the management server group. In order to construct the proposed system, a dispersion defense method of a management program in consideration of the P2P method, a server management priority to prevent a malfunction in recovery processing, and a dynamic management extension method to deal with problems that occur simultaneously on multiple servers are proposed and introduced. Furthermore, an experimental system using the proposed method is constructed. Experiments to reproduce several types of service program trouble and network trouble are conducted. The experiments examine the ability of the proposed system to recover to a normal condition from a condition in which trouble occurs, and the recovery times of the target server function and the problem target server are measured and analyzed.

The remainder of the present paper is organized as follows. Section 2 describes the construction of the proposed method, the operating procedure of an executer system to prevent the dispersion of the management program in consideration of the P2P method, setting files that are necessary for management, and the virtual server startup method considering the load state of the management server group. Section 3 describes the processing procedure of the management program for monitoring the target server, the detailed operation in the server management priority, the method for dealing with server trouble, and the features of the dynamic management extension method. Section 4 describes the construction and the specifications of the experimental system incorporating the proposed method and demonstrates that the proposed method can deal with various types of trouble, including the network and the service program on the target server. In addition, the recovery times required to deal with various types of trouble are measured and analyzed.

2. Outline of a server management system incorporating a P2P method

2.1. Server management method

Figure 1 shows the server-client method and the P2P method used for server management. In Fig. 1(a), one management server monitors a network and a condition offering services to clients on multiple target servers in the server-client method. The management server recovers the function for the target server by performing the recovery task depending on the type of trouble encountered if the management server detects trouble in the target server. In this method, a server administrator accesses one management server in order to update the management program or to examine the operating environment in the management server and the operating state for all target servers based on a recorded log file. However, a problem whereby the load of the management server increases in proportion to the number of target servers may occur because one management server monitors and controls multiple target servers. In addition, if the management server is abnormal, log files that record the operating state of target servers may disappear and it may not be possible to manage target servers. As such, the management server is constructed as a redundant system. Therefore, controlling the management server trouble may be complicated.

The P2P method does not have a specific management server, and the target server acts as a management server. In the proposed method, one management server monitors two target servers, as shown in Fig. 1(b). If the management server detects a problem in the target server, the management server monitoring the problem target server performs recovery of the function by performing the recovery job depending on the type of trouble encountered. The P2P method is superior to the server-client method with respect to the fault tolerance of the server. In addition, it is not necessary to consider the number of target servers. However, the P2P method has a problem in that the management program is installed on all management servers. Thus, the server administrator must access multiple management servers in order to update the management program or to examine the operating environment in the management server and the conditions for all target servers based on the recorded log file.

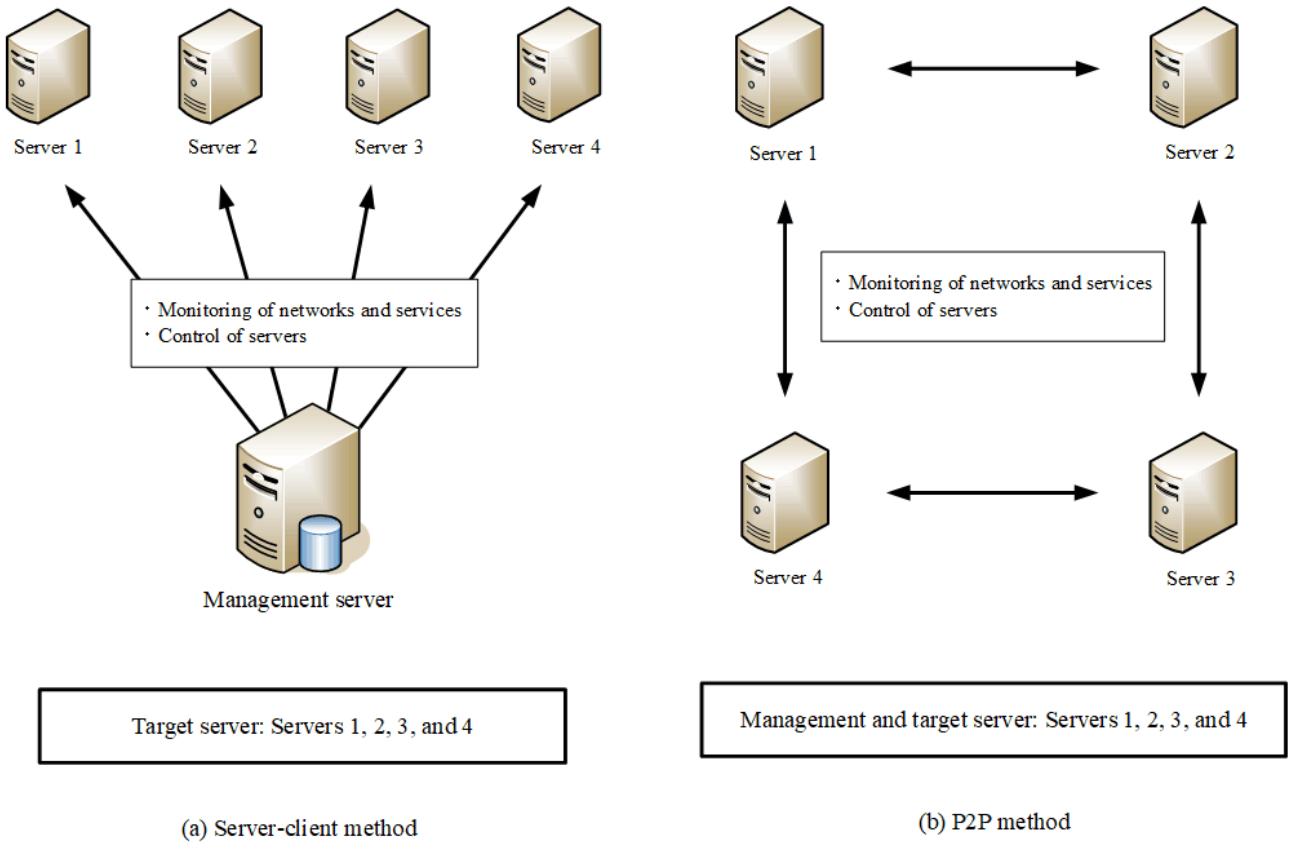


Fig. 1 Server management system constructed by server-client and P2P methods.

We compare the server-client method, which is the conventional method shown in Fig. 1, with the proposed P2P method. As the conventional method, MSBS shown in Reference 13 is used. In Tables 1 and 2, the proposed method is denoted as “P2P”, and the conventional method is denoted as “MSBS”. The proposed method and MSBS use a virtual server to recover the function of a troubled server. Here, since the conventional method requires a specific management server, the number of servers in the conventional method is one more than that in the proposed method.

Table 1 shows the number of target servers for management per management server depending on the number of target servers. In the case of three or more target servers, the proposed method manages only two target servers, whereas the number of target servers for management also increases according to the number of target servers in the conventional method. For this reason, an increase in the load for management on the management server and the network bandwidth of the server are of concern in the conventional method.

Table 1 Number of target servers per management server.

	Number of target servers					
	2	**3**	**4**	**・・・**	***n*-1**	***n***
P2P	1	2	2	・・・	2	2
MSBS	2	3	4	・・・	*n*-1	*n*

Table 2 shows the maximum startup number of virtual servers per management server depending on the number of troubled servers. Here, the total number of target servers is *n*. Since there are multiple management servers in the proposed method, for the case in which up to half of the target servers experience trouble, the maximum startup number of virtual servers per management server is one and increases in proportion to the number of troubled servers in the conventional method. In addition, even if more than half of the target servers fail, the maximum startup number of virtual servers per management server in the proposed method is smaller than that in the conventional method, except in the case of *n*-1 failures.

Table 2 Maximum startup number of virtual servers per management server.

	Number of troubled servers								
	1	**2**	**3**	· · ·	$\frac{n}{2}$	$\frac{n}{2}+1$	$\frac{n}{2}+2$	· · ·	***n*-1**
P2P	1	1	1	· · ·	1	2	3	· · ·	*n*-1
MSBS	1	2	3	· · ·	$\frac{n}{2}$	$\frac{n}{2}+1$	$\frac{n}{2}+2$	· · ·	*n*-1

2.2. Basic management configuration of P2P server management

As an example, Fig. 2 shows the basic management configuration in a general server system. The server system consists of two file servers, two DNS servers, and six servers belonging to the server group. The file and DNS servers are based on a redundant system. In the P2P method, six servers belonging to the server group are set as target servers offering services to clients and management servers monitoring the target servers. Servers 1 through 6 connect to the file server through the network file system (NFS) connection, and a shared area is allocated on each server. Each server offers services A through E to clients, and servers 3 and 4 offer the same services. The proposed system can be composed of servers providing different services to clients.

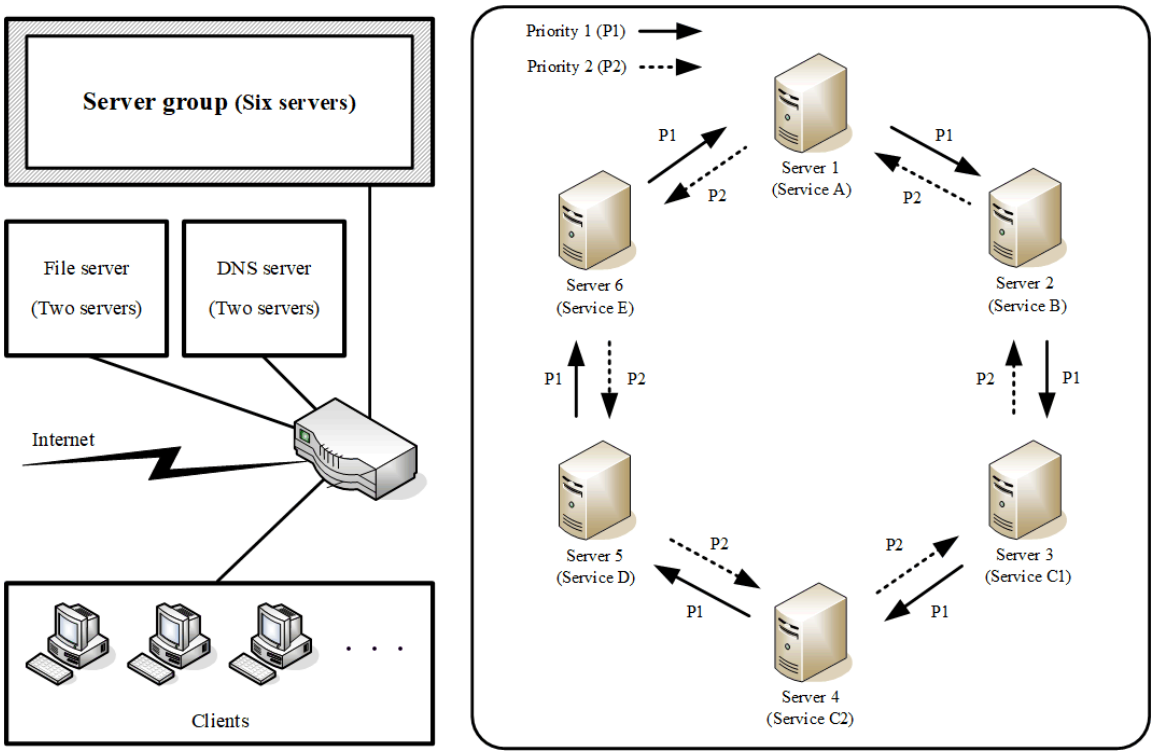


Fig. 2 Basic management configuration of P2P server management.

Two management programs, which have server management priority, are executed on each management server. As shown in the figure, Server 1 manages Server 2 with management priority 1 (P1) and manages Server 6 with management priority 2 (P2), and Server 2 is managed by Server 1 with P1 and by Server 3 with P2. The proposed method adopts management priority because two servers manage one server and this priority clarifies the priority of the recovery operation when trouble is detected in the target server.

2.3. Executer system

The P2P method has a problem regarding dispersion of the management program because it is necessary for the management program to be installed on all management servers. In the P2P method, an executer system is developed and operated in order to solve the problem. The construction and function of the executer system are shown in Fig. 3. An executer program is installed on all management servers, and all management servers are connected to a file server through NFS in order to realize centralized management of the management program, management data, and log files. The management server automatically executes the executer program at the start time of the management server. The executer program examines the existence of a flag file that is saved to a local disk on the server executing the executer program.

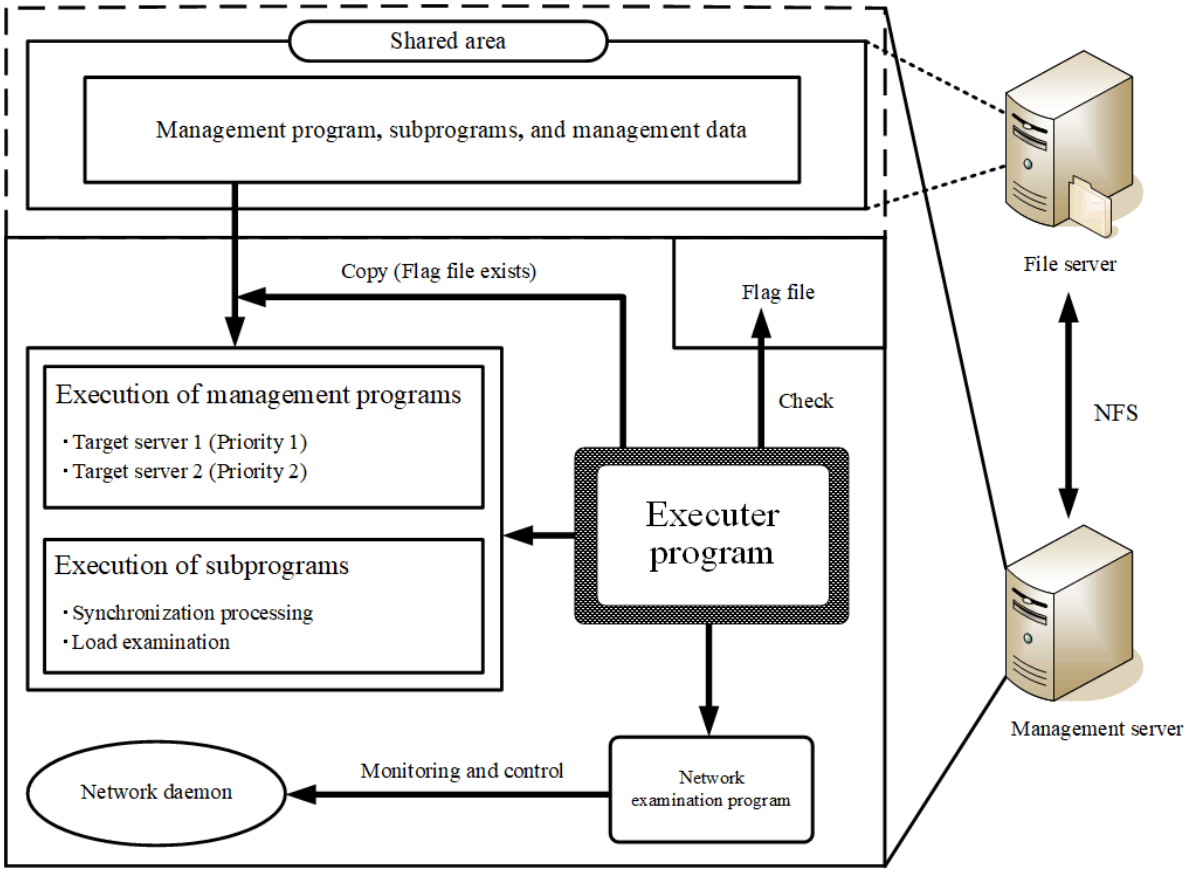


Fig. 3 Construction of the executer system.

If the flag file exists, the executer program copies the management program, subprograms, and management data, which are saved in the shared area, to the local disk and executes the management program and subprograms. The management program is executed in order to manage two target servers according to the server management priority. Regarding subprograms, a program for performing synchronization in management of the target server and a program for examining the load state of the management server are executed. If the flag file is deleted, the executer program stops all management programs and subprograms and stops the operation of the management server. The executer system can realize a function

for dealing with updating and modifying the management program by editing the program on the file server. Thus, the executer system can solve the problem of dispersing the management program.

In addition, the executer program executes a network examination program as a background job, and this program regularly monitors a network daemon, which is a program that provides a network service on the server. This monitoring function is added because the management server cannot perform recovery operation using the network if the network daemon on the target server is abnormal. If the network examination program detects network trouble on the server, the program executes recovery by restarting the daemon.

2.4. Management data

The management program is executed using the management data as an argument. The necessary information for management is recorded in the management data, as shown in detail in Fig. 4. A virtual IP address is recorded in the first line of the management data. The target server information is recorded from the second line to the sixth line. The virtual server information, which is used to recover a function if trouble occurs in the target server, is recorded in the seventh and eighth lines. In order to recover the function of the troubled server, the virtual IP address recorded in the first line of the data is set to the virtual server. Here, the name of the service daemon, which is a program that provides services to clients, is recorded in the fifth line. Since one server offers multiple services to clients, the names of multiple service daemons can be recorded using space separation, e.g., postfix dovecot. In the server type recorded in the sixth line, the server that mainly performs data transfer using the network is denoted by “NETWORK”, and the server that uses the CPU is denoted by “CPU”. Moreover, the server that performs both data transfer and uses CPU is denoted by “CPU_NET”. Here, management refers to monitoring the network and the state of the service program on the target server, recovering the server function by starting the virtual server, setting the virtual IP address to the virtual server, and recovering the target server by restarting the problem service program or the problem target server.

Management data	Mail.dat	Web.dat
1st line : **Virtual IP address (VIP)**	172.21.14.201	172.21.14.202
2nd line : **Host name**	Mail	Web
3rd line : **IP address (IPR)**	172.21.14.1	172.21.14.2
4th line : **Mac address**	00:00:00:00:00:01	00:00:00:00:00:02
5th line : **Service daemon**	postfix dovecot	httpd
6th line : **Server type**	NETWORK	CPU_NET
7th line : **Virtual server host name**	Mailb	Webb
8th line : **Virtual server IP address (IPV)**	172.21.14.101	172.21.14.102

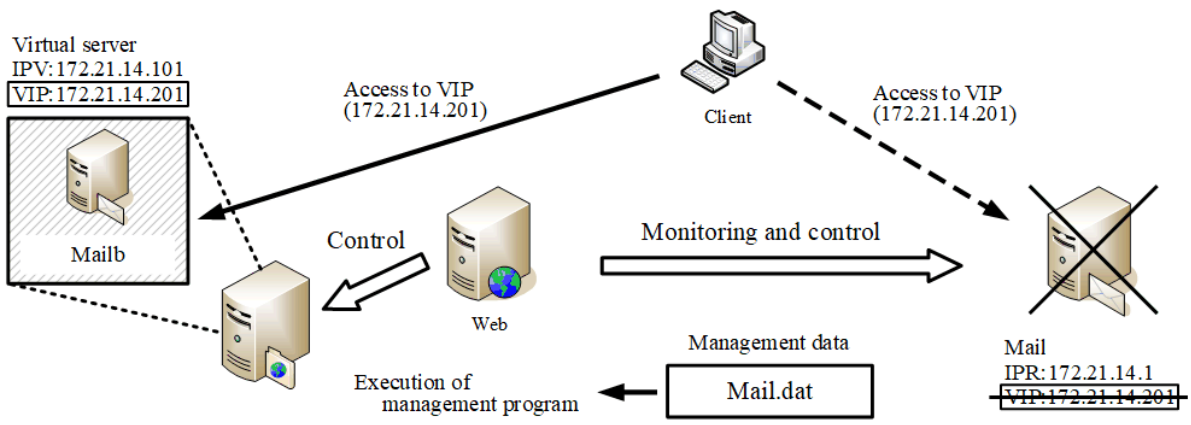


Fig. 4 Details of management data.

A mail server and a web server provide services to clients. If the target server host name is Mail, as shown in the figure, the management data becomes Mail.dat, and the management program on the web server, which is the management server, is executed using Mail.dat as an argument. The client accesses the virtual IP address "172.21.14.201" set in the mail server. If a problem is detected in the mail server, a mailb server, which is a virtual server having the function of the mail server, is started on the real server that the web server judged as optimal among all management servers. Then, the virtual IP address 172.21.14.201, which is recorded in the first line, is set to the mailb server by the management program. As a result, the function of the mail server is recovered because access from clients is provided by the virtual server. The management program need not be modified according to the target server by using the management data as an argument.

2.5. Virtual server startup method considering load state

Figure 5 shows a virtual server startup method in consideration of the load state of the management server group in order to recover the function of a troubled server. Each management server periodically performs measurements of server load by the subprogram shown in Fig. 3. Regarding the server load, we consider the memory, the CPU, and the network usage. Figure 5(a) shows the coefficients and calculation formulas required for calculating the predicted load. Here, *i* is the number of measurements, *mi* is the available memory size of the server, *ci* is the idle rate of the CPU, and *ni* is the total transfer bytes transmitted and received by the network. Based on these measured values, average load values *A*m, *A*c, and *A*n are calculated. From the average load value acquired in two stages in each management server, the predicted load is calculated based on the difference. The predicted load $L\mathrm{m}_n$ of the memory is defined as follows:

$$L\mathrm{m}_n\ [\mathrm{KB}] = 2 \times A\mathrm{m}_n - A\mathrm{m}_{n-1}. \tag{1}$$

The predicted load $L\mathrm{c}_n$ of the CPU is given as follows:

$$L\mathrm{c}_n\ [\%] = 2 \times A\mathrm{c}_n - A\mathrm{c}_{n-1}. \tag{2}$$

The predicted load $L\mathrm{n}_n$ of the network is defined as follows:

$$L\mathrm{n}_n\ [\mathrm{KB}] = 2 \times A\mathrm{n}_n - A\mathrm{n}_{n-1}. \tag{3}$$

Since the predicted loads obtained from equations (1), (2), and (3) may have negative values and exceed upper limits due to fluctuations, the minimum value is set to zero and the maximum value is such that *L*m is the physical memory size of the management server, *L*c is the number of CPU cores × 100, and *L*n is the rated transfer speed × 2. Since each management server has different resources, normalization is required for comparison. For the purpose of standardization, the following coefficients are required: $M_{\max}$ is the maximum available memory size among all management servers, $M_{\min}$ is the minimum usable memory size, C_{cor} is the number of CPU cores in each management server, $C_{\max}$ is the maximum value of $L\mathrm{c} \times C_{\mathrm{cor}}$ in each management server, $C_{\min}$ is the minimum value, N_{rat} is the total number of bytes transferred per second for rated transmission and reception of the network, $N_{\max}$ is the maximum value of N_{rat} - *L*n in each management server, and $N_{\min}$ is the minimum value. Here, if a router with 1,000 Mbps is taken as an example, since the rated transfer rate can be achieved by 128×10^3 KB in one second, N_{rat} is 256×10^3 KB from the total value of transmission and reception. The calculation method for normalizing the load values of the memory, the CPU, and the network is shown below. For normalization, the calculation formula differs depending on whether each maximum value and minimum value are different. The normalized load value $N\mathrm{m}_n$ of the memory in each management server is given by

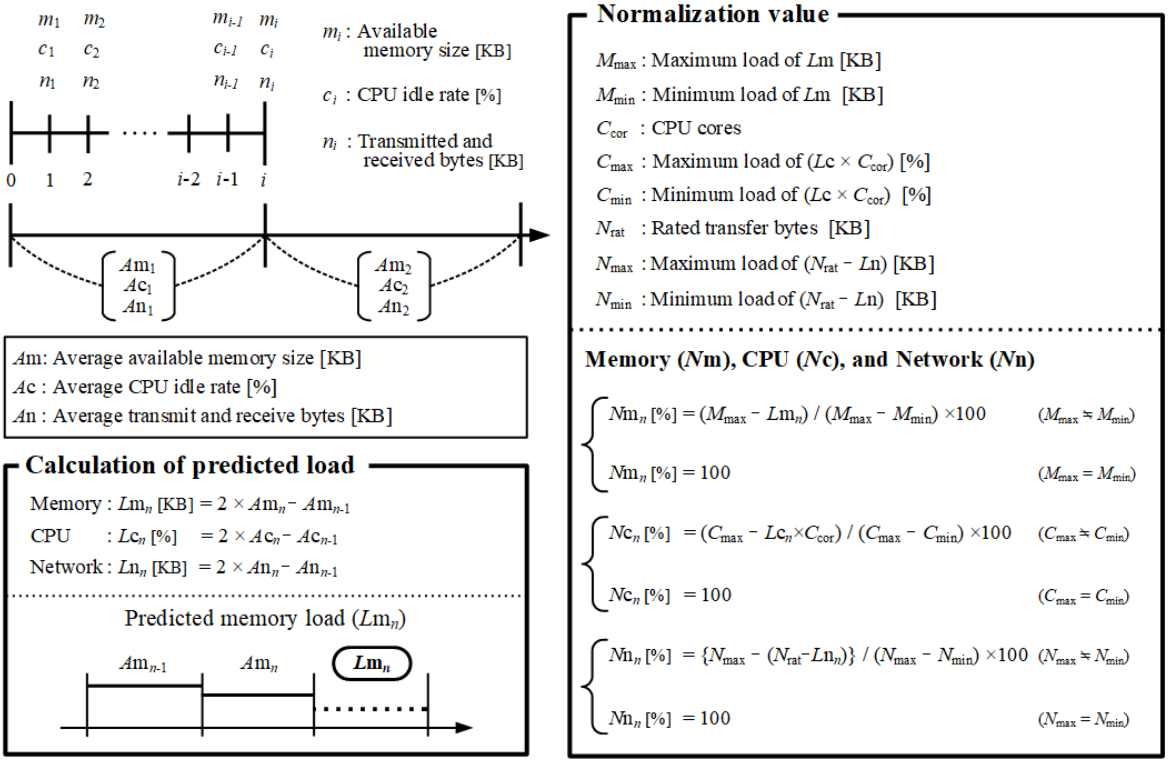
$$N\mathrm{m}_n\ [\%] = \frac{M_{\max} - L\mathrm{m}_n}{M_{\max} - M_{\min}} \times 100 \quad (M_{\max} \neq M_{\min}), \tag{4}$$

$$N\mathrm{m}_n\ [\%] = 100 \qquad (M_{\max} = M_{\min}). \tag{5}$$

The normalized load value $N\mathrm{c}_n$ of the CPU is calculated by

$$N\mathrm{c}_n\ [\%] = \frac{C_{\max} - L\mathrm{c}_n \times C_{\mathrm{cor}}}{C_{\max} - C_{\min}} \times 100 \quad (C_{\max} \neq C_{\min}), \tag{6}$$

$$Nc_n\,[\%] = 100 \qquad (C_{\max} = C_{\min}). \tag{7}$$

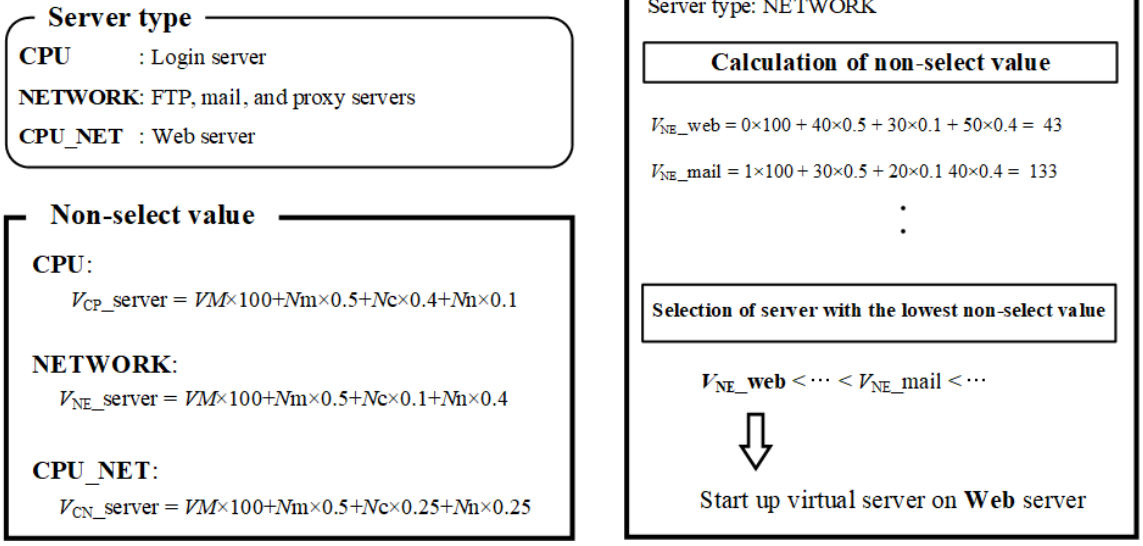


(a) Calculation of predicted load and its normalization

Real server condition file

Management server	*VM*	Memory (*Nm*) [%]	CPU (*Nc*) [%]	Network (*Nn*) [%]
Web	0	40	30	50
Mail	1	30	20	40

VM: Number of virtual servers running on management server



(b) Selection of a real server for the starting virtual server in consideration of the type of failed server

Fig. 5 Virtual server startup method considering load state.

The normalized load value Nn_n of the network is given as

$$Nn_n\,[\%] = \frac{N_{\max} - (N_{\mathrm{rat}} - Ln_n)}{N_{\max} - N_{\min}} \times 100 \quad (N_{\max} \neq N_{\min}), \tag{8}$$

$$Nn_n\,[\%] = 100 \qquad (N_{\max} = N_{\min}). \tag{9}$$

The normalized value has a minimum value of zero and a maximum value of 100. The load value standardized for each management server and *VM*, which is the number of virtual servers running on the management server, are recorded in a real server condition file.

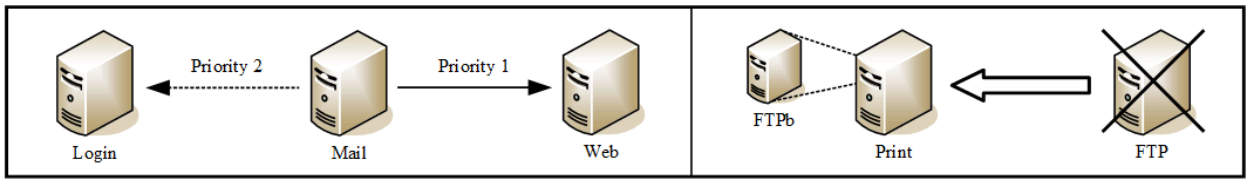
In this system, a virtual server is used to recover the function of the failed target server. Therefore, it is necessary to select the management server for starting the virtual server. A non-select value of each management server corresponding to the server type of the failed target server is calculated based on the real server condition file. The non-select value indicates that it is not appropriate for starting a virtual server if the value is large. The calculation method is shown in Fig. 5(b). We define three server types: the CPU-type server, which mainly uses the CPU, the NETWORK-type server, which mainly provides data, and the CPU_NET-type server, which both uses CPU and provides data. Here, a web server is defined as the CPU_NET-type server because it provides HTML data and image data and uses CGI. Here, *N*m, *N*c, and *N*n are weighted according to the server type, and the total value obtained from these is set to 100 at the maximum. In addition, *VM* is weighted 100 times in order to increase the non-select value. Since the memory load *N*m is important in every server type, *N*m is 0.5 times. For the CPU-type server, *N*c is 0.4 times, and *N*n is 0.1 times. For the NETWORK-type server, *N*c is 0.1 times, and *N*n is 0.4 times. In the CPU_NET-type server, *N*c is 0.25 times, and *N*n is 0.25 times. The figure shows the case in which the FTP server, which is the NETWORK-type server, fails as an example. Non-select values $V_{\mathrm{NE_}}$web and $V_{\mathrm{NE_}}$mail are calculated from values of the real server condition file, and the web server with the lowest value is selected.

2.6. Management file

The proposed system uses a group file, a separation file, an operation file, a trouble file, and the real server condition file, which are shown in Fig. 6. These files are saved in the shared area on the file server, and each management server uses these files through NFS connection. The group file contains the number, the host name of the target server, and the file name of the management data. The operation file, which contains the server management priority, the host name of the management server, the host name of the target server, and the file name of the management data, is constructed based on the group file.

In Fig. 6, the mail server manages the web server with P1 and manages a login server with P2 based on the information indicated in bold boxes in the operation file. The management program is executed based on the management data, as shown in Fig. 4. In addition, the proposed system records the name of the virtual server to recover the server function, the name of the target server generating the trouble, and the name of the management server starting the virtual server in the trouble file when trouble is detected in the target server. In this figure, an FTPb server, which is a virtual server, is started on a print server in order to recover the function of an FTP server, which is experiencing trouble. The recovery condition is indicated in bold boxes in the trouble file. Here, the real server condition file shown in Fig. 5(b) is used for selecting the management server to start the virtual server.

Each file is related. If trouble is detected in the target server, the server information is deleted from the group file and is recorded in the separation file. Thereafter, the operation file is modified based on the group file. The management construction is dynamically modified based on the operation file. If the target server is recovered, the server information is deleted from the trouble file and the separation file. Thereafter, the information is returned to the group file, and the operation file is modified based on the group file. The proposed system dynamically modifies the management configuration based on the operation file.



Group file

Number	Target server	Management data
1	Mail	Mail.dat
2	Web	Web.dat
3	Proxy	Proxy.dat
4	Login	Login.dat

Operation file

Priority	Management server	Target server	Management data
1	Mail	Web	Web.dat
1	Web	Proxy	Proxy.dat
1	Proxy	Login	Login.dat
1	Login	Mail	Mail.dat
2	Mail	Login	Login.dat
2	Web	Mail	Mail.dat
2	Proxy	Web	Web.dat
2	Login	Proxy	Proxy.dat

Separation file

Number	Target server	Management data
8	FTP	FTP.dat

Trouble file

Virtual server (Backup server)	Trouble server	Management server
FTPb	FTP	Print

Real server condition file

Management server	VM	Memory [%]	CPU [%]	Network [%]

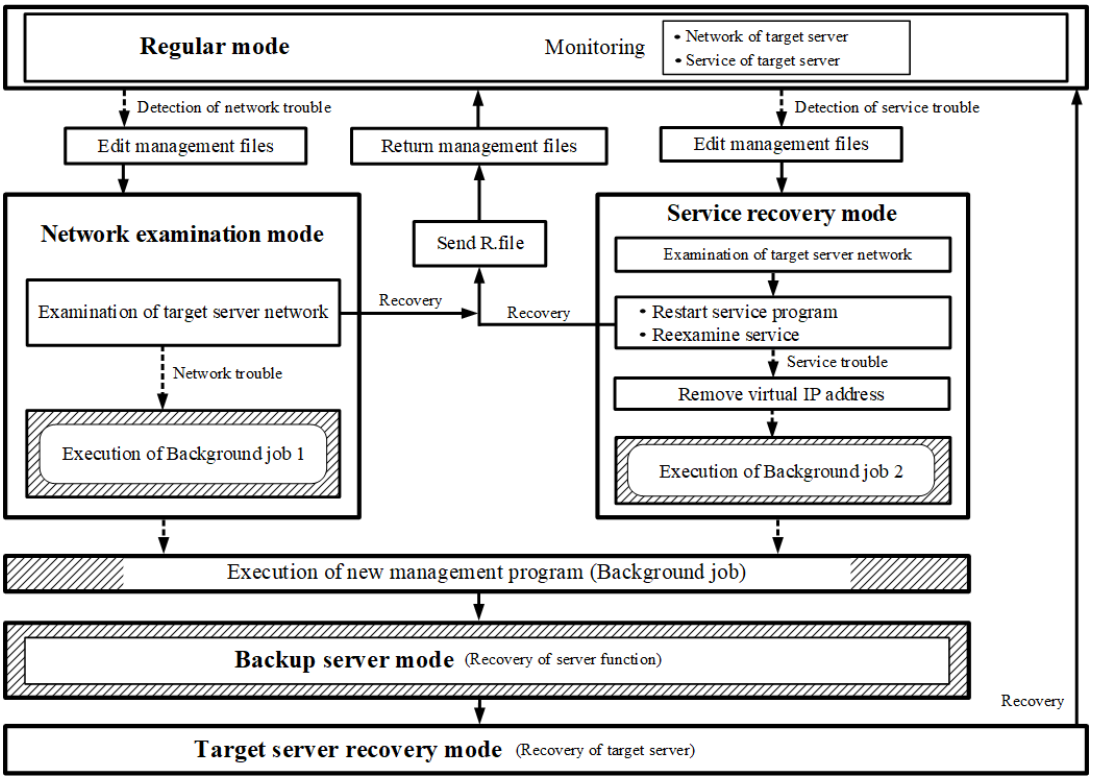
Fig. 6 Management files.

3. Management and recovery operation of the server management system incorporating a P2P method

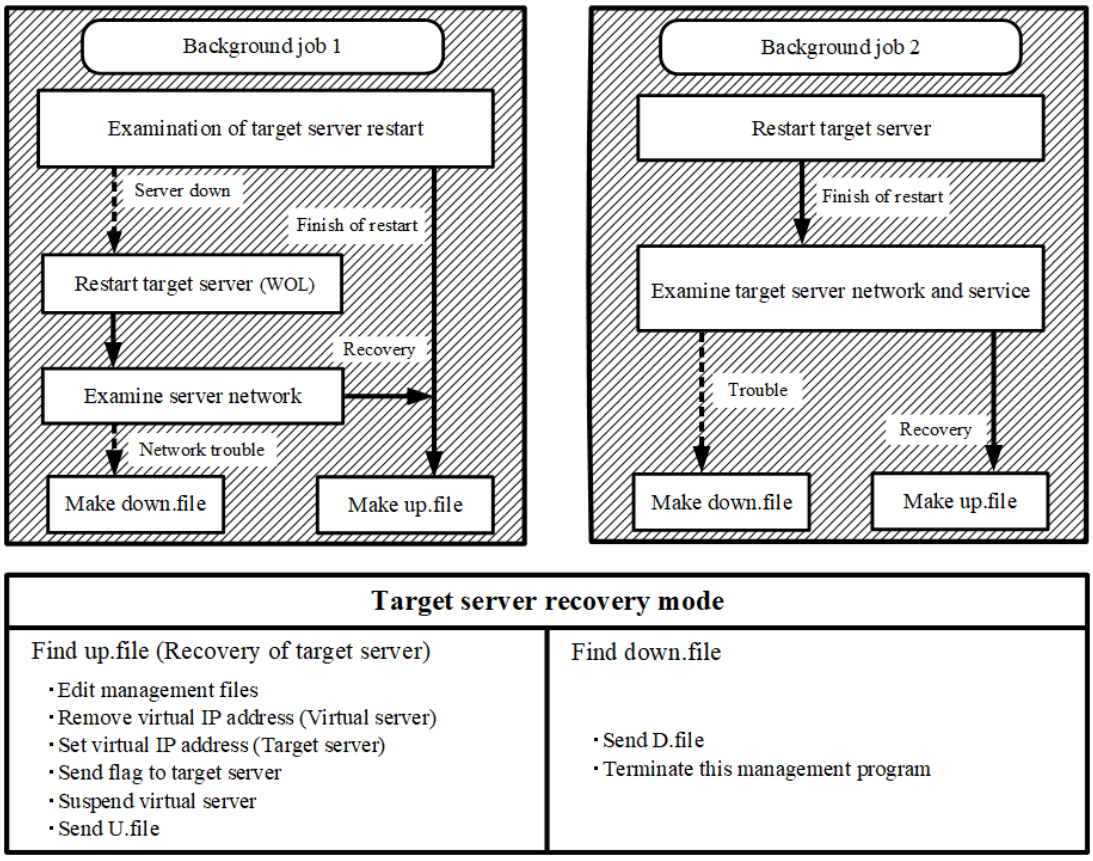
Since a shared area is required, each server is connected to the file server through NFS connection. Therefore, in consideration of the load of the file server, inter-process communication between management servers is adopted. The proposed system minimizes the access to the file server, except for the startup of the management system and the recovery operation of the target server function.

3.1. Operation flow of the management program

The operation flow of the management program is shown in Fig. 7. The management program shown in Fig. 7(a) adopts a mode segmentation method [13],[14] and is configured in five different modes: Regular, Network examination, Service recovery, Backup server, and Target server recovery. The mode is a software system used to execute a specific task, and each mode can be executed independently. The operation in this figure shows the execution state of the management server operating with P1. In the Regular mode, the management program regularly monitors the network and the service offer condition on the target server. The proposed system uses the command “ping” in UNIX to monitor the network on the target server and uses the Expect programming language to monitor the service condition [13],[14]. This language enables the correct monitoring of the service offer state on the target server because this language has a time-out function and a function to perform processing based on the response from the target server accessing a service port offering the service. If trouble is detected in the target server, the control state of the management program shifts from the Regular mode to another mode based on necessary information depending on the type of trouble encountered.



(a) Operation flow for dealing with various types of trouble by the mode segmentation method



(b) Role of the background job

Fig. 7 Operation flow of the management program.

If trouble is detected in the target server network in the Regular mode, the server information is deleted from the group file shown in Fig. 6 and the related management file is edited. The control state shifts to the Network examination mode. This mode examines the network. If the network is judged to be normal, an R.file is sent to the management server operating with P2 in order to report that the network is recovered, and the control state returns to the Regular mode after the related management files are returned. If the network is judged to be abnormal, Background job 1, which performs the recovery operation of the target server and the examination of the recovery state, is executed as a background process. The control state shifts to the Backup server mode after executing the new management program as a background process based on the management file. The management program, which is operated with P1, executes a management program with P2 on the new target server. We refer to this method as the dynamic management extension method.

In addition, if the service trouble on the target server is detected in the Regular mode, the server information is deleted from the group file, and the related management file is edited. The control state shifts to the Service recovery mode. This mode restarts the service program after the target server network is examined and reexamines the service offer state. If the state is judged to be normal, the R.file is sent to the management server operating with P2 to report that the service condition is recovered and the control state returns to the Regular mode after the related management files are returned. If the state is judged to be abnormal, Background job 2, which performs the recovery operation of the target server and the examination of the recovery state, is executed as a background process. The control state shifts to the Backup server mode after executing the new management program as a background process based on the management file. Background jobs 1 and 2 are executed as a background process because performing the state examination and the recovery job for the target server is time consuming. In the Backup server mode, the virtual server having the function of the target server as a backup server is started. In order to change client access from the target server generating the trouble to the virtual server, the virtual IP address is set to the virtual server by the management program after the virtual server has been started. The server function is recovered by the recovery job. Selection of the management server for starting the virtual server is determined based on the non-select value in Fig. 5. In the Target server recovery mode, the results of Background job 1 or 2 are awaited, and the operation is performed based on the results.

The roles of the background job and the detailed operations of the Target server recovery mode are shown in Fig. 7(b). In Background job 1, the target server in which network trouble is detected is examined in order to determine whether the target server is in the restart state, and if the target server network is judged to be normal after restart, the up.file is created. However, if the target server is judged not to be in the restart state, the target server is forcefully restarted by means of wake on LAN (WOL) using the MAC address recorded in the management data shown in Fig. 4. The target server network is reexamined after restart. If the target server network has recovered, the up.file is created, whereas if the target server network has not recovered, the down.file is created, and Background job 1 is finished thereafter. In Background job 2, the problem target server is restarted after removing the virtual IP address from the server. Thereafter, both the network and the service on the target server are examined. If the target server service has recovered, the up.file is created, whereas if the target server service has not recovered, the down.file is created and Background job 2 is finished thereafter.

In the Target server recovery mode, if the up.file is found, this mode judges that the target server is recovered. The related information is edited, and the target server is returned to the original condition by setting the virtual IP address to the target server after removing the virtual IP address from the virtual server to be suspended later. A flag file is sent to the target server in order to execute the management program. The control state returns to the Regular mode after an U.file is sent to the management server operating with P2 in order to report that the server is recovered. If the down.file is found, the target server is judged not to be recoverable, and a D.file is sent to the management server operating with P2 in order to report that the server cannot be recovered. The management program, which has become unnecessary, is terminated.

3.2. Operation flow of each management priority

An operation flowchart of two server management priorities is shown in Fig. 8. We explain the operation using only

four servers. Servers S1, S3 and S4 are management servers, and S2 is the target server. The virtual server, S2b, is the backup server for recovering the S2 function. We herein assume that the operation of S1, S3 and S4 is normal. Here, (P1) and (P2) indicate a state in which recovery operation is being performed with their respective priorities. Server S1 executes the management program with P1, and S3 executes the management program with P2. Both S1 and S3 regularly monitor the network and the service offer state of S2. In the proposed method, in order to synchronize the monitoring timing of all of the management servers, the synchronization processing program shown in Fig. 3 is executed. The S.file, which is a synchronization file, is regularly broadcasted from the management server that is recorded at the head of the group file shown in Fig. 6. Thus, servers S1 with P1 and S3 with P2 synchronously monitor S2.

If trouble is detected in S2, S1 executes the background job that performs the recovery operation of S2. S1 executes new management programs that S1 manages S3 with P1 and S3 manages S1 with P2 by the dynamic management extension method. Thereafter, S1 controls S2b to start on S4 because the non-select value of S4 is the lowest and sets the virtual IP address to S2b. The function of S2 is recovered by means of this operation. Thereafter, S1 enters a wait state in order to find the up.file or the down.file, which is created by the background job. If S3 detects trouble in S2, then S3 enters a wait state in order to receive the U.file or the D.file, which is sent by S1. In this state, S3 regularly examines the network and the service offer state of S1. If S1 is judged to be in an abnormal state, then S3, rather than S1, performs an operation with P1. For the case in which S1 is in the normal state, the up.file is created by the background job if S2 is judged to recover the function. If S1 finds the up.file, the U.file is sent to S3 by inter-process communication, and both S1 and S3 synchronously monitor S2 again when S3 receives the U.file. If S2 is judged not to be recovered by the background job, the down.file is created. If S1 finds the down.file, the D.file is sent to S3 by inter-process communication. When S3 receives the D.file, the management programs to manage S2 are then terminated. Through the above-mentioned process, the management configuration is automatically modified through the process.

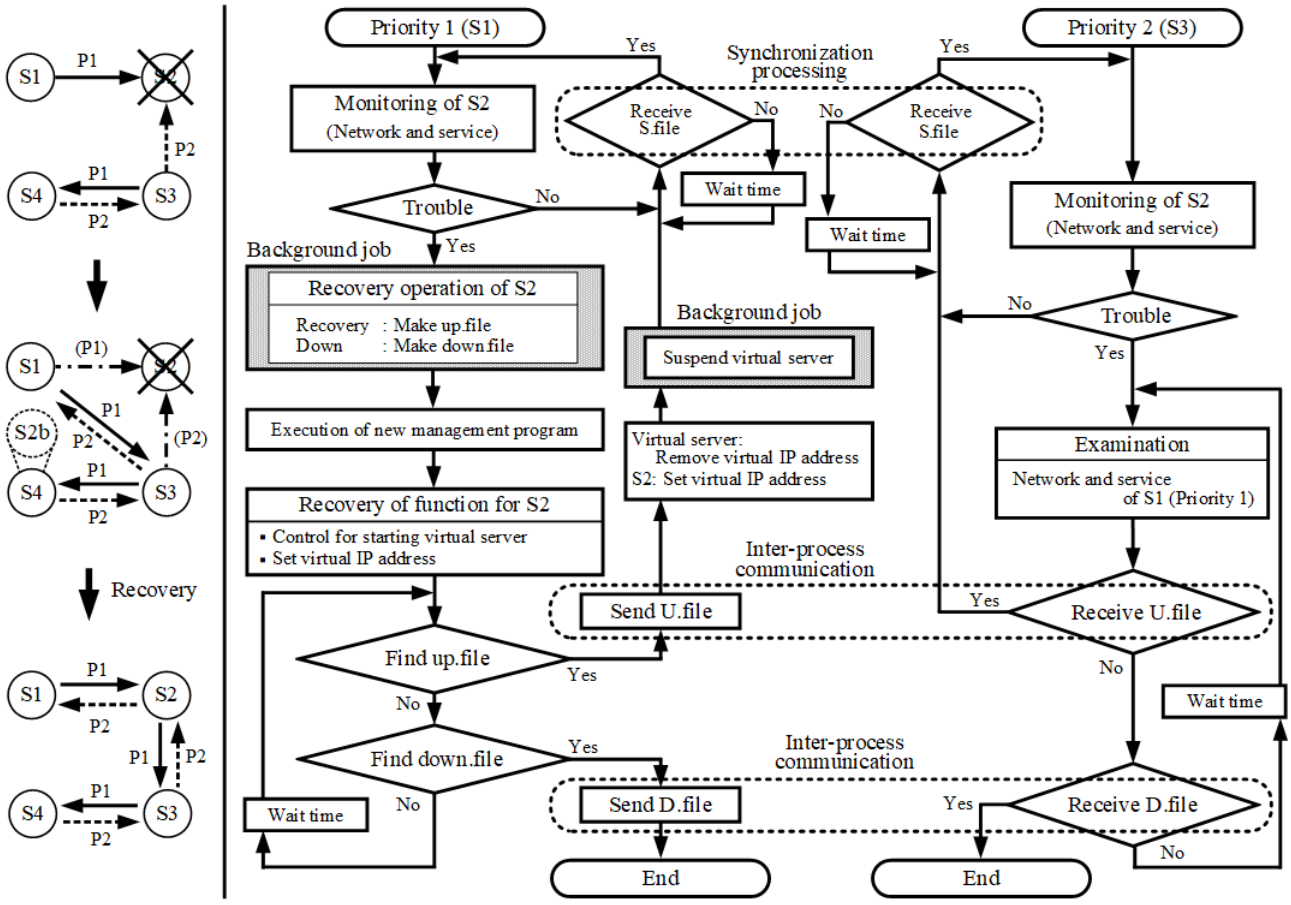


Fig. 8 Flow chart of each management priority.

3.3. Server function recovery method and dealing with the server after recovery

Figure 9 shows how to recover the server functions in the case of target server failure. S1 through S6 operate as both management servers and target servers. Servers S1b through S4b in the figure denote virtual servers for recovering the server function of each server. System data to construct virtual servers are saved on the file server. Each management server uses system data through NFS connection.

The recovery method for the case shown in Fig. 9(a) is described below. If management server S1 operating with P1 detects trouble in target server S2, management programs that S1 manages S3 with P1 and S3 manages S1 with P2 are executed by the dynamic management extension method. Thereafter, virtual server S2b is started on S6 by the control of S1 because S6 has the lowest non-select value, and sets the virtual IP address to S2b. As a result, the function of S2 is recovered. If S2 is judged not to be recoverable by S1, the management programs operating in (P1) and (P2) are terminated. Here, the management program operating with P1 on S6 does not detect trouble in S2. Thus, the management program operating with P2 on S3 reports the modification to S6. The proposed method does not perform management of S2b because S2b only performs the role of S2 during the repair of S2.

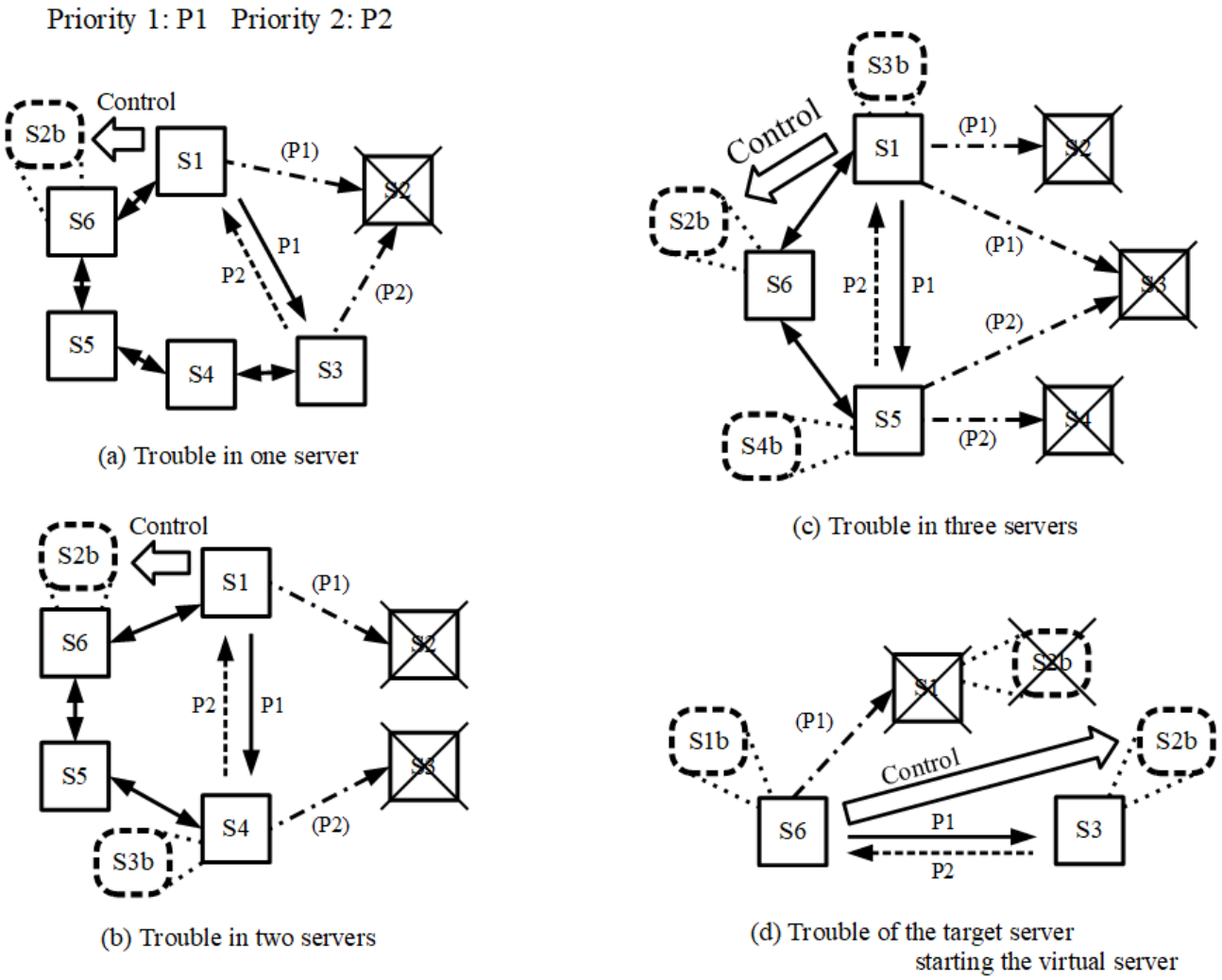


Fig. 9 Recovery methods for dealing with various types of trouble.

Recovery using two management servers operating with P1 and P2 shown in Fig. 9(b) is described below. Management server S1 operating with P1 detects trouble in S2, and virtual server S2b is started on S6 by the control of S1 because S6 has the lowest non-select value. Server S1 sets the virtual IP address to S2b and recovers the function of S2.

However, since management server S2 operating with P1 is in an abnormal state, management server S4 operating with P2 detects trouble in S3, and virtual server S3b is started on S4, which has the lowest non-select value at this point. Server S4 sets the virtual IP address to S3b and recovers the function of S3. After failure detection, S1 starts management of S4 with P1, and S4 starts management of S1 with P2 by the dynamic management extension method. If S2 and S3 are judged not to be recoverable by the management servers, the management programs operating in (P1) and (P2) are terminated.

Recovery using three management servers operating with P1 and P2 shown in Fig. 9(c) is described below. Management server S1 operating with P1 detects trouble in S2, and virtual server S2b is started on S6 by the control of S1 because S6 has the lowest non-select value. Server S1 sets the virtual IP address to S2b and recovers the function of S2. However, since management server S3 operating with P1 is in an abnormal state, management server S5 operating with P2 detects trouble in S4, and virtual server S4b is started on S5, which has the lowest non-select value at this point. Server S5 sets the virtual IP address to S4b and recovers the function of S4. After S1 and S5 detect trouble in S2 and S4, new management programs whereby S1 manages S3 and S5 manages S3 are executed. Management server S1 operating with P1 detects trouble in S3. S1 starts the recovery operation for S3. Virtual server S3b is started on S1, which has the lowest non-select value at this point. After S1 and S5 detect trouble in S3, new management programs whereby S1 manages S5 and S5 manages S1 are executed.

In the proposed method, the virtual server may start on the target server because each server has functions of both management and target servers. Figure 9(d) shows how to recover the server function when the target server starting the virtual server breaks down. Target server S1 starts virtual server S2b in order to recover the server function of S2. Management server S6 operating with P1 detects trouble in S1, and virtual server S1b is started on S6, which has the lowest non-select value. Server S6 sets the virtual IP address to S1b and recovers the function of S1. S6 controls S2b to start from S3, which has the lowest non-select value at this point. Server S3 can start virtual server S2b because the system data of S2b are saved in the shared area on the file server.

In a server system that actually operates, the probability that more than half of the servers in the system will fail at the same time is considered to be very low. The proposed system can deal with even such a case. If five of the six servers fail, one server that is operating starts five virtual servers to recover the function of the failed servers. However, this is conditional upon whether the running server has sufficient resources to deal with such a situation.

Figure 10 shows a processing procedure for the case in which the target server is recovered in the state shown in Figs. 9(b) and 9(d). P1 and P2 enclosed by boxes indicate the management program which is started by the recovered target server. The case in which S2 and S3 simultaneously experience faults and then only S2 is recovered is shown in Fig. 10(a). If (P1) judges that target server S2 has recovered, (P1) sends the flag file to the server and management programs are started on the server. (P1) performs setting and removing the virtual IP address on S2 and S2b, and suspends S2b. P1 enclosed by a box executes the management program, which manages S2 with P2, on target server S4, and the management program operating with P2 that manages other than S2 is terminated. The management program P2 managing S3 is also a termination target. However, the program is not terminated because the program is performing the recovery processing. P2 enclosed by a box terminates the management program, which manages S4 with P1, on S1. (P1), which has been in the recovery processing state, returns to P1.

For the case in which S1 experiences a fault starting S2b and then S1 is recovered is shown in Fig. 10(b). If (P1) judges that target server S1 has recovered, (P1) sends the flag file to the server and management programs are started on the server. (P1) sets and removes the virtual IP address on S1 and S1b and suspends S1b. The management program, which is P1 enclosed by a box, terminates the management program, which manages S6 with P2, on S3. The management program, which is P2 enclosed by a box, terminates the management program, which manages S3 with P1, on S6. (P1) and (P2) which have been in recovery processing states return to P1 and P2, respectively. Here, S2b does not return to the activation from S1 because the processing time is considered.

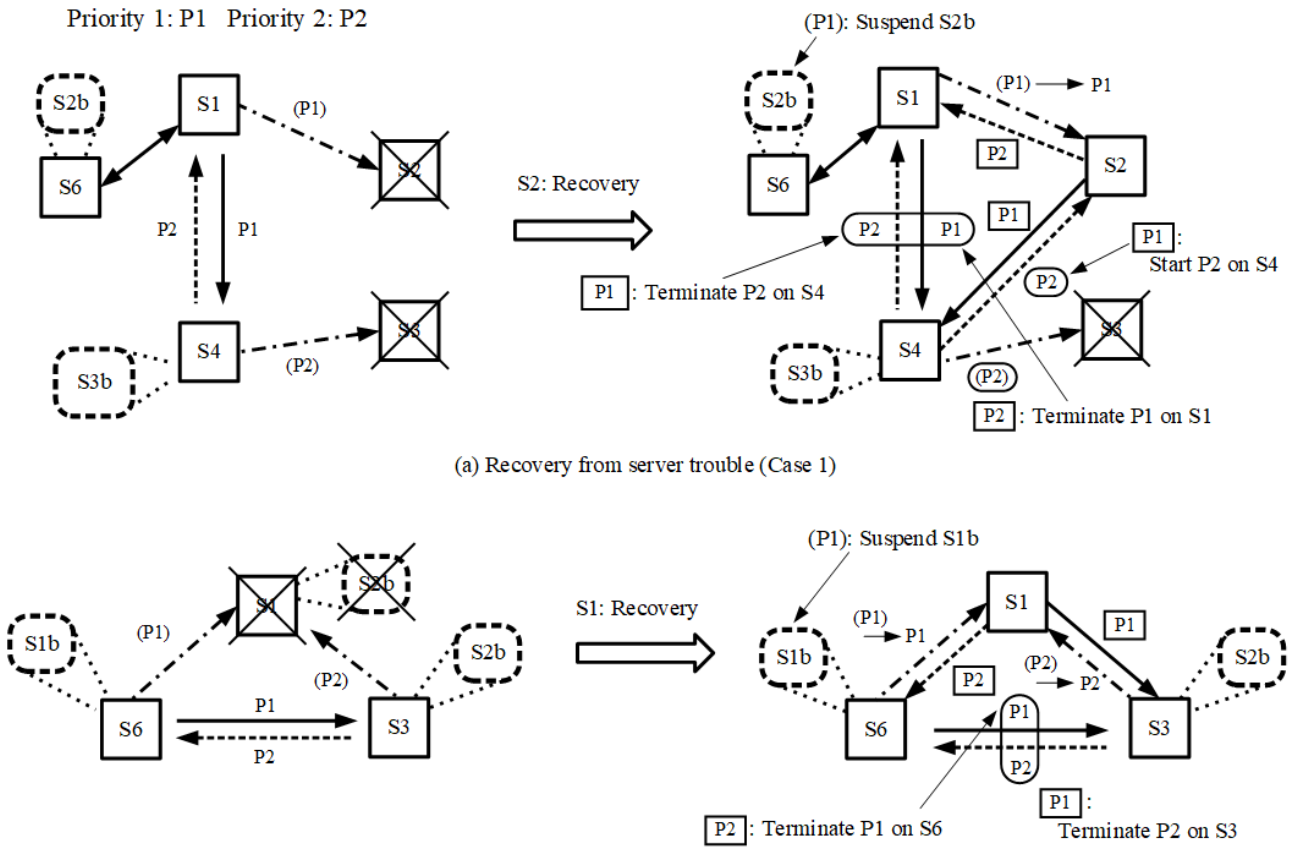


Fig. 10 Processing after recovery of the target server.

4. Operation experiment on the server management system incorporating a P2P method

4.1. Experimental system

The construction of the experimental server management system incorporating the P2P method is shown in Fig. 11. The experimental system consists of six servers (Mail, login, FTP, proxy, web 1, and web 2), file server 1, file server 2, a client, a router with a 1000BASE-T function, a 1000BASE-T switching HUB, and a 100BASE-TX switching HUB. File server 1 is used in the P2P management system, and file server 2 is used to offer services to clients. The six servers have functions of both management and target servers. The executer program shown in Fig. 3 is installed on each management server and is executed at the start time of each management server. In the network environment of the experimental system, the access speed from clients is set to one-tenth the transfer speed between servers, as in a general network system. The management program, the group file, the operation file, the separation file, the trouble file, the real server condition file, management data, system log files, and virtual server system data are saved in the shared area on file server 1. Each management server uses these files through NFS connection. In the experimental system, flag files on each management server are created or deleted by file server 1. If the flag file is created in the local disk, management programs are executed based on the operation file, and subprograms, which are the synchronization processing program and the server load measurement program, are also executed at the same time. The operating state of the management program on each management server is recorded in the system log file. If the flag file is deleted, the executer program stops all management programs and subprograms starting on the management server and stops the operation of the management server.

In this experiment system, we use the command “sar” in UNIX for load measurement. The load measurement is performed within the server monitoring interval time, and the real server condition file shown in Fig. 6 is in the latest state at the time of monitoring.

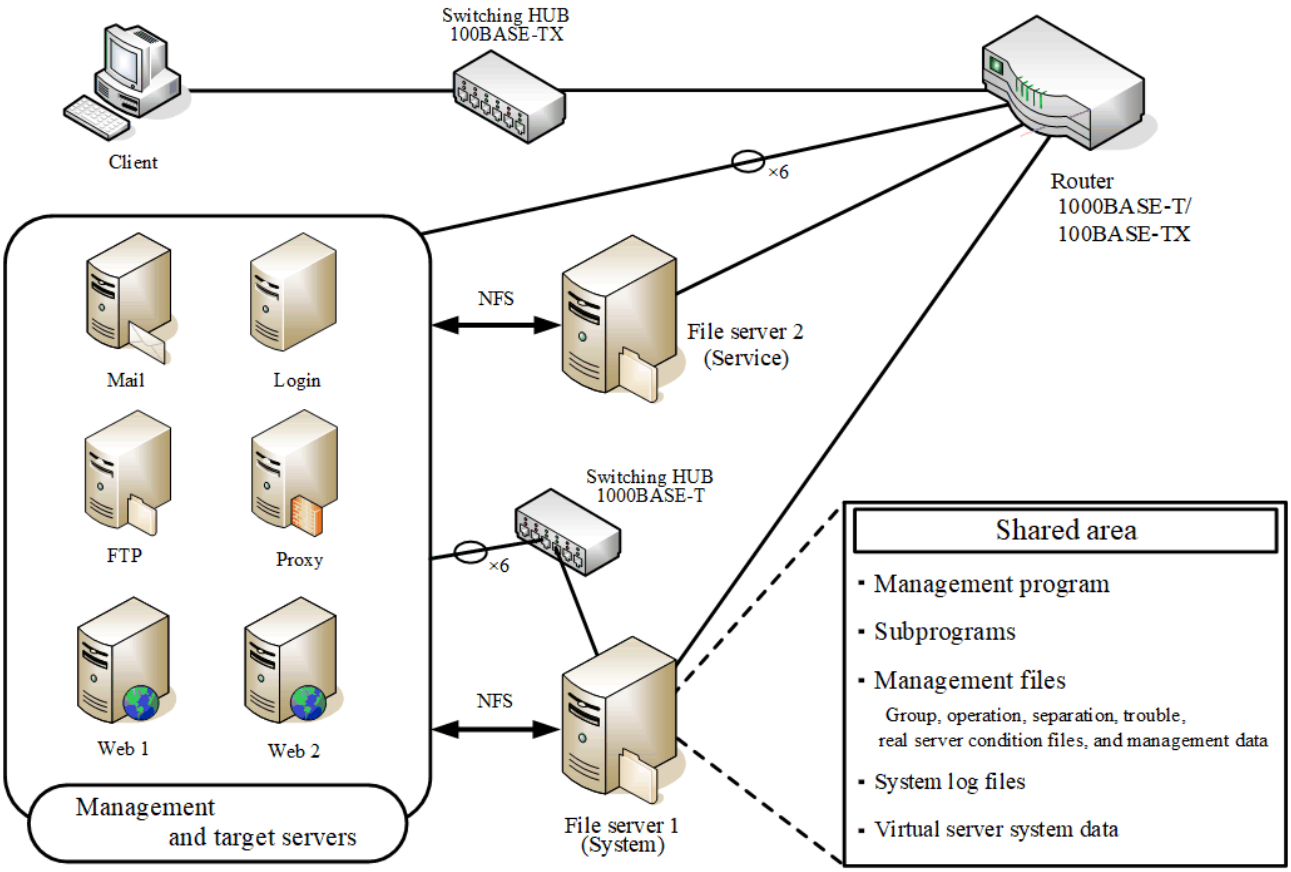


Fig. 11 Experimental system.

The specifications of the experimental system, the characteristics of each server as fundamental data, and the setting parameters when the system operates are listed in Table 3. The specifications of the six management servers are similar, and the CPU is a Core i7-3770. The memory size is 8,192 MB, in consideration of the start of virtual servers. A Kernel-based Virtual Machine (KVM), which is virtualization software, is installed in each management server. Since each management server operates in an environment of a character-based user interface, the virtual server is started in a no-window state on the display. File server 1 has a Core i7-4770 CPU and 8,192 MB of memory. One CPU and 2,048 MB of memory are assigned to each virtual server. CentOS (64-bit), which is often adopted as an OS for servers, is installed on each server.

The characteristics of each server are shown as fundamental data. These data are the average values of multiple experimental results. The average times in Table 3 are given by the command “time” in UNIX. Here, *T*rs, which is the start time of each management server, is 56.67 s, and *T*rr, which is the restart time of each management server, is 58.51 s. Moreover, the network examination time when the target server network is normal is 0.01 s, and the service examination time *T*se varies depending on the type of service provided, 0.06 s for the FTP server, 0.23 s for the login server, 0.36 s for the mail server, 0.59 s for the proxy server, and 1.59 s for the web server. In the virtual server, the start time under the normal condition is 42.84 s, *T*v, which represents the time to activate the suspended condition, is 4.67 s. For this reason, the start time for the virtual server is shortened by this method. Therefore, the virtual server is always in a waiting state with a suspended condition in this system.

In the setting parameters, *T*im, which represents the inter-monitoring time in the target server management, is 10.00 s. Moreover, *T*nd, which is the network examination time when the target server network is abnormal, is 2.00 s. In addition, *T*sw, which is the maximum waiting time for service examination, is 3.00 s because the maximum value of *T*se

is 1.59 s, and *T*rc, which is set to approximately 1.5*T*rr, is the maximum time required in order to determine whether the target server is in the restart state and is 88.00 s.

Table 3 Specifications, characteristics, and setting parameters of the experimental system.

Specifications	Management server (Target server)	File 1 (System)
CPU	Core i7-3770 3.40 GHz (TB: 3.90 GHz)	Core i7-4770 3.40 GHz (TB: 3.90 GHz)
Memory [MB]	8,192	
Hard disk	SATA 2 (7,200 rpm)	
Virtualization software	KVM 1.5.3	
System software	postfix, sshd, httpd, vsftpd, squid	nfsd
OS	CentOS 7.3 (64-bit)	

Specifications	Virtual server
CPU	1
Memory [MB]	2,048
OS	CentOS 7.3 (64-bit)

Characteristics	Management server (Target server)	
Start time [s]	*T*rs	56.67
Restart time [s]	*T*rr	58.51
Network examination time [s]		0.01
Service examination time [s]	*T*se	0.06 - 1.59

Characteristics	Virtual server	
Start time [s]		42.84
Time to activate suspended server [s]	*T*v	4.67

Setting parameters		
Inter-monitoring time [s]	*T*im	10.00
Network examination time (Disconnection) [s]	*T*nd	2.00
Maximum waiting time for service examination [s]	*T*sw	3.00
Maximum judgment time for restart [s]	*T*rc	88.00

4.2. Log files to record the operating status

If trouble occurs in the target server, the proposed system records the operating status, such as the type of trouble detected and the operating state of the virtual server, with the time in the system log file. In addition, the operating state of the background job to perform the recovery job for the target server is recorded in the log file. In the proposed system, the log data recorded by the foreground job and that recorded by the background job are recorded in the log file. In the log data recorded by the background job, the string "_ _" is added to the top of the content in order to distinguish the log data.

Figure 12 shows the contents of the log file and the comments to represent important contents. System log (P1) is recorded by the management server operating with P1, and System log (P2) is recorded by the management server operating with P2. In the measurement, the contents of the log file are used. Since System log (P1) shown in the figure is experimental, numerical values are recorded in one-second intervals in the log file during the state awaiting the S.file shown in Fig. 8, and the word "Wait" is recorded in one-second intervals in the log file during the state awaiting an examination result from the background job. In System log (P2), the word "P2_wait" is recorded in one-second intervals in the log file during the state awaiting the S.file. In the access log in the client, the client accesses the server setting the virtual IP address, and the results to be accessed in one-second intervals by a client are recorded in the access log file. The client accesses the server using the Expect programming language to examine the correct service state. The status "OK", which represents the condition in which services are offered on the target server, and the status "XX", which represents the condition in which services cannot be offered, are recorded in the access log file together with the time. The target server, which is a real server, is an FTP server. The IP address is denoted as IPR, and the virtual IP address is denoted as

VIP. The IP address of the virtual server is denoted as IPV.

In the figure, the network trouble is reproduced by shutting down the FTP server when the inter-monitoring time exceeds five seconds. In System log (P1), trouble occurs at 12:53:45 in the inter-monitoring time, and the network trouble is detected at 12:53:52.13. The virtual server is started at 12:53:54.43 after the reexamination of the network trouble at 12:53:54.18, and startup is completed at 12:53:59.10. The server function is recovered at 12:53:59.22 after setting the virtual IP address. The difference between the time at which the server function is recovered and the time at which the network trouble is detected is the recovery time of the server function. In the access log in the client, trouble in the server is detected at 12:53:45.41, and the service is recovered at 12:53:59.56. In System log (P1), the recovery job of the target server is started at 12:53:54.19 by the background job. The target server is judged to be in a stop condition at 12:55:22.19, and forcible start is attempted by WOL. Thereafter, the target server is judged to be started normally, and the server is recovered at 12:56:19.96 after setting the virtual IP address. The server operating with P1 sends the U.file to the server operating with P2 at 12:56:19.98. The difference between the time at which the target server is recovered and the time at which the network trouble is detected is the recovery time of the target server. Although the recovery job of the target server is performed at 12:56:19.96, the abnormality of the service offer state is not found at 12:56:20.78 in the access log in the client. In System log (P2), the server operating with P2 enters the state awaiting a result from the server operating with P1 after the network trouble is detected at 12:53:52.25. The server operating with P2 returns the monitoring state of the target server after receiving U.file at 12:56:20.07.

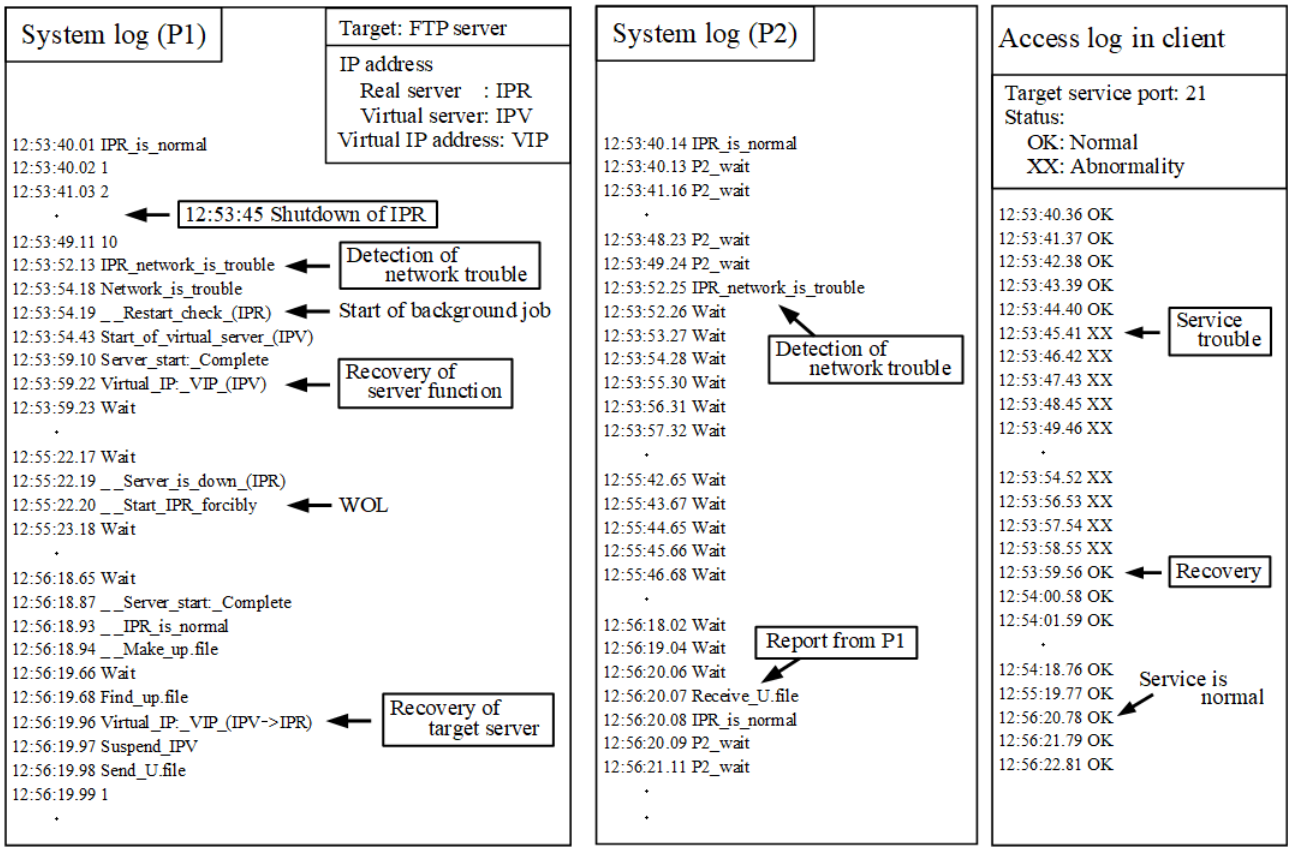


Fig. 12 Log files to record the operating status.

4.3. Experimental results in the case of target server trouble

In the experiment, four servers (Mail, FTP, login and web 1) are used. Experiments to reproduce trouble on the network or the service on target server FTP are performed. The inter-monitoring time is 10 s. Trouble is generated when the inter-monitoring time exceeds five seconds, and the time required for the recovery operation is measured. Here, the

login server is set to have 512 MB of memory, and the other servers are set to have 1 GB of memory by the command "stress" installed on each management server. Moreover, the measurement is conducted in a state without access from a client to the management server. The service trouble is reproduced by stopping the service program on the FTP server, and the network trouble is reproduced by restarting the FTP server or shutting down the FTP server.

As shown in Fig. 13, four types of trouble are defined. Here, St1 represents the condition in which the service function is recovered by restarting the service program. St2 represents the condition in which the service function is recovered by restarting the target server because the service function is not recovered by restarting the service program. Nt1 represents the case of an intentional restart of the target server, and Nt2 represents the condition in which the target server is shut down. Nt1 occurs when the server administrator restarts the server by pushing the restart button of the server in order to recover to a normal condition from a condition in which trouble occurs.

The Recovery time of server function column of the table in Fig. 13 indicates the time until the server function is recovered by the virtual server after trouble has occurred, and the Recovery time for target server column of the table in the figure is the time until the target server has recovered to a normal state after trouble has occurred. The client accesses the server in one-second intervals, and the Access disconnection time from the client column of the table in the figure indicates the time until the access by the client is recovered from the server access disconnection state. The recovery time of the server function and the recovery time for the target server are measured based on the log file that is recorded by the proposed system. The numerical value in parentheses represents the analytical time and is calculated based on the characteristics and setting parameters, which are listed in Table 3.

In the Recovery time of the server function column of the table in Fig. 13, *T*s2f is the recovery time for dealing with trouble in the case of St2, *T*n1f is the recovery time for dealing with trouble in the case of Nt1, and *T*n2f is the recovery time for dealing with trouble in the case of Nt2. The mode shift for the case of St2 is Regular mode → Service recovery mode → Backup server mode → Target server recovery mode → Regular mode. Here, *T*sw, which is the maximum waiting time for service examination in the Service recovery mode, and *T*v, which is the start time of the virtual server in the Backup server mode, are the main processing elements of *T*s2f. The analytical time is defined as follows:

$$T\mathrm{s2f} = T\mathrm{sw} + T\mathrm{v}. \tag{10}$$

In *T*s2f, the measured time is 8.56 s, and the analytical time is 7.67 s. The mode shift for the cases of Nt1 and Nt2 is Regular mode → Network examination mode → Backup server mode → Target server recovery mode → Regular mode. Here, *T*nd, which is the network connection examination time in the Network examination mode, and *T*v in the Backup server mode are the main processing elements of *T*n1f and *T*n2f. The analytical time is given as follows:

$$T\mathrm{n1f} = T\mathrm{n2f} = T\mathrm{nd} + T\mathrm{v}. \tag{11}$$

The measured time *T*n1f is 7.47 s, and the measured time *T*n2f is 7.46 s. The analytical time is 6.67 s. In the Recovery time of the server function column of the table in the figure, the difference between the measured time and the analytical time was less than 1 second.

In the Recovery time for the target server column of the table in Fig. 13, the recovery time for dealing with trouble in the case of St1 is *T*s1t, and the recovery time for dealing with trouble in the case of St2 is *T*s2t. Moreover, the recovery time for dealing with trouble in the case of Nt1 is *T*n1t, and the recovery time for dealing with trouble in the case of Nt2 is *T*n2t. The mode shift for the case of St1 is Regular mode → Service recovery mode → Regular mode. Here, the service program is restarted, and service reexamination is then performed in Service recovery mode after service abnormality is detected. The service examination time, *T*se, is the time required by the main processing elements of *T*s1t. The analytical time is defined as follows:

$$T\mathrm{s1t} = T\mathrm{se}. \tag{12}$$

In *T*s1t, the measured time is 0.31 s, and the analytical time is 0.06 s. In the case of St2, *T*rr, which is the restart time of the target server, is included in *T*s2t because the service offer state is recovered by restarting the target server. The analytical time is defined as follows:

$$T\mathrm{s2t} = T\mathrm{sw} + T\mathrm{rr} + T\mathrm{se}. \tag{13}$$

In *T*s2t, the measured time is 62.98 s, and the analytical time is 61.57 s. Since the target server is restated when the inter-monitoring time exceeds five seconds in the case of Nt1, the analytical time is given as follows:

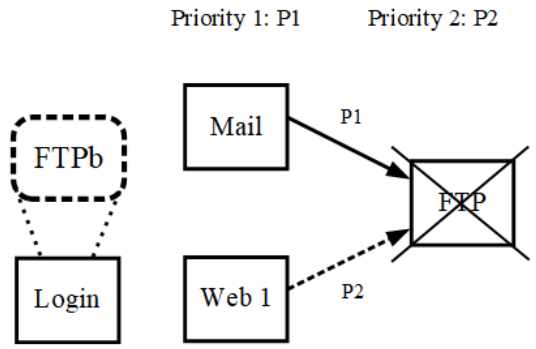
$$T\mathrm{n1t} = T\mathrm{rr} + T\mathrm{se} - (\frac{T\mathrm{im}}{2} + T\mathrm{nd}). \quad (14)$$

In *T*n1t, the measured time is 52.61 s, and the analytical time is 51.57 s. In the case of Nt2, *T*rc, which is the maximum time in order to determine whether the target server is in the restart state, and *T*rs, which is the server start time, are included in *T*n2t. The analytical time is defined as follows:

$$T\mathrm{n2t} = T\mathrm{nd} + T\mathrm{rc} + T\mathrm{rs} + T\mathrm{se}. \quad (15)$$

In *T*n2t, the measured time is 146.76 s, and the analytical time is 146.73 s. In the Recovery time for the target server column of the table in the figure, the difference between the measured time and the analytical time was approximately 1 second. The server restart time *T*rr greatly affects the recovery time in the case of St2 because the service function is recovered by server restart. The maximum time, *T*rc, and the server start time, *T*rs, greatly affect the recovery time in the case of Nt2 because the target server is restarted after examining whether the target server is in the restart state.

Here, in the Recovery time of the server function column and the Recovery time of the target server column, there is a difference between the measured time and the analytical time. We believe that the total time of the processing elements to be finished within 1 second influences this difference because this time is not included among the elements of the analysis time.



St1: Service program trouble (Recovery by service restart)

St2: Service program trouble (Recovery by server restart)

Nt1: Intentional restart of target server

Nt2: Shutdown of target server

		Target server: FTP		Value : Measured time [s] (Value) : Analytical time [s]			
Trouble list		**Recovery time of server function**		**Recovery time for target server**		**Access disconnection time from client**	
Service program	**St1**			***T*s1t**	0.31 (0.06)	***T*s1c**	5.36 (5.06±5.00)
	St2	***T*s2f**	8.56 (7.67)	***T*s2t**	62.98 (61.57)	***T*s2c**	13.24 (12.67±5.00)
Network	**Nt1**	***T*n1f**	7.47 (6.67)	***T*n1t**	52.61 (51.57)	***T*n1c**	13.73 (13.67±5.00)
	Nt2	***T*n2f**	7.46 (6.67)	***T*n2t**	146.76 (146.73)	***T*n2c**	14.14 (13.67±5.00)

Fig. 13 Recovery times in the case of trouble in one server.

In the Access disconnection time from client column of the table in Fig. 13, the recovery time for dealing with trouble in the case of St1 is *T*s1c, and the recovery time for dealing with trouble in the case of St2 is *T*s2c. The recovery time for dealing with trouble in the case of Nt1 is *T*n1c, and the recovery time for dealing with trouble in the case of Nt2 is

*T*n2c. Inter-monitoring time *T*im is included in each recovery time. Here, *T*s1c is the range of only the inter-monitoring time because the sum of the network examination time, the service examination time, and the restart time of the service is less than 1 second. The analytical time is defined as follows:

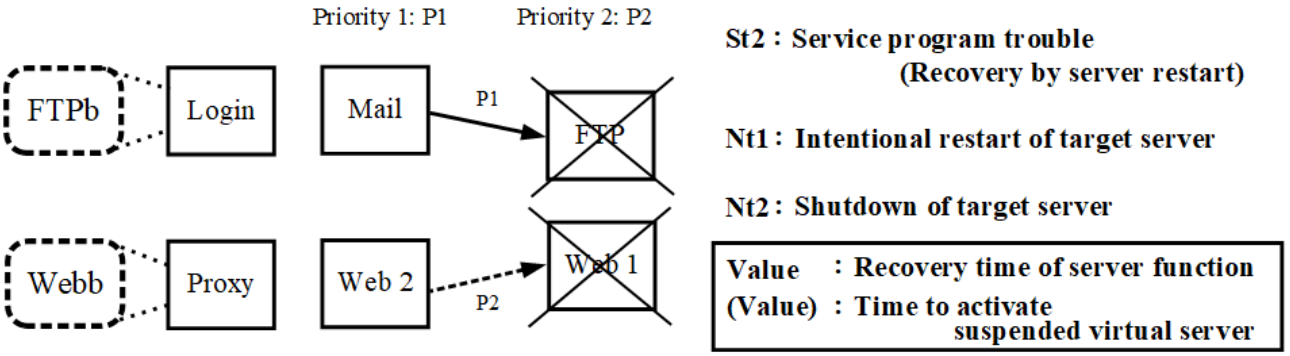
$$Ts1c = Ts1t + \frac{Tim}{2} \pm \frac{Tim}{2}, \tag{16}$$

$$Ts2c = Ts2f + \frac{Tim}{2} \pm \frac{Tim}{2}, \tag{17}$$

$$Tn1c = Tn2c = Tnd + Tn1f + \frac{Tim}{2} \pm \frac{Tim}{2}. \tag{18}$$

In *T*s1c, the measured time is 5.36 s, and the analytical time is 5.06 ± 5.00 s. In *T*s2c, the measured time is 13.24 s, and the analytical time is 12.67 ± 5.00 s. In *T*n1c, the measured time is 13.73 s, and the analytical time is 13.67 ± 5.00 s. In *T*n2c, the measured time is 14.14 s, and the analytical time is 13.67 ± 5.00 s. The measured time for each trouble is in the range of the analytical time.

The recovery time for the case in which trouble occurs simultaneously on two servers is shown in Fig. 14. The experimental condition is the same as in the case of trouble in one server shown in Fig. 13, and three types of trouble are defined in the figure. Numerical values that are not in parentheses indicate the recovery times of the server function, and numerical values in parentheses indicate the start times of the virtual server. The start time of the virtual server indicates the time to activate the suspended virtual server. Experiments to examine trouble in nine combinations of servers were performed. Here, the login server and the proxy server are set to have 512 MB of memory, and the other servers are set to have 1 GB of memory.

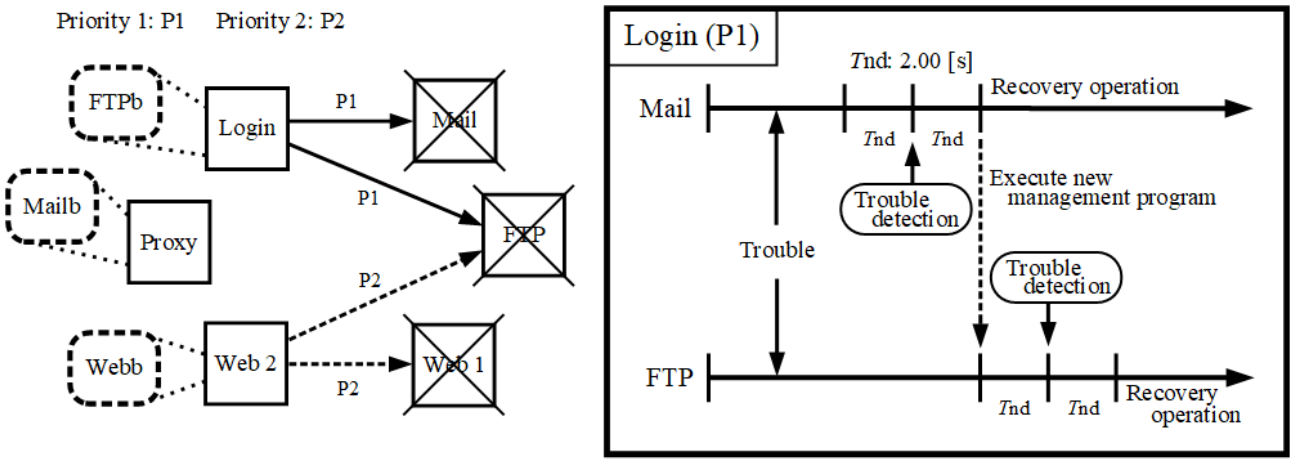


Measured time [s]		Target server: Web 1		
		St2	Nt1	Nt2
Target server: FTP	St2	11.49 (7.84) / 11.55 (7.82)	10.27 (7.72) / 10.85 (7.17)	10.17 (7.70) / 10.61 (6.94)
	Nt1	10.79 (7.15) / 10.13 (7.65)	10.84 (8.18) / 10.25 (7.70)	10.72 (8.06) / 10.11 (7.57)
	Nt2	11.01 (7.35) / 10.09 (7.61)	10.84 (8.31) / 10.31 (7.73)	10.61 (8.06) / 10.25 (7.69)

Fig. 14 Recovery times for cases in which trouble occurs simultaneously in two servers.

In the case of St2, trouble is generated simultaneously on two servers. The recovery time for the server function of the FTP server is 11.55 s, and that of web server 1 is 11.49 s. The start time of virtual server FTPb is 7.82 s, and that of virtual server webb is 7.84 s. In the case of Nt1, trouble is generated simultaneously on two servers. The recovery time for the server function of the FTP server is 10.25 s, and that of web server 1 is 10.84 s. The start time of virtual server FTPb is 7.70 s, and that of virtual server webb is 8.18 s. In the case of Nt2, trouble is generated simultaneously on two servers. The recovery time for the server function of the FTP server is 10.25 s, and that of web server 1 is 10.61 s. The start time of virtual server FTPb is 7.69 s, and that of virtual server webb is 8.06 s. Here, in the recovery time of the server function, the time is approximately 3 seconds longer than the result shown in Fig. 13. We consider that these results occur because the startup processing of the virtual server overlapped and the start time is longer than *T*v shown in Table 3.

The recovery time for the case in which Nt2 trouble occurs simultaneously on three servers is shown in Fig. 15. The experimental condition and the definition of the recovery time and the disconnection time are the same as in Fig. 13. Here, the proxy server is set to have 512 MB of memory, and the web server 2 is set to have 1 GB of memory. Moreover, the login server is set to have 2 GB of memory. The login server operating with P1 and the web server 2 operating with P2 perform recovery operations of the server function and the target server. For the case in which network trouble Nt2 occurs simultaneously in three servers, if the login server detects network trouble in the mail server, the management program is operated to manage the FTP server after a lapse of *T*nd for the network reexamination. Thereafter, the program detects network trouble in the FTP server after a lapse of *T*nd for the network examination.



Trouble: Nt2 (Shutdown of target server)

Target server	Recovery time of server function [s]	Time to activate suspended virtual server [s]	Access disconnection time from client [s]
Mail	11.17	8.50	17.59
Web 1	11.39	8.50	17.99
FTP	9.95	7.36	20.71

Fig. 15 Recovery times for cases in which trouble occurs simultaneously in three servers.

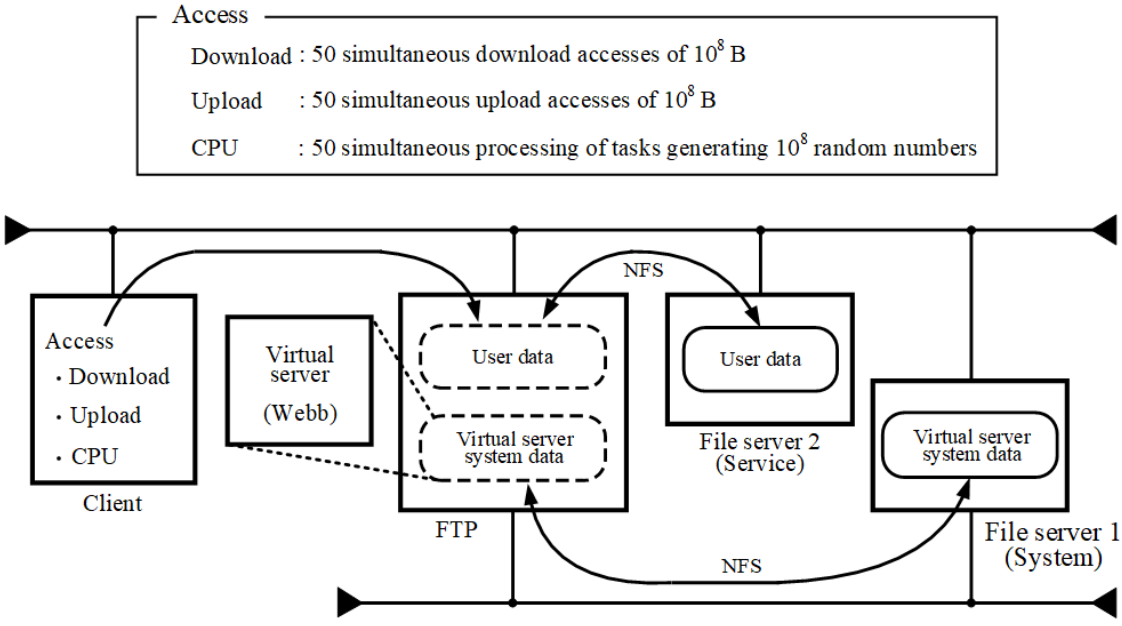
The recovery times for the server functions of the mail server, web server 1, and the FTP server are 11.17 s, 11.39 s, and 9.95 s, respectively. The start times for virtual servers of the mailb server, the webb server, and the FTPb server are 8.50 s, 8.50 s, and 7.36 s, respectively. The start time for the virtual server indicates the time to activate the suspended virtual server. The access disconnection times from the clients of the mail server, web server 1, and the FTP server are

17.59 s, 17.99 s, and 20.71 s, respectively. Here, the start time of the virtual server is approximately 4 seconds longer than *T*v shown in Table 3. Therefore, the recovery time of the server function is also approximately 4 seconds longer than the result shown in Fig. 13. We consider that these results occur because the startup processing of the virtual server overlapped. Virtual server FTPb is approximately 1 second shorter than the startup time of the other virtual servers because this server starts approximately 4 seconds later, thereby reducing overlap. Furthermore, in the access disconnection time from the client, the difference between the mail server and the FTP server depends on the time difference of the network trouble detection.

4.4. Influence of client access

In the proposed system, the system data of virtual servers is saved to the shared area on file server 1 and user data used to offer services to clients are saved on file server 2. The management server accesses file server 2, when clients require data in the management server. Access for data demands is assumed to influence the start time of the virtual server. The influence on the start time is shown in Fig. 16. The client applies an access load to management server FTP by download access, upload access, and CPU access. Download and upload accesses are performed using file transfer protocol. In the FTP server, user data targeted for downloading and uploading are accessed by file server 2 through NFS connection. In the experiment, the start time of the virtual server is measured during the state in which the client applies an access load to the FTP server. In the access load of downloading and uploading, 50 simultaneous accesses of 10^8 B data are performed. In the CPU access load, 50 simultaneous random number operations are performed 10^8 times.

As shown in Table 3, start time *T*v in a state without access from the client is 4.67 s. Start times in the download load and the upload load are 4.82 s and 4.79 s, respectively. Access for data demands does not influence start times. In the CPU load, *T*v is 6.78 s. An increase of approximately 2 seconds is observed due to the load. However, this influence is not a critical problem when we consider the frequency of server trouble and the initial cost of the server system.



		No access	Download	Upload	CPU
Time to activate suspended virtual server [s]	*T*v	4.67	4.82	4.79	6.78

Fig. 16 Influence of the start time of the virtual server on server load.

5. Conclusions

We herein proposed a server management system incorporating a P2P method for constructing a high-availability server system. In order to construct the proposed system, an executer system to prevent the dispersion of a management program in consideration of a P2P method, a server management priority to prevent a malfunction in recovery processing, and a dynamic management extension method to deal with problems that occur simultaneously on multiple servers were proposed and described herein. In addition, we adopted a virtual server startup method that considers the load state of management servers and the server type of the troubled server. Furthermore, an experimental server system using the proposed method was constructed, and the architecture was presented. Experiments to reproduce several types of trouble in the server system were conducted, and the recovery times of the target server function and the target server, which experienced trouble, were measured and analyzed. The proposed system was found to be able to recover the server function or the target server even when troubles simultaneously occur on several target servers.

References

[1] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. C.-Pastor, and A. Monje, "On the Optimal Allocation of Virtual Resources in Cloud Computing Networks," *IEEE Trans. Comput.*, Vol. 62, No. 4/6, 2013, pp. 1060-1071.

[2] M. Kitamura and C. Fujihashi, "Analysis of Response Time Characteristics of a Real Computer by Heterogeneous-Task Bursty-Arrival Models," *IEICE Trans.*, Vol. J81-B-I No. 4, 1998, pp. 243-253.

[3] Y. Zhu, W. Wu, J. Willson, L. Ding, L. Wu, D. Li, and W. Lee, "An Approximation Algorithm for Client Assignment in Client/Server Systems," *Proc. IEEE INFOCOM 2014*, Vol. 3, 2014, pp. 2777-2785.

[4] Y. Yu, C. Qian, "FTDC: A fault-tolerant server-centric data center network," *Proc. IEEE Conf.*, Vol. 2016 No. IWQos, 2016, pp. 1-2.

[5] Z. Li, Y. Yang, "RRect: A Novel Server-centric Data Center Network with High Availability," *Proc. IEEE Conf.*, Vol. 2016 No. ICPP, 2016, pp. 41-46.

[6] K. Ali Bashir, Y. Ohsita, M. Murata, "A Distributed Virtual Data Center Network Architecture for the Future Internet," *IEICE Technical Report*, Vol. 114 No.478, 2015, pp. 261-266.

[7] D. Irie, H. Miwa, "Network Design Method by Finding Server Placement and Protected Links to Keep Connectivity to Servers Against Link Failures," *Proc. IEEE Conf.*, Vol 2016 No. INCos, 2016, pp. 439-444.

[8] H. Otsuka, K. Joshi, M. Hiltunen, S. Daniels, and Y. Matsumoto, "Online Failure Prediction with Accurate Failure Localization in Cloud Infrastructures," *IEICE Technical Report*, Vol. 113 No. 496, 2014, pp. 7-12.

[9] J. Xiao, B. Wu, L. Zhang, H. Wen, X. Jiang, and P.-H. Ho, "Joint Design on DCN Placement and Survivable Cloud Service Provision over All-Optical Mesh Networks," *IEEE Trans. Commun.*, Vol. 62 No. 1, 2014, pp. 235-245.

[10] Q. Zhang, M.-F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable Virtual Data Center Embedding in Clouds," *Proc. IEEE INFOCOM 2014*, Vol. 1, 2014, pp. 289-297.

[11] M.-G. Rabbani, M.-F. Zhani, and R. Boutaba, "On Achieving High Survivability in Virtualized Data Centers," *IEICE Trans. Commun.*, Vol. E97-B No. 1, 2014, pp. 10-18.

[12] W.-L. Yeow, C. Westphal, and U.-C. Kozat, "Highly Available Virtual Machines with Network Coding," *Proc. IEEE INFOCOM 2011*, Vol. 1, 2011, pp. 386-390.

[13] M. Kitamura, "Configuring a Low-cost, Power-saving Multiple Server Backup System: Experimental Results," *IEICE Trans. Commun.*, Vol. E95-B No. 1, 2012, pp. 189-197.

[14] M. Kitamura, "Configuration of a Power-saving High-availability Server System Incorporating a Hybrid Operation Method," *JISSJ*, Vol. 10 No. 2, 2015, pp. 1-17.

[15] T.Yagata, "Oracle consult speaks! Super fast! Various fault detection architecture supporting Oracle Coherence high availability," *Oracle*, 2011.11, http://www.oracle.com/technetwork/jp/ondemand/middleware/application-grid/e-13-coherence-1484719-ja.pdf

[16] T. Ono and K. Ueda, "Failure Detection and Monitoring Method for High Availability Distributed Cluster," *IEIEC Technical Report*, Vol. 116 No. 484, 2017, pp. 137-142.

[17] H. Wang and H. Nakazato, "Failure Detection in P2P-Grid System," *IEICE Trans. Inf. & Syst.*, Vol. E98-D No.12, 2015, pp. 2123-2131.

[18] H. Yamamoto, "Automatic Disaster Recovery System Talk of Tsukuyomi," *Cybozu*, 2013.9, http://blog.cybozu.io/entry/5799

Authors Biography

Mitsuyoshi KITAMURA

He received his B.E. degree from Tokyo Polytechnic University in 1984. In 1990, he became a research assistant at Tokyo Polytechnic University, where he became an assistant professor in 2003. His current interests are the optimum design and construction of server-client systems and the analysis of various characteristics of server systems.

Yoshihisa UDAGAWA

He received his Ph.D. in aeronautical science from the University of Tokyo in 1982. From 1982 to 2010, he worked on and led various projects related to databases and Web-based system developed for industrial use. Since April 2010, he has served as a professor in the computer science department at Tokyo Polytechnic University. His research interests include system architecture, data mining, and quality management of systems development.

Hitoshi NAKAGOME

He received his B.E. degree in 2016 from Tokyo Polytechnic University, where he is currently working toward the M.S. degree. He is engaged in developing software for experimental systems and analyzing performance data.

Youta SHIMIZU

He received his B.E. degree in 2017 from Tokyo Polytechnic University, where he is currently working toward the M.S. degree. He is in charge of implementation of experimental hardware systems and analyses of power-saving data.