PAPER

Load Fluctuation-Based Dynamic File Allocation with Cost-Effective Mirror Function

Jun TAKAHASHI[†], **Akiko NAKANIWA**[††a)], ***Regular Members***,
Yasutomo ABE[††], ***Student Member***, **Hiroyuki EBARA**[††],
and **Hiromi OKADA**[††], ***Regular Members***

SUMMARY Mirroring of network servers has been considered to be effective for load balancing. However, the cost of setting up new mirror servers is enormously high. In this paper, we propose a dynamic file allocation model with a simple mirroring function for handling significant changes of network traffic in the Internet. According to the load fluctuation, we can dynamically reallocate files using this model. We show that our model accomplishes satisfactory performance and reduces cost by adding a simple mirroring function to all existent servers instead of setting up mirror servers afresh.

key words: *load balancing, simple mirroring function, dynamic file allocation, Internet, distributed networks*

1. Introduction

There has been an explosive increase in the number of users in multimedia networks, particularly on the Internet. Furthermore, network services which enable users to connect to the Internet using mobile computers including cellular phones are spreading rapidly. This is causing network servers to become overloaded, and we are faced with several essential issues, such as the decline of reliability, the increase of access delay, and so on. In finding a solution for these issues, the mirroring of network servers has been considered and various studies relative to these issues have been carried out widely. Several mirror servers, which store copies of files and have the same function as network servers, have been set up in networks for the purpose of load balancing.

Distributed allocation of mirror servers results in prompt access by users and improvement of system reliability. The authors have studied the reliability-based optimal allocation of mirror servers and concluded that there are benefits in assigning as many mirror servers as possible in terms of system reliability [10], [12]. However, it is also true that distributed mirror allocation causes the cost of setting up mirror servers and managing the whole system to increase. Also, there have been many works on file allocation which have not taken into consideration the load fluctuation across time [1], [4]–[12]. These studies consider the situation that the network traffic pattern is steady. However, in fact, the user access pattern changes from moment to moment.

Considering these issues, we propose a dynamic file allocation model. This model is characterized by the following two points; a simple mirroring function and the dynamic file allocation according to the load fluctuation, using this function.

First, we add the simple mirroring function to all existent servers. In this study, we add disk space, which enables the dynamic updating of files, to each server instead of setting up mirror servers afresh. This means that each server has both (A) disk space which holds specific files for a long period and (B) disk space where it is permitted to update files dynamically. We assume that spaces (A) and (B) are called static space and dynamic space, respectively. Both the static space and the dynamic space, particularly the dynamic space, have capacity restrictions. It is assumed that each server deletes preferentially from the oldest files or files that have been least accessed by users, if the total size of files exceeds the capacity of the dynamic space at the server.

Second, we introduce a dynamic file allocation algorithm into this model. In practical systems, it is important how to allocate files dynamically on networks according to the temporal fluctuation of user access. This algorithm works only for the file allocation in the dynamic space. We describe the details of the algorithm in Sect. 2.3.

In this paper, we show that this model can accomplish sufficient performance and reduce cost by adding a simple mirroring function to all existent servers instead of setting up servers afresh. This model enables us to evaluate various characteristics concerning load fluctuation.

In this paper, we introduce the optimal dynamic file allocation model with simple mirroring function in Sect. 2. In Sect. 3 we show the numerical results obtained by the dynamic file allocation method. Finally, we conclude this paper in Sect. 4.

Manuscript received March 28, 2002.
Manuscript revised July 24, 2002.

[†]The author is with Fujitsu Kansai-Chubu Net-Tech Limited.

[††]The authors are with Kansai University, Suita-shi, 564-8680 Japan.

a) E-mail: akiko@rcss.kansai-u.ac.jp

2. Dynamic File Allocation Model with Simple Mirroring Function

2.1 System Model

Figure 1 shows an example of a system model used in this work. We give a topology of a network by an adjacency matrix **A**, whose elements have weights relating to the distance between a pair of nodes. We calculate the shortest path matrix $\mathbf{Q}(q_{ij})$, and formulate the cost, the reliability, and the delay using the matrix **Q**. This enables us to deal with various kinds of network structures, including tree structures and mesh structures. In addition, we may take into account problems related to communication costs and delays by using the weight of the distance.

There exist n servers in the system, and each server is denoted by N_i $(1 \leqq i \leqq n)$. It is assumed that all nodes originally function as both routers and servers, which hold their own original files as well as copies of the original files on other servers. It is assumed that each user in the system is connected to one of the servers, called a local node. The users connected with the server N_i are called *i-users*. Server N_i has the storage capacity of B_i. As mentioned before, each server has dynamic space and static space. S_CAPA_i is the capacity of the static space at the server N_i and D_CAPA_i is the capacity of the dynamic space at the server N_i. That is, $B_i = S\_CAPA_i + D\_CAPA_i$.

The number of distinct files in the system is m, and each file is denoted by M_k $(1 \leqq k \leqq m)$. The maximum number of copies of the homogeneous file M_k that can exist in the whole system is denoted by r_k. Also, it is assumed that the size of the file M_k is denoted by F_k. Each server can store these files as long as the total size of files does not exceed the storage capacity. Each server has only one copy of the same kind of file.

File access flow to server N_i at time t is denoted by $\alpha_i(t)$. Also, let $P_{ik}(t)$ denote the access probability from *i-users* for the files M_k at time t $(\sum_k P_{ik}(t) = 1)$. The parameter $P_{ik}(t)$ determines the traffic matrix in the system at time t.

Moreover, we define λ_i as the failure rate of a server N_i, μ_i as the service rate, and e_{ij} as the communication rate of a link between servers N_i and N_j. In this paper, it is assumed, for analytical simplicity, that the communication rate of each link has the identical value E. We also do not consider the case of the failure of individual links in networks.

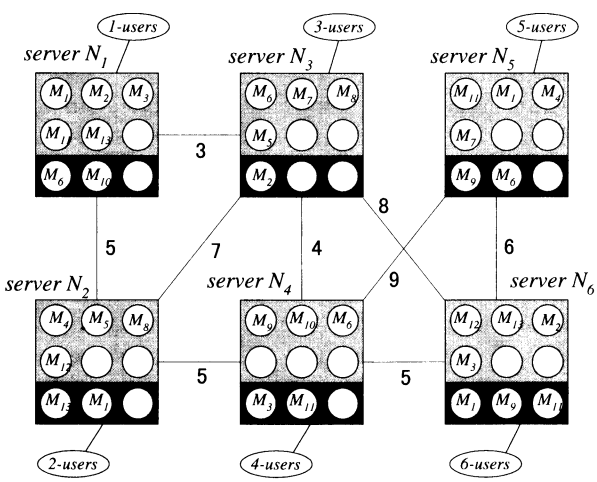


Fig. 1 An example of a system model ($n = 6$).

2.2 Formulation

Our objective is to dynamically find the optimal file allocation such that the total cost in the whole system per unit time is minimized. In this model, we formulate the optimization problem of dynamic file allocation, in which the total cost in the whole system is minimized subject to the system reliability and the communication delay as restrictive conditions. We define two types of 0-1 variables and formulate this optimization problem as a 0-1 integer programming problem using these 0-1 variables

- The variables on the allocation of files in static space X_{ik} [1].

$$X_{ik} = \begin{cases} 1 & \text{(the file } M_k \text{ is stored in} \\ & \text{the static space of the server } N_i) \\ 0 & \text{(otherwise)} \end{cases} \tag{1}$$

- The variables on the allocation of files in dynamic space $Y_{ik}(t)$.

$$Y_{ik}(t) = \begin{cases} 1 & \text{(the file } M_k \text{ is stored in} \\ & \text{the dynamic space of the} \\ & \text{the server } N_i \text{ at time } t) \\ 0 & \text{(otherwise)} \end{cases} \tag{2}$$

2.2.1 Primary Conditions

Under the assumptions presented in Sect. 2.1, we add the restriction on the maximum number of copies and the capacity restriction as primary conditions.

1. Restriction on the maximum number of copies
   It is assumed that the maximum number of copies of the homogeneous file M_k that can exist in the whole system is r_k, that is

$$1 \leqq \sum_i (X_{ik} + Y_{ik}(t)) \leqq r_k \tag{3}$$

2. Capacity restriction
   Each server N_i can store files so long as the total size of files does not exceed the capacity of the

node, that is

$$\sum_k X_{ik}F_k \leqq S\_CAPA_i$$
$$\sum_k Y_{ik}(t)F_k \leqq D\_CAPA_i \quad (4)$$

2.2.2 Objective Function

We formulate the total cost per unit time in the whole system as the objective function. In this paper, we define the total cost as the cost required to manage the whole system per unit time. It is assumed that the total cost $C(t)$ consists of the storage cost $CS(t)$, the communication cost $CT(t)$ and the reallocation cost $CR(t)$. The total cost is as follows.

$$C(t) = CS(t) + CT(t) + CR(t) \quad (5)$$

Next, we give full details of the storage cost, the communication cost and the reallocation cost.

1. Storage Cost $CS(t)$
   We assume that the storage cost is required for each server to store files per unit time. It is charged on the basis of unit time, and is not dependent on the frequency of accesses. In this study, the storage cost depends on the size of all files stored at a server. Let Cs be the storage cost coefficient, which has a fixed value that is common for all servers.
   $$CS(t) = Cs\sum_i\sum_k \left(X_{ik} + Y_{ik}(t)\right)F_k \quad (6)$$
2. Communication Cost $CT(t)$
   We assume that the communication cost is required to communicate between a user site and his/her requested file site per unit time. For example, when an *i-user* accesses a file M_k, the communication cost will be the cost of communicating between an *i-user* and a server that stores the file M_k. If a local server N_i stores the file M_k, the communication cost is the cost required to communicate only between an *i-user* and the local server. Also, if a file requested from a user is not stored in the local server and several remote servers store the file, the user accesses the servers that are the shortest distance from the local server. Let Ctt be the communication cost coefficient that has a fixed value that is common for all files.
   Using this coefficient and 0-1 variables X_{ik} and $Y_{ik}(t)$, we formulate the communication cost as follows.
   $$CT(t) = \sum_i\sum_k\Big(\left(1-(1-X_{ik})(1-Y_{ik}(t))\right) + (1-X_{ik})(1-Y_{ik}(t)) \cdot \left(1-(1-X_{hk})(1-Y_{hk}(t))\right)q_{ih}\Big) \cdot CttF_k\alpha_i(t)P_{ik}(t) \quad (7)$$
   Here, it is assumed that the index h equals the value that minimizes the following function on the positive side. That is, the function $h_{\min}(h)$ in Eq. (8) is the function that indicates the subscript of the server which minimize the right-hand side of this equation, that gives the distance from the local server. q_{ih} is an element of the shortest path matrix $\mathbf{Q}$ and denotes the distance between a local server S_i and a remote server S_h.
   $$h_{\min}(h) = (1-(1-X_{hk})(1-Y_{hk}(t))) \cdot q_{ih} \quad (8)$$
3. Reallocation Cost $CR(t)$
   We assume that the reallocation cost is required for each server to reallocate files according to the load fluctuation per unit time. In this paper, the reallocation of files means the addition or deletion of files in each server. This cost includes the cost to process the reallocation such as addition and deletion of files and the cost to transmit files to be reallocated. The reallocation cost is as follows.
   $$CR(t) = \sum_i\sum_k Cr|Y_{ik}(t) - Y_{ik}(t-1)|F_k \quad (9)$$
   Here, Cr is the reallocation cost coefficient which is common for all files.

2.2.3 Restrictive Conditions

We formulate the system reliability and the communication delay as the restrictive conditions for the optimal dynamic file allocation problem.

1. System Reliability $R(t)$
   The system reliability is the reliability per unit time in the whole system. In this paper, we define the system reliability as the mean of the success rate of each access from each user to each file. We consider all required servers to complete the access from each user to each file, and calculate the reliability of the combination of these servers using the reliability of each server as the success rate of each access.
   Using common probability formulas [2], [3], the reliability of a server S_i is formulated as follows.
   $$R_i(t) = \exp\Bigg(-\lambda_i\sum_j\sum_k \frac{\alpha_j(t)P_{jk}F_k}{\mu_i} \cdot \Big(\Delta_{ji}\Big(1-(1-X_{jk})(1-Y_{jk}(t)) - \beta\Big) + (1-\Delta_{ji})\Big((1-X_{jk})(1-Y_{jk}(t))\Delta_{ih} + \beta\Big)\Big)\Bigg) \quad (10)$$

Here, Δ_{ji} is defined as follows (Kronecker's delta); if $j = i$, the value of Δ_{ji} is equal to 1; otherwise, the value of Δ_{ji} is equal to 0. Index h is equal to the value that minimizes Eq. (8) on the positive side.

In Eq. (10), the term $\Delta_{ji}\big(1-(1-X_{jk})\,(1-Y_{jk}(t))-\beta\big)$ refers to the case of accesses from local users, and the term $(1-\Delta_{ji})\,\big((1-X_{jk})\,(1-Y_{jk}(t))\,\Delta_{ih}+\beta\big)$ refers to the case of accesses from remote users.

We may guess that remote access causes greater load since several nodes have to relay the access. We introduce β as the coefficient that represents the difference in the amount of load between local access and remote access due to this fact.

Using this server reliability, we calculate the system reliability. In the following, we describe the procedure of the calculation.

Step.1 We calculate *R_path* as the reliability of all servers that are required to be available for the user to succeed in accessing each file as follows. In the case that there exist several servers which store the file requested by the user, we calculate *R_path* for all paths from the user to these servers.

$$R\_path = \prod_i R_i(t) \tag{11}$$

Step.2 In the case that there is only one server which stores the requested file, we calculate $R\_temp(j, M_k)$ as the success rate of the access from *j-user* to the file M_k as follows.

$$R\_temp(j, M_k) = R\_path \tag{12}$$

Otherwise, $R\_temp(j, M_k)$ is as follows.

$$R\_temp(j, M_k) = 1 - \prod (1 - R\_path) \tag{13}$$

Step.3 We calculate the mean of $R\_temp(j, M_k)$ as the system reliability per unit time $R(t)$.

$$R(t) = \frac{\sum_j \sum_k R\_temp(j, M_k)}{n \cdot m} \geqq R_{\min} \tag{14}$$

Here, we define $R_{\min}$ as the minimum system reliability. The system reliability has to be larger than or equal to $R_{\min}$.

2. Communication Delay $DT(t)$

The communication delay is the time spent in the communication required for a user to access a file. The communication delay is denoted by $DT(t)$:

$$DT(t) = \sum_i \sum_k \Big(\big(1 - (1 - X_{ik})\,(1 - Y_{ik}(t))\big) + (1 - X_{ik})\,(1 - Y_{ik}(t)) \cdot \big(1 - (1 - X_{hk})\,(1 - Y_{hk}(t))\big) q_h \Big) \cdot Dtt\alpha_i(t) P_{ik}(t) \leqq DT_{\max} \tag{15}$$

Here, we define $DT_{\max}$ as the maximum communication delay, and $DT(t)$ must not exceed $DT_{\max}$. Dtt is the communication delay coefficient which has a fixed value that is common for all files. It is also assumed that index h equals the value that minimizes the function $h_{\min}(h)$ (in Eq. (8)) on the positive side, too.

2.3 Dynamic File Allocation Algorithm

In the dynamic file allocation problem, it is important how files are optimally allocated in networks according to temporal fluctuation of user accesses. If we keep the initial file allocation regardless of load fluctuation, user accesses become more likely to rely on the remote servers and the communication cost increases. Also, this causes user accesses to be concentrated at particular servers. Therefore, the allocation of files needs to be updated according to the load fluctuation. In this model, we introduce a dynamic file allocation algorithm for this purpose.

If we apply conventional methods directly to the dynamic file allocation, the optimization of file reallocation in the entire system has to be executed at every unit time to deal with load fluctuation. In our proposed dynamic file allocation model with simple mirroring function, we need to reallocate files only in limited space, that is, dynamic space. We aim to reduce the reallocation cost that is required to reallocate files by using this simple mirroring function instead of replacing all files in the entire system.

In the following, we describe the dynamic file allocation algorithm using the simple mirroring function in our proposed model.

2.3.1 Initial File Allocation

First, it is assumed that the file allocation in the static space is performed randomly. On the other hand, in the dynamic space, no files are allocated at the beginning. Then as long as the total size of files does not exceed the capacity of the dynamic space at each server, those files are stored according to their popularity among local users. In other words, the files that are the most frequently accessed are kept on the server.

2.3.2 File Reallocation in the Dynamic Space

After the initial allocation of files is determined, the user accesses momentarily change as time goes by. Each server starts to store files according to their popularity among its local users, as the user access pattern varies. If the dynamic space at the server is not still filled with

files, those files can be stored without any problems; otherwise, each server has to delete preferentially starting from files accessed the least among the files stored at the server, and replaces an old one with the new one. This is the way each server reallocates files. In this model, we consider the cost required for reallocating files. We may guess that replacing all files is not always the optimal choice.

2.3.3 Dynamic File Allocation Algorithm

In the following, we give a description of the algorithm of dynamic file allocation. As described in Sect. 2.2, we formulate the optimization problem of dynamic file allocation as a 0-1 integer programming model. However, the 0-1 integer programming model is famous for its combinatorial explosion and it is well-known that when using an exact method, the scale of problems which can be practically solved is limited and it takes a very long time to solve even small problems. In this model, we optimize the dynamic file allocation using the Greedy Method [8] as the approximate method. Namely, we apply the Greedy Method to the dynamic space of each server to dynamically optimize file allocation using a simple mirroring function. In the Greedy Method, we take the access frequency to files as the local critical values, that is $\alpha(t) \times P_{ik}(t)$, which is denoted by $Af_{ik}(t)$ hereafter. In [8], it has been ensured that this Greedy Method is a very efficient algorithm in terms of both accuracy and computing time.

Step.0 [Initial allocation] We fix the variable $Y_{ik}(t) = 1$ in the order of the value of the corresponding access frequency $Af_{ik}(t)$, so long as all restrictive conditions are satisfied.

Step.1 As one unit time passes, execute the dynamic file allocation, the procedures of which are described from step 2.

Step.2 Calculate the access frequency $Af_{ik}(t)$ for each file at time t.

Step.3 Sort the variables $Y_{ik}(t)$ in the decreasing order of the corresponding $Af_{ik}(t)$.

Step.4 Repeat the procedures from steps 5 to 9 until the last variable of the sorted sequence in the previous procedure.

Step.5 Fix the variables $Y_{ik}(t) = 1$ for the file whose access frequency $Af_{ik}(t)$ is maximum at this time.

Step.6 Check whether the capacity restrictive condition is satisfied or not. If the capacity restrictive condition is not satisfied, we fix the variables to 0 for the file whose access frequency is minimum, and we fix the variables $Y_{ik}(t) = 1$ for the file whose access frequency $Af_{ik}(t)$ is maximum. Fix $Af_{ik}(t) = 0$.

Step.7 Check whether the rest of the restrictive conditions are satisfied or not. If the restrictive conditions are not satisfied, we return the value of the variables $Y_{ik}(t)$ to 0, and go back to step 5.

Step.8 Compare the variable $Y_{ik}(t)$ with the variable $Y_{ik}(t-1)$. If their values are different, we calculate the reallocation cost. Otherwise, we do not calculate the reallocation cost.

Step.9 Calculate the total cost $C(t)$ for the current allocation determined in the previous procedures, and commit the file allocation with the total cost to the memory.

Step.10 We select the file allocation that has the minimum total cost among all allocations calculated in step 9, and consider it as the optimal solution at time t.

3. Numerical Results and Considerations

3.1 File Allocation Methods for Performance Comparison

In order to demonstrate the usefulness of our optimal dynamic file allocation model, we compare three models, that is the proposed simple mirroring model (I), the conventional model (II) and the static allocation model (III). In the static allocation model (III), the initial file allocation is determined randomly, and then the allocation of files is fixed regardless of the temporal fluctuation of user accesses. In the following, we give details of the conventional model (II).

Conventional Model

In contrast to the proposed simple mirroring model, in which we reallocate files only in limited space, that is dynamic space, the conventional model reallocates files in the entire system. Namely, in the conventional model, the optimization of file reallocation in the entire system (i.e., reconfiguration) is executed at every unit time to deal with load fluctuation. Similarly to the simple mirroring model, we apply the Greedy Method [8] to the optimization of file allocation.

Here, we similarly consider the total cost as the objective function of the optimization. The total cost in this model includes the reconfiguration cost instead of the reallocation cost in our proposed model, as well as the storage cost and the communication cost. The reconfiguration cost is the cost of reconfiguring the entire file allocation in the system per unit time and consists of the reallocation cost and the management cost of coping with the reconfiguration.

The conventional model first determines the initial file allocation randomly. Next, at every unit time, the entire file allocation is reconfigured according to the access pattern at that time so that the total cost is minimized subject to the system reliability and the communication delay.

3.2 System Parameters for Numerical Results

We quantitatively show the general characteristics on the load balancing by the dynamic distributed allocation of files. Here, we consider the system parameters as follows. There exist 20 servers ($n = 20$), and the capacity of each server is 200 [Gbyte] ($B_i = 200000, 1 \leqq i \leqq 20$). Here, the rate of the dynamic space is 25% (50 [Gbyte]). Also, there exist 500 kinds of files in the whole system ($k = 500$), and the size of each file is 1 [Gbyte] ($F_k = 1000, 1 \leqq k \leqq 500$). The number of copies of files that are allocated in the whole system at each server on the initial allocation is 3 ($r_k = 3, 1 \leqq k \leqq 500$). Moreover, in this study, we introduce the following log-normal distribution (Eq. (16)) to the access distribution to each server (Fig. 2). The access distribution is expressed by using a function of log-normal distribution. This function is appropriate to assume for the situation such as video on demand, where the user access flow to a file is likely to take the largest value when it becomes available and then it becomes smaller with time. Also, in this situation, the time between when the file becomes available and when the user access flow takes the largest value and the value it takes when it is the maximum are not uniform. This function is capable of representing a range from a burst of access to relatively gentle access.

$$\alpha_i(t, t_i) = \frac{A_i}{\sqrt{2\pi}\sigma(t-t_i)} \exp\left(\frac{-(\log(t-t_i)-\gamma)^2}{2\sigma^2}\right) \quad (t \geqq t_i) \tag{16}$$

If $t < t_i$, $\alpha_i(t, t_i)$ is equal to 0. Here, we define t_i as the time of initial generation on each access flow, A_i as the variables that indicate a scale of each server, γ as the mean, and σ as the variance.

In addition, to make the change of popular order of files natural to some degree, the log-normal distribution is also applied to the access probability $P_{ik}(t)$. First, we generate a variety of access distributions of the number of all combinations of servers and files and assign them one by one. Next, the values of each access distribution at each time are compared and the popular order is applied to $P_{ik}(t)$. Eventually, according to the Zipf' law, the access probability to the file M_k from *i-users* is assigned based on the order.

We obtain the concrete assignment of files to servers as the optimal solution at each time. On the basis of this assignment, we calculate the total cost, the system reliability, the communication delay, and so forth. We describe our considerations regarding the optimization results using these criterions.

3.3 Performance Comparison

Our objective is to reveal that the proposed dynamic file allocation model with simple mirroring function can reduce the total cost while keeping the system reliability sufficiently high and the communication delay sufficiently low. It can be easily imagined that when not using a simple mirroring function, file reallocation in the entire system at every unit time causes a serious increase in the cost. Here, we show the characteristics of the cost, system reliability, and delay when using our proposed model, the conventional model and the static allocation model, respectively. By comparing our proposed model with the conventional model, we show that without much degradation of system reliability and delay, the cost is decreased when using our proposed simple mirroring model.

3.3.1 Cost Characteristics on Load Fluctuation

Here, we show the characteristics of the total cost on the load fluctuation.

In Fig. 3, we show the total cost in three models vs. unit time. As seen in this figure, the total cost in model (II) is much larger than the others. This is because the cost of reconfiguring the entire file allocation is extremely high. When we compare model (I) to model (III), the cost of the latter is seen to be larger. This is because more users tend to access the remote servers

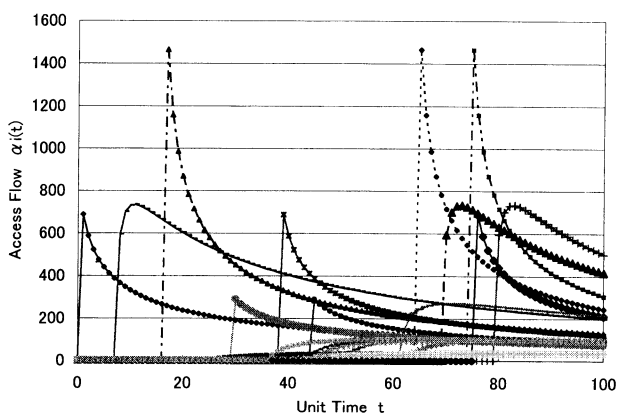


Fig. 2 Access flows at each server.

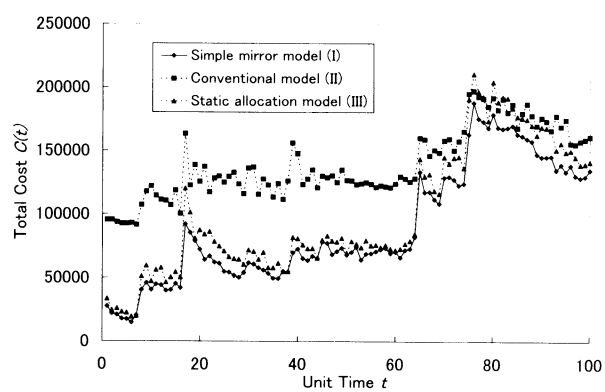


Fig. 3 Cost characteristics on load fluctuation.

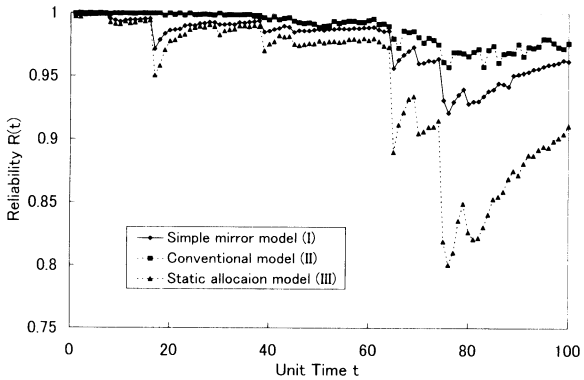


Fig. 4 Reliability characteristics on load fluctuation.

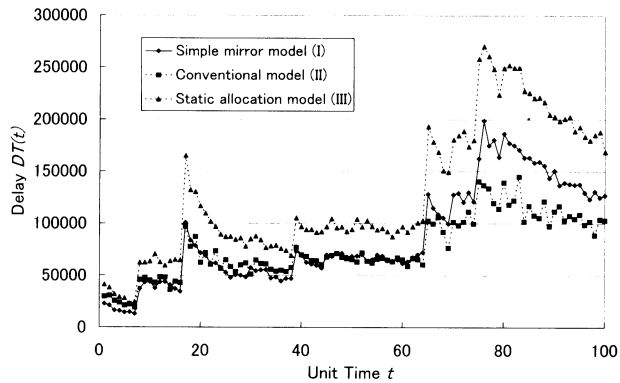


Fig. 5 Delay characteristics on load fluctuation.

and the communication cost increases due to the fixed file allocation. Consequently, it is obvious that our dynamic file allocation model can reduce costs by adding a simple mirroring function.

3.3.2 Reliability Characteristics on Load Fluctuation

We show the characteristics of the system reliability on the load fluctuation.

In Fig. 4, we show the system reliability in the three models vs. unit time. As shown in this figure, when the access frequency increases as time passes, the system reliability in model (III) decreases markedly. It is considered that the load of user accesses cannot be distributed due to the fixed allocation of files. However, the load of user accesses can be distributed in the case of using the simple mirroring. The system reliability in the case of model (II) is the best, since the complete optimization is executed at each time. On the other hand, the system reliability in the case of model (I) is somewhat inferior to that in the case of model (II). However, we can still maintain sufficiently high system reliability with our proposed model (I), considering the reallocation of files in the limited space compared to model (II).

Consequently, while our proposed model is slightly inferior to the conventional model in terms of system reliability, the fact that cost performance is fairly good indicates that our proposed simple mirroring model is sufficiently useful.

3.3.3 Delay Characteristics on Load Fluctuation

We show the influence of the characteristics of the communication delay on the load fluctuation.

In Fig. 5, we show the communication delay in the three models vs. unit time. As we may see from this figure, the communication delay in model (III) is longer than those of the other models. We consider that this is because more users tend to access the remote servers due to the fixed allocation of files. On the other hand, we can imagine that in model (I) and model (II), files

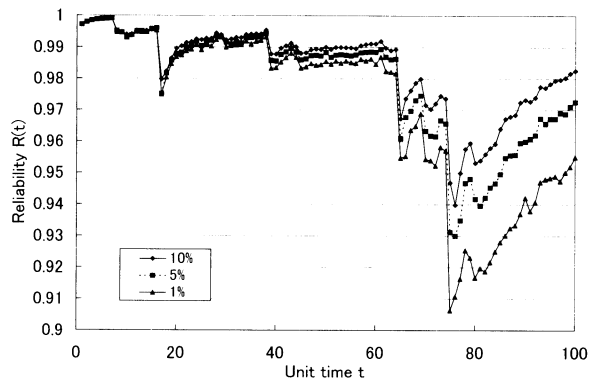


Fig. 6 Reliability characteristics with different size of dynamic space.

which are more frequently requested by users are allocated at their local servers. Our proposed model can also keep the communication delay sufficiently small.

3.4 Characteristics on Simple Mirroring Function

In this section, in order to show more details of the performance of the simple mirroring model, we compare characteristics between models with differing sizes of the dynamic space. Here, we compare three cases, i.e., 10.0% of the total storage capacity (20 [Gbyte]), 5.0% (10 [Gbyte]), and 1.0% (2 [Gbyte]). Similarly, we evaluate the system reliability, the cost, and the delay.

First, in Fig. 6, we show the system reliability in the three cases vs. unit time. As seen in this figure, the performance in the case of a larger dynamic space is better than those of the other cases. This is because it is possible to execute the reallocation of files that is more beneficial for user accesses due to an increase in the number of files that can be reallocated as the dynamic space becomes larger.

Second, in Fig. 7, we show the total cost in the three cases vs. unit time. As seen from this figure, the performance in the case of larger dynamic space is slightly better for the similar reason.

Third, in Fig. 8, we show the communication delay in the three cases vs. unit time. Similarly in this figure,

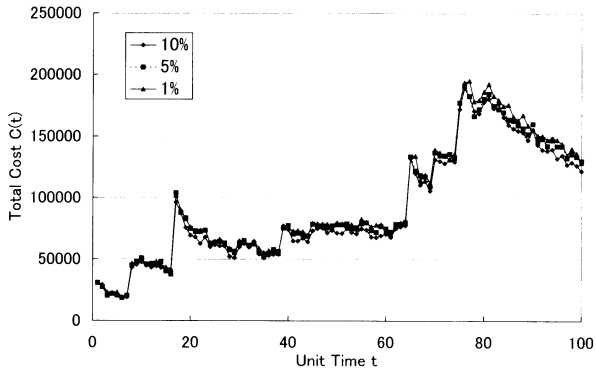


Fig. 7 Cost characteristics with different size of dynamic space.

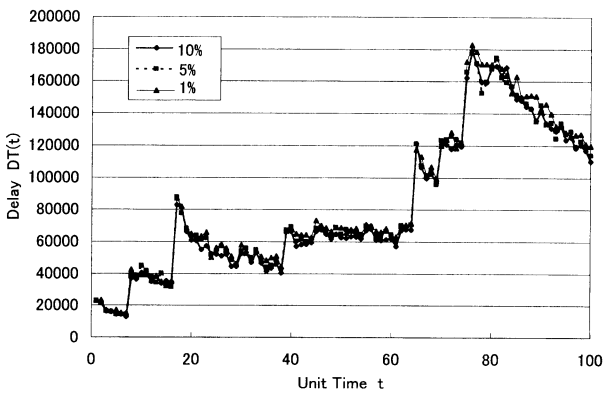


Fig. 8 Delay characteristics with different size of dynamic space.

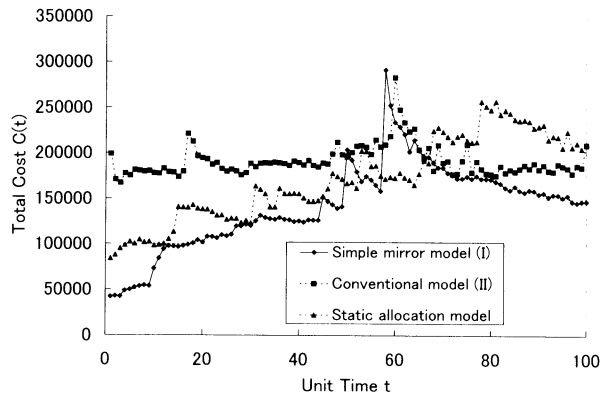


Fig. 9 Cost characteristics in access pattern 2.

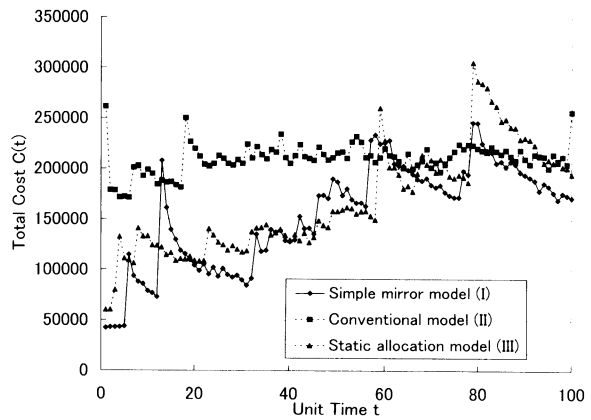


Fig. 10 Cost characteristics in access pattern 3.

the performance in the case of a larger dynamic space is somewhat better. We may consider that this is because fewer users need to access the remote servers due to an increase in the number of files that can be reallocated as the dynamic space becomes larger.

As a result, we may conclude that since there are hardly any differences in the cost among the cases, it would be better to assign as large a dynamic space as possible to realize better performance.

3.5 Reconfiguration Point of File Allocation

In the previous section, we showed that our proposed dynamic file allocation model is sufficiently useful. However, as seen from the results above, our proposed model becomes less useful in terms of cost, system reliability, and delay as time goes by. It would be even better to reconfigure the entire file allocation periodically. In this section, we attempt to find the best timing for the reconfiguration. The characteristics shown in Figs. 3, 4, and 5 are based on one access pattern (pattern 1). Here, we generate randomly two other access patterns in order to investigate more generally how we should determine the reconfiguration point.

First, in the case of pattern 1, the cost performance in our proposed model degrades significantly from time $t = 65$. It is considered that this is because there is a limit when dealing with the temporal fluctuation of user accesses by reallocating files only within the limited space, that is, the dynamic space.

Next, the time when the performance of our model reaches the limit, that is, "limit point" in the other patterns, can be seen in Figs. 9 and 10. The limit point in pattern 2 is near $t = 60$, and in pattern 3, it is near $t = 57$. In this study, we consider that it might depend on the accumulation of access flow when the performance of our model is degrading. Therefore, we calculate all access flows $\alpha(t)$ at each time and the total value until each time is defined as the accumulation of access flow $\alpha_{total}(t)$. In Fig. 11, we show the accumulation of access flow in each access pattern.

According to each limit point described above, we can locate the reconfiguration point in each access pattern in Fig. 11. Namely, we can obtain the accumulation of access flow $\alpha_{total}(t)$ corresponding to the limit point in each pattern. In pattern 1, $\alpha_{total}(t)$when $t = 65$ is nearly equal to 200000, in pattern 2, $\alpha_{total}(t)$ when $t = 60$ is nearly equal to 182000, and in pattern 3, $\alpha_{total}(t)$ when $t = 57$ is nearly equal to 200000. Therefore, in our proposed dynamic file allocation model, it is considered that the entire file allocation, including

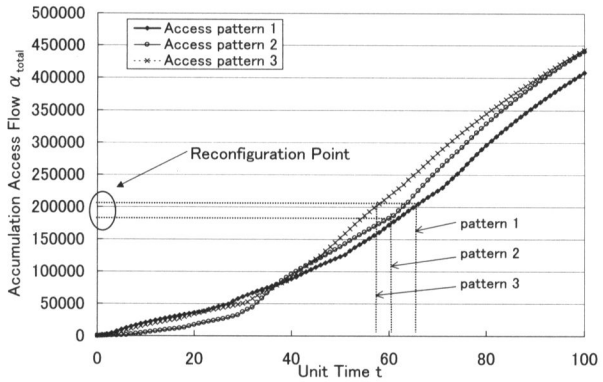


Fig. 11 Accumulation of access flow.

the static space, should be reconfigured when the accumulation of access flow reaches about 190000.

4. Conclusions

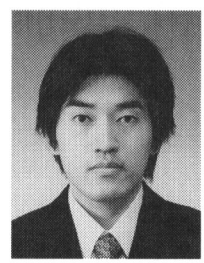
In this paper, we have proposed a dynamic file allocation model with a simple mirroring function. In this model, we have formulated the dynamic file allocation problem such that the total cost is minimized subject to the reliability and the delay, as a 0-1 integer programming model. As a result, we have shown that our model enables cost reduction while accomplishing satisfactory performance. We can conclude that our optimal dynamic file allocation model with a simple mirroring function is very useful in evaluating various essential issues associated with the load fluctuation in multimedia networks such as the Internet. In addition, we have found the reconfiguration point of the entire file allocation to make our model even more useful.

Acknowledgements

The part of this research was financially supported by the Kansai University Special Research Fund, 2002 and the Kansai University Grant-in-Aid for the Faculty Joint Research Program, 2002.

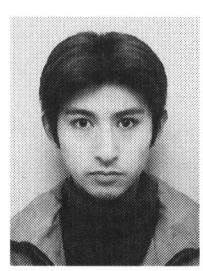
References

[1] W.W. Chu, "Optimal file allocation in a multiple computer system," IEEE Trans. Comput., vol.C-18, no.10, pp.885–890, Oct. 1969.
[2] A.K. Verma and M.T. Tamhankar, "Reliability-based optimal task-allocation in distributed-database management systems," IEEE Trans. Reliability, vol.46, no.4, pp.452–459, Dec. 1997.
[3] S.M. Shatz and J. Wang, "Models & algorithms for reliability-oriented task-allocation in redundant distributed-computer systems," IEEE Trans. Reliability, vol.38, no.1, pp.16–27, April 1989.
[4] C.C. Bisdikian and B.V. Patel, "Issues on movie allocation in distributed video-on-demand systems," Proc. IEEE ICC'95, pp.250–255, 1995.
[5] C.C. Bisdikian and B.V. Patel, "Cost-based program allocation for distributed multimedia-on-demand systems," IEEE Multimedia, pp.62–72, Fall 1996.
[6] S.A. Barnett and G.J. Anido, "A cost comparison of distributed and centralized approaches to video-on-demand," IEEE J. Sel. Areas Commun., vol.14, no.6, pp.1173–1183, Aug. 1996.
[7] A. Nakaniwa, H. Ebara, and H. Okada, "File allocation designs for distributed multimedia information networks," IEICE Trans. Commun., vol.E81-B, no.8, pp.1647–1655, Aug. 1998.
[8] A. Nakaniwa, M. Onishi, H. Ebara, and H. Okada, "File allocation in distributed multimedia information networks," Proc. IEEE GLOBECOM'98, pp.740–746, Nov. 1998.
[9] A. Nakaniwa, M. Onishi, H. Ebara, and H. Okada, "Sensitivity analysis of file allocation for distributed information networks," Proc. IEEE ICC'99, pp.1339–1345, June 1999.
[10] A. Nakaniwa, J. Takahashi, H. Ebara, and H. Okada, "Reliability-based optimal allocation of distributed mirror servers for Internet," Proc. IEEE GLOBECOM2000, pp.1571–1577, Nov. 2000.
[11] A. Nakaniwa, M. Onishi, H. Ebara, and H. Okada, "Sensitivity analysis in optimal design for distributed file allocation systems," IEICE Trans. Commun., vol.E84-B, no.6, pp.1655–1663, June 2001.
[12] A. Nakaniwa, J. Takahashi, H. Ebara, and H. Okada, "Reliability-based mirroring of servers in distributed networks," IEICE Trans. Commun., vol.E85-B, no.2, pp.540–549, Feb. 2002.

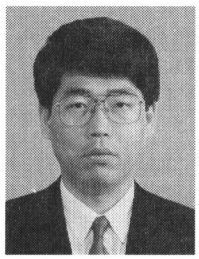


Jun Takahashi was born in 1977. He received his B.S. and M.S. degrees in Electronics Engineering from Kansai University, Japan, in 2000 and 2002, respectively. Currently he is with Fujitsu Kansai-Chubu Net-Tech Limited. His research interest is the design of reliable networks.

Akiko Nakaniwa received her B.S., M.S., and Ph.D. degrees in Electronics Engineering from Kansai University, Japan, in 1997, 1999, and 2002, respectively. Currently she is a post-doctoral fellow in the Research Center of Socionetwork Strategy, Kansai University. Her research interest includes the optimization design of distributed networks. She is a member of IEEE.



Yasutomo Abe received his B.S. degree in Electronics Engineering from Kansai University, Japan, in 2002. Currently he is studying in the graduate school of Kansai University. His research interest includes load-balancing.



Hiroyuki Ebara was born in 1958. He received his B.S., M.S., and Ph.D. degrees in Communications Engineering from Osaka University, Osaka, Japan, in 1982, 1984, and 1987, respectively. In 1987 he became an Assistant Professor at Osaka University. Since 1994 he has been with Kansai University, where he is currently an Associate Professor. His main research interests lie in computational geometry, combinatorial optimization, and parallel computing. Dr. Ebara is a member of IEEE, ACM, and SIAM.

Hiromi Okada received his B.S., M.S., and Ph.D. degrees in Communications Engineering, from Osaka University, Japan, in 1970, 1972, and 1975, respectively. From 1975 to 1983, he was an Assistant Professor at Osaka University. From 1983 to 1987, he was an Associate Professor at Kobe University. From 1987 to 1996, he was with Osaka University as an Associate Professor. Since April 1996, he has been with the department of Electronics Engineering, Faculty of Engineering, Kansai University as a Professor. His current research interests include ATM multicasting networks, wireless ATM networks, ad-hoc wireless networks, performance evaluation and optimization of distributed information networks. He is the author of several books entitled "Information Networks" (in Japanese, Baifukan, Tokyo), "Introduction of Computer Systems" (in Japanese, Shokodo, Tokyo) and so on. He is a member of IEEE, IPS (Information Processing Society) Japan.