論文 / 著書情報
Article / Book Information

論題(和文)	
Title(English)	Local Optimal File Delivery Scheduling in a Hop by Hop File Delivery System on a One Link Model
著者(和文)	鶴見宏美, 宮田高道, 山岡克式, 酒井善則
Authors(English)	Hiromi Tsurumi, Takamichi Miyata, Katsunori Yamaoka, Yoshinori Sakai
出典(和文)	, vol. E92-B, No. 01, pp. 34-45
Citation(English)	IEICE Transactions on Communications, vol. E92-B, No. 01, pp. 34-45
発行日 / Pub. date	2009, 1
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2009 Institute of Electronics, Information and Communication Engineers.

IEICE TRANS. COMMUN., VOL.E92–B, NO.1 JANUARY 2009

PAPER Special Section on Networking Technologies for Dependable Networks

Local Optimal File Delivery Scheduling in a Hop by Hop File Delivery System on a One Link Model

Hiromi TSURUMI[†a)], ***Student Member***, **Takamichi MIYATA**[†b)], **Katsunori YAMAOKA**[†c)], ***and*** **Yoshinori SAKAI**[†d)], ***Members***

SUMMARY Many content distribution systems such as CDN and P2P file sharing have been developed. In these systems, file-type contents require downloads to be completed before they can be played and they have no value before the download finishes. Therefore, a user's satisfaction depends on the length of the service latency. That is, the length of time from when the user issued a request until the user received an entire file. Reducing the sum of that time is necessary for the whole delivery system to satisfy users and maintain dependability on system performance. We discuss a hop-by-hop file delivery system suitable for delivering file contents whereby the sum of service latency is reduced by using the request conditions. Moreover, we propose a file delivery scheduling algorithm for a one-link model given that the content request frequency is unknown. The algorithm is based on a local optimal strategy. We performed a characteristic analysis by computer simulation. The results showed that our algorithm performs at nearly the theoretical efficiency limit of the hop-by-hop system when the request frequency distribution of each content has a deviation.

key words: *file delivery, scheduling, optimization, hop by hop*

1. Introduction

With the significant increase in network traffic, cache technologies, which can enhance the performance of systems by load balancing, have been implemented on a P2P (Peer-to-Peer) file-sharing system, CDN (Content Delivery Network), and Web server proxy system, for example [1]–[5]. In these systems, multiple copies of content from an original node (i.e., node means "peer" in the P2P file sharing system, and "server" in CDN and Web server proxy system) that has the original content are stored in caches of some other nodes. Then, the number of nodes, which can serve the same content, increases. Therefore, by storing many multiple copies of the same content in geographically dispersed caches, we can reduce the server's load, network congestion, and latencies experienced by users.

Storing multiple copies of content in caches of multiple nodes, which are located on a delivery path to users, has been implemented as a method of dispersing content on many kinds of systems. That method has been investigated in reported in studies, e.g., Freenet [1] and Winny, which are kinds of P2P file sharing systems, the transparent proxy system, which is a kind of structure of a Web server proxy system and CDN [4], [5]. In these systems, nodes that can provide content are capable of intercepting user requests and forwarding the requests to another node if the requested content is not present in their local cache. When content is found on a cache of a node, that content is delivered hop by hop through some nodes from which the request arrives (i.e., delivery path), and stored in their caches. This method effectively disperses many multiple copies over the network by following demands for the content. To avoid causing ambiguity, we call this delivery method "hop-by-hop content delivery," and it is illustrated in Fig. 1.

In the system that uses hop-by-hop content delivery, there are two basic types of content-delivery methods: media streaming, which includes video file streaming and live video streaming, and file downloading. While streaming enables the buffered downloaded content to be played before the download is completed, file downloading requires that the content be completely downloaded before it can be played, and it has no value before this. Therefore, the user's satisfaction depends on the length of the service latency, that is, the length of time from when he or she issued the request until he or she received the entire file, and reducing the sum of that time (a comprehensive performance indicator with mean service latency per request) is necessary for the whole delivery system to satisfy users. Some of the most important techniques to improve the total service latency are caching algorithms that decide how to maintain files to achieve high hit rates, file-placement algorithms that decide where we should place files, and file-delivery schedulings that decide how to deliver files. However, in this paper, we address the problems of file-delivery scheduling of hop-by-hop file delivery ("hop-by-hop content delivery" for files) to reduce the

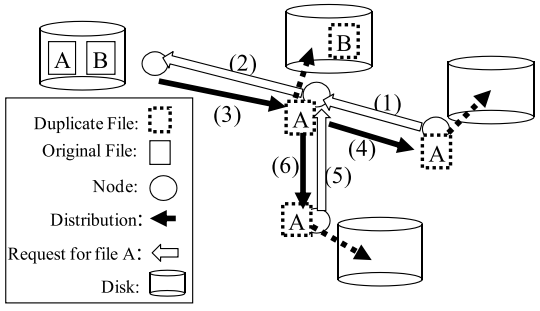


Fig. 1 Hop-by-hop file delivery system.

Manuscript received April 30, 2008.
Manuscript revised August 21, 2008.

[†]The authors are with Tokyo Institute of Technology, Tokyo, 152-8552 Japan.
a) E-mail: hiro@net.ss.titech.ac.jp
b) E-mail: miyata@ss.titech.ac.jp
c) E-mail: yamaoka@ss.titech.ac.jp
d) E-mail: ys@ss.titech.ac.jp

DOI: 10.1587/transcom.E92.B.34

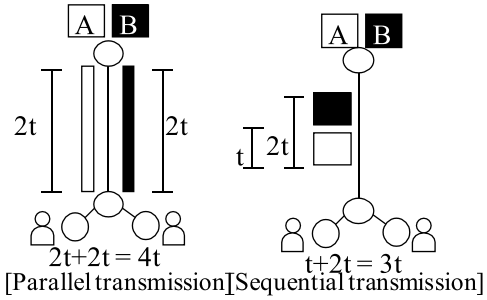


Fig. 2 Transmission methods.

total service latency. We leave the other components to other research [5], [6].

When there is a node at which a large number of requests arrive and which should deliver some of each file corresponding to each request through the same link, the link becomes a bottlenecked link. On bottlenecked links, the scheduling problem is important because it governs deciding which files should be delivered and how to deliver them, and that has a direct effect on the total service latency. Poor file-delivery scheduling can result in considerably longer service latencies, which impair dependability on system performance, while good scheduling can shorten those latencies and efficiently utilize resources such as network bandwidth. In this paper, we address the problems from the following perspectives.

1. *File transmission method*: Two file-type transmission methods can be used when one node is processing multiple requests for different files simultaneously. In parallel transmission, the bandwidth is divided to execute parallel file delivery. In sequential transmission, the files are delivered sequentially. Sequential transmission is better suited for delivering file content because the total service latency is generally shorter. We illustrated this in Fig. 2.
2. *Order in which the files are transmitted*: With sequential transmission, the total service latency for the whole delivery system depends on the order in which the files are transmitted when one node is processing multiple requests for different files simultaneously, because the value of serving the file is determined by the popularity of the file. For example, suppose node 1 has file A and file B, and node 1 receives more requests for file A than file B from node 2, which is located beyond a bottlenecked link almost simultaneously. In this case, if both file sizes are the same, node 1 should deliver file A first and then file B to node 2 to minimize the total service latency experienced by users. Moreover, if file A is very popular, i.e., many requests of file A will arrive afterward and file A will be sent in the order at each node on the delivery path, we can derive a large benefit from the cache. That is because file A can be dispersed faster over the network and users can get it from any cache of nodes near users or from their local caches directly with a considerably shorter service latency.
3. *Dynamic scheduling*: In an actual network environment, request arrival times are not known in advance, and the file transmission order that has already been decided will not always be appropriate afterward. Therefore, the scheduling should be changed appropriately in the middle of transmission to reduce the total service latency.

2. Related Work

Techniques to improve the total service latency include caching algorithms [7], [8], content-placement algorithms [5], [6], and content-delivery schedulings [9]–[18]. Our review in this section focus on the studies on content-delivery scheduling.

Some studies on the content-delivery scheduling focus on either a parallel transmission that downloads a content from multiple servers [9], [10] or transmission that downloads a content from single server [11]–[18].

[9] and [10] focus on parallel file delivery scheduling that downloads a file from multiple servers with caches in a P2P file-sharing system over links that have asymmetric upload and download bandwidths like those of an asymmetric digital subscriber line (ADSL). In the network, a file is divided into blocks, and each node has a portion of the blocks and nodes exchange blocks with other nodes. This method of file delivery reduces the completion time at which each node finishes collecting all blocks of a file. However, it does not account for the popularity of each block of a file. Therefore, this method does not work well for our purpose, although our method does regard one block as one file.

[11] and [12] focus on file delivery from single server and dynamic file delivery scheduling in a P2P file sharing system and in CDN with cache. In [11], there are nodes that have video files (servers), and servers provide those video files for nodes that request the video files. Requests arrive at the servers dynamically. Service latency is reduced by changing the server that provides the video file and by avoiding congestion at the server during downloading. In [12], the authors focus on the tree-based file delivery in a P2P file sharing system in which the links have different capacities, and they propose efficient algorithms to dynamically reorganize the structure of network so as to enhance distribution efficiency. These papers focus only on the server and network structure, and do not consider the order in which the files are transmitted.

[15]–[19] investigate file delivery scheduling with the shortest remaining processing time (SRPT) algorithm at the Web server or satellite link to determine the serving order, where SRPT is primarily the optimal scheduling algorithm to minimize the mean response time of jobs at a single processor by changing the processing order of jobs. SRPT determines the order according to the remaining processing time of the jobs and gives higher priority to the job that has shorter remaining processing time. SRPT can be extended

to scheduling on a Web server by regarding the remaining file transmission time as remaining processing time. However, it is not effective to adapt SRPT to scheduling on a hop-by-hop file delivery system because the value of serving a file is determined by the popularity of the file and we should consider the benefits of caches of nodes that have finished downloading files. Moreover, because SRPT gives the optimal mean service latency on Web server, the authors of [15]–[19] mainly focus on the trade-off between the fairness of each serving and mean service latency. However, in this paper, we only focus on the scheduling minimizing the total service latency to clarify the characteristic of hop-by-hop file delivery system.

[13] and [14] investigate scheduling on the link between the satellite and satellite dish. [13] focuses on utilization of satellite broadcast links and modem unicast links by considering the benefits of caches and the popularity of each file. In online scheduling to utilize broadcast links and to improve service latency, they determine the order of transmitted files. However, they do not consider interruption of file transmission with online scheduling. Therefore, their method is not suitable for a hop-by-hop file delivery system.

3. Problem Definition

In a CDN or transparent proxy system, there are intermediate nodes that have a cache respectively and have roles of both server and client. Each original server, which has original files, has the role of server, and each node on which users are placed has the role of client. However, in major P2P file-sharing systems, each node has a cache and has roles of both server and client. Therefore, we assume that each intermediate node, which is located on delivery path in hop-by-hop file delivery system, has a cache function and file-delivery capability from its local cache to another node. In actual delivery systems, cache-storage capacity is limited, so some suitable algorithms are used for removing files from a cache. Here we assume unlimited cache capacity (i.e., stored files are never removed) because our focus is the scheduling algorithm for file delivery.

Moreover, to avoid causing ambiguity, we define a "requesting node" as one that has users who request files, and a "providing node" as one that provides each file corresponding to each request. In this network, there are initially no duplicate files on any node and only the original providing nodes have the original files. A requested file is sent from the original providing node to the requesting node. A copy of the file is stored using the cache function at each node on the delivery path. Subsequently, if a node has a neighbor node that has a requested file in its cache, the node can get the file from the neighbor node rather than the original providing node.

3.1 Discussion about Performance Indicator

To satisfy each user in the system that is opened to the general public, reducing mean service latency per request is important because file downloading requires that the content is completely downloaded before it can be played. Partially downloaded data has no value, as mentioned above in Sect. 1. Shortening the mean service latency improves usability for each user of the system in the long term.

We treat the total service latency of all requests instead of mean service latency per request as a performance indicator of file-delivery scheduling because total service latency is a performance indicator coequal with mean service latency per request, i.e., there is a relationship in which, "mean service latency per request $\times$ the number of requests = total service latency of all requests." When mean service latency increases and the number of requests is fixed, the total service latency also increases coequally. Therefore, they are coequal performance indicators, which denote coessential performance of the system.

Another well-known performance indicator is "fairness," which is treated in [15]–[19]. The authors of [15], [16] propose SRPT scheduling in consideration of the trade-off between "service latency" and "fairness." They claim that although SRPT, which gives priority to servings of small files, can minimize service latency, it may cause an unfair situation, "starvation," where servings of large files are harmed due to giving priority to servings of small files. However, the influence of the trade-off is not always large when we consider an actual network environment. That is, there are some cases where a scheduling can produce both small service latency and high fairness although the scheduling is only intended to minimize service latency. For example, there was analytical verification that servings of large files are seldom harmed in terms of fairness when SRPT is applied under lower-load situations [18], [19]. Moreover, under a transient overload, there was verification by an implementation study that servings of large files by SRPT experience only negligibly higher service latency than those by FAIR scheduling, which is a traditional scheduling on a web server and which allocates resources fairly among servings [17]. In particular, the authors claim that the effects are accentuated under heavy-tailed file-size distribution (approximated file-size distribution in an actual network) [17].

These examples demonstrate that the relationship between service latency and fairness is not generically strong, and latency and fairness can be investigated independently. Therefore, our proposed scheduling, which focuses on only service latency could be enhanced to produce both good service latency and good fairness when our proposed scheduling produces good results and they are utilized. In this paper, we focus on minimizing the total service latency and leave investigating fairness to future work.

3.2 Transmission Method

There is a sequential transmission method, as mentioned above, when a node is handling multiple requests. Sequential transmission is suitable for file delivery. Sequential transmission methods are categorized as interruption disabling or interruption enabling. Interruption enabling meth-

ods allow the interruption of file transmission even if the transmission has not finished. That is, transmission can be interrupted and another file can then be transmitted before the current file transmission is completed. The interruption-disabling methods do not allow file transmission to be interrupted before the current file transmission has been completed. That is, another file cannot be transmitted before the current file transmission has finished.

The theoretical optimal value of the total service latency (the minimum value of the sum of the service latencies) can be calculated if all the request arrival times are known in advance. This optimal value is given when we use the interruption disabling method. When there are more interruptions during a file transmission, the total service latency for the whole delivery system is longer. The proof is shown as Appendix A. However, in an actual network environment, the request arrival times are not known in advance. Furthermore, the number of outstanding requests changes moment to moment. This causes the "appropriate order of file transmission update problem" described in the next section, and there are some cases in which the total service latency is shorter with interruptions than without them. Therefore, we discuss the interruption enabling method and describe a file-delivery scheduling algorithm that interrupts a current file delivery in such a way that the total service latency for the whole delivery system is reduced.

4. File-Delivery Scheduling

The requirements for file-delivery scheduling are best understood through examples. First, we consider the case in which a providing node has multiple requests for certain files, as illustrated in Fig. 3. The providing node, node 1 in this case, has to determine the order for delivering each file so as to minimize the total service latency. For example, suppose it receives two requests for file A, one from user 2 and one from user 3, and one request for file B from user 1. If the files sizes are the same, it should deliver file A first and then file B to minimize the total service latency.

Next, let us consider the case in which the providing node receives a request for a file while it is transmitting another file. For example, suppose that users 2 and 3 each request file B while the node is transmitting file A to node2 for user 1. If a remaining part of file A which is not yet transmitted is large enough, it should suspend transmission of file A and transmit file B to node2 for users 2 and 3 to minimize the total service latency.

Finally, we consider the case in which there are intermediate nodes on the delivery path, and a copy of each file arriving at all of these nodes is stored there because of a behavior of hop-by-hop file-delivery systems. This means that it is possible to suspend delivery of a file stored on an intermediate node to expedite delivery of another file and thereby minimize the total service latency. For example, still looking a Fig. 3, let us consider the case in which users 2 and 3 request file A and users 1 and 4 request file B at the same time. First, the providing node (again node 1) sends file A

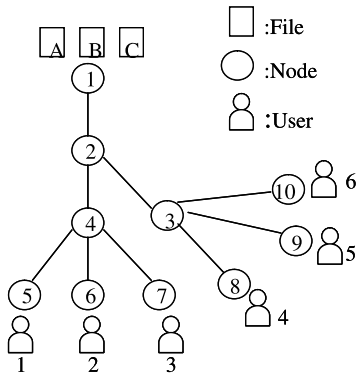


Fig. 3 Multi-link network.

to node 2. Node 2 then sends the file to node 4, and, at the same time, the providing node sends file B to node 2. If a new request does not arrive, node 2 next sends the file to node 3 for user 4. If a new request does arrive, the node 2 should determine whether delivery of file B be suspended or not. For example suppose a request for file A from user 5 arrives. Node 2 should deliver file A to node 3 prior to sending file B to node 3 to minimize the total service latency.

These examples show that we need to consider the order in which the files for each outstanding request are delivered, whether to suspend delivery of the current file in order to deliver a newly requested file, and how the file passing at nodes on the delivery path should be scheduled. From these considerations, we identified three requirements.

1. Determine the order of file delivery so that the total service latency is minimized.
2. Suspend current file delivery and deliver a newly requested file if doing so will reduce the total service latency.
3. Schedule file passing at nodes on the delivery path so that the total service latency is minimized.

Moreover, there are two methods for resuming transmission of suspended files: resume at the point where transmission was suspended or resend the entire file. The resumption model has better transmission efficiency because the node does not have to send data that has already been sent. However, the resumption model requires the system to keep track of the states for each uncompleted file transmission. This means the system bears a heavy load when many files are suspended simultaneously. Since the reset model does not impose this requirement, the system does not bear a heavy load even if many files are suspended simultaneously. However, the node needs to resend data it had already sent, so the transmission efficiency is worse. We therefore focused on the resumption model in order to reduce the total service latency for the whole delivery system.

If the number of suspensions is large, the behavior of the system is similar to that of one using parallel transmission. That is, the transmission completion time is later. This adversely affects the system from the viewpoint of the total service latency. Therefore, we should appropriately deter-

mine when to suspend current file delivery in order to expedite delivery of another file in response to a new request.

5. File-Delivery Scheduling Algorithm for One-Link Network

For a basic study, we analyzed the performance characteristics of our file-delivery scheduling method for a one-link network (Fig. 4) without an intermediate node on the path between the providing and requesting nodes, i.e., we focused on only one link between neighboring nodes on the delivery path. Thus, we did not consider how the files are passed at the nodes on the delivery path (requirement (3) above). Therefore, our analysis only focused on requirements (1) and (2). When scheduling is performed effectively on a one-link network, it can be extended to a version adapted for a multi-link network and performs effectively by considering requirement (3). We discuss the feasibility of our proposed one-link scheduling algorithm in Sect. 5.2 below.

We made three assumptions in our basic analysis. First, the load on the transmission link is constant over time, regardless of cross traffic, et al. Second, transmission of suspended files is resumed at the point where transmission was suspended, which results in higher transmission efficiency and a shorter total service latency for the whole delivery system. Finally, the request frequency distribution for each file is unknown.

5.1 Details of Algorithm for One-Link Network

To describe the file-delivery scheduling algorithm clearly, we define parameters below. Let $U = \{file\,1, file\,2, \cdots, file\,m\}$ be a set of files that the providing node has, and each $file\,i$ has two parameters: the remaining file size f_i, which has not been transmitted, and the number of requests r_i. Let F_o be a set of files, which has been requested before the algorithm starts, i.e., $F_o \subseteq U$ and $r_i \neq 0$, for $\forall file\,i \in F_o$.

First, to satisfy requirement (1), the providing node calculates total service latency g for F_o and determines the file delivery order so that g (Seq. (1)) takes a minimum value g_{min}:

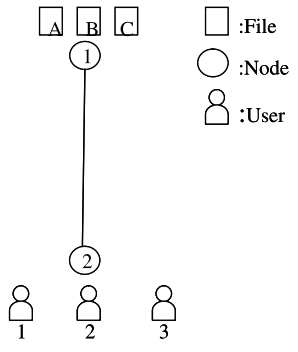


Fig. 4 One-link network.

$$g = \sum_{i=1}^{n} r_{\Phi_T(i,F_o)} \sum_{j=1}^{i} \frac{f_{\Phi_T(j,F_o)}}{b}, \tag{1}$$

where $\Phi_T(I, S)$ is a file number χ of $file\,\chi$, which is in the I-th place in the transmission order of all files in a subset $S \subseteq U$. Equation (1) takes a minimum value when $\Phi_T(I, F_o) = \Phi_D(I, F_o)$, for $1 \leq I \leq n$, where n is the number of elements of F_o, i.e., $|F_o|$, and $\Phi_D(I, S)$ is a file number χ of $file\,\chi$ whose place is I-th in the descending order of $\frac{r_\chi}{d_\chi}$ of all files in a subset $S \subseteq U$, where $d_i = \frac{f_i}{b}$ and b is the bandwidth of the transmission link. (The proof is given in Appendix B.) Relationships among these parameters are shown in Fig. 5.

Now, requests will arrive at a providing node after transmission of a file of F_o has started. Let $file\,T \in U$ be a file that is being transmitted, and let $F \subseteq U$ be a set of files in which $r_i \neq 0$, $f_i \neq 0$ and $i \neq T$, for $\forall file\,i \in F$, during the transmission of $file\,T$.

Next, to satisfy requirement (2), when a new request arrives, the providing node decides whether to suspend the current file delivery, i.e., transmission of $file\,T$, and transmit the requested $file\,A$ ($file\,A \in F$ or $F \leftarrow F + \{file\,A\}$, i.e., F is replaced by $F + \{file\,A\}$, if $file\,A$ is the first requested file, i.e., $file\,A$ had no request before the new request came) instead of $file\,T$. Since we assume that the request frequency distribution of each file is unknown, the providing node calculates the local optimal solution for the period between the moment when a request arrives and the moment when the next request arrives. The result is used to determine whether to suspend the current file delivery, i.e., transmission of $file\,T$, and start delivery of the newly requested file, i.e., $file\,A$. The determination is made using Eqs. (2) and (3) and Relationship (4).

$$\begin{aligned} w' &= \frac{f_A + f_T}{b} r_T + \left(t_s + \frac{f_A}{b}\right) r_A - t_s \\ &\quad + \sum_{i=1}^{l-1} r_{\Phi_T(i,F')} \left(\sum_{j=1}^{i} \frac{f_{\Phi_T(j,F')}}{b} + \frac{f_T}{b} + \frac{f_A}{b} \right) \\ w &= \frac{f_T}{b} r_T - t_s \end{aligned} \tag{2}$$

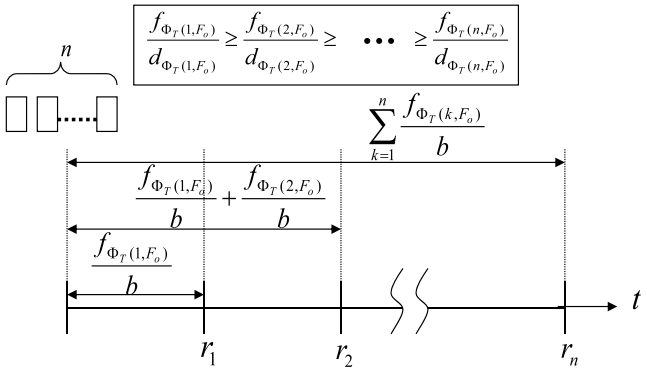


Fig. 5 Relationships among parameters for minimizing total service latency.

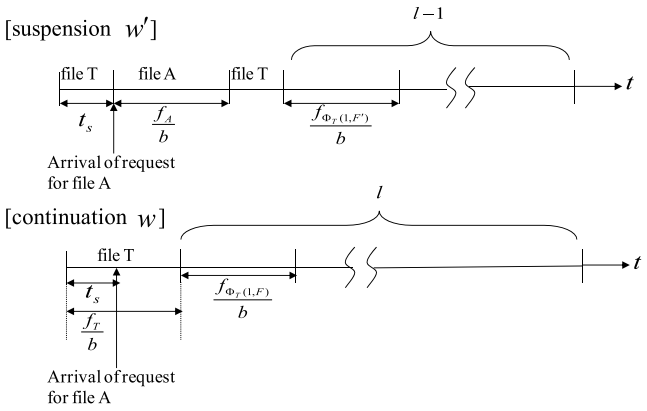


Fig. 6 Relationships among parameters for deciding whether to suspend current file delivery when new request arrives.

$$+\sum_{i=1}^{l} r_{\Phi_T(i,F)}\left(\sum_{j=1}^{i}\frac{f_{\Phi_T(j,F)}}{b}+\frac{f_T}{b}\right) \tag{3}$$

$$w_{min} > w'_{min} \tag{4}$$

The w' and w respectively denote the total service latency for the whole delivery system, when the current file delivery is suspended and when it is not, from the time when transmission of $file\,T$ starts until the time when transmissions of all files of $F+\{file\,T\}$ end under the situation where each r_i of $file\,i$ in $F+\{fileT\}$ will not change afterward. We determine the delivery order so that each of them takes a minimum value: w'_{min} and w_{min}. Equation (2) takes a minimum value when $\Phi_T(I,F') = \Phi_D(I,F')$, for $1 \le I \le l-1$, where F' is a set of files, which is $F - \{file\,A\}$. Equation (3) takes a minimum value when $\Phi_T(I,F) = \Phi_D(I,F)$, for $1 \le I \le l$. The t_s is the time from starting transmission until transmission is suspended because a new file request arrives and l is the number of elements of F, i.e., $|F|$. Relationships among these parameters are shown in Fig. 6.

If Relationship (4) holds when the current file delivery is suspended and a newly requested file is delivered, the total service latency would be reduced by suspending the delivery. In this case, $F \leftarrow F - \{file\,A\} + \{file\,T\}$ if $f_T \neq 0$, or $F \leftarrow F - \{file\,A\}$ if $f_T = 0$, and $T \leftarrow A$. In addition, this new delivery of $file\,T$ could also be suspended by the above determination using Equations (2) and (3) and Relationship (4) due to another newly requested file.

5.2 Feasibility of Algorithm for One-Link Network

Our proposed algorithm for the one-link network, which is described above, can be effectively applied to a bottlenecked link between a neighboring node and a super node, which is one-link. The super node is like a hub at which a large number of requests arrive and must serve a large number of files. Moreover, the algorithm also can be applied to a delivery path which is configured using multi-link.

The algorithm requires only two parameters: the number of requests and remaining file size which is not yet transmitted. Remaining transmission time is predicted by the latter parameter when cross traffic has little influence. Therefore, by using values of those parameters that a super node has, the algorithm can be implemented in file delivery on a link between the super node and neighboring node. On the link, the super node is regarded as the providing node, and the neighboring node is regarded as the requesting node.

Moreover, each node on a delivery path can get values of the parameters because the delivery path in a hop-by-hop file-delivery system is a path through which requests have passed once, as described in Sect. 3. That is, each node can get the number of requests for each requested file by counting when each request arrives and each node knows each remaining file size because nodes transmit the files. Therefore, by using the values, the algorithm can be adapted for and be implemented in file delivery on multi-link of a delivery path where each link between neighboring nodes on the delivery path is regarded as one-link, and each node is regarded as performing both roles: providing node and requesting node. This implementation reduces total service latency so that it can satisfy a part of requirement (3). However, the implementation is just one example of implementation for a multi-link network. The implementation can be enhanced by considering peculiar elements of multi-link networks, e.g., time when each intermediate node on the delivery path starts file transmission to the next node (starts when whole file is received completely or not. We leave problems of the multi-link algorithm to future work.

6. Simulation and Performance Analysis

We evaluated the performance of the algorithm described in Sect. 5 by computer simulation.

6.1 Simulation Conditions

We used the one-link network shown in Fig. 4. The providing node had files, and the requesting nodes had clients. The requesting nodes sent requests for each file independently so that the requests arriving at the providing node followed a Poisson distribution. To take into account the effect of the deviations in popularity among files, the average arrival rate of the requests for each file was given by one of three distributions: equivalent, uniform, or Pareto. To investigate scheduling effectiveness when the load on the providing node is high, we chose 1.80 [requests/sec] as average requests rate [20]. Note that a Pareto distribution approximates the distribution of the number of requests for each content on the Internet [20]. The cumulative distribution function of the Pareto distribution is expressed by

$$F(x) = 1 - \left(\frac{k}{x}\right)^{\alpha}, x \ge k. \tag{5}$$

To take into account different file types, e.g, audio, video, and document, we selected a lognormal distribution as the file size distribution. The cumulative distribution function of the Pareto distribution is expressed by

$$F(x) = \int_0^x \frac{1}{\sqrt{2\pi\sigma y}} exp\left[\frac{-(log\, y - \zeta)^2}{2\sigma^2}\right] dy \qquad (6)$$

and we set ζ to 1.2 and σ^2 to 0.2 [21].

We compared three file delivery scheduling algorithms with our algorithm:

1. file delivery scheduling algorithm with FIFO.
2. file delivery scheduling algorithm with SRPT.
3. file delivery scheduling algorithm which calculates the optimal solution when all arrival times of requests are known in advance.

Here, 1) is a scheduling method whereby files are delivered in the order in which requests arrive at the node. 2) is an SRPT scheduling applied to file delivery. Recall that SRPT is the optimal scheduling policy to minimize the mean response time of jobs at single processor by changing the processing order of jobs. 3) is a scheduling which calculate the minimum sum of service latencies in the whole hop-by-hop file delivery system when the arrival times of all requests are known before the algorithm starts. Obviously, this would not be possible in an actual network, but it does show the theoretical efficiency limit of a hop-by-hop file-delivery system.

6.2 Simulation Results

Figures 7, 8 and 9 plot the average service latency per request against the number of files to be sent when the average arrival rates were equivalent, followed a uniform distribution, or followed a Pareto distribution for the proposed, FIFO, SRPT, and optimal solutions. The relationship between the average service latency per request and number of files was almost linear for all three distributions. The results of the proposed algorithm were better than FIFO.

The results of the proposed algorithm, SRPT and optimal scheduling in Fig. 7 are almost same when the average arrival rates were equivalent, and the proposed algorithm's solutions include worse part than those of SRPT. This is because the requests have the little influence of rescheduling requests on the results when the average arrival rates for individual file requests are equivalent and the number of the requests has no deviation over time; Hence, our algorithm's scheduling behavior resembles that of SRPT by mainly being affected by file size. Although our scheduling focuses on both file size and the number of requests while SRPT scheduling focuses on only file size, the small influence of requests causes this similarity. Moreover, when the average request arrival rates are equivalent, the number of suspensions is larger than for the other distributions (Fig. 10), and the suspensions created by our algorithm negatively affect the results (As mentioned in Sect. 1, when the transmitting completion orders of each file are the same, the total service latency when there are suspensions is longer than when there are no suspensions.).

As shown in Figs. 8 and 9, when the average arrival rate follows a Pareto or uniform distribution, our scheduling

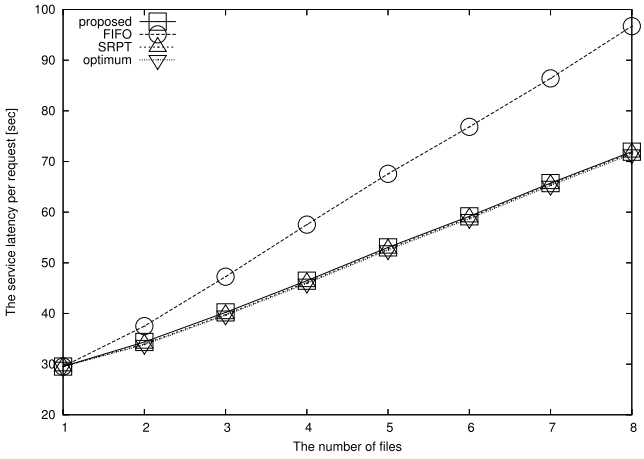


Fig. 7 Equivalent: arrival rate = 1.80 [request/sec].

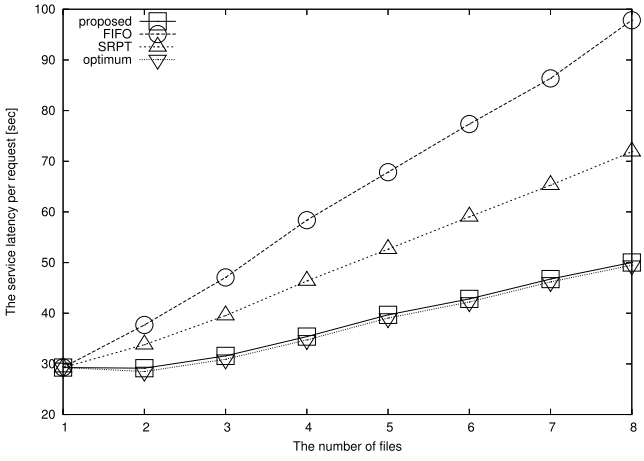


Fig. 8 Uniform distribution: arrival rate = 1.80 [request/sec].

algorithm's solution is close to the optimal one and further from those of FIFO and SRPT. This means our algorithm can appropriately change the file transmission order to minimize the total service latency by following both the popularity and file size of each content. If the average arrival rate follows a Pareto distribution, the fewer the files, the farther the solution is from the optimum one. This is because all average request arrival rates per file are easy to apply to the long tail of the Pareto distribution (the portion of the tail that corresponds to few requests and does not make much difference in the values) when there are few files. Therefore, the difference in the values decreases and the results approach those when the average arrival rates are equivalent. However, when the number of files to be sent is large, our algorithm is effective even for the Pareto distribution, because the difference in the values increases with the number of requests for each file. Since a Pareto distribution approximates the distribution of the number of requests for content on the Internet, our algorithm should be effective in an actual network.

Figures 11, 12 and 13 plot the average service latency per request of each scheduling against the load when the av-

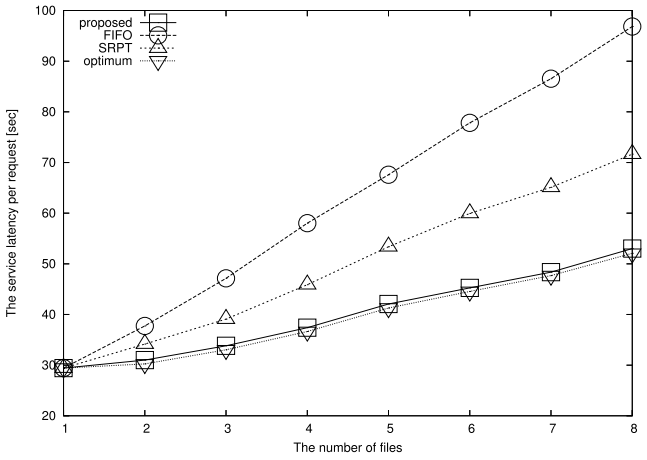


Fig. 9 Pareto distribution: arrival rate = 1.80 [request/sec].

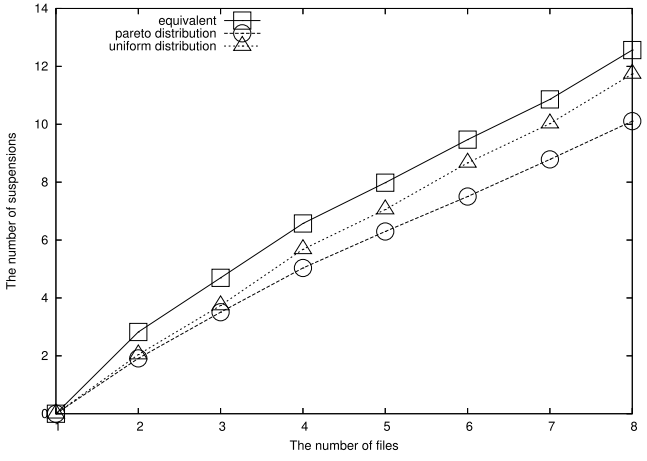


Fig. 10 Distribution of requests vs. number of suspensions.

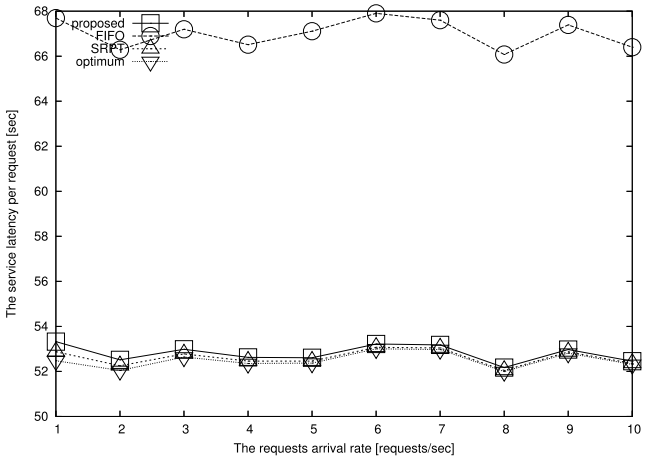


Fig. 11 Equivalent: the number of files = 5.

erage arrival rates were equivalent, followed a uniform distribution, or followed a Pareto distribution, and the number of file was 5. The results of the proposed algorithm, SRPT and optimal scheduling in Fig. 11 are almost same when the

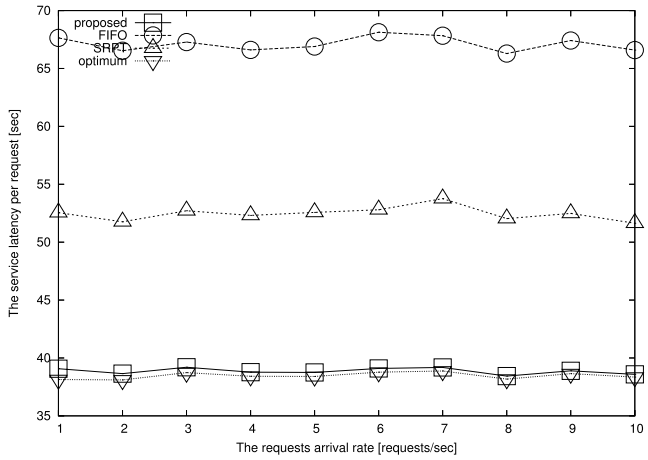


Fig. 12 Uniform distribution: the number of files = 5.

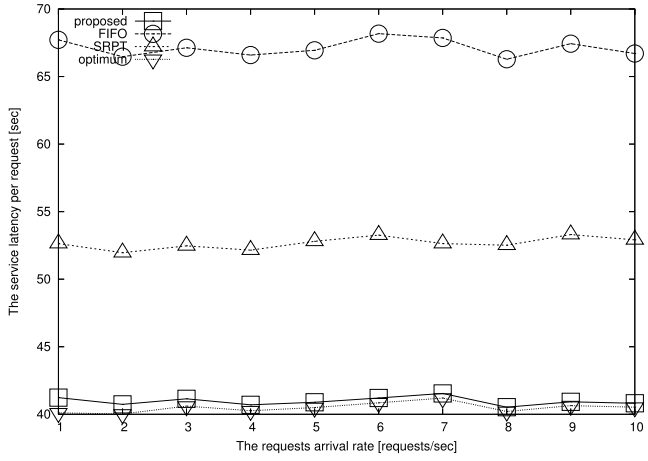


Fig. 13 Pareto distribution: the number of files = 5.

average arrival rates were equivalent, and the proposed algorithm's solutions include worse part than those of SRPT. This is because of the same reason mentioned above, the little influence of rescheduling requests on the results when the average arrival rates for individual file requests are equivalent and the number of the requests has no deviation over time. However, in Figs. 12 and 13, our scheduling algorithm's solution is close to the optimal one for the whole load range, whereas FIFO and SRPT give solutions far from the optimum one. We also evaluated the performance of them with the various number of files by computer simulation and the trend of the results are same as the figures.

These results show that our algorithm performed at nearly the theoretical efficiency limit of a hop-by-hop system by using a local optimal strategy when the request frequency distribution for each file had a deviation, even though we did not consider the request frequency distribution.

7. Conclusion

We discussed delivery of file-type content that cannot be

played before it has been completed downloaded in a hop-by-hop file delivery system. We proposed a new file delivery scheduling method for the system. As a basic study, we developed a new file delivery scheduling algorithm that works when the frequency of requests for each file is unknown in the one-link model. We evaluated its performance characteristics in a simulation. The results indicate that the algorithm, which uses the local optimal solution until the next request arrives, performed nearly at the theoretical efficiency limit of the hop-by-hop system when the request frequency distribution for each content had a deviation.

In the future, we will investigate a dynamic algorithm for varying conditions such as the request frequency distribution. We will also investigate fairness. We will also validate an extended version of the algorithm adapted for the caching algorithm and multi-link network. We mentioned in Sect. 5 that our proposed algorithm for the one-link network can be adapted for and implemented in an actual network without any change. Moreover, the algorithm can be enhanced by considering peculiar elements of a multi-link network. We will enhance it by considering the following perspectives.

1. *Time of starting file transmission*: Consider which starting time is the best choice when each intermediate node on the delivery path starts file transmission to the next node. That is, the node starts file transmission to the next node when the node gets the whole file from the previous node on the delivery path or the node gets a part of the file from the previous node on the delivery path.
2. *Delivery path routing*: Consider which next node is the best choice to receive a file when some nodes on the delivery path are very crowded or the number of hops of a delivery path is very large. Moreover, we should investigate how nodes not on the delivery path can know the information to decide that.
3. *Pair of providing node and requesting node*: Consider which providing node is the best choice when many nodes have copies of original content. Moreover, consider which requesting node has the highest priority to receive content when a node has caching and forwarding capability and another node does not have that.

References

[1] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "A distributed anonymous information storage and retrieval system," Designing Privacy Enhancing Technologies, LNCS 2009, pp.46–66, Springer-Verlag, 2001.

[2] J.S.K. Chan, V.O.K. Li, and K.-S. Lui, "Performance comparison of scheduling algorithms for peer-to-peer collaborative file distribution," IEEE J. Sel. Areas Commun., vol.25, no.1, pp.146–154, Jan. 2007.

[3] S. Ganguly, A. Saxena, and S. Banerjee, "Fast replication in content distribution overlays," IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.25, no.4, pp.2246–2256, March 2005.

[4] X. Jia, D. Li, H. Du, and J. Cao, "On optimal replication of data object at hierarchical and transparent Web proxies," IEEE Trans. Parallel Distrib. Syst., vol.16, no.8, pp.673–685, Aug. 2005.

[5] P. Rodriguez, C. Spanner, and E.W. Bjersack, "Analysis of Web caching architectures: Hierarchical and distributed caching," IEEE/ACM Trans. Netw., vol.9, no.4, pp.404–418, Aug. 2001.

[6] E. Cronin, S. Jamin, C. Jin, A.R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the Internet," IEEE J. Sel. Areas Commun., vol.20, no.7, pp.1369–1382, Sept. 2002.

[7] J.E. Smith and J.R. Goodman, "Instruction cache replacement policies and organizations," IEEE Trans. Comput., vol.C-34, no.3, pp.234–241, March 1985.

[8] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: Design implementation, and performance," IEEE Trans. Knowl. Data Eng., vol.11, no.4, pp.549–562, July/Aug. 1999.

[9] J.S.K. Chan, V.O.K. Li, and K.-S. Lui, "Performance comparison of scheduling algorithms for peer-to-peer collaborative file distribution," IEEE J. Sel. Areas Commun., vol.25, no.1, pp.146–154, Jan. 2007.

[10] P. Felber and E.W. Biersack, "Cooperative content distribution: Scalability through self-organization," in Self-star Properties in Complex Information Systems, ed. O. Babaoglu, et al., pp.343–357, Springer-Verlag, Berlin, Heidelberg, 2005.

[11] Y. Cai, A. Natarajan, and J. Wong, "On scheduling of peer-to-peer video services," IEEE J. Sel. Areas Commun., vol.25, no.1, pp.140–145, Jan. 2007.

[12] M. Schiely, L. Renfer, and P. Felber: "Self-organization in cooperative content distribution networks," IEEE International Symposium on Network Computing and Applications, pp.109–118, July 2005.

[13] M. Agrawal, A. Manjhi, N. Bansal, and S. Seshan, "Improving Web performance in broadcast-unicast networks," IEEE Infocom, pp.229–239, March 2003.

[14] E. Modiano, "Scheduling algorithms for message transmission over a satellite broadcast system," IEEE MILCOM 97, vol.2, pp.628–634, Nov. 1997.

[15] L. Cherkasova, "Scheduling strategy to improve response time for Web applications," High-Performance Computiong and Networking International Conference and Exhibition, LNCS, vol.1401, pp.305–314, 1998.

[16] M.A. Bender, S. Chakrabarti, and S. Muthukrishnan, "Flow and stretch metrics for scheduling continuous job streams," Proc. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.270–279, 1998.

[17] B. Schroeder and M. Harchol-Blter, "Web servers under overload: How scheduling can help," ACM Trans. Internet Technology, vol.6, no.1, pp.20–52, Feb. 2006.

[18] N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling: Investigating unfairness," ACM Sigmetrics/Performance, pp.279–290, 2001.

[19] A. Wireman and M. Harchol-Balter, "Classifying scheduling policies with respect to unfairness in an M/GI/1," ACM Sigmetrics International Conference on Measurment and Modeling of Computer Systems, pp.238–249, June 2003.

[20] M. Nabe, K. Baba, M. Murata, and H. Miyahara, "Analysis and modeling of WWW traffic for designing Internet access networks," IEICE Trans. Commun. (Japansese Edition), vol.J80-B-I, no.6, pp.428–437, June 1997.

[21] M. Andreolini and R. Lancellotti, "Analysis of peer-to-peer systems: Workload characterization and effects on traffic cacheability," IEEE MASCOTS'04, pp.94–105, Oct. 2004.

[22] H. Tsurumi, T. Miyata, K. Yamaoka, and Y. Sakai, "A basic study of a file delivery scheduling in a hop by hop file delivery system on one link model," IEEE PACRIM'07, pp. 446–449, Aug. 2007.

Appendix A: Proof of Disadvantage of Interruption

The theoretical optimal value of the total service latency (the minimum value of the sum of the service latencies) can be calculated if all the request arrival times are known in advance. This optimal value is given when we use the interruption disabling method. The total service latency for the whole delivery system is longer when there are more interruptions of file transmission.

[Proof]

Label files according to the order of transmission finish time, i.e., $file\ 1$, $file\ 2$, …, $file\ m$. Let the sizes of the files be f_1, f_2, …, f_m, and the whole download periods of files with no interruption be h_1, h_2, …, h_m, respectively. When bandwidth b is constant, $h_i = \frac{f_i}{b}$. The transmission finish times of each file when there is no interruption are thus denoted as t_1, t_2, …, t_m, respectively, and the transmission finish times of each file when there are interruptions are $\acute{t}_1$, $\acute{t}_2$, …, $\acute{t}_m$, respectively.

Let α_i be the transmitting period during $t'_{i-1} < t < t'_i$ except the transmitting period of *file i* and $\alpha_{i,j}$ be the transmitting period of $file\ j$ during α_i.

The transmission finish time of the first file when an interruption is allowed is the sum of the transmitting periods of $file\ 1$ and the other interrupted files. Therefore, the transmission finish time of the first file when an interruption is allowed ($\acute{t}_1$) is expressed as

$$\acute{t}_1 = h_1 + \alpha_1 \tag{A·1}$$

The transmission finish time of the second file when an interruption is allowed is the sum of the transmitting periods of $file\ 1$, $file\ 2$, and the other interrupted files. Therefore, the transmission finish time of the second file when an interruption is allowed ($\acute{t}_2$) is expressed as

$$\acute{t}_2 = h_1 + h_2 + (\alpha_1 - \alpha_{1,2}) + \alpha_2 \tag{A·2}$$

Similarly, the transmission finish time of the n-th file when an interruption is allowed ($\acute{t}_n$) is expressed as

$$\begin{aligned}\acute{t}_n &= \sum_{i=1}^{n} h_i + (\alpha_1 - \alpha_{1,2} - \alpha_{1,3}, \ldots, -\alpha_{1,n}) \\ &\quad +(\alpha_2 - \alpha_{2,3} - \alpha_{2,4}, \ldots, -\alpha_{2,n}) \\ &\quad +, \ldots, +(\alpha_{n-1} - \alpha_{n-1,n}) + \alpha_n \\ &= \sum_{i=1}^{n} h_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_{i,j}\end{aligned} \tag{A·3}$$

The interruption finish time of the n-th file when an interruption is not allowed (t_n) is expressed as

$$t_n = \sum_{i=1}^{n} h_i \tag{A·4}$$

Here, because $0 \le \sum_{j=i+1}^{n} \alpha_{i,j} \le \alpha_i$, one gets $\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_{i,j} \ge 0$. Therefore, we get Eq. (A·5) as follows:

$$\acute{t}_n = \sum_{i=1}^{n} h_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_{i,j} \ge \sum_{i=1}^{n} h_i = t_n \tag{A·5}$$

Therefore, the relation between t_i and $\acute{t}_i$ is always expressed by Eq. (A·6).

$$\acute{t}_n \ge t_n \tag{A·6}$$

Here, $t_a(j,i)$ is the arrival time of the i-th request for $file\ j$, $\acute{e}_j$ is the number of requests which arrived before $\acute{t}_j$ of the number of the requests for $file\ j$, and l is the total number of the files for which requests arrived.

The sum of service times $\acute{s}$ is

$$\acute{s} = \sum_{j=1}^{l} \sum_{i=1}^{\acute{e}_j} (\acute{t}_j - t_a(j,i)) \tag{A·7}$$

On the other hand, the sum of service times s is Eq. (A·8) when an interruption is not allowed and the order of file transmission is the same as above.

$$s = \sum_{j=1}^{l} \sum_{i=1}^{e_j} (t_j - t_a(j,i)) \tag{A·8}$$

Here, according to Eq. (A·6), the transmission finish time of each file when an interruption is allowed is later than or the same as when an interruption is not allowed. Therefore, the relation between $\acute{e}_j$ and e_j is expressed by Eq. (A·9).

$$\acute{e}_j \ge e_j \tag{A·9}$$

According to Eq. (A·6), Eq. (A·7), Eq. (A·8), and Eq. (A·9), the relation between $\acute{s}$ and s can be expressed as

$$\acute{s} \ge s \tag{A·10}$$

Therefore, the theoretical optimal value of the sum of service latencies is only attainable when there is no interruption. If the transmitting periods of files except the transmitting periods of files which finished transmission before t_n increase before t_n, then $\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_{i,j}$ in Eq. (A·3) increases. According to Eq. (A·7), this causes the sum of service latencies in the whole system to grow.

The above indicates that we cannot expect the interruption enabling method to reduce the total transmission time if file transmissions can be interrupted.

[End of Proof]

Appendix B: Proof of Local Optimal Solution

The algorithm described in Sect. 5 should determine the transmission order of files so that g, w, and w' (Eqs. (1), (2), and (3), respectively) take minimum values. That is, the algorithm should determine the file transmission order so that the total service latency for the whole delivery system takes a minimum value when the number of requests for each file remains unchanged. We found the determination method;

otherwise we must do a full-search [22] to determine that. The method is as follows.

The g takes a minimum value when $\Phi_T(I, F_o) = \Phi_D(I, F_o)$, for $1 \leq I \leq n$. The w takes a minimum value when $\Phi_T(I, F) = \Phi_D(I, F)$, for $1 \leq I \leq l$, and the w' takes a minimum value when $\Phi_T(I, F') = \Phi_D(I, F')$, for $1 \leq I \leq l-1$. The proof is as follows.

[Proof]

Label files in accordance with the descending order of $\frac{r_i}{d_i}$, i.e., $file\,1, file\,2, \cdots, file\,m$. Let the number of requests for the files be $r_1, r_2, \cdots, r_m$, and the total download times for the files without interruption be $d_1, d_2, \cdots, d_m$, respectively. If bandwidth b is constant and the remaining size of $file\,i$ is f_i, $d_i = \frac{f_i}{b}$.

[Step 1. When the number of files is 2]

Let the total service latency be s' when the files are transmitted in order $file\,1$, $file\,2$, and let it be s'' when the files are transmitted in order $file\,2$, $file\,1$.

$$s' = d_1 r_1 + (d_1 + d_2) r_2 \tag{A·11}$$

$$s'' = d_2 r_2 + (d_1 + d_2) r_1 \tag{A·12}$$

$$s'' - s' = d_2 r_1 - d_1 r_2 \geq 0 \left(\because \quad \frac{r_1}{d_1} \geq \frac{r_2}{d_2} \right) \tag{A·13}$$

Therefore, if the number of files is two, the total service latency takes a minimum when the files are transmitted in descending order in terms of $\frac{r_i}{d_i}$.

[Step 2. When the number of files is $k+1$]

We assume that the total service latency takes a minimum when the number of files is k and the files are transmitted in descending order in terms of $\frac{r_i}{d_i}$ (Assumption 1). Under this assumption, label files in accordance with the descending order of $\frac{r_i}{d_i}$, i.e., $file\,1, file\,2, \cdots, file\,k$.

Let the $(k+1)-th\ file$ be $file\,x$, the number of requests for the file be r_x, and the total download period for all files without interruption be d_x.

B.1 When $\frac{r_x}{d_x} \geq \frac{r_1}{d_1}$ ($\frac{r_x}{d_x}$ is the largest value for all files)

Under this condition, the minimum total service latency when $file\,x$ is first transmitted, s', is calculated by minimizing the sum of the service latency of the k files transmitted after the first file is transmitted. Given Assumption 1 above, the transmission order of files that results in the minimum total service latency is that corresponding to the descending order of $\frac{r_i}{d_i}$, i.e., $file\,x, file\,1, file\,2, \cdots, file\,k$.

$$s' = d_x r_x + \sum_{i=1}^{k} r_i \left(d_x + \sum_{j=1}^{i} dj \right) \tag{A·14}$$

Next, we calculate the minimum total service latency, s'', when $file\,x$ is not transmitted first. Let the file that is transmitted first be $file\,a$ $(1 \leq a \leq k)$. Under this condition, according to Assumption 1 and $\frac{r_x}{d_x} \geq \frac{r_1}{d_1}$, $file\,x$ is selected to be transmitted next. The remaining files are transmitted in the order $file\,1, file\,2, \cdots, file\,a-1, file\,a+1, file\,a+2, \cdots, file\,k$.

$$s'' = d_a r_a + (d_x + d_a) r_x + \sum_{i=1}^{a-1} r_i \left(d_x + d_a + \sum_{j=1}^{i} d_j \right) + \sum_{i=a+1}^{k} r_i \left(d_x + \sum_{j=1}^{i} d_j \right) \tag{A·15}$$

$$s'' - s' = (d_a r_x - d_x r_a) + \left(\sum_{i=1}^{a} d_a r_i - \sum_{i=1}^{a} d_i r_a \right) \geq 0 \left(\because \quad \frac{r_x}{d_x} \geq \frac{r_a}{d_a},\ \frac{r_i}{d_i} \geq \frac{r_a}{d_a}\ (1 \leq i \leq a) \right) \tag{A·16}$$

Therefore, if Assumption 1 and $\frac{r_x}{d_x} \geq \frac{r_1}{d_1}$ hold, the total service latency takes a minimum when $file\,x$ is transmitted first and the order of the k remaining files transmitted after the first file is $file\,1, file\,2, \cdots, file\,k$, i.e., in descending order corresponding to $\frac{r_i}{d_i}$.

Therefore, if Assumption 1 and $\frac{r_x}{d_x} \geq \frac{r_1}{d_1}$ hold, and the number of files is $k+1$, the total service latency takes a minimum when the $k+1$ files are transmitted in descending order corresponding to $\frac{r_i}{d_i}$.

B.2 When $\frac{r_x}{d_x} < \frac{r_1}{d_1}$ ($\frac{r_1}{d_1}$ is the largest value for all files)

Under this condition, the minimum total service latency when $file\,1$ is transmitted first, s', is calculated by minimizing the sum of the service latency of the k files transmitted after the first file is transmitted. Given Assumption 1 above, the transmission order of $k+1$ files that results in the minimum total service latency is that corresponding to the descending order of $\frac{r_i}{d_i}$, i.e., $file\,1, file\,2, \cdots, file\,b-1, file\,x, file\,b, file\,b+1, \cdots, file\,k$.

$$s' = \sum_{i=1}^{b-1} r_i \sum_{j=1}^{i} d_j + \left(d_x + \sum_{i=1}^{b-1} d_i \right) r_x + \sum_{i=b}^{k} r_i \left(d_x + \sum_{j=1}^{i} d_j \right) \tag{A·17}$$

Next, we calculate the total service latency when $file\,1$ is not transmitted first: s_1'' when $\frac{r_1}{d_1} \geq \frac{r_a}{d_a} \geq \frac{r_x}{d_x}$ and s_2'' when $\frac{r_1}{d_1} \geq \frac{r_x}{d_x} \geq \frac{r_a}{d_a}$. Let the first transmitted file be $file\,a$ $(1 < a \leq k$ or $file\,a$ is $file\,x)$.

$$s_1'' = d_a r_a + \sum_{i=1}^{a-1} r_i \left(d_a + \sum_{j=1}^{i} d_j \right) + \sum_{i=a+1}^{b-1} r_i \sum_{j=1}^{i} d_j + \left(d_x + \sum_{i=1}^{b-1} d_i \right) r_x + \sum_{i=b}^{k} r_i \left(d_x + \sum_{j=1}^{i} d_j \right) \tag{A·18}$$

$$s_2'' = d_a r_a + \sum_{i=1}^{b-1} r_i \left(d_a + \sum_{j=1}^{i} d_j \right) + \left(d_a + d_x + \sum_{i=1}^{b-1} d_i \right) r_x$$

$$+\sum_{i=b}^{a-1} r_i\left(d_a + d_x + \sum_{j=1}^{i} d_j\right) + \sum_{i=a+1}^{k} r_i\left(d_x + \sum_{j=1}^{i} d_j\right) \quad \text{(A·19)}$$

$$s_1'' - s' = s_2'' - s' = \sum_{i=1}^{a} d_a r_i - \sum_{i=1}^{a} d_i r_a \geq 0$$

$$\left(\because \quad \frac{r_i}{d_i} \geq \frac{r_a}{d_a}\ (1 \leq i \leq a)\right) \quad \text{(A·20)}$$

Therefore, if Assumption 1 and $\frac{r_x}{d_x} < \frac{r_1}{d_1}$ hold, the total service latency takes a minimum when $file\,1$ is transmitted first and the k remaining files are transmitted corresponding to the descending order of $\frac{r_i}{d_i}$.

Therefore, if Assumption 1 and $\frac{r_x}{d_x} < \frac{r_1}{d_1}$ hold, and the number of files is $k+1$, the total service latency takes a minimum when the $k+1$ files are transmitted in descending order corresponding to $\frac{r_i}{d_i}$.

B.3 Conclusion

As shown in Steps 2-1 and 2-2, if the total service latency for transmitting k files takes a minimum when the files are transmitted in descending order corresponding to $\frac{r}{d}$, the total service latency for transmitting $k+1$ files also takes a minimum when the files are transmitted in descending order corresponding to $\frac{r_i}{d_i}$.

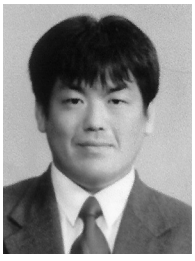
[Step 3. When there is any number of files]

As proved by mathematical induction in Steps 1 and 2, if there is any number of files, the total service latency takes a minimum when the files are transmitted in descending order corresponding to $\frac{r_i}{d_i}$.

[End of Proof]

Hiromi Tsurumi received a B.E. degree from the Tokyo Denki University in 2006. She joined the Department of Communications and Integrated Systems at the Tokyo Institute of Technology in 2006 and is currently undertaking a master course of Tokyo Institute of Technology. Her research interests include scheduling, optimization, techniques for file delivery, and communications.

Takamichi Miyata received the B.Eng. and M.Eng. degrees from the University of Toyama, Toyama, Japan in 2001 and 2003, respectively, and Dr.Eng. degree from Tokyo Institute of Technology, Tokyo, Japan in 2006. In 2006, he joined Dept. of Communications and Integrated Systems at Tokyo Institute of Technology as an assistant professor. His current research interests include image processing, and image coding.



Katsunori Yamaoka received the B.E., M.E., and Ph.D. degrees from the Tokyo Institute of Technology in 1991, 1993 and 2000, respectively. He left Ph.D. program in 1994 and joined the Tokyo Institute of Technology as an assistant professor at that time. In 2000, he joined the National Institute of Multimedia Education (NIME) in Japan as an associate professor. Since 2001, he has been an associate professor at the Tokyo Institute of Technology. He has also been a visiting associate professor of the National Institute of Informatics (NII) in Japan since 2004. His research interests are network QoS control for multimedia.

Yoshinori Sakai received the B. Eng., and the Dr.Eng. In electrical engineering from the University of Tokyo, Tokyo, Japan in 1969 and 1974 respectively. From 1974 to 1987, he was employed by Nippon Telegraph and Telephone Public Corporation (NTT). From 1987 to 1990, he was an Associate Professor at Tokyo Institute of Technology, where he is a Professor. He designed many communication systems, for example digital transmission system over existing FDM network, facsimile communication network and teleconference system at NTT lab. His current research interests include image information retrieval, network streaming and content distribution over the Internet. He received the best author award from the ITV Japan, the best paper award from IIEEJ Japan and the achievement award from IEICE. He is now Director, Publications of IEICE.