Dynamic parallel downloading with network coding in λ-grid networks

Kouji Hirata[1], Khamisi Kalegele[2], Yoshinobu Higami[1], Shin-ya Kobayashi[1]
[1,2]Graduate School of Science and Engineering, Ehime University, Ehime, Japan
Email: [1]{hirata, higami, kob}@cs.ehime-u.ac.jp, [2]kalegs@koblab.cs.ehime-u.ac.jp

***Abstract*— In λ-grid networks, data files for job execution are stored on file servers as replicas, and computing servers, which execute jobs, download these replicas in parallel to reduce downloading time. However, parallel downloading raises blocking probability of lightpath establishments because it uses many links and thus wavelength resources are wasted. To resolve this problem, we propose a dynamic parallel downloading scheme with network coding which encodes data at intermediate nodes. The proposed scheme performs network coding by regarding file servers as intermediate nodes. In this scheme, a file is divided into multiple blocks. A file server creates an encoded block from those blocks and stores it as a replica. Computing servers download encoded blocks from multiple file servers in parallel. Through simulation experiments, we show that the proposed scheme can improve the blocking probability and the downloading time efficiently.**

***Index Terms*— λ-grid, file replication, parallel downloading, network coding, WDM**

I. Introduction

Grid computing integrates geographically distributed computing resources, such as CPU and storage, through communication networks. Data grids which are one of the grid computing techniques manage distributed data and provide high performance computing [7], [13]. Fig. 1 shows a model of a data grid network. In data grid networks, there are many sites each of which consists of a file server (storage) and a computing server. File servers store data files needed for job execution. Note that, to distribute loads, data files are replicated on multiple file servers. When a job is generated according to a user request at a site, the computing server at the site downloads these replicas from file servers in parallel to reduce the downloading time [4], [9]. Then it executes the job. However, even if parallel downloading is used, the transmission rate is insufficient because the size of files transmitted in data grid networks is very huge.

To enhance the transmission rate of data grids, λ-grids which employ wavelength division multiplexing (WDM) and lightpaths have been proposed [8], [16]. The WDM increases the capacity of a fiber optic link by simultaneously transmitting multiple signals with different

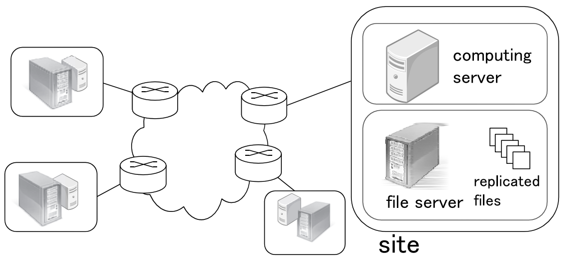


Figure 1. A data grid network.

wavelengths over a single fiber. The lightpaths guarantee bandwidth for data transmission, and thus reliable data transmission is realized. In λ-grid networks, before transmitting data, a lightpath is established between sites with a wavelength. Note that lightpaths cannot be established with the same wavelength in the same link at the same time. If the lightpath establishment is blocked, the file transmission is also blocked. As a result, the computing server cannot execute the job. Thus blocking of lightpath establishments is one of the significant issues in λ-grid networks.

In λ-grid networks as well as conventional data grid networks, parallel downloading can be used to reduce downloading time. However, parallel downloading increases the blocking probability of lightpath establishments because it uses more links than single downloading and thus more wavelength resources are wasted [9], [24]. Furthermore, the blocking probability increases with the number of file servers used for each parallel downloading session because the number of lightpaths to be established increases. Generally, downloading a file from too many file servers is not effective in terms of the blocking probability. Thus, to use parallel downloading efficiently, we have to select file servers by means of appropriate server selection schemes [5], [12], [24], [26] when there is a file download attempt. For example, in [24], the authors proposed a novel server selection scheme for parallel downloading in λ-grid networks. In this scheme, file servers are selected in consideration of network resource availability. This scheme distributes loads for links and suppresses the generation of bottleneck links. As a result, this scheme reduces the blocking probabilities of lightpath establishments and job execution requests.

Note that the performance of file server selection

Manuscript received February 1, 2010; revised February 23, 2010; accepted March 8, 2010.
This research was partially supported by Grant-in Aid for Young Scientists (Start-up) of the Japan Society for the Promotion of Science under Grant No. 20800039.

schemes depends on how many replicas of each file are stored. The blocking probability decreases with the increase of the number of replicas of each file for the following reason. If there exist a few replicas of a target file in a network, download requests for the file concentrate in specific file servers and lightpaths with long hops tend to be established, so that wavelength resources are wasted. On the other hand, if many file servers have replicas of the target file, computing servers can select file servers from many candidates and thus download requests are distributed. Furthermore, lightpaths with short hops tend to be established. As a result, wavelength resources are used effectively and the blocking probability of lightpath establishments decreases. However, the size of files is generally very huge and the storage size is limited, so that file servers hardly store many replicas. Therefore we may not be able to efficiently use parallel downloading in λ-grid networks.

In this paper, we propose a parallel downloading scheme with network coding to enhance the performance of server selection schemes in λ-grid networks. The main idea of network coding [1] is that an intermediate node encodes several incoming data to one or multiple outgoing data, instead of simply forwarding data. In the proposed scheme, network coding is done on an overlay network by regarding file servers as intermediate nodes. A file is divided into multiple blocks and a file server creates an encoded block from these blocks and stores it. Computing servers download encoded blocks from multiple file servers in parallel. Because file servers store only an encoded block as a replica for each file, the proposed scheme enables file servers to store many replicas. Furthermore, an original file can be recovered from encoded blocks, regardless of selection of file servers from which these encoded blocks are downloaded. Therefore the replicas are downloaded with low wavelength resources, and thus the blocking probability of lightpath establishments is expected to decrease. As a result, the blocking probability of job execution requests is also expected to decrease.

Network coding was first proposed in [1]. Since then, it has been applied to multicast communications [19], mobile ad-hoc networks [17], sensor networks [2], [11], and P2P [10], [18]. However, to the best of our knowledge, application of network coding to λ-grid networks has not been considered. Therefore this combination is a creative approach. Note that, in [10], the authors proposed a scheme for content distribution of large files in P2P that is based on network coding. In this scheme, a file is divided into multiple blocks and peers store those blocks. Whenever a peer needs to forward a block to another peer, it produces an encoded block of all the blocks it currently stores. The concept of this scheme may be similar to our proposed scheme. However, our proposed scheme differs from it because our proposal considers characteristics of λ-grid networks as follows. In λ-grid networks, the data size is very huge and thus the number of files which can be stored on file servers is limited. Thus, unlike [10], in our proposed scheme file servers store at most one encoded block for each file to reduce storage consumption. Furthermore, in λ-grid networks, file servers used for downloading are selected when a user request arrives, and are not changed during downloading. According to those characteristics, our proposed scheme develops a new efficient parallel downloading structure with network coding for λ-grid networks.

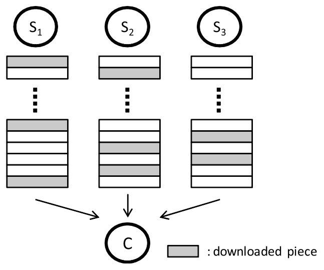


Figure 2. Dynamic parallel downloading (S_i denotes a file server and C denotes a computing server).

The rest of this paper is organized as follows. In Section II, we describe file replication and parallel downloading. Section III discusses our parallel downloading scheme. In Section IV, the performance of the proposed scheme is discussed with the results of simulation experiments. Finally, we conclude the paper in Section V.

II. File replication and parallel downloading

The size of data transmitted in data grid networks is generally huge. Thus when a computing server downloads a file, large amounts of bandwidth could be consumed. To resolve this problem, files are replicated on multiple file servers. File replication reduces the bandwidth consumption and helps in load balancing. In data grid networks, replicas are dynamically created and deleted on each site according to replication schemes [3], [6], [21]–[23] or caching schemes [14], [20]. In replication schemes, a file server decides when and where to create a replica of one of its files as necessary. On the other hand, in caching schemes, when a computing server downloads a file, a file server in the same site tries to store it. If there is no storage space, the file server replaces stored files with the new file, based on information such as usage of files.

Replicas are downloaded in parallel to reduce the downloading time. Parallel downloading scheme is divided into two categories: static and dynamic schemes [9]. In the static parallel downloading, before the download of a target file starts, a computing server estimates the throughput of file servers. The computing server then requests a portion of the file proportional to the estimated throughput, from each file server. However, the static parallel downloading is not flexible because the portion of the file has to be assigned in advance. As shown in Fig. 2, in the dynamic parallel downloading, a file is divided into many small pieces of equal size on file servers. A computing server downloads one of the pieces from each file server. When the download of a piece from a file server finishes, then the computing server downloads another piece from the file server, and this

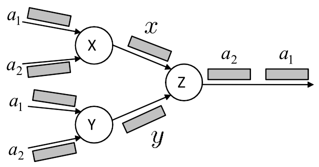


Figure 3. Network coding.

process continues until the whole file is downloaded. The dynamic parallel downloading is more flexible than the static parallel downloading because pieces are selected dynamically according to network conditions. For example, in λ-grid networks, computing servers can download a target file from remaining file servers even if lightpath establishments for some file servers are blocked. Thus in this paper we focus on the dynamic parallel downloading.

III. Parallel downloading with network coding

A. Network coding

Network coding is a novel mechanism to improve link utilization by allowing intermediate nodes to encode data. This paper uses linear network coding [15] on finite field GF(q), where q denotes the size of finite field. In linear network coding, each data is regarded as an element in GF(q). We show an example of network coding in Fig. 3. In this figure, we assume that data a_1 and a_2 arrive at nodes X and Y. We also assume that nodes X and Y have coding vectors $\boldsymbol{c_X} = (c_1, c_2)$ and $\boldsymbol{c_Y} = (c_3, c_4)$, respectively, where $c_j \in$ GF(q) ($j = 1, 2, 3, 4$). In this case, data a_1 and a_2 are linearly encoded to data $x = c_1a_1 + c_2a_2$ and $y = c_3a_1 + c_4a_2$ at nodes X and Y, respectively. Node Z receives x and y. By solving a system of linear equations, node Z can retrieve the original data a_1 and a_2. Note that x and y have to be linearly independent to each other to recover the original data.

In general, a set of N data a_n ($n = 1, 2, \dots, N$) arriving at an intermediate node is linearly encoded to single output data $a_{\text{out}} = \sum_{n=1}^{N} c_n a_n$ ($c_n \in$ GF(q)), where $\boldsymbol{c} = (c_1, c_2, \dots, c_N)$ denotes a coding vector constructed at the node. The receiver node can recover the original data by receiving N encoded data which are linearly independent to each other.

B. Proposed scheme

1) Overview: The proposed scheme applies network coding to a λ-grid network by regarding file servers as intermediate nodes. Specifically, network coding is done on an overlay network as shown in Fig. 4. In the proposed scheme, a file is divided into multiple blocks. As necessary, file servers create encoded blocks from either these blocks or encoded blocks of other file servers and store them. Note that file servers store only an encoded block as a replica for each file. When a job request arrives, a computing server downloads encoded blocks needed for the request in parallel, and recovers the original file. Then the computing server uses the recovered file to execute the job.

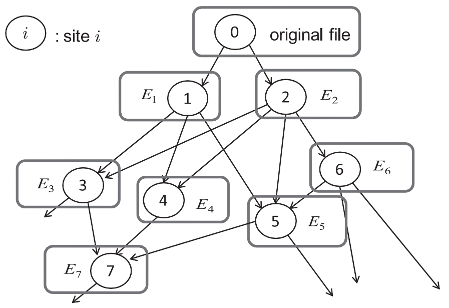
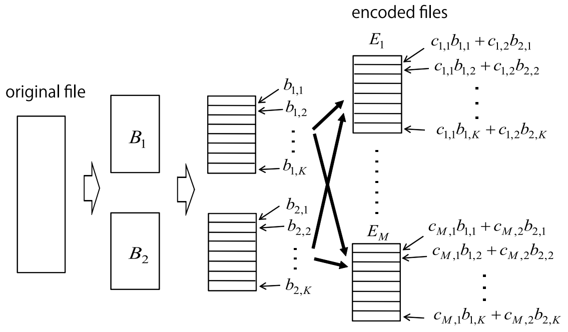


Figure 4. Overlay network (E_i denotes an encoded block stored on file server i).



encoded files
original file
B_1
B_2
$b_{1,1}$
$b_{1,2}$
$b_{1,K}$
$b_{2,1}$
$b_{2,2}$
$b_{2,K}$
E_1
$c_{1,1}b_{1,1}+c_{1,2}b_{2,1}$
$c_{1,1}b_{1,2}+c_{1,2}b_{2,2}$
$c_{1,1}b_{1,K}+c_{1,2}b_{2,K}$
E_M
$c_{M,1}b_{1,1}+c_{M,2}b_{2,1}$
$c_{M,1}b_{1,2}+c_{M,2}b_{2,2}$
$c_{M,1}b_{1,K}+c_{M,2}b_{2,K}$

Figure 5. Making procedure of encoded blocks ($N = 2$).

2) Procedure of making encoded blocks: In the proposed scheme, encoded blocks are made at file server m ($m = 1, 2, \dots, M$) as follows, where M denotes the number of file servers in the network. We first consider the case whereby encoded blocks are created from an original file. In this case, the original file is divided into N ($\leq M$) blocks B_n ($n = 1, 2, \dots, N$) of the same size, and then each block is divided into many small pieces $b_{n,k}$ ($k = 1, 2, \dots, K$) of equal size, where K denotes the number of pieces. At file server m, these pieces are encoded into $e_{m,k} = \sum_{n=1}^{N} c_{m,n} b_{n,k}$, where $c_{m,n} \in$ GF(q) denotes the nth element of coding vector $\boldsymbol{c_m} = (c_{m,1}, c_{m,2}, \dots, c_{m,N})$. File server m stores encoded block $E_m = \{e_{m,1}, e_{m,2}, \dots, e_{m,K}\}$, which is N times smaller than the original file, instead of the original file. We illustrate an example of the above procedure in Fig. 5, where $N = 2$.

We can also create an encoded block from other encoded blocks which is made from the same original file. Specifically, a set of N' encoded pieces $e_{n,k}$ ($n = 1, 2, \dots, N'$) can be encoded to $e_{m,k} = \sum_{n=1}^{N'} c_{m,n} e_{n,k}$ at file server m, where $N' \leq N$. For example, when an encoded block $E_m = \{e_{m,1}, e_{i,2}, \dots, e_{m,K}\}$ at file server m is created from an encoded block $E_i = \{e_{i,1}, e_{i,2}, \dots, e_{i,K}\}$ at file server i and an encoded block $E_j = \{e_{j,1}, e_{j,2}, \dots, e_{j,K}\}$ at file server j, $e_{m,k} = c_{m,1}e_{i,k} + c_{m,2}e_{j,k}$.

3) Dynamic parallel downloading with network coding: For any k, N pieces $b_{n,k}$ of the original file can be recovered from *any* N linearly independent pieces $e_{m,k}$. When a user request arrives, a computing server downloads these encoded pieces with dynamic parallel

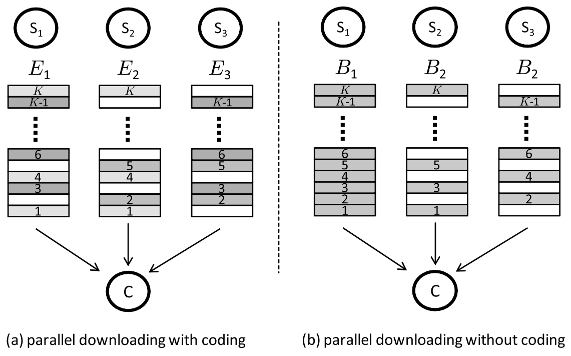

S1
S2
S3
E1
E2
E3
B1
B2
B2
K
K-1
6
5
4
3
2
1
C
(a) parallel downloading with coding
(b) parallel downloading without coding

Figure 6. Parallel downloading with/without coding ($N = 2$).

downloading as shown in Fig. 6 (a). Note that in this figure, C denotes the computing server, and S_1, S_2, and S_3 denote file servers and they have linearly independent encoded pieces. As we can see from this figure, for each k, we select N file servers to download N encoded pieces, if encoded pieces of a file can be downloaded from more than N file servers. By doing so, the proposed scheme can efficiently download the file from more than N file servers.

As mentioned in Section II, replicas are dynamically created and deleted according to replication or caching schemes. In this paper, we assume that the proposed scheme uses a caching scheme in which a file server creates and stores an encoded block whenever a computing server at the same site downloads a file. Specifically, when a user request arrives at a site, a computing server downloads a file and a file server replicates and stores an encoded block according to the following three scenarios. Note that, in this paper, we assume that each computing server knows which file servers have replicas of a target file by using a replica catalog which has location information of replicas [7].

1) If the file server at the same site has the original of the target file, the computing server downloads it.
2) If the file server at the same site does not have the original and the number of file servers which have the desired encoded blocks is less than N, the computing server downloads the original. Subsequently, the downloaded file is encoded and stored on the file server at the same site.
3) If the file server at the same site does not have the original and the number of file servers which have the desired encoded blocks is equal to or more than N, the computing server downloads those encoded blocks with dynamic parallel downloading as shown in Fig. 6 (a). Subsequently, these encoded blocks are re-encoded again and stored on the file server at the same site.

Note that in scenarios 2) and 3), if storage of the file server is full when the file server tries to store the new encoded block, the file server replaces one or more stored blocks with the new encoded block based on information such as usage of files. Note also that, in scenario 3), N' described in Section III-B.2 is equal to N.

C. Advantage of the proposed scheme

The proposed scheme can perform dynamic parallel downloading efficiently because each file server stores only an encoded block which is N times smaller than the original file. In order to explain the effectiveness of the proposed scheme, we assume two conventional non-encoded schemes. In the first scheme, each file server stores the whole copy of each file. Thus each piece can be downloaded from any file servers as shown in Fig. 2. However, the number of replicas stored on file servers tends to be small because the size of files are huge and the storage size of each file server is limited. Thus lightpaths with long hops tend to be established because there are few replicas of each file in the network. Therefore wavelength resources are wasted. On the other hand, in the proposed scheme, file servers can store more replicas because it stores only encoded blocks as replicas. As the second scheme without network coding, we also consider the case whereby each file server stores a non-encoded block B_n (see Fig. 5). In this case, the downloading performance mainly depends on the location of blocks as shown in Fig. 6 (b). Specifically, while blocks stored on many file servers can be downloaded quickly, downloading of blocks stored on a few file servers takes much time. Furthermore, all kinds of blocks are needed to recover an original file, i.e., B_1 to B_N. Thus even if blocks other than B_n are collected, the original file cannot be recovered without B_n. On the other hand, the proposed scheme can recover the original file from any blocks as long as they are linearly independent to each other. As a result, the proposed scheme can use dynamic parallel downloading more efficiently than those non-encoded schemes.

Note that the proposed scheme has some disadvantages. Specifically, the proposed scheme divides each file into N blocks, and thus N has to be decided in advance. Furthermore, at least N file servers are needed for each parallel downloading because at least N linearly independent blocks are needed to recover the original file.

The proposed scheme can be applied to conventional data grid networks without optical networking. However, the proposed scheme is more suitable for λ-grid networks. In conventional data grid networks, the transmission rate varies according to network conditions. Thus the proposed scheme may not be effective in conventional grid networks, because downloading from at least N file servers is needed. Specifically, downloading time depends on the throughput of the slowest file server when the network has few replicas. On the other hand, because lightpaths guarantee the bandwidth in λ-grid networks, the downloading time of each file server is almost the same, regardless of network conditions. Thus the proposed scheme is effective in λ-grid networks and this paper focuses on λ-grid networks.

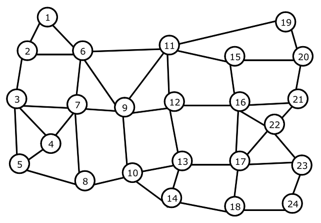


Figure 7. Network model.

IV. Performance evaluation

A. Model

To evaluate the performance of the proposed scheme, we conduct simulation experiments with a network model shown in Fig. 7. It consists of 24 nodes and 43 bi-directional links. Each node has a site, and it plays both roles as a computing server and as a file server. To establish lightpaths, we use wavelength routing with backward reservation [25]. We assume that there are no wavelength converters at any nodes. We also assume that each file server has a linearly independent coding vector. For simplicity, we assume that the propagation delay of each link is equal to 1 [msec], and the processing time of signaling at each node is equal to 0.1 [msec]. We also assume that the bandwidth D of each wavelength is equal to 10 [Gbps]. The total number W of available wavelengths is set to be 32 and the number F of different files is set to be 24, unless stated otherwise. Every original file is stored on one of file servers and the size S of storage of each file server is set to be 10 [Tbyte], unless stated otherwise. We define ρ as the offered load per wavelength:

$$\rho = \frac{8 \times \lambda \times L \times 10^3}{D \times W},$$

where λ [1/sec] denotes the average arrival rate of job execution requests at each site and L [Tbyte] denotes the average file size. For each scenario, we collect 30 independent samples from simulation experiments, and 95% confidence intervals are shown in each figure (even though most of them are invisible).

For the sake of comparison, we use two schemes without network coding as described in Section III-C. Specifically, in the first scheme, each file server stores the whole file, and in the second scheme, each file server stores a non-encoded block B_n. In this paper, to dynamically create and delete replicas, we use a simple caching scheme which replaces one or more stored replicas which have the oldest access time with a new replicas. Note that in the second scheme, a block to be stored are randomly selected from blocks of a downloaded file.

As mentioned earlier, downloading a file from too many file servers is not effective in terms of the blocking probability. Thus in this paper we apply a simple server selection scheme, which restricts the maximum number P of file servers used for each parallel downloading session as follows. In this scheme, a computing server first tries to establish a lightpath for each P file servers with the target file (or replicas) based on the shortest path routing when a

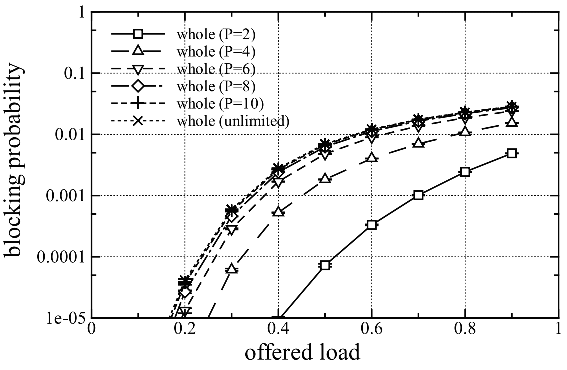


Figure 8. Blocking probability of the first scheme.

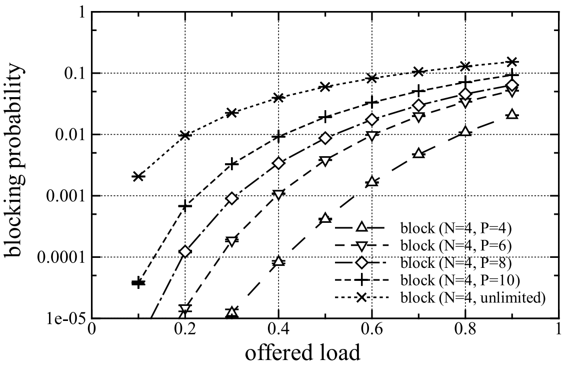


Figure 9. Blocking probability of the second scheme ($N = 4$).

job execution request arrives. If there are more than P file servers, P file servers which located near (in terms of the number of hops) from the computing server are selected. Note that in the second scheme, we select each block B_n in a round-robin fashion. If any lightpaths for file servers are blocked, the computing server downloads the file from remaining file servers in parallel. The job execution request is blocked if the number of established lightpaths is less than the minimum required number of lightpaths for the download, i.e., one in scenarios 1) or 2), and N in scenario 3). Note that in the first scheme the required number is one. Note also that, in the second scheme the required number is one or N, like the proposed scheme.

B. Performance in a homogeneous model

In this section, we evaluate the performance of the proposed scheme in a homogeneous model. In this model, the size L_f of file f ($f = 1, 2, \cdots, F$) is fixed to 2 [Tbyte]. We assume that user requests of job execution at each site is generated according to a Poisson process with rate λ, and the target file is independently chosen equally likely among all possible files except original files stored at the site.

1) Impact of the maximum number P of file servers: First, we examine the impact of the maximum number P of file servers used for each parallel downloading session. Figs. 8-10 show the blocking probability of job execution requests as a function of the offered load ρ for the first scheme which stores the whole file, the second scheme with $N = 4$ which stores a non-encoded block,

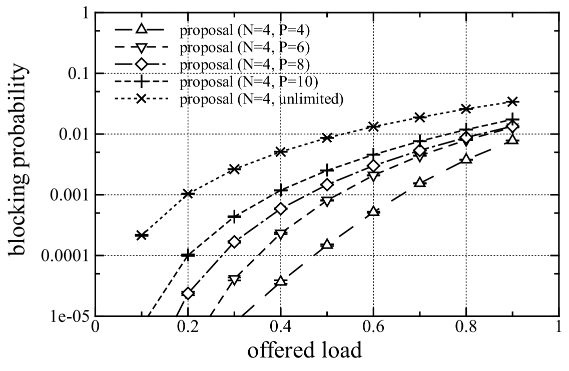


Figure 10. Blocking probability of the proposed scheme ($N = 4$).

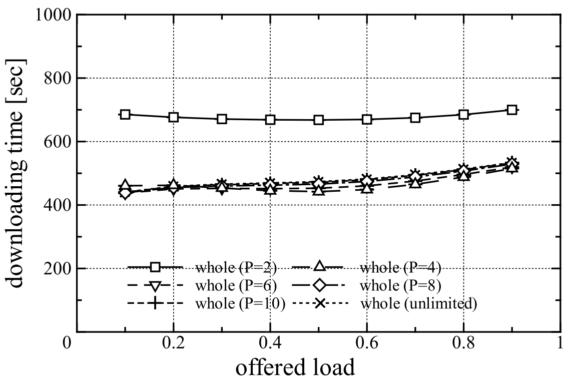


Figure 11. Average downloading time in the first scheme.

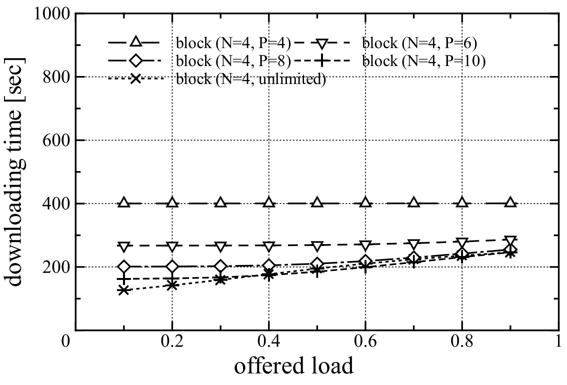


Figure 12. Average downloading time in the second scheme ($N = 4$).

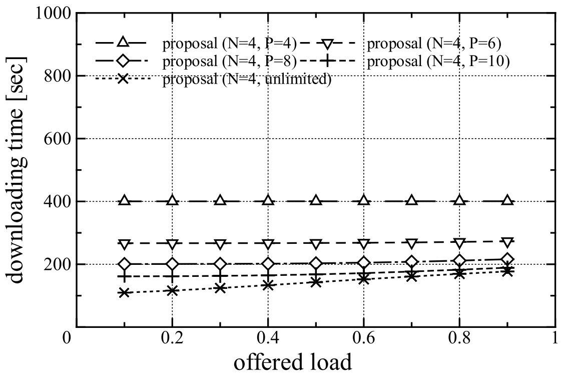


Figure 13. Average downloading time in the proposed scheme ($N = 4$).

and the proposed scheme with $N = 4$, respectively. Also, Figs. 11-13 show the average downloading time as a function of the offered load ρ for the first scheme, the second scheme with $N = 4$, and the proposed scheme with $N = 4$, respectively. Note that the blocking probability BP_r of job execution requests is defined as

$$\mathrm{BP}_r = \frac{\text{\# of blocked job execution requests}}{\text{\# of job execution requests}}.$$

Note also that the downloading time denotes the difference between the start time and finish time of the file transmission. We assume that, in the first scheme, downloading time is 0 when a replica is downloaded from the same site because it has the whole replica. In those figures, "whole ($P = p$)" represents the result of the first scheme with $P = p$ and "block ($N = n$, $P = p$)" represents the result of the second scheme with $N = n$ and $P = p$. Also, "proposal ($N = n$, $P = p$)" represents the result of the proposed scheme with $N = n$ and $P = p$. The schemes labeled with "whole (unlimited)", "block ($N = n$, unlimited)" and "proposal ($N = n$, unlimited)" mean that there are no limits on the number of file servers used for each parallel downloading session. Note that the second scheme and the proposed scheme with $N = 4$ does not have the case of $P = 2$ because at least four file servers are needed to recover an original file.

As we can see from Figs. 8-10, the blocking probability increases with P in all schemes. On the other hand, as shown in Figs. 11-13, the average downloading time basically decreases with P, because the number of file servers used for each parallel downloading session increases. We observe that, in the first scheme, the average downloading time is the largest when $P = 2$ and the average downloading time does not decrease when P is more than 4. This result implies that each file is stored on only four or five file servers on average. In the second scheme and the proposed scheme, the average downloading time decreases with the increase of P even when P is more than 4, because the size of blocks is smaller than the whole file and thus more blocks can be stored at file servers.

2) Effectiveness of the proposed scheme: Figs. 14 and 15 show the blocking probability BP_r of job execution requests and the average downloading time as a function of the maximum number P of file servers, respectively for $\rho = 0.5$. We observe that the proposed scheme can reduce the blocking probability more efficiently than the first and second schemes. We also observe that the performance of the proposed scheme with $N = 4$ is more efficient than that with $N = 2$, because the increase in N means the increase in file servers which have a target file. Specifically, the proposed scheme with large N can easily select nearby file servers, and therefore wavelength resources are not wasted. Furthermore, the average downloading time in the proposed scheme is smaller than that in the first scheme except when $P = 2$. Note that in the first scheme, downloading time is 0 when a computing server downloads a replica from a file server located in the same site. Thus when $P = 2$, the

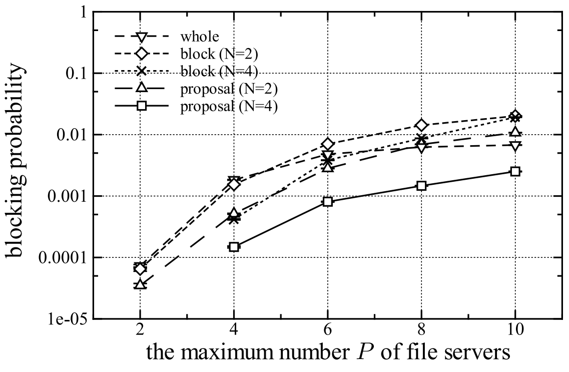


Figure 14. Blocking probability ($\rho = 0.5$).

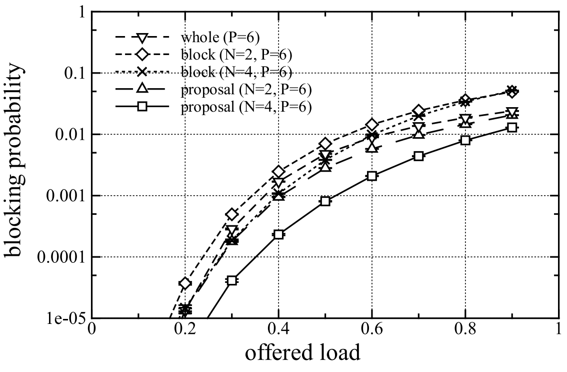


Figure 16. Blocking probability.

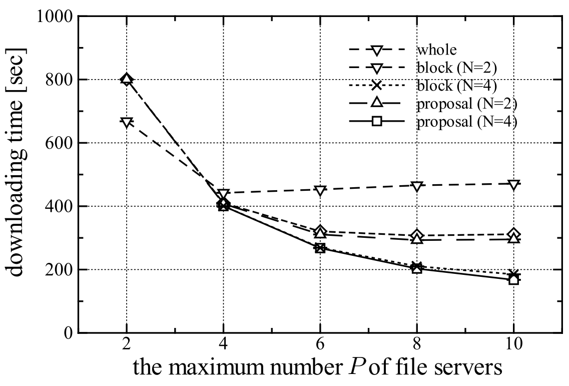


Figure 15. Average downloading time ($\rho = 0.5$).

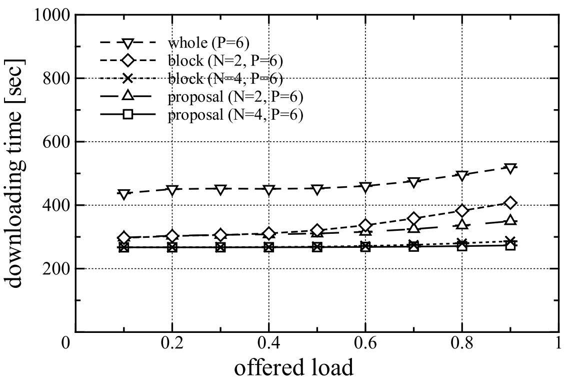


Figure 17. Average downloading time.

downloading time in the first scheme is smaller than that in the proposed scheme. Furthermore, the downloading time in the proposed scheme is slightly smaller than that the second scheme because of the advantage described in Section III-C. We also observe that the proposed scheme with $N = 4$ reduces the average downloading time more efficiently than that with $N = 2$. Specifically, the proposed scheme with $N = 4$ shows the best performance in terms of both of the blocking probability and the average downloading time.

Figs. 16 and 17 show the blocking probability BP_r of job execution requests and the average downloading time as a function of the offered load ρ, respectively for $P = 6$. We observe that the performances of the proposed schemes with network coding are better than the schemes without network coding and the performance of the proposed scheme with $N = 4$ is excellent, regardless of the offered load.

Next, we examine the blocking probability BP_l of lightpath establishments. Note that this blocking probability is different from the blocking probability BP_r of job execution requests discussed above and it is defined as

$$\mathrm{BP}_l = \frac{\text{\# of blocked lightpaths}}{\text{\# of lightpath establishment requests}}.$$

A job execution request is blocked when one or more of its lightpaths are blocked. Fig. 18 shows the blocking probability of lightpath establishments as a function of the maximum number P of file servers for $\rho = 0.5$. As we can see from this figure, the proposed scheme can reduce the blocking probability of lightpath establishments efficiently and with $N = 4$, it shows excellent performance. We also observe that the blocking probability of each scheme increases with P. The reason is that many lightpaths exist in the network and thus wavelength resources are wasted when P is large. As a result, as shown in Fig. 14, the blocking probability BP_r of job execution requests also increases with P.

Fig. 19 shows the blocking probability BP_l of lightpath establishments as a function of the offered load ρ for $P = 6$. We observe that the proposed scheme with $N = 4$ has the best performance, similar to the result in Fig. 16. This result implies that the proposed scheme uses wavelength resources efficiently.

3) Impact of the number of wavelengths: Next, we examine the performance of the proposed scheme against the number W of available wavelengths. Fig. 20 shows the blocking probability BP_r of job execution requests as a function of W for $\rho = 0.5$. When the number of available wavelengths is small, the blocking probabilities of the proposed schemes with $N = 2$ and $N = 4$ are the almost same. The reason is that the proposed scheme tends to simultaneously establish many lightpaths and thus bottleneck links are often generated. We also observe that the blocking probability of the proposed scheme improves dramatically with the increase of W because bottleneck links are rarely generated.

Fig. 21 shows the average downloading time as a function of W for $\rho = 0.5$. As we can see from this figure, the proposed scheme with $N = 4$ shows the excellent

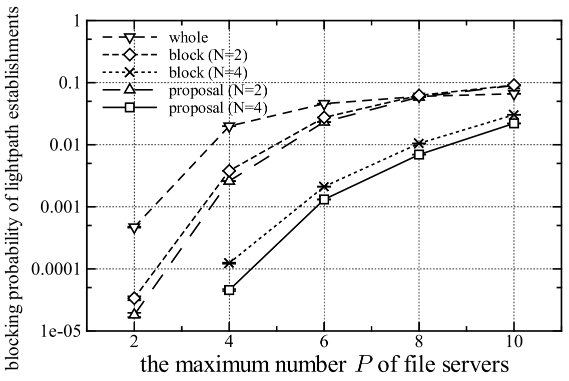


Figure 18. Blocking probability of lightpath establishments ($\rho = 0.5$).

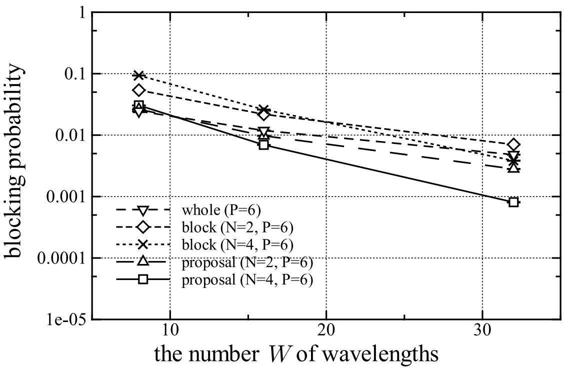


Figure 20. Blocking probability ($\rho = 0.5$).

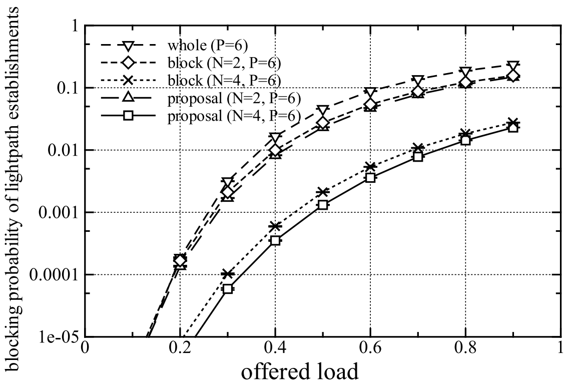


Figure 19. Blocking probability of lightpath establishments.

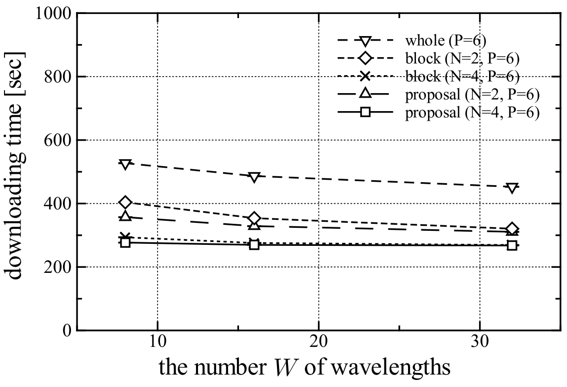


Figure 21. Average downloading time ($\rho = 0.5$).

performance. Therefore, we conclude that the proposed scheme keeps the superior performance, regardless of the number W of wavelengths.

4) Impact of the number of different files: We examine the impact of the number F of different files. Fig. 22 shows the blocking probability BP_r of job execution requests as a function of the number F of different files for $\rho = 0.5$. Also, Fig. 23 shows the average downloading time against F for $\rho = 0.5$. We observe that the blocking probability and the average downloading time of each scheme increase with F. This is because the number of replicas of each file which is stored in the network decreases. We also observe that the proposed scheme with $N = 4$ can efficiently reduce the blocking probability and the average downloading time. We conclude that the proposed scheme keeps the superior performance even if the number F of different files increases.

5) Impact of the storage size: Finally, we evaluate the performance of the proposed scheme against the storage size S of each file server. Figs. 24 and 25 show the blocking probability BP_r of job execution requests and the average downloading time as a function of S, respectively for $\rho = 0.5$. We observe that the blocking probability and the average downloading time are nonincreasing functions of the storage size. We also observe that the proposed scheme improves the blocking probability and the average downloading time efficiently. Note that the blocking probability of the proposed scheme with $N = 4$ touches bottom at $S = 14$ because each file server has replicas of all files when S is equal to or larger than 14. Thus the performance of the proposed scheme with $N = 2$ is better than that with $N = 4$ when S is large.

C. Performance in a more realistic scenario

We evaluate the performance of our scheme in a more realistic scenario. For this purpose, we assume that the mean arrival rate λ_i of job execution requests from site i is proportional to the ratios in Table I. We also assume that the target file is chosen in proportion to the ratios in Table II. Note that values in those tables are generated randomly. Furthermore, we assume that the size L_f of file f follows an exponential distribution with mean $L = 2$ [Tbyte], where L_f is normalized in such a way that the total file size is equal to $\sum_{f=1}^{F} L_f = 48$.

Figs. 26 and 27 show the blocking probability BP_r of job execution requests and the average downloading time as a function of the maximum number P of file servers, respectively for $\rho = 0.5$. We observe that the proposed scheme can reduce the blocking probability and the downloading time more efficiently than the schemes without network coding, similar to the results in the homogeneous model. We also observe that the performance of the proposed scheme with $N = 4$ is more efficient than that with $N = 2$.

Figs. 28 and 29 show the blocking probability BP_r of job execution requests and the average downloading time as a function of the offered load ρ, respectively for $P = 6$. As shown in Fig. 28, the blocking probability of

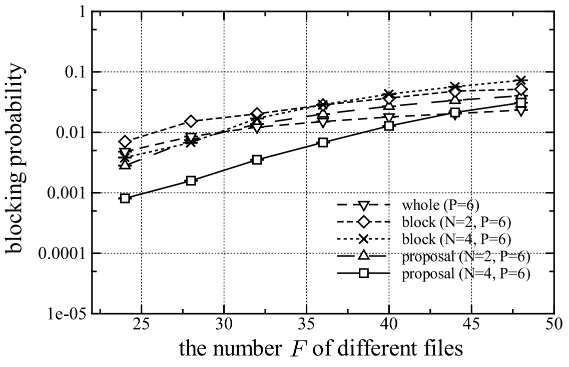


Figure 22. Blocking probability ($\rho = 0.5$).

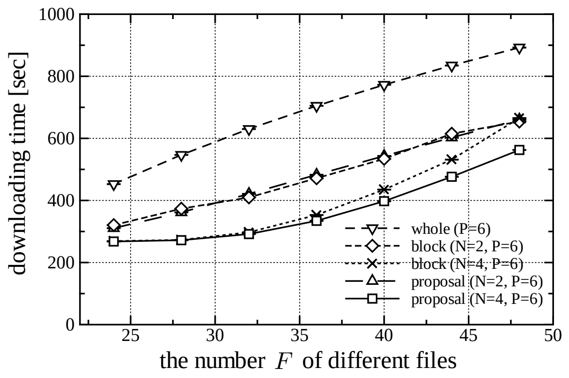


Figure 23. Average downloading time ($\rho = 0.5$).

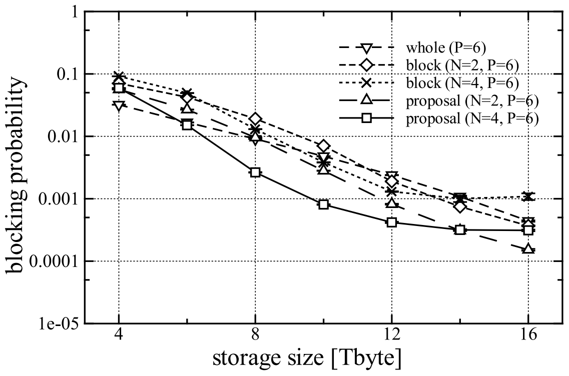


Figure 24. Blocking probability ($\rho = 0.5$).

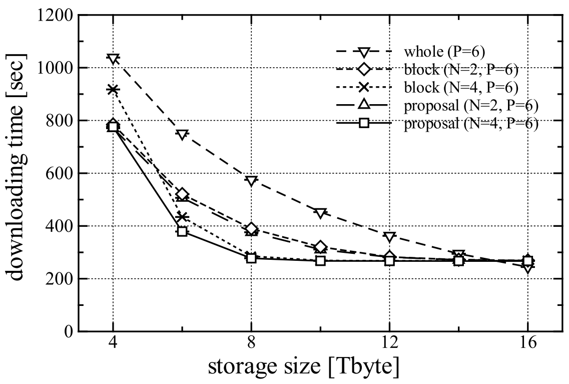


Figure 25. Average downloading time ($\rho = 0.5$).

the proposed scheme is smaller than that of the schemes without network coding. Furthermore, as we can see from Fig. 29, the average downloading time in the proposed scheme is also smaller than that in schemes without network coding. Generally, the proposed scheme shows the excellent performance in terms of both the blocking probability and the average downloading time in this scenario, similar to the results in the homogeneous model.

V. Conclusion

This paper proposed a parallel downloading scheme with network coding in λ-grid networks. The proposed scheme allocates encoded blocks to file servers with network coding, and thus reduces the storage and effectively uses dynamic parallel downloading. Through simulation experiments, we showed that the blocking probability and the downloading time were improved significantly by the proposed scheme. In this paper, we assumed that the proposed scheme worked with a simple caching scheme. However, other file replication schemes can also be combined with the proposed scheme. In the future work, we are going to evaluate the performance of the proposed scheme with those file replication schemes. The result of this ongoing research will be reported somewhere else.

Acknowledgment

We would like to thank Professor Tetsuya Takine of Graduate School of Engineering, Osaka University and Dr. Takahiro Matsuda, who is an Associate Professor of Graduate School of Engineering, Osaka University, for their helpful advices.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204-1216, 2000.

[2] E. Ayday, F. Delgosha, and F. Fekri, "Location-aware security services for wireless sensor networks using network coding," in *Proc. IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 1226–1234.

[3] W. H. Bell et al., "Evaluation of an economy-Based file replication strategy for a data grid," in *Proc. IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2003)*, Tokyo, Japan, May 2003, pp. 661-668.

[4] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads," in *Proc. IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 275–283.

[5] R.-S. Chang, M.-H. Guo, and H.-C. Lin, "A multiple parallel download scheme with server throughput and client bandwidth considerations for data grids," *Future Generation Computer Systems*, vol. 24, no. 8, pp. 798–805, 2008.

[6] R.-S. Chang, H.-P. Chang, and Y.-T. Wang, "A dynamic weighted data replication strategy in data grids," in *Proc. the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, Doha, Qatar, Mar. 2008, pp. 414–421.

TABLE I.
RATIO OF JOB EXECUTION REQUESTS FROM SITE i.

i	1	2	3	4	5	6	7	8	9	10	11	12
ratio	0.065	0.030	0.060	0.061	0.070	0.015	0.026	0.059	0.021	0.043	0.037	0.048
i	13	14	15	16	17	18	19	20	21	22	23	24
ratio	0.028	0.039	0.073	0.070	0.049	0.055	0.011	0.047	0.001	0.019	0.011	0.062

TABLE II.
RATIO OF TARGET FILES.

No.	1	2	3	4	5	6	7	8	9	10	11	12
ratio	0.042	0.037	0.018	0.037	0.018	0.032	0.025	0.074	0.064	0.039	0.066	0.022
No.	13	14	15	16	17	18	19	20	21	22	23	24
ratio	0.015	0.076	0.025	0.031	0.047	0.076	0.040	0.052	0.047	0.002	0.064	0.051

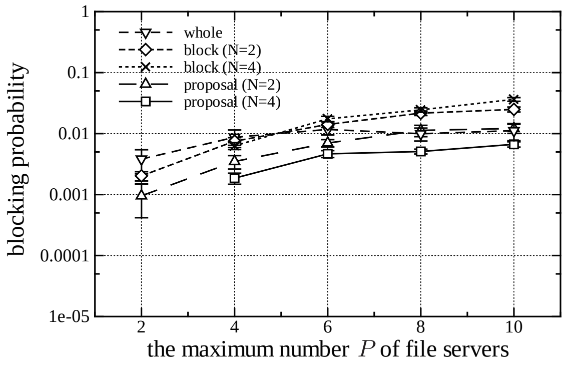


Figure 26. Blocking probability ($\rho = 0.5$).

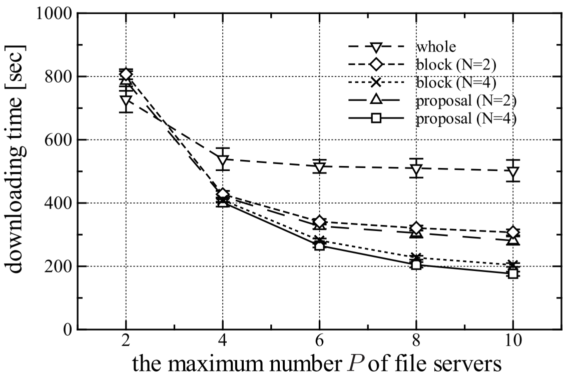


Figure 27. Average downloading time ($\rho = 0.5$).

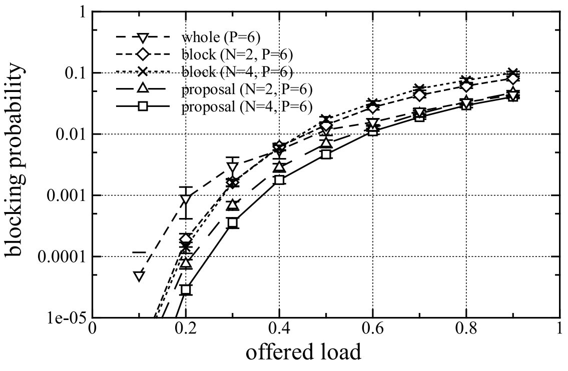


Figure 28. Blocking probability.

[7] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The data grid: towards an architecture for the distributed management and analysis of large scientific datasets," *Journal of Network and Computer Applications*, vol. 23, no. 3, pp. 187-200, 2000.

[8] S. Figueira et al., "DWDM-RAM: enabling grid services with dynamic optical networks," in *Proc. IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, Chicago, IL, Apr. 2004, pp. 707–714.

[9] C. Gkantsidis, M. Ammar, and E. Zegura, "On the effect large-scale deployment of parallel downloading", in *Proc. IEEE Workshop on Internet Applications*, San Jose, CA, Jun. 2003, pp. 79–89.

[10] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE INFOCOM*, Atlanta, GA, Mar. 2005, pp. 2235-2245.

[11] Z. Guo, P. Xie, J.-H. Cui, and B. Wang, "On applying network coding to underwater sensor networks," in *Proc. 1st ACM International Workshop on Underwater Networks*, Sep. 2006, pp. 109–112.

[12] Y. Higashi, S. Ata, I. Oka, and C. Fujiwara, "Topology-aware server selection method for dynamic parallel downloading," in *Proc. IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, Jan. 2005, pp. 325–330.

[13] W. hoschek, F. J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data management in an international data grid project," in *Proc. the First IEEE/ACM International Workshop on Grid Computing*, Bangalore, India, Dec. 2000, pp. 77–90.

[14] S. Jiang and X. Zhang, "Efficient distributed disk caching in data grid management," in *Proc. IEEE International Conference on Cluster Computing*, Williamsburg, VA, Dec. 2003, pp. 446–451.

[15] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 371–381, 2003.

[16] M. D. Leenheer, C. Develder, T. Stevens, B. Dhoedt, M. Pickavet, and P. Demeester, "Design and control of optical grid networks," in *Proc. Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS 2007)*, Raleigh, NC, Sep. 2007, pp. 107–115.

[17] T. Matsuda, T. Noguchi, and T. Takine, "Broadcasting with randomized network coding in dense wireless ad hoc networks," *IEICE Transactions on Communications*, vol. E91-B, no. 10, pp. 3216-3225, 2008.

[18] K. Nguyen, T. Nguyen, and S. Cheung, "Peer-to-peer streaming with hierarchical network coding," in *IEEE International Conference on Multimedia and Expo*, Beijing, China, Jul. 2007, pp. 396–399.

[19] T. Noguchi, T. Matsuda, and M. Yamamoto, "Performance evaluation of new multicast architecture with network coding", *IEICE Transactions on Communications, Special Issue on Content Delivery Networks*, vol. E86-B, no. 6, pp. 1788-1795, 2003.

[20] E. Otoo and A. Shoshani, "Accurate modeling of cache replacement policies in a data grid," in *Proc. the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, CA, Apr. 2003, pp. 10–19.

[21] S.-M. Park, J.-H. Kim, Y.-B. Ko, and W.-S. Yoon, "Dynamic data grid replication strategy based on Internet hierarchy," in *Proc. Second International Workshop on Grid and Cooperative Computing (GCC 2003)*, Shanghai, China, Dec. 2003, pp. 838–846.

[22] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid," in *Proc. the International Grid Computing Workshop*, Denver, CO, Nov. 2001, pp. 75–86.

[23] Q. Rasool, J. Li, G. S. Oreku, and E. U. Munir, "Fair-share replication in data grid," *Information Technology Journal*, vol. 7, no. 5, pp. 776–782, 2008.

[24] R. Usui, H. Miyagi, Y. Arakawa, S. Okamoto, and N.

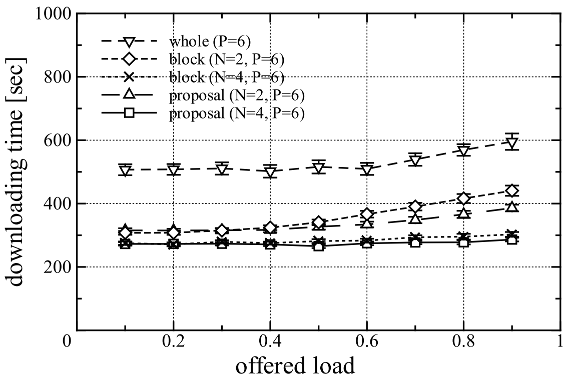


Figure 29. Average downloading time.

Yamanaka, “A novel distributed data access scheme considering with link resources and metric in lambda grid networks,” in *Proc. the 14th Asia-Pacific Conference on Communications*, Tokyo, Japan, Oct. 2008.

[25] X. Yuan, R. Melham, and R. Gupta, “Distributed path reservation for multiplexed all-optical networks,” *IEEE Transactions on Computers*, vol. 48, no. 12, pp. 1355–1363, 1999.

[26] A. Zeitoun, H. Jamjoom, and M. El-Gendy, “Scalable parallel-access for mirrored servers,” in *Proc. the 20th IASTED International Conference on Applied Informatics*, Innsbruck, Austria, Feb. 2002, pp. 93–98.

Kouji Hirata received the B.E. and M.E. degrees in communications engineering from Osaka University in 2003 and 2005, respectively. He received the Ph.D. degree in electrical, electronic, and information engineering from Osaka University in 2008. At present, he is an Assistant Professor in the Department of Electrical and Electronic Engineering and Computer Science at the Graduate School of Science and Engineering of Ehime University. His research interests include optical networks and its performance evaluation. He is a member of IEEE, IPSJ and IEICE.

Khamisi Kalegele received his M.E in Computer Sciences from Ehime University in March 2010. He did his B.Sc in Computer Systems Engineering at the University of Dar es salaam in Tanzania from 1999 to 2003. He has 3 years of experience in GSM radio access network planning, dimensioning and optimization. His interests lie within things related to Information Systems. He is a member of ACM and IPSJ.

Yoshinobu Higami received his B.E., M.E., and D.E. degrees from Osaka University in 1991, 1993 and 1996, respectively. After serving as a research fellow of the Japan Society for the Promotion of Science, he joined Department of Computer Science, Ehime University in 1998. Currently he is an associate professor at Graduate School of Science and Engineering, Ehime University. In 1998 and 2006, he was also an honorary fellow at University of Wisconsin-Madison, U.S.A. He received the IEICE Best Paper Award in 2005. His research interests include test generation, design for testability and fault diagnosis of logic circuits. He is a senior member of the IEEE and a member of IEICE and IPSJ.

Shin-ya Kobayashi received the B.E. degree, M.E. degree, and Dr.E. degree in Communication Engineering from Osaka University in 1985, 1988, and 1991 respectively. He is a Professor at Graduate School of Science and Engineering, director of Center for Information Technology, and special aide to the president, Ehime University. His research interests include distributed processing, and parallel processing. He is a member of the Information Processing Society of Japan, the Institute of Electrical Engineers of Japan, IEICE, IEEE, and ACM.