PAPER

Joint Bandwidth Scheduling and Routing Method for Large File Transfer with Time Constraint and Its Implementation

Kazuhiko KINOSHITA[†a)], ***Member***, **Masahiko AIHARA**[††], **Shiori KONO**[††], ***Nonmembers***, **Nariyoshi YAMAI**[†††], ***and*** **Takashi WATANABE**[††], ***Members***

SUMMARY In recent years, the number of requests to transfer large files via large high-speed computer networks has been increasing rapidly. Typically, these requests are handled in the "best effort" manner which results in unpredictable completion times. In this paper, we consider a model where a transfer request either must be completed by a user-specified deadline or must be rejected if its deadline cannot be satisfied. We propose a bandwidth scheduling method and a routing method for reducing the call-blocking probability in a bandwidth-guaranteed network. Finally, we show their excellent performance by simulation experiments.

key words: *scheduling, routing, file transfer, time constraint*

1. Introduction

In recent years, various types of data have become available in large quantities via large high-speed computer networks. Users hope to be able to access these data files routinely and rapidly by fast transfer. With the increasing diversity of applications, new service guarantees must be offered to users. Real time applications, such as multimedia services or stock market information services require an immediate transfer, whereas backup applications may require a large bandwidth, but not necessarily immediate applications. In the latter case, data transfer completion time is the key service guarantee that users want. In general, any applications that need coordinated use of several resources can benefit from being deadline-aware [1].

There are many studies on file transfer, but most of them focus on shortening the average transfer completion time [2] [3], [4]. In such studies, it is difficult to predict and/or guarantee transfer completion times because they depend strongly on the network conditions [5], [6]. It is not appropriate for the network in which users require service guarantee. Some existing works focus on bandwidth reservation[7], [8]. However, they do not consider strict time constraint.

To overcome this problem, some studies have introduced a model whereby a transfer request must either be completed by a user-specified deadline or rejected if the deadline cannot be met [9], [10]. Note that in this model, it is not necessary to shorten the transfer time below its deadline and it is preferable to accept more requests, thereby reducing the number of rejected requests. To handle many requests that meet their deadline and to reduce the call-blocking probability, it is important to consider the bandwidth allocation for each request.

In this file transfer model, a dynamic bandwidth assignment method called *ChangeRates* has been proposed [11]. *ChangeRates* achieves a reduction in call-blocking probability by considering the minimum bandwidth that will meet the deadline. Moreover, some studies have demonstrated about scheduling with hard deadline [12]–[14].

Some CPU scheduling disciplines can be applied to the bandwidth allocation for a link. In file transfer with time constraint, however, routing is another important factor. A file transfer requires the bandwidth in all links along its route, so that a shorter route makes the amount of the necessary bandwidth smaller.

On the other hand, although many routing algorithms have been proposed, they do not consider time constraint of file transfer. In other words, they find a route based on the number of hops, current available bandwidth and so on. Therefore, a new routing method should be studied.

In this paper, we propose a joint bandwidth scheduling method and routing method for reducing the call-blocking probability in a bandwidth-guaranteed network. The main idea of the proposed methods is considering the bandwidth that will be released by future completion of ongoing transfer [15]. This paper is its extended version.

The remainder of the paper is organized as follows. Section 2 presents the method for transfer within a deadline, using an existing bandwidth assignment method. In Sect. 3, we propose a bandwidth scheduling and routing method. Section 4 evaluates their performance by simulation experiments. Section 5 makes some conclusions.

2. File Transfer with Time Constraints

2.1 Problem Formulation

A transfer request with constraint $R_i (i = 1, 2, \ldots)$ is defined by parameters as follows.

$$R_i = (s_i, d_i, A_i, F_i, D_i) \qquad (1)$$

Here, s_i is the source node, d_i is the destination node,

Manuscript received March 31, 2017.
Manuscript revised July 24, 2017.
Manuscript publicized September 4, 2017.

[†]The author is with Tokushima University, Tokushima-shi, 770-8506 Japan.
[††]The authors are with Osaka University, Suita-shi, 565-0871 Japan.
[†††]The author is with Tokyo University of Agriculture and Technology, Fuchu-shi, 183-0054 Japan.
a) E-mail: kazuhiko@is.tokushima-u.ac.jp

DOI: 10.1587/transcom.2017EBP3121

A_i is the request arrival time, F_i is the file size, and D_i is the deadline of the request. Note that as time passes, F_i and D_i will decrease. Therefore, we use $F_i(t)$ to describe the remaining size of the transfer, and $D_i(t)$ to indicate the remaining time to deadline. We suppose that if the transfer request is finished by the deadline defined by the user, the user is sufficiently satisfied.

For each request R_i, $MinRate_i(t)$ is defined as the minimum average transfer rate that will meet the request's deadline [19]. It can be determined from the file size $F_i(t)$ and the deadline $D_i(t)$ as follows.

$$MinRate_i(t) = \frac{F_i(t)}{D_i(t)} \tag{2}$$

An accepted request can meet its deadline when the allocated bandwidth is $MinRate$ or more. With more bandwidth allocated, the $MinRate$ will decrease monotonically.

In addition, $MaxRate_i(t)$ is defined as the maximum bandwidth that can be allocated to R_i. This is given by the minimum available bandwidth from among all links within the path. The available bandwidth is the difference between the link capacity and the bandwidth allocated for other requests. The value of $Rate_i$, the bandwidth actually allocated for R_i, must be decided in the range of $MaxRate_i$ or below. The available bandwidth for each link will vary according to the routing method.

2.2 Transfer Model

In this paper, we assume the following network characteristics. The bandwidth allocated to each transfer connection is guaranteed, as in Software Defined Network (SDN) and Wavelength Division Multiplex (WDM). There is a database for managing essential information, such as network topology, link capacity, and ongoing requests, which is used to find paths and allocate bandwidths. The access networks are so fast that they can never become bottlenecks. In general, core networks have wider bandwidth than access networks. However, a link in core networks is shared by much more flows than that in access networks, so that each flow cannot obtain enough throughput in core networks.

In such a network, we consider the following transfer model. First, a user requests a transfer within an allowable deadline. For this new transfer request, a feasible route that has a bandwidth sufficient to meet the request's deadline is searched for. After the request's route is decided, the allocation of adequate bandwidth is considered. If the route is not found or the allocation fails, the request is rejected. In the best-effort model, all transfer requests are accepted. However, even when the user defines the deadline for transfer requests, the transfers may not be completed by the deadline if the network is congested. Therefore, in this work, we assume that the network is able to guarantee the bandwidth for transfer requests. In related work [11], in an optical network with WDM, dynamic circuit switching (DCS [18]) is used to guarantee bandwidth. Using DCS, an arbitrary bandwidth can be allocated on demand to each transfer flow. Moreover, SDN and Network Functions Virtualization (NFV) enable networks to be flexibly designed and managed. Consequently, SDN/NFV are suitable for the model being considered here as it requires flexible path and bandwidth control. This transfer model can be applied to the network that must guarantee the quality of service, for example, Software-Defined Data Center (SDDC).

2.3 Existing Methods

At first, we introduce two typical bandwidth assignment methods.

- *Max*: always assigns the maximum available bandwidth $MaxRate$ on demand.
- *Min*: always assigns the minimum rate to meet the deadline $MinRate$ on demand.

Max's advantage is that the whole bandwidth is used and almost no bandwidth remains idle, but it tends to trigger resource competition at high network loads and to rejection of future requests.

To the contrary, file transferring is inefficient and takes more time when using *Min*. However, some bandwidth will remain idle for future requests, and the method can handle many requests in parallel.

In a combination method of *Min* and *Max*, the bandwidth offered to reuquest is min($MaxRate$, $\rho \times MinRate$), where $\rho \geq 1$. For larger values of ρ, $\rho \times Min$ would perform close to *Max*, while for smaller values of ρ, it would perform closer to *Min*.

These methods do not consider the dynamic network state. In [19], ChangeRates method is proposed as an efficient bandwidth allocation method that changes the assigned bandwidth dynamically. The method can decrease the call-blocking probability, while simultaneously guaranteeing $MinRate$ and allocating the remaining bandwidth.

Instead of dividing the bandwidth equally for all ongoing requests, ChangeRates varies the bandwidth allocation according to the ratios of $MinRate$ of the requests using the link.

The bandwidth is allocated by selecting the minimum $Rate_j$ calculated in each link j as follows. Here, C_j is link capacity and $\sum MinRate_j$ is the sum of $MinRate$ of requests using the link j.

$$Rate_j = MinRate_i \frac{C_j}{\sum MinRate_j} \tag{3}$$

[12] studied about an off-line algorithm and an on-line algorithm to schedule packets optimally for multihop wired networks. [13] developed a new protocol to meet application deadlines for cloud data center networks. [14] proposed a flow scheduling method and a routing method for data center networks. [16] studied about a switch buffer control to meet flow deadlines. [17] introduced a deadline-aware advance reservation model for media production networks. [20] showed our fundamental idea. However, they do not consider routing problem.

3. Proposed Method

3.1 Overview

In this section, we propose a routing method and a scheduling method. These methods are invoked only when a new request has arrived. First, a route for a new request is searched. In the proposed routing method, once a route is decided, it never changes until transfer completion. After the routing, the proposed scheduling method is invoked next. It is dynamic bandwidth scheduling, so that the bandwidth allocated to all ongoing requests in the network are flexibly changed. In these methods, if the route cannot be searched or the bandwidth scheduling cannot meet an ongoing requests' deadline the arrived request is rejected. The proposed routing method is designed for the best use of the proposed scheduling method. Therefore, in the following subsections, we explain the scheduling method first.

3.2 Scheduling Method

In the existing method described above, a new request may be rejected without considering the bandwidth that will be released by future transfer completions. This means that it always allocates at least $MinRate$ bandwidth. However, even if the allocated bandwidth is temporarily less than $MinRate$, the transfer can be completed by the deadline.

We propose a bandwidth scheduling method that can meet a request's deadline without always allocating $MinRate$. The proposed method considers the bandwidth that will be freed when an ongoing request will have completed.

Since the length of the transfer time is not need to be considered and all requests may complete just before their deadlines, we consider scheduling a request's bandwidth starting with the request with the latest deadline as long as more bandwidth than $MinRate$ would be allocated later.

Specifically, while requests $R_i (i = 1, \ldots, n)$ are ongoing, suppose that a new request R_{n+1} arrives. First, a path for R_{n+1} is searched. In our path search, a cost from R_{n+1}'s arrival time of A_{n+1} to R_{n+1}'s deadline of D_{n+1} is calculated for each link. After that Dijkstra's algorithm finds the cost optimal path. Specific procedures of this routing is explained in 3.3.

Next, the proposed method schedules starting with the request with the latest deadline. Assume $D_i < D_{i+1}$ for simplicity without loss of generality. The request with the latest deadline is the newly arrived R_{n+1}. Our method considers it first. As there are no requests besides R_{n+1} in the interval from D_n to D_{n+1}, R_{n+1} can occupy the whole bandwidth. f_{n+1} is defined as the maximum file size that R_{n+1} can transfer in $[D_n, D_{n+1}]$. F'_{n+1} and $MinRate'_{n+1}$ in the interval before $[D_n, D_{n+1}]$ are considered as follows (Fig. 1).

$$F'_{n+1} = F_{n+1} - f_{n+1} \tag{4}$$

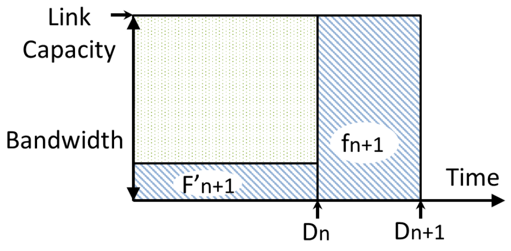

Fig. 1 Bandwidth allocation.

$$MinRate'_{n+1} = \frac{F'_{n+1}}{D_n} \tag{5}$$

In the former interval, the changed $MinRate'_{n+1}$ is provisionally allocated to R_{n+1}. Next, for R_n, because R_{n+1} was allocated provisionally in $[D_{n-1}, D_n]$, the bandwidth obtained by subtracting $MinRate'_{n+1}$ from the link capacity is allocated to R_n. f_n, F'_n and $MinRate'_n$ before $[D_{n-1}, D_n]$ are calculated in the same way. Then, $MinRate'_n$ is provisionally allocated to R_n in the former interval. This is applied to R_{n-1}, R_{n-2} sequentially, to complete the schedule for all requests equally.

Subsequently, we explain an operation when a request cannot meet its deadline. Even if a sufficient bandwidth cannot be allocated to a request before its deadline in this method, it may be possible to meet its deadline by changing the provisional scheduling. Specifically, assume that $R_i (i = 1, \ldots, n + 1)$ cannot meet its deadline. First, $R_j (j = i + 1, \ldots, n + 1)$ that uses the same link as R_i uses from R_{n+1} is searched. Next, the proposed method checks whether R_j has surplus bandwidth that R_j can use in interval $[D_k, D_{k+1}] (k = 1, \ldots, j - 1)$. If R_j has such surplus bandwidth, R_j's bandwidth in $[D_k, D_{k+1}]$ is increased, and R_j's bandwidth before D_k is decreased. Finally, the additional bandwidth that was originally allocated to R_j is reallocated to R_i. These operations are applied to $R_n, R_{n-1}, \ldots, R_{i+1}$.

In this process, if it is found that the new request cannot be completed by its deadline, it is rejected.

After the scheduling procedure, bandwidth allocation to all requests is completed. Note here that there may be some bandwidth which are not allocated to any ongoing requests at the scheduling procedure. While file transferring, such surplus bandwidth is added to ongoing requests based on WFQ (Weighted Fair Queuing) method [21]. WFQ varies the bandwidth for requests according to the ratios of $MinRate$ of the requests using the link. This method is similar to ChangeRate [19]. But, the amount of remaining bandwidth is considered instead of link capacity. First, the bandwidth by selecting the minimum $Rate_k$ calculated as follows is allocated to R_k. Here, C'_k is remaining link capacity.

$$Rate_k = MinRate_i \frac{C'_k}{\sum MinRate_k} \tag{6}$$

After this allocation, available bandwidth still remains because each request uses its own route. Next, the requests that can get no available bandwidth in all link in the path is

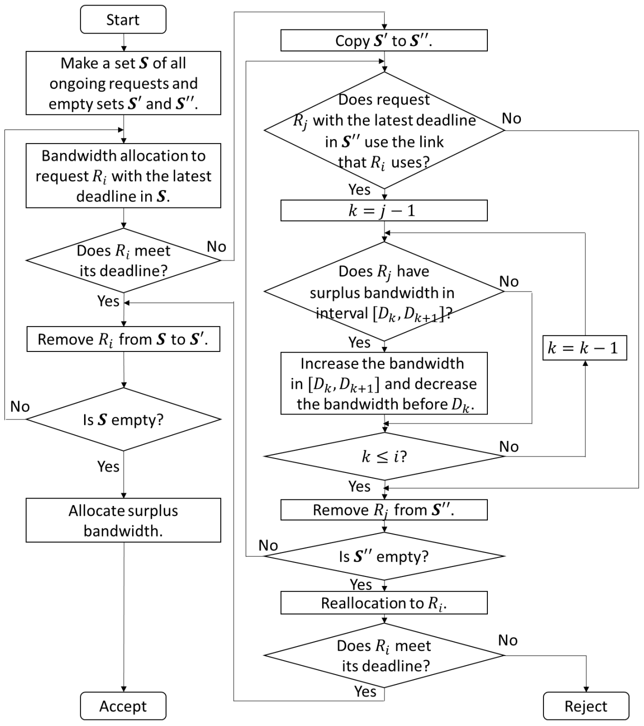
Start
Make a set S of all ongoing requests and empty sets S' and S''.
Bandwidth allocation to request R_i with the latest deadline in S.
Does R_i meet its deadline?
No
Yes
Remove R_i from S to S'.
Is S empty?
No
Yes
Allocate surplus bandwidth.
Accept
Copy S' to S''.
Does request R_j with the latest deadline in S'' use the link that R_i uses?
No
Yes
$k = j - 1$
Does R_j have surplus bandwidth in interval $[D_k, D_{k+1}]$?
No
Yes
Increase the bandwidth in $[D_k, D_{k+1}]$ and decrease the bandwidth before D_k.
$k = k - 1$
$k \leq i$?
No
Yes
Remove R_j from S''.
Is S'' empty?
No
Yes
Reallocation to R_i.
Does R_i meet its deadline?
No
Yes
Reject

Fig. 2 Flowchart of the proposed scheduling.

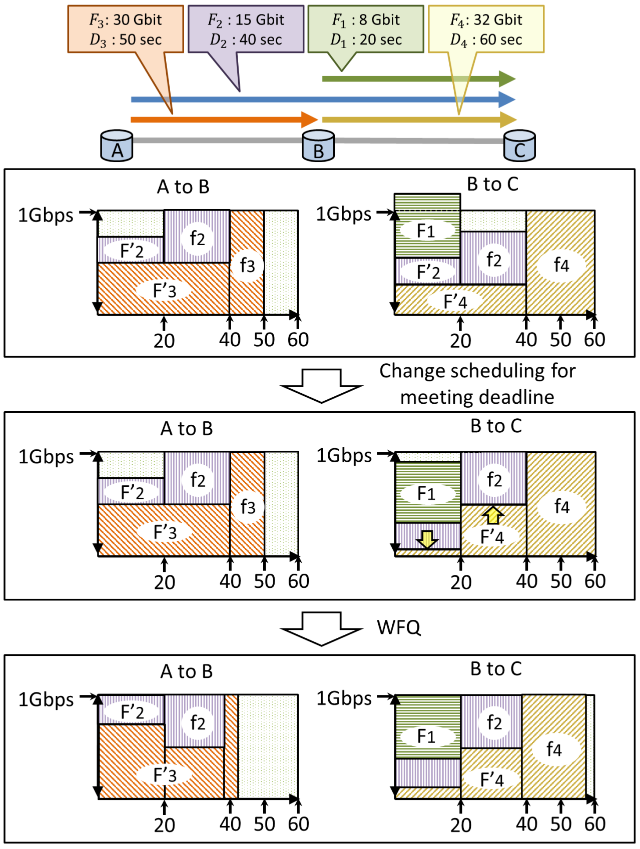
F_3: 30 Gbit D_3 : 50 sec
F_2 : 15 Gbit D_2 : 40 sec
F_1 : 8 Gbit D_1 : 20 sec
F_4: 32 Gbit D_4 : 60 sec
A
B
C
A to B
B to C
1Gbps
F'2
f2
f3
F'3
F1
f4
F'4
20
40
50
60
Change scheduling for meeting deadline
WFQ

Fig. 3 Example of the proposed scheduling method.

excluded from calculation of $\sum MinRate_k$. The allocation is repeated until all request can get no available bandwidth. Figure 2 shows the flowchart of the proposed scheduling.

We demonstrate how the proposed scheduling method works using Fig. 3. There are 4 requests that have different file size and deadlines. First, R_4 that has the latest deadline is scheduled. f_4 and F'_4 are calculated and $MinRate'_4$ is allocated to R_4. In the same manner, R_3, R_2 and R_1 are scheduled.

But R_1's file size is so large that sum of $MinRate'$ exceeds the link capacity. In other words, R_1 cannot complete before its deadline. However, at link B–C, there is the surplus bandwidth in the interval from 20 s to 40 s because of the difference between $MinRate'_3$ and $MinRate'_4$. R_1 and R_2 cannot use this surplus bandwidth because of the restriction of the deadline and link capacity, but R_4 can use this. Therefore, R_1's deadline can be met by increasing R_4's bandwidth in [20, 40] and reducing R_4's bandwidth in [0, 20]

Since a new request has been accepted, the proposed method varies the surplus bandwidth by WFQ. The surplus bandwidth is allocated to all requests according to the rate calculated by Eq. (6). But there is still the surplus bandwidth in link A–B since $Rate_2$ in link B–C is smaller than $Rate_2$ in link A–B. Consequently, WFQ method is applied to the requests again and R_3 can get the bandwidth.

The proposed method decides the scheduling of r ongoing requests. Scheduling is partitioned by each deadline, so that there are r slots. The dominant procedure is "Increase the bandwidth in $[D_k, D_{k+1}]$ and decrease the bandwidth before D_k." in Fig. 2. The time complexity of this procedure is $O(m)$, since a route of each request has at most m hops, where m is the number of links in the network. As a result, the time complexity of the proposed method is $O(mr^2)$. In our assumued application, the order of the time constraint is minute. Compared with it, the computation time is small enough. For each slot, a data including ID of the request which uses the link and the volume of the assigend bandwidth should be registered. Its size is $O(\log r)$. As a result, the space complexity of the proposed method is $O(mr^2 \log r)$. This is small enough for the memory size of present standard systems.

3.3 Routing Method

We also propose a routing method designed for the best use of the proposed scheduling method. When a new request R_i has arrived, cost d_j for link L_j is defined as follows.

$$d_j = \frac{F_j + F_i}{M_j} \tag{7}$$

M_j, F_j and F_i mean the remaining bandwidth except sum of $MinRate$ of ongoing requests from link capacity, the sum of the remaining transfer file size of ongoing requests using link L_j, and the transfer file size of the request R_i, respectively. Dijkstra's algorithm finds the route that minimizes the total cost.

This method has the following three features that can enhance the effect of the dynamic bandwidth allocation method

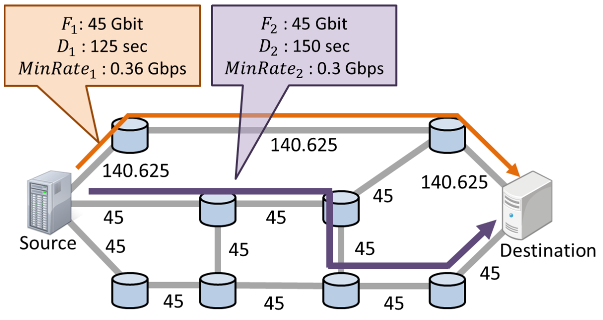

Fig. 4 Example 1 of the proposed routing method.

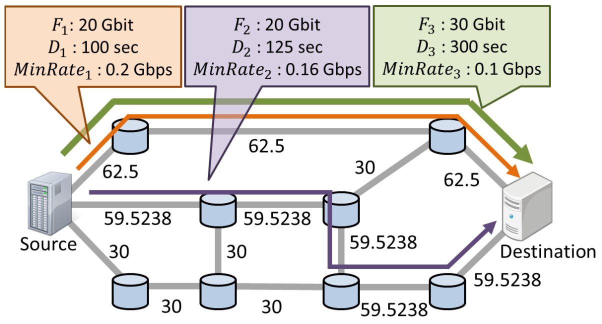

Fig. 5 Example 2 of the proposed routing method.

and reduce the call-blocking probability.

- Since the numerator includes F_i, the number of hops of a route tends to be short. A request whose file size is large uses large bandwidth resources, so that a route of such a request should be short to reduce the bandwidth resources.
- Since the numerator contains F_j, a new route avoids links used for ongoing requests with larger remaining transfer file size. If a link used for a few requests is selected, allocated bandwidth by WFQ becomes larger because a remaining bandwidth is large and sum of other requests' $MinRate$ is small. More bandwidth will be allocated for a request that requires bandwidth for a long time, and the effect of the dynamic bandwidth allocation method is enhanced.
- Since M_j is in the denominator, a new route avoids links with smaller remaining bandwidth. Consequently, a new request is easy to be accepted. Furthermore, available bandwidth in each link will be left equally likely, and flexibility of future requests' routing keep high.

Figures 4 and 5 show an execution example of the proposed routing method. For simplicity, requests are made through the same source and destination, and all links have an identical capacity of 1 Gbps.

In Fig. 4, suppose that R_1 is ongoing and a new request R_2 has arrived. According to the definition of link costs, the costs of the links on R_1's route are $\frac{45+45}{1-0.36} \fallingdotseq 140$ and other costs are $\frac{0+45}{1} = 45$. By using Dijkstra's algorithm, the disjoint route shown by blue arrow is chosen. The links of this route have larger available bandwidth, so that it is easy to accept future requests.

Figure 5 shows the routing when R_3 has arrived 25 seconds later. In the same manner, the costs of the links that no requests use are $\frac{0+30}{1} = 30$. The costs of the links on R_1's route are $\frac{20+30}{1-0.2} = 62.5$ and the costs of the links on R_2's route are $\frac{20+30}{1-0.16} \fallingdotseq 60$. As a result, the same route as R_1 uses is chosen. R_3's file size is larger than R_1 and R_2, so that the shortest route is chosen.

4. Performance Evaluation

4.1 Simulation Model

We evaluated the performance of the proposed methods by simulation experiments. In the experiments, the network had Waxman's random topology [22] with 100 nodes and about 300 links. Each link had a uniform capacity of 1 Gbps. Transfer requests were generated via a Poisson arrival process with an average arrival rate of λ. The call-blocking probability was used as a performance measure. The source and destination nodes for each request were selected randomly. Simulation results were averaged over 30 kinds of random topology generated by the same condition. We picked up the values of the arrival rate between 3.0 and 3.5 where the change of graphs are notable. We evaluated the combinations of the following scheduling methods and routing methods.

- Proposed Scheduling: Dynamic bandwidth allocation was performed using the proposed scheduling method in 3.2.
- Existing Scheduling: Bandwidth allocation was performed using the ChangeRates method [19].
- Proposed Routing : Routing was performed using the proposed routing method in 3.3.
- Existing Routing: Routing was performed using the path with the least hops.

4.2 Simulation Results

First, we evaluate the basic performance of each method. All requests involved a file size of 5 GB and a deadline of 200 seconds. Figure 6 shows the call-blocking probability as a function of arrival rates.

As shown in this figure, the proposed scheduling method and the proposed routing method effectively reduce the call blocking probability compared with the existing methods independent from call arrival rate. Both the proposed routing and the proposed scheduling is effective. Moreover, the combination of the proposed methods achieves the drastic improvement such as an order of magnitude smaller.

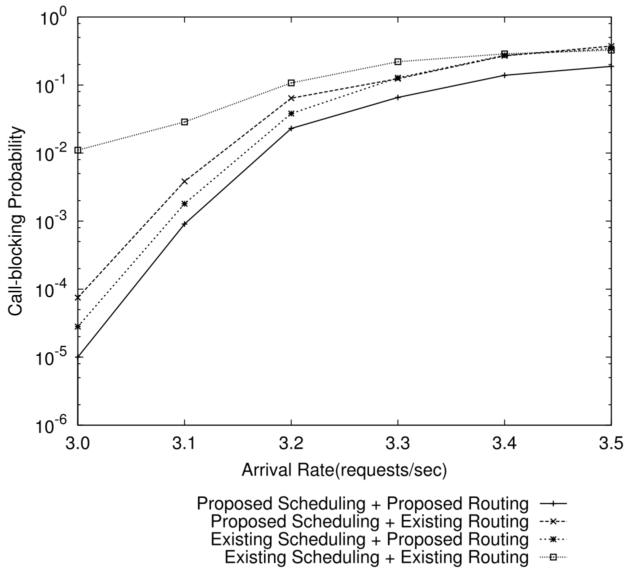

Fig. 6 Basic performance.

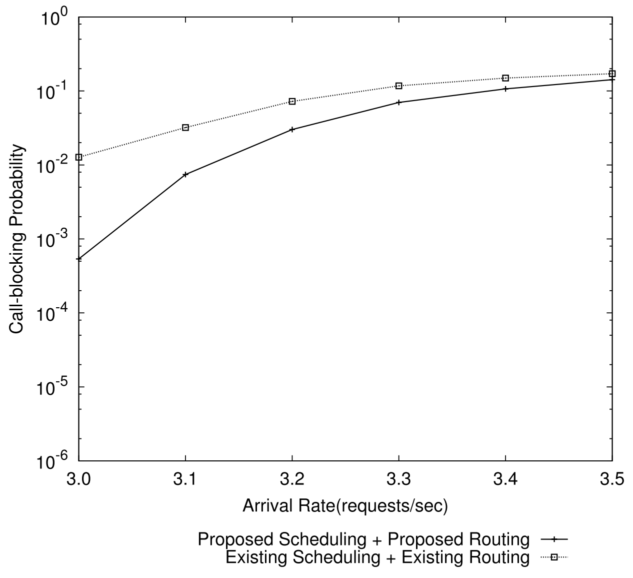

Fig. 8 Total call-blocking probability.

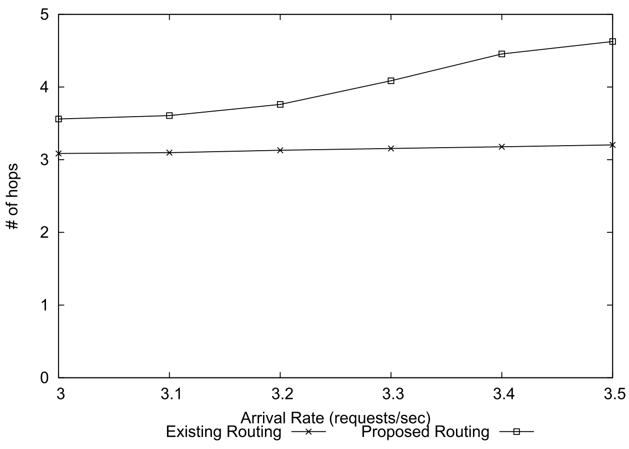

Fig. 7 The number of hops.

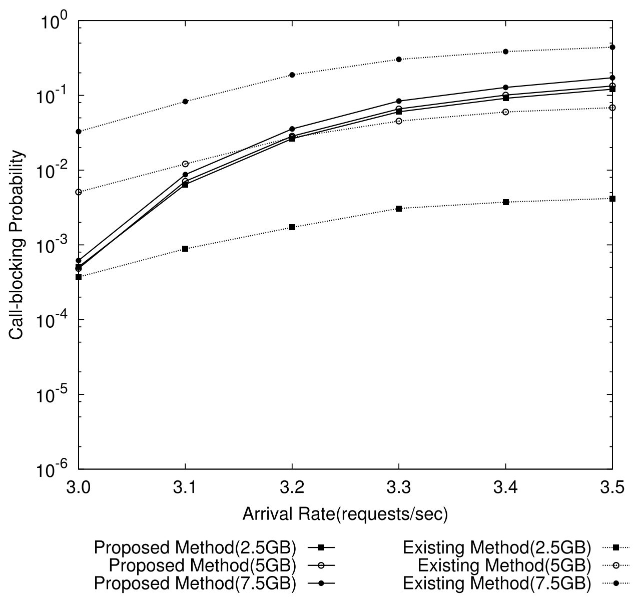

Fig. 9 Call-blocking probability of each class.

Figure 7 shows the mean number of hops by the proposed routing method and the existing routing method. From this graph, it is found that the path generated by the proposed routing was about 20–50% longer than that by the existing routing. It is reasonable, since the existing routing makes the least hop path. While the proposed routing generates longer path, it achieves smaller call blocking probability as shown in Fig. 6. It means that the proposed routing effectively avoids congested links.

Next, we evaluate the effect of differences in requested file size. Three classes of request that have the same deadline of 200 sec but different file size of 2.5 GB, 5 GB, 7.5 GB arrived. These classes of request is randomly chosen with equal probability. Figure 8 shows the total call-blocking probability and Fig. 9 shows the call-blocking probability of each class.

As shown in Fig. 8, the combination of proposed methods is also effective. Higher arrival rate makes smaller difference between the proposed methods and existing methods, since all links get congested and the effect of the scheduling and routing become smaller.

Moreover, as shown in Fig. 9, the proposed methods accepted all classes of requests fairly in spite of the different file size. This is an important feature for practical use.

4.3 Prototyping

To confirm the feasibility of the proposed methods, we made a prototype system as follows.

We used 7 PCs. Their spec is summarized in Table 1. 4 PCs constructed a software-defined network by using trema [23]. A PC included 4 virtual nodes. In the network, each link capacity was 10 [Mbps]. Other 3 PCs played a role of

```
12:07:53(1402369673.91845), request
10MB-1.txt 10.0
12:07:53(1402369673.95246), connected to
the server
12:07:54(1402369674.04069), 0.80000 Mbit,
0.12224 sec, 6.54 Mbps
12:07:54(1402369674.11989), 0.80000 Mbit,
0.07920 sec, 10.10 Mbps
12:07:54(1402369674.19933), 0.80000 Mbit,
0.07943 sec, 10.07 Mbps

(skip)

12:07:56(1402369676.73753), 0.80000 Mbit,
0.07918 sec, 10.10 Mbps
12:07:56(1402369676.81691), 0.80000 Mbit,
0.07938 sec, 10.08 Mbps
12:07:56(1402369676.91714), 0.80000 Mbit,
0.10022 sec, 7.98 Mbps
12:07:57(1402369677.01737), 0.80000 Mbit,
0.10024 sec, 7.98 Mbps
12:07:57(1402369677.11721), 0.80000 Mbit,
0.09984 sec, 8.01 Mbps
12:07:57(1402369677.21735), 0.80000 Mbit,
0.10014 sec, 7.99 Mbps

(skip)

12:07:59(1402369679.71951), 0.80000 Mbit,
0.10004 sec, 8.00 Mbps
12:07:59(1402369679.8196), 0.80000 Mbit,
0.10009 sec, 7.99 Mbps
12:07:59(1402369679.95515), 0.80000 Mbit,
0.13556 sec, 5.90 Mbps
12:08:00(1402369680.09073), 0.80000 Mbit,
0.13557 sec, 5.90 Mbps
12:08:00(1402369680.22637), 0.80000 Mbit,
0.13564 sec, 5.90 Mbps
12:08:00(1402369680.36202), 0.80000 Mbit,
0.13565 sec, 5.90 Mbps

(skip)

12:08:04(1402369684.15899), 0.80000 Mbit,
0.13561 sec, 5.90 Mbps
12:08:04(1402369684.29477), 0.80000 Mbit,
0.13577 sec, 5.89 Mbps
12:08:04(1402369684.43169), Total 10000000
bytes, 10.513 sec, 7.609 Mbps
```

Fig. 10 Experimental log in prototype system.

server and client.

The purpose of the prototyping is to confirm that the proposed scheduling algorithm works well and the assigned bandwidth (bitrate) is provided stably enough by using OpenFlow.

Table 1 PCs in prototyping system.

CPU	Intel Core i5-3470 3.2 [GHz]
# of cores	4
memory	8 [GB]
HDD	500 [GB]
OS	Ubuntu 12.04
software	OpenJDK 1.6.0_24

We generated 10 [MB] of file request several times and observed the transfer rate. Figure 10 shows its log including time, transfered file size, spent time for transmission, and throughtput. As shown in it, we confirmed that transfer rate changed dynamically according to the proposed method. Although the actual rate fluctuated, its range was less than 1%. It comes to the conclusion that the proposed method works well sufficiently.

5. Conclusion

This paper focused on transferring large files with time constraints. We proposed a joint bandwidth scheduling and routing method for reducing the call-blocking probability. Simulation results showed that the combination of the proposed methods achieved the drastic improvement such as an order of magnitude smaller. In addition, its feasibility was confirmed by prototyping.

For shorter time constraint, a distributed control proposed in [24] can be applied.

In a future work, we will apply this model to green ICT.

References

[1] T. Ferrari, "Grid Network Services Use Cases from the e-Science Community," Open grid forum informational document, 2007.
[2] H. Okada, T.N. Trung, K. Kinoshita, N. Yamai, and K. Murakami, "A cooperative routing method for multiple overlay networks," 6th Annual IEEE Consumer Communications and Networking Conference (CCNC 2009), pp.1–2, Jan. 2009.
[3] L. Toka, M. Dell'Amico, and P. Michiardi, "Data transfer scheduling for P2P storage," IEEE P2P, pp.132–141, Sept. 2011.
[4] Y. Chiu and D.Y. Eun, "Minimizing file download time in stochastic peer-to-peer networks," IEEE/ACM Trans. Netw., vol.16, no.2, pp.253–266, April 2008.
[5] S. Kamei, "Status and traffic issues of peer-to-peer technology," Computer Software, vol.22, no.3, pp.8–18, 2005.
[6] S. Gorinsky and N.S. V. Rao, "Dedicated channels as an optimal network support for effective transfer of massive data," 25th IEEE International Conference on Computer Communications (ICCC 2006), pp.1–5, April 2006.
[7] Y. Lin, Q. Wu, N.S.V. Rao, and M. Zhu, "On design of scheduling algorithms for advance bandwidth reservation in dedicated networks," IEEE INFOCOM, pp.1–6, April 2008.
[8] M. Veeraraghavan, H. Lee, E. Chong, and H. Li, "A varying-bandwidth list scheduling heuristic for file transfers," IEEE ICC, pp.1050–1054, June 2004.
[9] B. Chen and P. Primet, "Scheduling deadline-constrained bulk data transfers to minimize network congestion," Proc. 7th IEEE International Symposium Cluster Computing and the Grid (CCGRID), pp.410–417, May 2007.
[10] P. Dharam, C.Q. Wu, and Y. Wang, "Advance bandwidth reservation with deadline constraint in high-performance networks," International Conference on Computing, Networking and Communications

(ICNC 2014), pp.1041–1045, April 2014.
[11] D. Andrei, "Provisioning of deadline-driven requests with flexible transmission rates in WDM mesh networks," IEEE/ACM Trans. Netw., vol.18, no.2, pp.353–366, 2010.
[12] Z. Mao, C.E. Koksal, and N.B. Shroff "Optimal online scheduling with arbitrary hard deadlines in multihop communication networks," IEEE/ACM Trans. Netw., vol.24, no.1, pp.177–189, 2014.
[13] J. Hwang, J. Yoo, and N. Choi, "Deadline and incast aware TCP for cloud data center networks," Computer Networks, vol.68, pp.20–34, Aug. 2014.
[14] L. Wang, F. Zhang, and K. Zheng, "Energy-efficient flow scheduling and routing with hard deadlines in data center networks," 34th IEEE International Conference on Distributed Computing Systems, pp.248–257, 2014.
[15] M. Aihara, S. Kono, K. Kinoshita, N. Yamai, and T. Watanabe, "Joint bandwidth scheduling and routing method for large file transfer with time constraint," Proc. 8th IEEE/IFIP International Workshop on Management of the Future Internet (ManFI2016) in conjunction with IEEE/IFIP Network Operations and Management Symposium (NOMS2016), pp.1125–1130, April 2016.
[16] J. Zhang, "Deadline-aware bandwidth sharing by allocating switch buffer in data center networks," IEEE INFOCOM 2016, pp.1–9, April 2016.
[17] M. Barshan, H. Moens, J. Famaey, and F. De Turck, "Deadline-aware advance reservation scheduling algorithms for media production networks," Comput. Commun., vol.77, pp.26–40, March 2016.
[18] B. Mukherjee, "Architecture, control, and management of optical switching networks," IEEE/LEOS Photonics in Switching Conference, pp.43–44, Aug. 2007.
[19] D. Andrei, M. Barayneh, S. Sarkar, C.U. Martel, and B. Mukherjee, "Deadline-driven bandwidth allocation with flexible transmission rates in WDM networks," IEEE International Conference on Communications (ICC 2008), pp.5354–5358, June 2008.
[20] K. Katsumoto, K. Kinoshita, N. Yamai, and K. Murakami, "A bandwidth assignment method for downloading large files with time constraints," 5th International Conference on Emerging Network Intelligence (EMERGING 2013), Oct. 2013.
[21] J.F. Kurose and K.W. Ross, Computer Networking: A Top-Down Approach, 4/E, Addison-Wesley Computing, 2007.
[22] B.M. Waxman, "Routing of multipoint connections," IEEE J. Sel. Areas Commun., vol.6, no.9, pp.1617–1622, Dec. 1988.
[23] Trema, Full-Stack OpenFlow Framework in Ruby and C, https://trema.github.io/trema/
[24] K. Saito, K. Kinoshita, N. Yamai, and T. Watanabe, "A distributed bandwidth assignment method for large file transfer with time constraints," 17th Asia-Pacific Network Operations and Management Symposium (APNOMS2015), pp.279–284, 2015.

Kazuhiko Kinoshita was born in Osaka, Japan, on June 13, 1973. He received the B.E., M.E. and Ph.D degrees in information systems engineering from Osaka University, Osaka, Japan, in 1996, 1997 and 2003, respectively. From April 1998 to March 2002, he was an Assistant Professor at the Department of Information Systems Engineering, Graduate School of Engineering, Osaka University. From April 2002 to March 2008, he was an Assistant Professor at the Department of Information Networking, Graduate School of Information Science and Technology, Osaka University. From April 2008 to January 2015, he was an Associate Professor at the same University. Since February 2015, he has been a Professor at Tokushima University. His research interests include mobile networks, network management, and agent communications. Dr. Kinoshita is a member of IEEE.

Masahiko Aihara received the bachelor's degree in information systems engineering and master's degree in information networking from Osaka University, Osaka, Japan in 2015 and 2017, respectively. Since April 2017, he has been worked for Canon Inc.

Shiori Kono received his bachelor's degree in engineering and master's degree in information networking from Osaka University, Osaka, Japan, in 2013 and 2015, respectively. Since April 2015, he has been worked for Fujitsu Limited.

Nariyoshi Yamai received his B.E. and M.E. degrees in electronic engineering and his Ph.D. degree in information and computer science from Osaka University, Osaka, Japan, in 1984, 1986 and 1993, respectively. In April 1988, he joined the Department of Information Engineering, Nara National College of Technology, as a research associate. From April 1990 to March 1994, he was an assistant professor in the same department. In April 1994, he joined the Education Center for Information Processing, Osaka University, as a research associate. In April 1995, he joined the Computation Center, Osaka University, as an assistant professor. From November 1997 to March 2006, he joined the Computer Center, Okayama University, as an associate professor. From April 2006 to March 2014, he was a professor in the Information Technology Center (at present, the Center for Information Technology and Management), Okayama University. Since April 2014, he has been a professor in the Institute of Engineering, Tokyo University of Agriculture and Technology. His research interests include distributed system, network architecture and Internet. He is a member of IPSJ and IEEE.

Takashi Watanabe is a Professor of Graduate School of Information Science and Technology, Osaka University, Japan. He received his B.E., M.E. and Ph.D. degrees from Osaka University, Japan, in 1982, 1984 and 1987, respectively. He joined Faculty of Engineering, Tokushima University as an assistant professor in 1987 and moved to Faculty of Engineering, Shizuoka University in 1990. He was a visiting researcher at University of California, Irvine from 1995 through 1996. He has served on many program committees for networking conferences, IEEE, ACM, IPSJ (Information Processing Society of Japan), IEICE (The Institute of Electronics, Information and Communication Engineers, Japan). His research interests include wireless networking, mobile networking, ad hoc networks, sensor networks, ubiquitous networks, intelligent transport systems, specially MAC and routing. He is a member of IEEE, IEEE Communications Society, IEEE Computer Society as well as IPSJ and IEICE.