One-to-many file transfer using multipath multicast with gossiping

著者	Heira Kenji, Ogawa Kyohei, Tsuru Masato
journal or publication title	2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)
page range	188-191
year	2018-06
URL	http://hdl.handle.net/10228/00006939

doi: info:doi/10.1109/NETSOFT.2018.8460123

One-to-many file transfer using multipath-multicast with gossiping

Kenji Heira, Kyohei Ogawa
Computer Science and Systems Engineering
Kyushu Institute of Technology
Fukuoka, Japan
{heira,ogawa}@infonet.cse.kyutech.ac.jp

Masato Tsuru
Computer Science and Systems Engineering
Kyushu Institute of Technology
Fukuoka, Japan
tsuru@cse.kyutech.ac.jp

***Abstract*—With the recent progress of cloud and distributed computing technologies, data migration and replication among distributed data centers grows rapidly. To manage a simplified scenario that a single sender sends a large-sized file to multiple recipients, i.e., one-to-many file transfer, on a network with full-duplex links, we are developing the Multipath-Multicast (MPMC) file transfer. A file is appropriately divided into equally-sized blocks; different blocks are concurrently transmitted to the same recipient on multiple paths; while the same block is concurrently transmitted to multiple recipients by multicast, aiming at shorter reception completion times of all recipients. However, on large-scale complex network topologies, it is not easy to find a good block transfer schedule, i.e., that realizes the reception completion times of most recipients close to their lower-bounds in MPMC. In this report, therefore, a gossiping approach to allow block transfer among recipients is introduced into MPMC and evaluated through simulation on two real backbone topologies. Since unused capacities of links in the original basic MPMC can be utilized in the MPMC with gossiping, a good schedule can be found more easily compared with the basic MPMC even with the same simple greedy block allocation.**

***Index Terms*—OpenFlow, multipath transfer, multicast transfer, one-to-many file transfer, gossiping**

I. INTRODUCTION

As cloud and distributed computing technologies are widely spread, a rapid growth of data migration and replication either in a data center or among distributed data centers is becoming a problem. Therefore, it is of necessity that a single sender sends a large-sized file to multiple recipients, i.e., one-to-many file transfer, to complete the file reception by each recipient as fast as possible.

Transferring a file on a single multicast tree by "reliable" multicasting is an approach to efficient one-to-many file transfer. Although it can prevent wasting link capacities in general and some commercial products for multicast file transfer are available [1], the reception completion times are throttled by the recipient at the worst location. Addressing this weakness, the use of multiple multicast trees with appropriately grouping recipients was considered [2], but it does not utilize the aggregated capacity of multiple network paths. On the other hand, data transmission leveraging multiple paths has emerged and attracted attention for diverse applications [3]. For example, a multipath transmission scheme was proposed that employs a fountain code to encode transmission data and achieves higher reliability without retransmissions [4], but it does not utilize multicasting. For one-to-many file transfers in a data center, a P2P-based data dissemination scheme was proposed focusing on typical data center network topologies such as the FatTree and Multi-Routed Tree [5].

Based on these observations, we proposed a one-to-many file transfer scheme, Multipath-Multicast (MPMC), with the use of multiple paths and multicast to fully and efficiently utilize the link capacity at the same time. In particular, we focused on a network with full-duplex links in which different data can be transmitted in both directions of the same link simultaneously, implicitly assuming dedicated high-speed wired links. The MPMC was implemented based on the capability of flexibly controllable routing of OpenFlow [6]. To shorten the reception completion time of each recipient in MPMC, a good transfer schedule in space and time (i.e., route and phase) is essential. However, on large-scale complex network topologies, good schedules in MPMC are not easy to find. In this report, therefore, a gossiping approach to allow block transfer among recipients is introduced in MPMC and evaluated on two large-scale topologies through simulation. Although "gossiping" often represents a data exchange among recipients in an ad-hoc, autonomous, and unsynchronized manner, in our work, the block transfer among recipients is globally scheduled by controller to transfer a block from an already received recipient to an unreceived recipient as long as the link capacity allows.

The reminder of this paper is organized as follows. The MPMC one-to-many file transfer is outlined in Section 2. The MPMC with gossiping is introduced in Section 3 and evaluated in Section 4. Concluding remarks are given in Section 5.

II. MULTIPATH-MULTICAST FILE TRANSFER

The MPMC one-to-many file transfer was proposed in our previous work [6], in which a file is divided into equally-sized blocks; different blocks are concurrently transmitted to the same recipient on multiple paths to fully utilize the link capacity; and the same block is concurrently transmitted to multiple recipients by multicast to efficiently utilize the link capacity. An essential assumption is that all links are full-duplex so that different data can be transmitted in both directions of the same link simultaneously. In addition, for simplicity, a fixed, dedicated, error-free communication network is assumed. The block transfer in space and time is scheduled in a centralized manner meaning that the central scheduler (i.e., controller) knows the network topology, link capacities, and sender/recipients' locations.

The aim of scheduling is to achieve the reception completion time (**RCT**) of each recipient as short as possible, where RCT of recipient R is defined as the duration from the time when the sender starts sending the file to at least one recipient to the time when recipient R has received the entire contents of the file. The RCT inherently varies among recipients due to their heterogeneity in the network topology. The lower-bound of RCT of recipient R is equal to the file size divided by the **max-flow value** (i.e., the aggregate capacity of a concurrent transfer on possibly multiple paths) from sender S to R, and an "optimal schedule" will realize such RCT lower-bounds for every recipient. Therefore, if the number of recipients is one, this scheduling problem is solved simply by applying the standard **max-flow problem**. However, if the number of recipients is more than one, this problem is different from the well-known single-source multiple-sink max-flow problem. Although it can be formulated as a kind of optimization problem, an efficient algorithm to solve it is not trivial. Therefore we proposed a simple greedy block allocation to generate a schedule relying on the max-flow problem that can be solved by well-known algorithms efficiently, e.g., with a computational cost proportional to the number of links on the network.

The basic idea is as follows. Let S be the sender who possesses multiple blocks to be sent to multiple recipients. Block transfer is scheduled on a phase-by-phase basis, i.e., it is determined that which block is sent to which recipient on which path (route) from S in each phase, which is called the **block allocation**. The max-flow value and a set of paths from S to each recipient to realize the value are computed and leveraged in the block allocation. Those paths are called **the max-flow paths** although the set of the max-flow paths is not unique. To define a **phase**, let b_j be the max-flow value from the sender to recipient j and w_j be the total size of unreceived blocks for j at the beginning of the phase. The **remaining-data transmission time** D_j of recipient j is computed as $D_j = w_j/b_j$, i.e., the minimum necessary time to receive the entire file by j. A recipient that has the (non-zero) shortest remaining-data transmission time is selected as the **primary recipient** of the phase. In a tie-break, a pre-defined ordered list of recipients (called the **recipient order parameter**) is used to decide the primary recipient. All unreceived blocks for the primary recipient are allocated to its max-flow paths to send; the current phase ends at the time the file reception by the primary recipient has completed. In other words, the time duration of a phase is the remaining-data transmission time of the primary recipient of the phase.

On the other hand, recipients other than the primary recipient are called the **secondary recipients** of the phase. The secondary recipients are also sorted in the ascending order of their remaining-data transmission times as block allocation priority. In a tie-break, the recipient order parameter is used again to decide the order of the secondary recipients in the current phase. According to that order as priority, each secondary recipient is examined to allocate unreceived blocks on its max-flow paths that (partially) overlaps one of the max-flow paths of the primary recipient as long as unused capacities of links and unprocessed recipients exist in the current phase. In this way, the max-flow paths of the primary recipient will branch and reach some other secondary recipient, which forms multiple multicast trees. Such a simple and greedy block allocation without backtracking for each phase is repeated until all recipients have received all blocks, i.e., the one-to-many file transfer has completed.

An example of MPMC file transfer is illustrated in Fig.1. Hosts (S, R_1, R_2, R_3) and switches $(0, 1, 2, 3, 4)$ are connected at 1 [Gbps], respectively; and these switches are connected each other at 100 [Mbps]. S transfers the same file to R_1, R_2, and R_3. The max-flow value b_j from S to R_j are $b_1 = 200$, $b_2 = 300$, $b_3 = 200$ [Mbps], respectively. The max-flow paths to each recipient can be a set of unit-capacity paths (called as **unit-paths**). In this case, the unit capacity is 100 [Mbps] and the **normalized max-flow value** M_j is defined by $b_j/100$, i.e., the number of the unit-paths to realize the max-flow value; $M_1 = 2, M_2 = 3, M_3 = 2$. The file to send is divided by the least common multiple (LCM) of the normalized max-flow values of all recipients. Thus, the file is divided into six blocks in this case. The recipient order parameter is set to (R_1, R_2, R_3),

A snapshot of block transfer in the 2nd phase and a total block allocation over three phases are illustrated in Fig.1 and the left of Fig.2, respectively. At the beginning of the 1st phase, the remaining-data transmission time of each recipient is in the following order: $D_2 < D_1 = D_3$. Therefore, the primary recipient of the 1st phase is R_2. In the 1st phase, R_2 has completed the file transfer, while R_1 receives blocks $0, 1, 2, 3$ and R_3 receives blocks $0, 1, 4, 5$ as its secondary recipients in this schedule. At the beginning of the 2nd phase, the remaining-data transmission time R_1 and R_3 are the same. Therefore the primary recipient of the 1st phase is R_1 according to the pre-defined recipient order parameter. In the 2nd phase, R_1 has completed the file transfer, while R_3 receives blocks 3 only as its secondary recipient. Six blocks are transferred on three unit-paths to the primary recipient (R_2) in the 1st phase, while two blocks are transferred on two unit-paths to the primary recipient (R_1) in the 2nd phase. Therefore, the time duration of the 2nd phase is half of that of the 1st phase. Finally, in the 3rd phase, the last recipient R_3 has completed the file transfer but this schedule is not optimal. R_1 and R_2 achieve their lower-bounds of RCT but R_3 does not. On more large-scale complex network topologies, a good schedule is sometimes difficult to obtain based solely on the simple greedy block allocation.

Of further note is on our preliminary MPMC system for Ethernet-based networks [6]. An MPMC controller was implemented on Trema [8], which is a Ruby-based framework

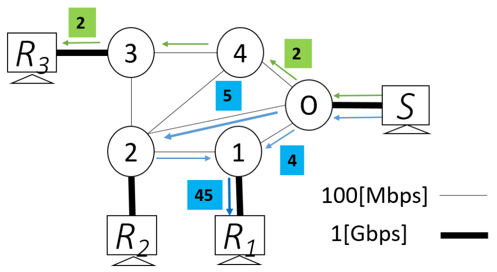

Fig. 1. An example of block transfer in the 2nd phase.

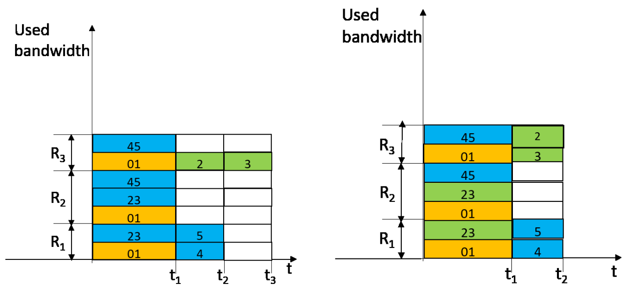

Fig. 2. A block transfer schedule (Left:Basic, Right:Gossiping).

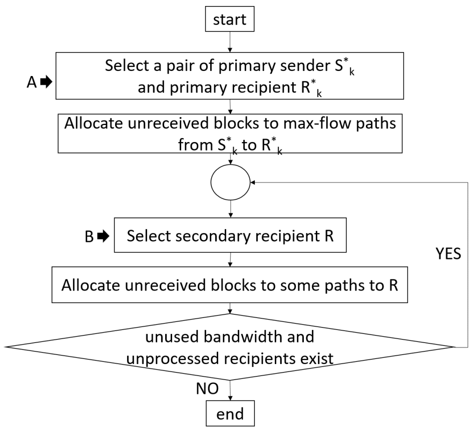
start
A
Select a pair of primary sender S*k and primary recipient R*k
Allocate unreceived blocks to max-flow paths from S*k to R*k
B
Select secondary recipient R
YES
Allocate unreceived blocks to some paths to R
unused bandwidth and unprocessed recipients exist
NO
end

Fig. 3. Block allocation process in phase k

for OpenFlow Controller (OFC), to manage different types of OpenFlow Switchs (OFSs) including hardware and software ones. The sender agent on the sender host and the recipient agent on each recipient host were also implemented. In our OpenFlow-based implementation, each unit-path is flexibly routed as a UDP flow and the multicast forwarding is realized using the packet-copy on OFS. The MPMC controller collects the network topology, accepts a sending request and receiving requests from the sender and the recipient agents, decides a schedule, configures each OFS, and starts the block transfers by directing the sender and the recipient agents. In the current system, lost packets will be retransmitted by unicast after finishing all phases.

III. Multipath-multicast file transfer with gossiping

In the same manner as the original MPMC (called the **basic MPMC**), let S be **the original sender** who originally possesses multiple blocks to be sent to multiple recipients. Block transfer is scheduled on a phase-by-phase basis. In contrast to the basic MPMC, however, a recipient can retrieve some blocks from not only S but also other recipients who already possess such blocks in case of gossiping. Therefore, the max-flow value from a recipient as potential sender to another recipient should be considered, and the remaining-data transmission time of each recipient R with a potential sender R' is computed. Each phase ends at the time when the primary recipient of the current phase has received all blocks, and then a potential sender and a recipient for the next phase will be selected, until all recipients have received all blocks, i.e., the file transfer has completed. Note that the primary sender of the 1st phase is always original sender S and the primary recipient of the 1st phase is the same one in the basic MPMC.

The flowchart of a certain phase is shown in Fig.3 and two main functions are described in the following subsections.

A. Select a pair of the primary sender and the primary recipient

To define a phase, each pair of a potential sender (either the original sender S or a recipient who has already received all blocks in the preceding phases) and a potential recipient (a recipient who has not received all blocks yet) is examined in terms of the remaining-data transmission time of the pair. A pair with the shortest remaining-data transmission time is selected as the primary sender S^*_k and the primary recipient R^*_k of phase k. If there are two or more pairs having the shortest remaining-data transmission time, the one with the smallest number of hops from S^*_k to R^*_k is chosen. In a tie-break, the recipient order parameter is used to decide S^*_k and R^*_k.

B. Select the secondary recipient

The secondary recipients of phase k are sorted in the ascending order of the remaining-data transmission time of the recipient with the primary sender S^*_k as allocation priority. In a tie-break,I the one with the smallest number of hops from S^*_k to recipient R is chosen. If a tie-break still happens, the recipient order parameter is used again to decide the order of the secondary recipients of the current phase. Similarly to the basic MPMC, according to that order as priority, each secondary recipient R is examined to allocate unreceived blocks on its max-flow paths from S^*_k that (partially) overlaps one of the max-flow paths of R^*_k from S^*_k as long as unused capacities of links and unprocessed recipients exist in the current phase. However, in contrast to the basic MPMC, a block can be transferred to secondary recipient R not only from S^*_k but also any recipient who already possesses the block and is directly connected to a switch on one of the max-flow paths of R.

C. Example

To explain the proposed MPMC with gossiping, the same topology of Fig.1 is used where R_1, R_2, and R_3 have the max-flow values of $b_1 = 200$, $b_1 = 300$, and $b_1 = 200$

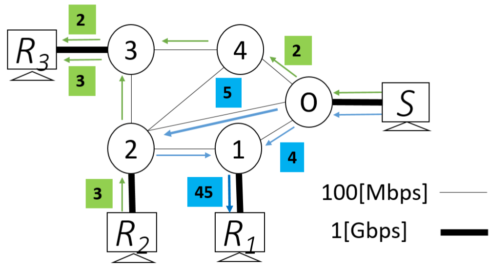

Fig. 4. An example of block transfer with gossiping in the 2nd phase.

[Mbps], respectively. The file is divided into six blocks B_j ($j = 0, 1, ..., 5$). The recipient order parameter is set to (R_1, R_2, R_3). In the first phase, since only the original sender S possesses all the blocks, S becomes the primary sender. Since R_2 has the shortest remaining-data transmission time from the primary sender, all blocks are transferred to R_2 as the primary recipient of the phase. At the same time, B_0, B_1, B_2, B_3 are delivered to R_1 by multicast transfer, and B_0, B_1, B_4, B_5 are delivered to R_3.

In the second phase, a pair of the primary sender and the primary recipient of the phase should be decided. There are two potential senders; the original sender S and recipient R_2 who has already received all blocks in the first phase. Among multiple pairs that have the shortest remaining-data transmission time, the pairs (S, R_1), (R_2, R_1), and (R_2, R_3) that have the short hop number are chosen. According to the given recipient order parameter, the pairs (S, R_1) and (R_2, R_1) are chosen. Finally, since the original sender S is generally prioritized as the primary sender, the pair (S, R_1) is selected so that all unreceived blocks of R_1 are transferred from S. At the same time, some blocks are transferred from S to R_3 as the secondary recipient. Notably, in contrast to the basic method, some blocks can also be transferred from R_1 to R_3 using the unused capacities on the paths from R_1 to R_3. The schedule over two phases is shown in the right of Fig.2. A snapshot of block transfer in the 2nd phase is shown in Fig.4 where the link from switch 2 to 3 is utilized.

IV. PERFORMANCE EVALUATION

A. Schedule search

For both MPMC methods (with and without gossiping), given the recipient order parameter, one schedule instance is generated as explained in the previous sections. By varying the recipient order parameter at random N times, at most N different schedules are generated. We define **Maximum RCT** and **Average RCT** to measure the performance of a schedule. The maximum reception completion time (Maximum RCT) is the longest value of the RCTs of all recipients and the average RCT is the average value of the RTCs of all recipients. The recipient order parameter strongly affects the resulting block allocation and thus the generated schedule instance especially when a number of recipients have the same max-flow value. The best one schedule is chosen from at most N schedule instances so as to minimize the maximum RCT first and then to minimize the average RCT among those instances, called **schedule search** with the number N of schedule generations.

In the following subsections, the basic and the proposed gossiping MPMC methods are simulated on two large-scale topologies [7] (resenting real backbone networks) but in an ideal condition, i.e., with no packet loss and no link propagation delay. These methods are compared in terms of the RCTs of recipients realized by the schedule obtained by "schedule search" with different N of schedule generations (N is 10, 25, and 50). To get statistical results, in each setting, the simulation experiments are repeated by 50 times to get the best, average, and worst case performance of each method with each of different schedule generations.

B. Evaluation at Columbus

The performance was evaluated for transferring 100 [MB] on a topology of Fig. 5 (top) in which 69 switches are connected each other at 100 [Mbps]; one selected switch is connected to the sender and each of other switches is connected to each different recipient at 1 [Gbps] (larger than its max-flow value). Two sender locations (no. 1 and no. 29) are randomly selected and examined. The maximum value and the average value of the lower-bounds of RCT per recipient are 8.00 and 4.44 [s] from the sender located at no. 1, respectively; 8.00 and 4.48 [s] from the sender located at no. 29. The file is divided into 12 and 6 blocks for senders at no. 1 and no. 29, respectively. This is because, in case of the sender located at no. 1 for example, the normalized max-flow values of recipient are 1, 2, 3, and 4.

The results on "The maximum RCT" are omitted to show because the values were equal to the lower-bound (8 [s]) in almost cases and there was not much difference between the basic method and the gossiping method. About "The average RCT" (averaged over 68 recipients), Fig.5 (bottom) shows the statistical performance results of the schedule obtained by schedule search for two sender locations. In the results, all of the best, average, and worst case performances among 50 experiential trails are clearly improved by the gossiping method. With a sufficient number of schedule generations (50), even the worst case performance in the gossiping method is close to its lower-bound (4.44 and 4.48 [s] for senders at no. 1 and no. 29, respectively), meaning that the RCT of each recipient is close to its lower-bound in the majority of recipients. With a small number of schedule generations (10), the average case performance in the gossiping method is still close to its lower-bound. The worst case performance (among 50 trials) is also significantly improved.

As a deeper look into the difference, a certain schedule of the basic method and another certain schedule of the gossiping method (both are generated using the same recipient order parameter) are compared in case of the sender location at no. 29. In the basic method, it takes 8 [s] over 6 phases, while in the gossiping method, it takes 8 [s] over 4 phases to finally complete the file transfer. Therefore, in terms of the maximum RCT, two methods show the same and optimal performance. However, in the basic method, there are 13 recipients who do

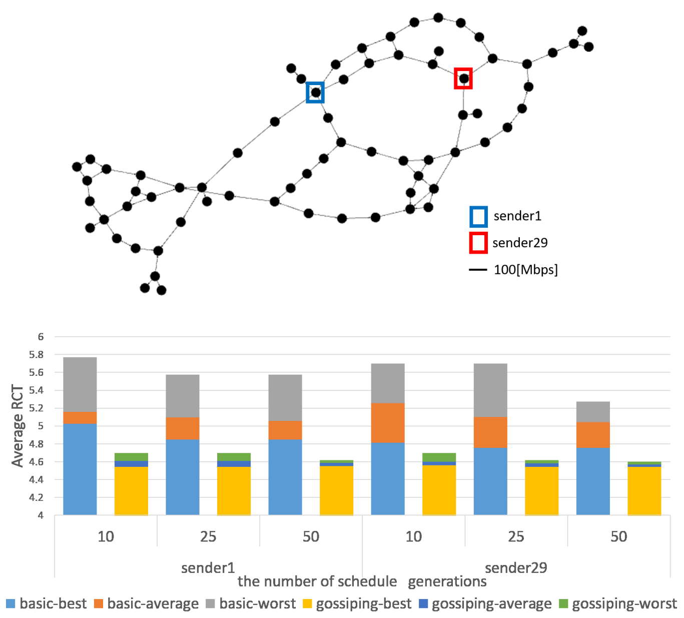

Fig. 5. Topology Columbus [7] and the average RCT over 68 recipients.

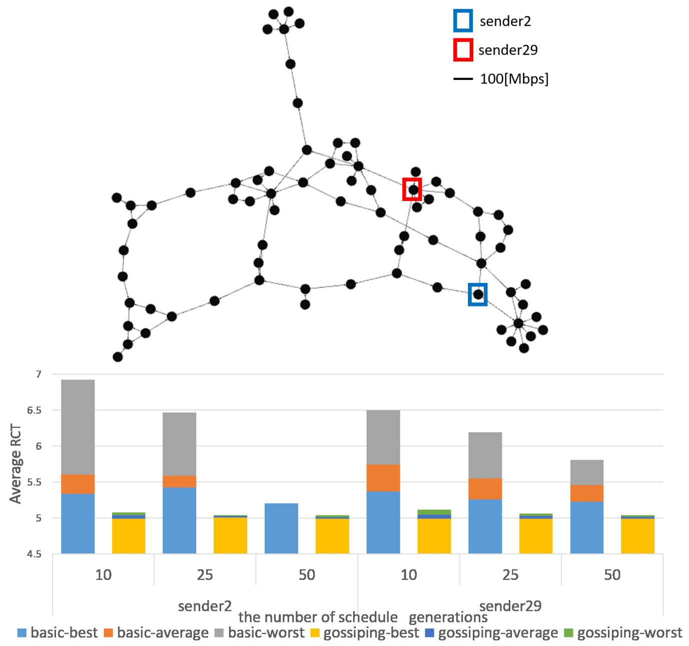

Fig. 6. Topology Switch [7] and the average RCT over 72 recipients.

not reach the lower-bound, while in the proposed method there are only 3 recipients who do not reach the lower-bound.

C. Evaluation at Switch

Another evaluation was conducted for transferring 100 [MB] on a topology of Fig. 6 (top) with 73 switches in a way similar to the previous Columbus case. Two sender locations (no. 2 and no. 29) are selected. The maximum value and the average value of the lower-bounds of RCT per recipient are 8.00 and 4.95 [s] from the sender at no. 2, respectively; 8.00 and 4.94 [s] from the sender at no. 29. For both sender locations, the file is divided into 6 blocks.

The results on the average RCT in Fig.6 (bottom) indicate that the performance characteristics are almost similar with the Columbus case. The advantage of the gossiping method to the basic method in terms of the average RCT is more significant in case with a small number of schedule generations (10).

V. CONCLUDING REMARKS

We have introduced a gossiping approach to allow inter-recipients transfer into MPMC-style one-to many file transfer so that unused capacities of links have more chances to leverage. The simulation results on two large-scale topologies suggest that a good schedule, i.e., by which most recipients achieve the RCTs close to their lower-bounds, is more easily found in the proposed gossiping method.

A short discussion on the proposed method in terms of computational costs and adaptability to changes is given below. In the block allocation, the max-flow problem should be solved to decide the max-flow paths between two hosts. Thus, if the number of involved hosts (the original sender and all the recipients) is large, it needs to be computed many times, i.e., for each potential pair among hosts. However, if the topology does not change, such computations can be conducted before-hand. To adapt dynamic changes of network topology including link capacities and switch/link availability, more work is needed. In addition, the one-to-many file transfer is assumed to start after all recipients join. A delayed join of recipients remains as future work. About the cost of schedule search, the experimental results suggest the proposed method accept a relatively small number of search.

We will investigate how a good schedule is found more effectively and how it depends on the network topologies including the locations of sender and recipients. We also plan to implement the proposed method with OpenFlow by enhancing our implementation of the basic method and demonstrate its feasibility on a wide-area OpenFlow testbed in Japan.

This research is supported by JSPS Grant-in-Aid for Scientific Research 16K00130.

REFERENCES

[1] K. Miller, K. Robertson, et al., "StarBurst Multicast File Transfer Protocol (MFTP) Specification," Internet Draft, draft-miller-mftp-spec-03.txt , April 1998.
[2] S. Bhattacharyya, J. Kurose, et al., "Efficient rate-controlled bulk data transfer using multiple multicast groups," IEEE/ACM Trans. Networking 11(6):895–907, 2003.
[3] S. K. Singh, T. Das, A. Jukan, "A Survey on Internet Multipath Routing and Provisioning," IEEE Commu. Surv. Tutorials 17(4):2157–2175, 2015.
[4] Y. Cui, X. Wang, et al., "FMTCP: A Fountain Code-Based Multipath Transmission Control Protocol," Proc. IEEE ICDCS'12, pp.366–375.
[5] Y. Zhao, J. Wu, and C. Liu, "On Peer-Assisted Data Dissemination in Data Center Networks: Analysis and Implementation," Tsinghua Science and Technology, 19(1):51–64, 2014.
[6] A. Nagata, Y. Tsukiji, M. Tsuru, "Delivering A File by Multipath-Multicast on OpenFlow networks," Proc. IEEE INCoS'13, pp.835—840.
[7] The Internet Topology Zoo, http://www.topology-zoo.org/ (accessed on Dec 18, 2017.)
[8] Trema, https://trema.github.io/trema/ (accessed on Dec 18, 2017.)