Coded-MPMC: One-to-Many Transfer Using Multipath Multicast With Sender Coding

MASAYUKI KURATA, MASAHIRO SHIBATA, AND MASATO TSURU
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Fukuoka 820-8502, Japan
Corresponding author: Masayuki Kurata (kurata.masayuki318@mail.kyutech.jp)
This work was partly supported by the Resilient Edge Cloud Designed Network (19304), The National Institute of Information and Communications Technology (NICT), Japan.

ABSTRACT One-to-many transfers in a fast and efficient manner are essential to meet the growing need for duplicating, migrating, or sharing bulk data among servers in a datacenter and across geographically distributed datacenters. Some existing works utilize multiple multicast trees for a one-to-many transfer request to increase network link utilization and its transfer throughput. However, since those schemes do not fully utilize the max-flow value of transmission from a single sender to each recipient, there is room for each recipient to retrieve data more quickly. Therefore, assuming fully-controlled networks with full-duplex links, we pose a problem to find a set of multicast flows with an allocation of block-wise transmissions by which each of multiple recipients with diverse max-flow values from the sender can utilize its own max-flow value. Based on that, assuming a sender-side coding capability on file blocks, we design a schedule of block transmissions over multiple phases by which each recipient can achieve a lower-bound of its file retrieval completion time, i.e., the file size divided by its own max-flow value. This paper presents the coded Multipath Multicast (Coded-MPMC) for one-to-many transfers with heuristic procedures to find a desired set of multicast flows on which block transmissions are scheduled. Through extensive simulations on large-scale real-world network topologies and different types of randomly-generated synthetic topologies, the proposed method is shown to design a desired schedule efficiently. A preliminary implementation on OpenFlow is also reported to show the fundamental feasibility of Coded-MPMC.

INDEX TERMS One-to-many transfer, multicast transfer, multipath transfer, max-flow value, sender coding, OpenFlow.

I. INTRODUCTION
With the popularization of cloud, distributed computing, and contents delivery networking technologies, there is a growing need for duplicating, migrating, or sharing bulk data among distributed servers or sites. The edge-cloud computing for emerging IoT technologies will further accelerate the need for delivery of a large file to a larger number of servers distributed over geographically-wider locations. Therefore, fast and efficient one-to-many transfers are demanded, in which a single sender delivers a file to multiple recipients over network paths not only in a datacenter but across geographically distributed datacenters [1].

For one-to-many transfers, transferring a file by “reliable” multicast is essentially adopted to prevent from wasting link capacities on the links shared by many paths to recipients and also from overloading the sender. However, in the case of using a single multicast tree (e.g., [2]), the completion time for each recipient to retrieve a file is prolonged by a recipient at the worst location. To address this limitation, the use of multiple multicast trees with appropriately grouped recipients was considered [3]. For one-to-many transfers in a datacenter, a P2P-based flexible data dissemination scheme was proposed focusing on typical datacenter network topologies such as the FatTree and Multi-Routed Tree [4].

The strong need for fast transfers of bulk data among globally-distributed datacenters has led to the deployment of dedicated high-speed backbone networks inter datacenters (e.g., [5]–[7]), which are centrally managed based on Software-defined network (SDN) technology including OpenFlow. Recent studies often utilize multiple multicast trees realized by a flexible and adaptive path routing based on SDN technology [8]–[10].

On the other hand, data transfers from a single sender to a single recipient leveraging multiple network paths have emerged and attracted attention for diverse applications [11].

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
For example, a multipath transmission scheme was proposed using a fountain code to encode transmission data and achieve higher reliability without retransmission [12]. The effective use of Multipath TCP (MPTCP) inter datacenters has also been studied [13]. File transfers on multiple network paths inter datacenters with fair resource allocation have been studied based on “multi-commodity max-flow problem” [14].

Block/packet-level coding techniques have been extensively studied in one-to-one, one-to-many, and many-to-many information dissemination to improve throughput, delay, and/or resilience in both multicast (e.g., [15], [16]) and multipath (e.g., [12]) settings. Coding not only at a sender node but also at network-internal relay nodes is often referred to as Network Coding, which can be used to maximize the information dissemination rate at the sender as remarked in the seminal work [17]. However, in practice, it is difficult to implement a network coding system on each network-internal relay node (i.e., switch) as well as to find an optimal coding scheme on a given network topology.

Motivated by these observations, we proposed the Multipath Multicast (MPMC) to transfer a large file from a single sender to multiple recipients on a bandwidth-guaranteed and fully-controlled OpenFlow network with full-duplex wired links [18]. A typical scenario to which MPMC will be applied is as follows. One-to-many transfers are performed one by one in a scheduled manner on a centrally-managed network; when one one-to-many transfer is in progress, no other one-to-many transfer is performed on the network unless some residual network resources are available. In such a scenario, a one-to-many file transfer task from a sender should be performed as quickly as possible, followed by a next transfer task from a different sender. Our essential interest is how to use multiple multicast flows from the sender to all or a part of recipients in transferring a single file, to maximally exploit the transmission capacities reserved for a specific one-to-many transfer in full-duplex links, i.e., to realize the max-flow value from the sender to each recipient. To do so, different parts of the file should be concurrently transmitted to the same recipient on multiple paths to fully utilize the link capacities; and the same data should be concurrently transmitted to different recipients on a multicast tree to efficiently utilize the link capacities.

We define the retrieval completion time (RCT) of a recipient as the duration from the time when the sender starts the file transfer until the recipient completes the file retrieval. Given a sender and a set of recipients in one-to-many transfer, RCTs generally differ among recipients because the location of recipients in relation to the sender’s location is heterogeneous in terms of network topologies and link capacities. Note that the flow completion time (FCT) [19] is often adopted as a performance metric in research on TCP flows and a similar concept to RCT. However, we adopt the RCT in this paper as a performance metric to emphasize the time of each recipient’s file retrieval completion; this is because our scheme deals with multiple multicast transmissions over multiple phases to deliver a file to multiple recipients and thus, the RCT is not directly bound with a single flow.

Figure 1 illustrates the advantage of MPMC transfer compared to multicast transfer. A file of size \( L \) is transferred from a single sender \( s \) to two recipients \( r_1 \) and \( r_2 \) over a network with relay nodes (i.e., OpenFlow switches) and bidirectional (i.e., full-duplex) links. In the network, the link capacity between relay nodes is one in each direction, and that between a relay node and a sender/recipient is sufficiently large.

In the multicast transfer as shown in Fig. 1(a), a single tree from \( s \) to \( r_1 \) and \( r_2 \) is constructed to transfer efficiently, and each recipient retrieves in RCT \( L \) using a single multicast flow whose flow value is one. On the other hand, MPMC transfer minimizes the RCT of each recipient by the multipath transfer for increasing the link capacity available and by the multicast transfer for decreasing the link capacity consumed by multiple recipients as shown in Fig. 1(b). In MPMC transfer, the file is divided into two original blocks “a” and “b” each of which is transmitted on different multicast trees, and each recipient retrieves in RCT \( L/2 \) at the same time, which is twice as fast as that in the multicast transfer and is equal to the theoretical RCT lower-bound. Note that the full-duplex transmission is essential, as shown in the link between nodes 1 and 2 in Fig. 1(b).

Our goal is to design a schedule of block transmissions on an appropriate set of multicast flows by which each of multiple recipients completes the file retrieval in its own lower-bound time, i.e., the file size divided by its own max-flow value from the sender. However, as described in Sec. III-A, such a schedule does not always exist in case that only original blocks are transmitted in MPMC (referred to as Basic-MPMC). Therefore, we have proposed an extension of Basic-MPMC with a sender-side coding scheme, referred to as Coded-MPMC [20]. In Coded-MPMC, a necessary number of coded blocks are generated from the original blocks.
of a file by using a maximum distance separable (MDS) coding by the sender. Both the original and coded blocks can be transmitted over one or multiple phases. Note that, since assuming the standard OpenFlow networks to apply our scheme, we could not consider the network coding but focus only on the sender coding. For the sender coding, throughout this paper, we adopt a systematic Reed-Solomon (RS) coding as an example of MDS coding to explain Coded-MPMC.

In Sec. V and VI, we present the Coded-MPMC scheduling. In Sec. V, we show the theoretical RCT lower-bound of every recipient. To the best of our knowledge, for a single one-to-many file transfer to multiple recipients on an arbitrary network topology with full-duplex wired links, no other work has fully exploited the max-flow value of each recipient and achieved the theoretical RCT lower-bound of every recipient.

The rest of the paper is organized as follows. In Sec. II, we introduce related work on transmission scheduling using single or multiple multicast trees in SDN. In Sec. III, we show the necessity of Coded-MPMC through a critical example of Basic-MPMC. In Sec. IV, we formulate the problem and present the Coded-MPMC scheduling. In Sec. V and VI, we propose a method with heuristic procedures to design an appropriate schedule in Coded-MPMC and check if the schedule is optimal through extensive simulations on a variety of topologies. In Sec. VII, an implementation of Coded-MPMC is introduced. Finally, we provide discussion on Coded-MPMC in Sec. VIII, and conclude in Sec. IX.

II. RELATED WORK

Avalanche [23] utilizes a minimal-sized multicast tree for a multicast request in typical datacenter topologies, which enables one-to-many transfers to all recipients with fewer resources. DCCAST [24] constructs a minimal-weighted multicast tree over a network graph, where each link is assigned a weight according to the total load of the already scheduled requests and a given new request. Such an assignment of weights enables to construct a minimal-sized multicast tree to avoid highly loaded links and allows multiple multicast requests to coexist on the network at the same time. However, since these approaches use the only single multicast tree for a multicast request, the transfer throughput is limited by a bottleneck link with the lowest capacity in the multicast tree or with congestion caused by competing flows.

Using multiple multicast trees for a multicast request increases network link utilization to improve the transfer throughput. In [25], each transfer request with a demanded deadline is completed by maximizing the transfer throughput of all requests using multiple multicast trees. For each request, transmission rates and the number of utilized trees are calculated based on the proposed linear programming formulation. In QuickCast [26], the recipients of a single one-to-many file transfer are partitioned into several groups using multiple multicast trees according to transmission rate allocation, which can efficiently and quickly accommodate coexisting multiple file transfers over the network with a series of file transfer requests dynamically arriving.

In recent works, multicast transfer schemes are also studied over reconfigurable datacenter networks with circuit switches that can reconfigure links and fifth-generation (5G) networks. DaRTree [27] dynamically changes specified links over a reconfigurable network, and allocates various rates to each multicast request per discrete time unit, i.e., timeslot, to complete the data transfer by the deadline. Timeslots are fully and efficiently utilized, and transfer throughput to each recipient is maximized. The aim of [28] is to reduce provisioning costs of Network Functions (NFs) with the capability of multicast replication, by minimizing the number of deployed NFs. When there are multicast requests that cannot meet the required transfer throughput between NFs in the designed single multicast tree, the scheduler allows multipath transfers between those NFs, which reduces the installation of extra NFs, and meets the demand.

Contrary to these multicast-based schemes and settings, which focus on scheduling multiple one-to-many transfer requests [24]–[28], MPMC deals with a single one-to-many transfer request and aims to design a block-wise transmission schedule using appropriate multiple multicast trees so that each recipient can retrieve the file using its own max-flow value on a static full-duplex network. Based on Basic-MPMC [18], we developed some variants in two directions; coding-based one [29] and gossiping-based one [30]. In [29], coded blocks are generated by combining two or more original blocks using exclusive OR (XOR) coding based on a reactive and opportunistic algorithm and are transmitted to recipients to recover (i.e., decode) their unreceived blocks. In [30], one or more blocks are transmitted from not only the sender but the recipients that have already retrieved all blocks to a recipient that has not yet received all blocks, to make use of unused residual links efficiently. Following them, a preliminary version of Coded-MPMC [20] was developed.
that can outperform the previous two variants, but it was also recognized that the preliminary Coded-MPMC is still unable to design an optimal schedule in some real topologies; therefore we focus on Coded-MPMC.

III. LIMITATION OF BASIC-MPMC

A. CRITICAL EXAMPLE

In both Basic-MPMC and Coded-MPMC, equally-sized blocks are transmitted to multiple recipients using multiple multicast trees over one or more phases. In Fig. 2, there are sender 1, recipient 2 with the max-flow value of three, and three recipients 1, 2, and 4 with the max-flow value of two, respectively. A file to transfer is divided into six blocks \{a, b, c, d, e, f\} because of LCM(2, 3) = 6. In any desired setting, in the 1st phase, 3 should complete the file retrieval by receiving all six blocks, while 1, 2, and 4 should complete four blocks using two multicast flows. In the 2nd phase, 1, 2, and 4 should receive two unreceived blocks to complete the file retrieval.

Figure 2(a) shows the set of multicast flows from 1 to some/all recipients and the set of received blocks at the end of the 1st phase. Since 1, 2, and 4 have not completed the file retrievals, Basic-MPMC proceeds to the 2nd phase. Figure 2(b) shows the set of multicast flows from 1 to some/all uncompleted recipients and the set of received blocks at the end of the 2nd phase. The number of blocks transmitted on each multicast flow is one; 1 and 2 receive blocks \{e, f\} using two multicast flows, and complete file retrievals, but 4 cannot receive block \{d\} using the residual link (3 → 4).

This is because the blocks received by those recipients are different in the 1st phase; 1 and 2 have \{a, b, c, d\}, whereas 4 has \{a, b, e, f\}. Note that, if there is a desired set of multicast flows by which 1, 2, and 4 receive the same blocks in the 1st phase, those recipients can complete at the same time in the 2nd phase. However, it is impossible to construct such multicast flows for the 1st phase. This is because those three multicast flows should deliver mutually different blocks so that recipient 3 receives all different blocks. Therefore, different pairs of two multicast flows should deliver different sets of four blocks, resulting in that at least one of three recipients (1, 2, and 4) should receive a set of blocks different from the other two recipients.

Figure 2(c) shows the time chart of Basic-MPMC transfer in each link. By letting \(L\) be the file size, 3 completes file retrieval in its lower-bound RCT \(L/3\) and 1 and 2 complete in their lower-bound RCT \(L/2\). However, 4 cannot receive block \{d\} in the 2nd phase, then completes in RCT \(2L/3\) that is not its lower-bound.

B. CODED-MPMC SOLUTION TO CRITICAL EXAMPLE

In Coded-MPMC, by generating a set of coded blocks from the original blocks using RS coding, both original and coded blocks can be transmitted. Each recipient can retrieve the file when it receives a necessary number of different blocks that are either original or coded.

Figure 3 shows an example of Coded-MPMC transfer in the same network as Fig. 2. The 1st phase is the same as that for Basic-MPMC in Fig. 2(a). Figure 3(a) shows the set of multicast flows from 1 to some/all recipients and the set of received blocks at the end of the 2nd phase in Coded-MPMC. In the 2nd phase, transmitting one coded block \{g\} enables all uncompleted recipients in the 1st phase to retrieve different six blocks using three multicast flows. As shown in Fig. 3(b), in Coded-MPMC, 4 also completes file retrieval in its lower-bound RCT \(L/2\); this is an optimal schedule.

IV. CODED-MPMC

A. NETWORK MODEL

We target one-to-many transfers on a fully-controlled network consisting of network nodes (i.e., routers or switches) with bandwidth-guaranteed full-duplex wired links connecting them. There are a sender host and multiple recipient hosts...
which are connected to the network. Each of those hosts is assumed to be attached to each different node through a link with a sufficiently large transmission capacity. In other words, any bottleneck of file transfers is not in those links attaching the sender/recipient hosts.

Our targeted communication network with full-duplex links is modeled as a connected and symmetric directed graph (digraph) $D(V, A)$; $V$ represents a set of nodes; and $A \subseteq V \times V$ represents physical directed connections of adjacent nodes. Let $c(u, v)$ be a positive integer value representing the capacity of link $(u, v) \in A$; that is, the capacity of each link is a multiple of the unit capacity 1. Digraph $D(V, A, C)$ is defined in association with static link capacities (mapping $C: A \rightarrow N^+$) where links $(u, v)$ and $(v, u)$ have the same link capacity ($c(u, v) = c(v, u)$). All links are assumed to be static, error-free, and mutually independent to simplify the file transfer process proposed in this paper.

Let $N = |V|$ be the number of nodes. All nodes are numbered (as index) from 0 to $(N - 1)$. Let $s$ be the index to which the sender host is attached; $R$ be the set of indexes of nodes to which the recipient hosts are attached. Let $K = |R|$ be the number of recipients; $K \leq N - 1$. In reality, a file is transferred by a sender host attached to node $s$, and is received by each recipient host attached to a node in $R$. However, since each host is attached to a corresponding node with a sufficiently large capacity, we can consider $s$ as a sender and $R$ as a recipient set hereinafter.

On given $D(V, A, C)$, a path is defined as a cascaded sequence $(u, w_1), (w_1, w_2), \ldots, (w_{k-1}, w_k), (w_k, v)$ of links from node $u$ to node $v$ without loop; its length is 1 if $w_1 = v$, and is $(k + 1)$ otherwise. Since only non-loop (non-cycle) paths are considered, all nodes are mutually different; that is, $u, w_j (j = 1, 2, \ldots, k)$, and $v$ are different nodes if the path length is $k + 1$. A unit flow $f_{uw}$ is a data flow from $u$ to $v$ along a path from $u$ to $v$ consuming the unit capacity at each link on the path. If a node is passed by a flow, it is to belong to the flow. Note that hosts attached to any node belonging to the flow (except for $u$) can receive data from $u$ on the flow. A set of unit flows $f_{u1}, f_{u2}, \ldots, f_{uk}$ is called “compatible” if the number of the unit flows traversing link $(p, q)$ does not exceed its capacity $c(p, q)$ for any link $(p, q)$ traversed by at least one of those flows. Compatible flows can transfer data simultaneously and independently. In the special case that all links have the unit capacity (i.e., $c(p, q) = 1$), it is equivalent to an arc-disjoint set of paths. Furthermore, for a data transfer from $u$ to $v$, multiple paths can be utilized simultaneously. The maximum number of compatible unit flows (along multiple paths in general) from $u$ to $v$ is called the max-flow value $m(u, v)$, which is the maximally possible aggregate capacity of a concurrent transfer on a set of unit flows. Finding the max-flow value $m(u, v)$ and the max-flow set (a set of $(u, v)$ unit flows from $u$ to $v$) is known as the max-flow problem. Note that, a given source-sink pair $(u, v)$, the max-flow set is not unique in general.

Multicast tree $T_{uw}(V', A')$ rooted at $u$ with leaf node set $W$ is defined as a subgraph of $D(V, A)$ (i.e., $V' \subseteq V, A' \subseteq A$) that satisfies, for each node $v \in V'$, there is a unique path on $T_{uw}(V', A')$ from $u$ to $v$; $W$ is the set of leaf nodes that have no downstream node, i.e., $W = \{w \in V' | \forall v \in V', ((w, v) \notin A')\}$. Unit multicast flow $F_{uw}(V', A')$ is a multicasting data flow from $u$ to $W$ along a multicast tree $T_{uw}(V', A')$ consuming the unit capacity at each link on the multicast tree. Note that any host attached to a node (either $\in W$ or $\in V' \setminus W$) on tree $T_{uw}$ can receive data on the flow. A set of unit multicast flows $F_{uw1}, F_{uw2}, \ldots, F_{uwk}, W_k$ is called “compatible” if the number of the unit flows flowing on link $(p, q)$ does not exceed its capacity $c(p, q)$ for any link $(p, q)$ flowed by at least one of those flows. Compatible multicast flows can transfer data simultaneously and independently. In the special case that all links have the unit capacity (i.e., $c(p, q) = 1$), it is equivalent to an arc-disjoint set of directed trees.

**B. MODEL OF BASIC-/CODED-MPMC**

On given connected and symmetric digraph $D(V, A, C)$, sender $s \in V$, and recipient set $R \subseteq V$, the MPMC $(D(V, A, C), s, R)$ problem is to design a schedule of one-to-many transfer on an appropriate set of unit multicast flows rooted at $s$ for distributing the file to each of recipients $r \in R$ as quickly as possible.

The retrieval completion time (RCT or RCT[$r$]) of recipient $r$ is defined as the duration from the time when sender $s$ starts sending the file to the time when recipient $r$ has received the entire contents of the file. Note that the propagation delay at the link and the processing delay at the node are ignored.
The RCT inherently varies among recipients due to their heterogeneity of the location in the network. The lower-bound of \( \text{RCT}[r] \) is equal to \( \frac{L}{m(s, r)} \) unit time, that is the file size \( L \) divided by the max-flow value \( m(s, r) \) (by defining the unit capacity as the unit data size per the unit time).

A schedule is called “optimal” if and only if the RCT\([r]\) is equal to its lower-bound time for every recipient \( r \). Let us classify all recipients by their own max-flow values. Assuming that those max-flow values \( m(s, r) \) range into \( h \) different values as \( m_1 > m_2 > \ldots > m_h \), the recipient set \( M(j) \) is defined as the set of all recipients whose max-flow value from the targeted sender is \( m_j \). We call it “the \( j^{th} \) group” which depends on the sender location. Therefore, in any optimal schedule, for \( j = 1, 2, \ldots, h \), the RCTs of all recipients in \( M(j) \) are identical, and equal to \( \frac{L}{m_j} \) and it is shorter than that of recipients in \( M(k) \) if \( j < k \).

To transfer a file on multiple unit flows, a file should be appropriately divided into equally-sized blocks. They are called “original” block to be distinguished from “coded” block appearing in Coded-MPMC. The file division number \( d \) should be the least common multiple of all the max-flow values \( m_j (j = 1, 2, \ldots, h) \) in any optimal schedule as stated below. It is worth noting that, if an optimal schedule in the MPMC \((D(V, A, C), s, R)\) problem is given, we can derive an optimal schedule in the MPMC \((D(V, A, C), s, R)\) problem for any \( R' \subset R \) by just removing unnecessary transmissions from that original schedule. In other words, our goal can be an optimal schedule in the MPMC \((D(V, A, C), s, V \setminus \{s\})\) problem. Actually, in Sec. VI, we experimentally show that our proposed scheme can design an optimal schedule in the MPMC \((D(V, A, C), s, V \setminus \{s\})\) problem for many different types of \( D(V, A, C) \) with any place of sender \( s \in V \).

To design an optimal schedule, the 1\(st\) phase is defined as the period from the start time of transfer to the completion time of retrieval by all the recipients in the 1\(st\) group. For \( j = 2, 3, \ldots, h \), the \( j^{th} \) phase is the period from the completion time of retrieval by recipients in \( M(j-1) \) to the completion time of retrieval by recipients in \( M(j) \). Thus, in any optimal schedule, the duration of the 1\(st\) phase is \( \frac{L}{m_1} \) and that of the \( j^{th} \) phase is \( \left( \frac{L}{m_j} - \frac{L}{m_{j-1}} \right) \). The file transfer proceeds in such a phase-by-phase manner until all recipients have retrieved the entire file if there are multiple recipient groups.

Let \( S_r(j) \) be the set of all blocks that are scheduled to be received by recipient \( r \) before the \( j^{th} \) phase. At the beginning of the 1\(st\) phase in any schedule, all recipients have no block, i.e., \( \forall r \in R(S_r(1) = \phi) \). On the other hand, at the beginning of the \( j^{th} \) phase in any “optimal” schedule, the number of blocks to be received in the previous phases by recipient \( r \) should be \( d \cdot \frac{m_i}{m_{j-1}} \) for any recipient \( r \in M(i) \) (\( i \geq j \)). Let \( R(j) \) be the set of all uncompleted recipients at the beginning of the \( j^{th} \) phase, i.e., recipients who need to receive additional blocks in the \( j^{th} \) phase; \( R(j) = \bigcup_{i=j}^{h} M(i) \).

For the \( j^{th} \) phase (\( j = 1, 2, \ldots, h \)),
- we determine a set \( \{F_s(V_k, A_k)|k = 1, 2, \ldots, t\} \) of unit multicast flows that are compatible and rooted at \( s \) with some leaf node set consisting of all or a part of uncompleted recipients (i.e., \( V_k \subset R(j) \)), and
- for each \( k = 1, 2, \ldots, t \), we determine a set \( \{B(k)^{i}\} \) of blocks to be transmitted on the \( k^{th} \) unit multicast flow \( F_s(V_k, A_k) \) to uncompleted recipients in (not only \( V_k \) but) \( V_k \).

In Basic-MPMC, \( B(k)^{i} \) consists of original blocks only; while in Coded-MPMC, \( B(k)^{i} \) can include coded blocks.

We call it block allocation \( \sigma = \{F_s(V_k, A_k), B(k)^{i}\} \) \( k = 1, 2, \ldots, t \). Block allocation \( \sigma \) for a phase is called “bandwidth-efficient (BE) for recipient \( r \)” if \( r \) receives its maximally possible number of blocks on the max-flow value \( m(s, r) \) of unit flows in that phase, that is, \( r \) can fully utilize a max-flow set from \( s \) to \( r \). Block allocation \( \sigma \) for a phase is called “globally bandwidth-efficient (GEB) if it is BE for every uncompleted recipient in that phase. A GEB block allocation should satisfy the following condition(s).

\[(C_1)\] In the 1\(st\) phase, any recipient \( r \) in the \( i^{th} \) group \((i = 1, 2, \ldots, h) \) should receive different blocks using the max-flow value \( m_i \) of unit multicast flows that are all or a part of \( \{F_s(V_k)|k = 1, 2, \ldots, t\} \). In other words, for any recipient in the \( i^{th} \) group, the number of unit multicast flows passing \( r \) should be \( m_i \), i.e., \(|\{k|r \in V_k\}| = m_i \) if \( r \in M(i) \).

Note that we can choose \( t = m_1 \) by a simple allocation in which \( \frac{d}{m_1} \) different blocks are allocated to each of \( m_1 \) unit multicast flows, i.e., \(|B(k)^{i}| = \frac{d}{m_1}, k = 1, 2, \ldots, m_1 \) such that \( B(k)^{i} \cap B(j)^{f} = \phi \) if \( k \neq f \).

For any subsequent phase, the block allocation should consider the set \( S_r(j) \) of blocks that are scheduled to be received by \( r \) in previous phases.

\[(C_2)\] In the \( j^{th} \) phase (\( j > 1 \)), any recipient \( r \) in the \( i^{th} \) group \((i = j, j + 1, \ldots, h) \) should receive different “unreceived” blocks that are not in \( S_r(j) \), using the max-flow value \( m_i \) of unit multicast flows that are all or a part of \( \{F_s(V_k)|k = 1, 2, \ldots, t\} \). That is, for any uncompleted recipient \( r \),
- if \( r \) is a recipient in the \( j^{th} \) group, the number of unit multicast flows passing \( r \) should be \( m_i \), i.e., \(|\{k|r \in V_k\}| = m_i \) if \( r \in M(i) \).

Note that the number of blocks to be transmitted on each unit multicast flow should be \( \left( \frac{d}{m_j} - \frac{d}{m_{j-1}} \right) \).

The number \( t \) of unit multicast flows in the \( j^{th} \) phase should be equal to or greater than \( m_j \).
C. CODED-MPMC SCHEDULING

A schedule for the Coded-MPMC(D(V, A, C), s, R) problem is designed as follows.

1) By solving the max-flow problem, the max-flow values for all recipients ((m(s, r)|r ∈ R) are computed. Then, all the recipients are classified into h different groups in terms of its max-flow value. M(j) is the recipient set in the jth group (j = 1, 2, . . . , h) with the max-flow value of m_j. The file division number d is LCM(m1, m2, . . . , m_h).

2) The file is supposed to be divided into d original blocks. Then, a necessary number e of coded blocks are to be generated from the original blocks using a (d + e, d)-systematic RS coding. Set C* of coded blocks to use is derived in the total schedule (s1, s2, . . . , s_h) as shown in Alg. 1 in Sec. IV and the number e is |C*| after the design of s_h is completed.

3) A GBE block allocation s1 for the 1st phase is found as shown in Alg. 2 in Sec. V to distribute the original blocks to all the recipients R so that all the recipients in the 1st group M(1) complete the file retrieval.

4) A GBE block allocation sj for the jth phase (j > 1) is derived from the block allocations for past phases (s1, s2, . . . , s_j−1) as shown in Alg. 1 to distribute the coded and/or original blocks to all the uncompleted recipients R(j) so that all the recipients in the jth group M(j) complete the file retrieval.

Note that, if block allocation σ is GBE for the jth phase (i.e., BE for ∀r ∈ R(j)), we can simply derive σ’ that is GBE for the (j + 1)th phase (i.e., BE for ∀r ∈ R(j + 1)). All or a part of the multiple unit multicast flows F_s1(V_k), F_s2(V_k), . . . , F_s_k(V_k) that cover the recipient set R(j) in σ can be used in σ’ to cover the recipient set R(j + 1) ⊂ R(j), and a set of coded blocks that is not used in the previous phases can be allocated to the unit multicast flows.

A trivial approach to guarantee for every recipient to receive blocks that were not received before is to use completely new (coded) blocks in each phase. However, there is a chance to reuse some blocks that were received by a set of recipients before but not received by another set of uncompleted recipients. Algorithm 1 is used to select blocks in determining which unit multicast flow transmits which block set. This process aims to reduce the number of coded blocks to generate by reusing an already delivered block on
Algorithm 2 Block Allocation Algorithm in the 1st Phase to Find a GBE-BA

Input: \( D(V, A, C), s, R \), Recipient order parameter

Output: \( \sigma_1 = \left\{(C^{(k)}_{s,V_k}(V_k,A_k), B^{(k)})\middle| k = 1, 2, \ldots, m_1 \right\} \)

1: Compute \( m(s, r) \) for \( r \in R \), and get \( h \) different max-flow values \( m_1 > m_2 > \ldots > m_h \);
2: \( d \leftarrow \text{LCM}(m_1, m_2, \ldots, m_h) \); \( d \) original blocks from a file
3: Define hash table \( BaOrder \) to store the recipient order of block allocation;
4: \( \text{for } i = 1, 2, \ldots, h \) do
5: \( M(i) \leftarrow \text{Recipients with max-flow value of } m_i \);
6: \( BaOrder[i] \leftarrow M(i) \) sorted by \( \text{path-length ascending order allocation} \) and recipient order parameter; \( \triangleright \) Sec. V-B
7: \( \text{end for} \)
8: \( \text{for } k = 1, 2, \ldots, m_1 \) do
9: \( W_k \leftarrow \phi \); \( V_k \leftarrow \phi \); \( A_k \leftarrow \phi \);
10: \( B^{(k)} = \frac{d}{m_1} \) blocks, where \( B^{(k)} \cap \bigcup_{i=1}^{k-1} B^{(i)} = \phi \);
11: \( \text{end for} \)
12: \( \sigma_1 = \left\{(C^{(k)}_{s,V_k}(V_k,A_k), B^{(k)})\middle| k = 1, 2, \ldots, m_1 \right\} \);
13: \( \text{for } i = 1, 2, \ldots, h \) do
14: \( \text{for all } r \text{ in } BaOrder[i] \) do
15: \( \text{old}_r \sigma_1 \leftarrow \sigma_1 \);
16: Select max-flow set \( P = \{f_1, f_2, \ldots, f_m\} \) from \( s \) to \( r \) using super-source \( s^* \); \( \triangleright \) Sec. V-D
17: Get set \( Z \) of path-blocks bipartite matching; \( \triangleright \) Sec. V-C
18: Modify each flow in \( P \) based on \( Z \); \( \triangleright \) Sec. V-C
19: \( \text{for all } (f_x, F^{(0)}) \text{ in } Z \) do \( \triangleright \) Block allocation for \( r \) do
20: \( \forall y \in \mathcal{V}_y \cup \bigcup_{(u,v) \notin f_x} \{u, v\}; \)
21: \( \forall y \leftarrow A_y \cup \bigcup_{(u,v) \in f_x} \{u, v\}; \)
22: Update \( W_y \);
23: \( \text{end for} \)
24: if \( |Z| < m_1 \) then \( \triangleright \) BA for \( r \) is non-BE
25: if Repeat count of deprioritized reallocation for \( r \) does not exceed threshold value then \( \triangleright \) Sec. V-B
26: \( \sigma_1 \leftarrow \text{old}_r \sigma_1 \);
27: \( r \) is pushed to the tail of \( BaOrder[i] \); \( \triangleright \) Sec. V-D
28: \( \text{continue} \);
29: end if \( \triangleright \) Deprioritized reallocation
30: Max-Flow Set Reselection; \( \triangleright \) Sec. V-D
31: \( \sigma_1 \leftarrow \text{old}_r \sigma_1 \);
32: Retry the block allocation for \( r \); \( \triangleright \) Line 17-22
33: if \( |Z| < m_1 \) then \( \triangleright \) Stop finding a GBE-BA
34: exit; \( \triangleright \)
35: end if
36: \( \text{end if} \)
37: \( \text{end for} \)
38: end for

a unit multicast flow if all uncompleted recipients in that multicast flow have not received the block.

V. GLOBALLY BANDWIDTH-EFFICIENT BLOCK ALLOCATION (GBE-BA)

A. PROCEDURE TO FIND GBE-BA

To design an optimal schedule, we should find GBE block allocation \( \sigma = \left\{(C^{(k)}_{s,V_k}(V_k,A_k), B^{(k)})\middle| k = 1, 2, \ldots, m_1 \right\} \) for the 1st phase that satisfies the condition \( C_1 \) defined in the previous section. However, since finding such \( \sigma \) is not trivial with a combinatorial explosion as increasing the size of the network, we propose a heuristic approach (Alg. 2) to construct appropriate \( \sigma \) that is GBE.

The basic idea is to determine a block allocation (BA) first to unit flows from sender \( s \) to a recipient who has the largest max-flow value \( m_1 \), i.e., a recipient who should complete the file retrieval first. Different blocks from all \( d \) blocks are evenly allocated to each of \( m_1 \) unit flows in a max-flow set for this recipient. This BA is at least BE.

For subsequent recipients, a per-recipient construction of a max-flow set with block allocation is repeated in descending order of the recipient’s max-flow value until the block allocation is performed for all recipients in a one-by-one manner. In this process, the max-flow set for the target recipient should be constructed to be compatible with the previous recipients’ max-flow sets. To do so, each unit flow for the target recipient is constructed by extending or branching some existing unit flow in an already constructed max-flow set for some previous recipient, which eventually results in incrementally constructing \( m_1 \) unit multicast flows rooted at sender \( s \). Note that this extension is not unique for some branch nodes. Since all recipients belonging to the same unit multicast flow receive the same set of blocks, this extension directly determines the blocks allocated to the target recipient. Therefore, to make a BE-BA for the target recipient, each flow in the max-flow set should correspond to (i.e., be extended from) each different unit multicast flow by solving “bipartite matching problem”, to allocate mutually different sets of blocks to all unit flows in the max-flow set as described in Sec. V-C.

Since the above approach strongly depends on the order of recipients for block allocation, we need to appropriately control the order. In particular, the order among multiple (possibly a large number of) recipients in the \( j \)th group \( M(j) \) is of essential importance to find a GBE-BA. Therefore, we introduced dynamic changes of the recipient order by allowing retrials in selecting a target recipient as described in Sec. V-B.

Furthermore, since we found that such heuristic controls of the order of recipients alone are insufficient to find a GBE-BA in more than half of examples of the realistic network topologies in Table 1 in Sec. VI-A, we also introduce dynamic changes of the max-flow set by allowing retrials in selecting unit flows in the max-flow set for a target recipient.
TABLE 1. Realistic network topologies in [21].

| Number of nodes | Name                                      |
|-----------------|-------------------------------------------|
| ≥ 40 AND < 50   | Bell Canada, China Telecom, LambdaNet,    |
|                 | PalmettoNet, RoEduNet, GEANT, Renater,   |
|                 | UnUNET >> Total 8 topologies              |
| ≥ 50 AND < 60   | IRIS Networks, Bell South, BTN, Timet,    |
|                 | DFN, Hibernia Atlantic, CESNET, CUDI,    |
|                 | SURFNET >> Total 9 topologies             |
| ≥ 60 AND < 70   | FORTINET, Internode, ASNET-AM, EISnet,   |
|                 | Missouri Network Alliance, GARR           |
|                 | >> Total 7 topologies                     |
| ≥ 70 AND < 80   | Columbus Networks, IntelLiber, Tecove (level3), Syringa Networks, Sinet, SWITCH, Uninet |
|                 | >> Total 7 topologies                     |
| ≥ 80 AND < 90   | Bestel, Velat, ULAKNET                   |
| ≥ 100           | Interoute, ITC Delacom, Ion, Tata,        |
|                 | GTS CI, Colt Telecom, US Carrier,         |
|                 | Kentucky DataLink, Cogent >> Total 9     |

TABLE 2. Random network topologies generated by [22].

| Name                        | Description                                                                 |
|-----------------------------|-----------------------------------------------------------------------------|
| Erdos-Renyi model [31]      | Topologies with 50 nodes are randomly generated with the 10% or 20% possibility of edge creation. >> Total 20 topologies |
| Waxman model [32]           | Topologies with 50, 75, or 100 nodes are randomly generated. >> Total 30 topologies |
| Barabasi-Albert model [33]  | Topologies with 50 nodes are generated by randomly attaching a new node with new links to 1, 2, 3, or 4 existing nodes. >> Total 40 topologies |
| AS-level network [34]       | Topologies with 50, 75, or 100 nodes are randomly generated. >> Total 30 topologies |

Algorithm 3 Optional Non-BE Checking for Block Allocation in the 1st Phase

1: ⊳ It is inserted between 36th and 37th lines in Alg. 2.
2: if CheckBlockAllocation() = Fail then
3:   if a past σ₁ is checked as Fail for r before then
4:     exit; ⊳ Stop finding a GBE-BA
5:   else
6:     σ₁ ← old σ₁;
7:     r is pushed to the tail of BOrder[i]; ⊳ In Alg. 2 next time, a new max-flow set P different from the failed P will be selected;
8:     end if
9:     end if

as described in Sec. V-D. Both the change of the order of recipients and the change of the max-flow set for a recipient are necessary to find a GBE-BA as shown in Table 3 in Sec. VI-B.

Lastly, note that, although it rarely happens, we also found that a BE-BA for a recipient with a larger max-flow value can prevent a posterior BA from being BE for another recipient with a smaller max-flow value. This cannot be detected by checking if a current BA is BE for every recipient with the same max-flow value. Therefore, in response to such a special case, we optionally check a necessary condition to finally realize a GBE-BA from the current BA configuration. This checking (Alg. 3) is introduced in the Appendix because it is not necessary in almost all cases.

B. CONTROL OF RECIPIENTS ORDER

In case that multiple recipients have the same max-flow value, a simple approach is to randomly change the order among such tied recipients and select the best order by trial and error. However, in general, it is challenging to examine all possible orderings of tied recipients to find a GBE-BA in complex large-scale network topologies with a large number of recipients.

To efficiently find a GBE-BA, we adopt a heuristic control of the recipient order of block allocation among recipients with the same max-flow value proposed in [20]. Concretely, we take priority to the recipients using a max-flow set that consists of flows along shorter-length paths. The idea comes from the fact that a BA for a recipient should have a small effect on BAs for the subsequent recipients. As explained in Sec. V-C, by allocating blocks to the max-flow set for a recipient, some links are newly allocated blocks. Allocating blocks to a link will reduce a possible chance to newly allocate different necessary blocks to that link or the residual capacity of that link for a subsequent recipient. Therefore, block allocation for a recipient who involves a small number of links is preferentially performed as follows.

Path-Length Ascending Order Allocation: For recipients with the same max-flow value, the block allocation is performed in ascending order of the total number of links traversed by unit flows in the max-flow set for each recipient. If there are multiple recipients whose max-flow set traverses the same number of links, the recipients are compared in terms of the maximum length of paths flowed along by unit flows in its max-flow set. Then, the block allocation is performed in ascending order of the maximum path length. If there remains a tie-break in some recipients, we use a pre-defined order “recipient order parameter,” which is randomly given in simulation.

However, since the use of this heuristic control alone does not always provide a good result, we allow a retrial for a recipient by dynamically postponing the block allocation for the recipient as follows.

Deprioritized Reallocation: When the block allocation for recipient r is performed, but a resulting BA is non-BE for r, its block allocation is postponed after the block allocation for all other recipients with the same max-flow value will have been performed. Such postponed retrials for r can be repeated until either a resulting BA becomes BE for r or the number of retrials for r exceeds a pre-defined threshold value.

C. BLOCK ALLOCATION FOR EACH RECIPIENT

To each (target) unit flow in a max-flow set for recipient r, a set of blocks should be allocated so as to be compatible with the max-flow sets for the previous recipients. This is equivalent to selecting an appropriate unit multicast flow from the already-constructed ones so that the target flow is branched or extended from the selected multicast flow. Note
that if recipient \( r \) already belongs to an already-constructed unit multicast flow, a unit flow along this unit multicast flow should be included in the max-flow set for \( r \). Let \( F \) be the set of all unit multicast flows to each of which a set of blocks are already allocated; \( P \) be the max-flow set for \( r \) with the max-flow value of \( m_i \) to each of which a set of blocks will be allocated. The block allocation procedure should find \( m_i \) different unit multicast flows in \( F \) so that each of \( m_i \) flows in the max-flow set can be extended from (or already included in) them.

To find \( m_i \) appropriate unit multicast flows for recipient \( r \), at first, we check which flow \( f_i \) in the max-flow set \( P \) can be extended from which unit multicast flow in \( F \) to deliver the blocks to \( r \). This is done by tracing back \( f_i \) from \( r \) to \( s \) to list possible branching nodes. Based on the above checking, a bipartite graph \( B(\mathcal{P}, F, \mathcal{Q}) \) is created consisting of the max-flow set \( \mathcal{P} = \{f_1, f_2, \ldots, f_m\} \) and the set \( \mathcal{F} = \{F^{(1)}, F^{(2)}, \ldots, F^{(m)}\} \) of unit multicast flows as two sets of vertices, and set \( \mathcal{Q} \) of directed edges from \( \mathcal{P} \) to \( \mathcal{F} \). A directed edge from \( f_i \) to \( F^{(j)} \) represents that \( r \) can belong to unit multicast flow \( F^{(j)} \) via unit flow \( f_i \) to receive block set \( B^{(j)} \). There are two cases; either in case that \( F^{(j)} \) can be newly extended for \( r \) by using residual links traversed by \( f_i \) or in case that \( f_i \) has already been included in \( F^{(j)} \) (i.e., \( r \) already belongs to \( F^{(j)} \)) due to an extension of \( F^{(j)} \) for previous recipients.

In the former case, let \( w \) be a branching node (i.e., \( w \) is the intersection of unit multicast flow \( F^{(j)} \) and unit flow \( f_i \)). The links (or one capacity on the links, more exactly) in the segment from \( w \) to \( r \) on flow \( f_i \) are newly used to extend a multicast flow with an allocation of set \( B^{(j)} \) of blocks. Based on the same idea of the path-length ascending order allocation, we reduce the number of such newly allocated links to remain more number of links with unused capacity for the use by the subsequent recipients. To do so, the number of links in the segment from \( w \) to \( r \) on \( f_i \) is assigned to the edge in \( \mathcal{Q} \) as the cost, and a desired matching from \( \mathcal{P} \) to \( \mathcal{F} \) is determined by using a standard algorithm for “Minimum Cost Bipartite Matching.” Note that the above matching procedure may change the actual route of flow \( f_i \). For example, suppose the original \( f_i \) (before matching) from \( s \) to \( r \) flows along multicast flow \( F^{(l)} \) and branches at node \( v \) toward \( r \); it meets another multicast flow \( F^{(l)} \) at node \( w \) in-between from \( v \) to \( r \). If \( f_i \) is matched to \( F^{(l)} \) in the matching, the modified \( f_i \) (after matching) from \( s \) to \( r \) flows along \( F^{(l)} \) and branches at node \( w \) toward \( r \). As a special case, nodes \( v \) and \( w \) can be the same node if the node \( (v = w) \) is an intersection of multicast flows \( F^{(l)} \) and \( F^{(l)} \). In our simulation, this special case often happened.

Let \( Z \) be the resulting set of “flow”-“multicast” matching that minimizes the cost, from which a BA for \( r \) is directly derived. Note that the number of flow-multicast matching \( |Z| \) is equal to \( m_i \) if and only if the BA is BE for \( r \). In other words, if \( |Z| < m_i \), the BA is non-BE.

### D. SELECTION OF MAX-FLOW SET

At different steps in the block allocation for recipient \( r \), max-flow set \( \mathcal{P} \) from \( s \) to \( r \) should be given. At first, a max-flow set is used to decide the order of recipients in Path-length ascending order allocation. This initial max-flow set is found using a standard algorithm with a breadth-first search (BFS).

Then, in allocating blocks for \( r \), a max-flow set from \( s \) to \( r \) used here may be different from the initial one in general. The pseudo-senders are defined as nodes included in the intersection of all the unit multicast flows in \( F \), i.e., the nodes belonging to all unit multicast flows. From the original \( D(V, A, C) \), we define \( D(V^*, A^*, C^*) \) by virtually connecting super-source \( s^* \) to each of all pseudo-senders with a sufficient capacity. A max-flow set from super-source \( s^* \) to \( r \) is computed on \( D(V^*, A^*, C^*) \) also using a standard algorithm with BFS. From the resulting virtual max-flow set from \( s^* \) to \( r \), we can derive a max-flow set from \( s \) to \( r \). The derived max-flow set is expected to be better than the above initial max-flow set in terms of reduction of the number of links newly used in allocating blocks for \( r \), which will increase the number of potential links available in allocating blocks for other subsequent recipients. Furthermore, the max-flow set from \( s \) to \( r \) may also be modified in the matching process as described in Sec. V-C.

### TABLE 3. Detailed analyses on simulation examples in three network topologies with several sender locations.

| Topology          | UUNET   | Interroute | Barabasi–Albert model |
|-------------------|---------|------------|-----------------------|
| Sender ID         | 4       | 11         | 0                     |
| Number of recipients | 48      | 109        | 49                    |
| Ratio of average RCT in Coded-MPMC to that in ST | 0.511   | 0.451      | 0.370                 |
| Number of original blocks | 420     | 12        | 180                   |
| Number of coded blocks when Alg. I is not used | 669     | 17        | 249                   |
| Number of coded blocks when Alg. I is used | 225     | 12        | 173                   |
| Number of times in which Deprioritized reallocation is performed | 1       | 1         | 4                     |
| Number of times in which Reselection of max-flow set is performed | 1       | 1         | 2                     |

---

M. Kurata et al.: Coded-MPMC: One-to-Many Transfer Using Multipath Multicast With Sender Coding
However, if a resulting BA on a given max-flow set cannot be BE for r, we should select another different max-flow set for r either by changing one unit flow in the max-flow set or by newly generating a different max-flow set as follows. Suppose a non-BE BA for r with the max-flow value of \( m_i \) is derived from \( F \) (the set of unit multicast flows), \( P \) (the current max-flow set), and \( Z \) (the flow-multicast matching from \( P \) to \( F \)).

Max-Flow Set Reselection: In case that only a single flow \( f \) in all \( m_i \) flows in \( P \) is unmatched in \( Z \), i.e., unallocated by the block allocation, the flow \( f \) is changed to a new flow. Let \( P_Z = P \setminus \{ f \} \), the set of all \( (m_i - 1) \) flows matched in \( Z \); \( F_Z \) be the set of all unit multicast flows with only one directed edge from some vertex in \( P \) on bipartite graph \( B(P, F, Q) \), i.e., the set of multicast flows that are uniquely determined to be extended; and \( F_2 \) be \( F \setminus F_Z \), the set of remaining unit multicast flows not included in \( F_Z \). To make a BE-BA, a new flow replacing \( f \) should extend one of unit multicast flows in \( F_Z \). Therefore, we define \( D(V^*, A^*, C^*) \) from \( D(V, A, C) \) as follows. \( C^* \) is defined by reducing the link capacities used by each multicast flow in \( F \) and also reducing the link capacities newly used to extend each of \( (m_i - 1) \) unit flows in \( P_Z \) from each corresponding matched multicast flow in \( F \). To define \( (V^*, A^*) \), virtual super-source \( s^* \) is connected to each node belonging to at least one unit multicast flows in \( F_Z \) with sufficient capacity. Then, the shortest virtual path to \( r \) from \( s^* \) on \( D(V^*, A^*, C^*) \) is computed by a standard algorithm; if it exists, it is guaranteed to be compatible with all \( (m_i - 1) \) unit flows in \( P_Z \). Finally, a new unit flow from \( s \) to \( r \) should be determined from the obtained virtual path, which is combined with \( (m_i - 1) \) flows in \( P_Z \) in order to construct a new max-flow set \( P' \) on the original \( D(V, A, C) \). Note that this new unit flow is not always uniquely determined. Let \( w \) be the node next to \( s^* \) in the virtual path. If \( w \) belongs to multiple unit multicast flows, the new unit flow is not determined in the route from \( s \) to \( w \) because it can flow one of multiple possible unit multicast flows from \( s \). However, this is not a problem. After the max-flow set reselection, we always do the block allocation based on bipartite graph matching from \( P' \) to \( F \), which finally solves such non-unique-ness.

On the other hand, in case that more than one flows are unmatched in flow-multicast matching \( Z \), a new max-flow set different from \( P \) is reselected. Let \( P_Z \) be the set of all flows matched in \( Z \); and \( P_Z' \) be \( P \setminus P_Z \), the set of all unmatched flows. To make a BE-BA, a new max-flow set should exclude the previously unmatched flows in \( P_Z' \) as much as possible. Therefore, from \( D(V, A, C) \), we define \( D(V^*, A^*, C^*) \) by virtually connecting super-source \( s^* \) to each node belonging to all (i.e., the intersection of) unit multicast flows in \( F \) with a sufficient capacity. We also consider \( D(V^*, A^*, C^*) \) with “link cost” by assigning a sufficiently large cost to each link that is included in some flow in \( P_Z' \) but not included any flow in \( P_Z \). This process prevents the previously unmatched unit flows from being reselected. Then, a new virtual max-flow set to \( r \) from \( s^* \) on \( D(V^*, A^*, C) \) with link cost is computed by a standard algorithm for the minimum-cost max-flow problem. Finally, a new max-flow set from \( s \) to \( r \) consisting of \( m_i \) compatible unit flows on the original \( D(V, A, C) \) is determined depending on which multicast flows in \( F \) are selected to accommodate the virtual max-flow set.

VI. SIMULATION EVALUATION

A. SIMULATION SETUP

Through simulation evaluation, we verify that our proposed block allocation algorithm (Alg. 2) can realize a GBE-BA for the 1st phase for various network topologies listed in Tables 1 and 2, and investigate more details of processes in the subsequent phases to show the effectiveness of the Coded-PMPC. These tables show realistic network topologies consisting of forty or more nodes in “The Internet Topology Zoo” [21] and three types of random network topologies generated by “NetworkX Python package” [22], respectively.

Each instance of network topologies in Table 1 or 2 is given as input for the simulation. We regard each link in the topology as a full-duplex link with both uniform and non-uniform settings for the link capacity. In the uniform case, the link capacity of each link is 1. In the non-uniform case, the link capacity is 1, 2, 3, or 1, 4, 10, corresponding to the number of duplicate edges defined in the GML file in [21].

In addition to the topology information, locations of a single sender (s) and recipients should be given as inputs. In this evaluation, we consider only the case that all nodes except for the sender are recipients, i.e., \( R = V \setminus \{ s \} \); that is, in reality, every node is directly connected to either sender or a recipient host with sufficiently large link capacity. This is because an optimal schedule in the MPMC (\( D(V, A, C) \), s, \( V \setminus \{ s \} \)) problem can easily lead to an optimal schedule in the MPMC (\( D(V, A, C) \), s, \( R \)) problem for any \( R \subset V \setminus \{ s \} \), as noted in Sec. IV-B.

The recipient order parameter for tie-break in the recipients with the same max-flow value is randomly given and sorted by path-length ascending order allocation. The threshold value for the maximal repeat count of deprioritized reallocation is set to 1 in this simulation. The obtained results suggest a single chance for such postponement is enough.

B. SIMULATION RESULTS

The essential result is that our scheme successfully found a GBE-BA in the 1st phase and generated an optimal schedule for every instance of realistic network topologies in Table 1 and random network topologies in Table 2 generated by ten random-seed values, regardless of the position of the sender and the uniformness of link capacities as explained in Sec. VI-A. This fact suggests that the Coded-PMPC actually benefits a sufficiently wide range of heterogeneous scenarios.

We illustrate examples of our simulation results in a topology with many high-degree nodes “UUNET” (Fig. 4),
of original blocks also depends on the sender location in each topology. For example, when the sender ID is 9 in UUNET, the file to transfer is divided into 1260 blocks since there are recipients with the max-flow value of 10, 9, 7, 6, 5, 4, 3, 2, and 1. The number of coded blocks used to generate the optimal schedule in Coded-MPMC is computed using Alg. 1, which reuses already distributed blocks as much as possible. Table 3 also shows that the use of Alg. 1 reduces the necessary number of coded blocks compared to using only new coded blocks without reusing already distributed blocks in all phases. This reduction in the necessary number of coded blocks benefits the coding process in real implementation to reduce the computational cost. The number of blocks is also discussed later in Sec. VIII-A.

We show examples of a GBE-BA in UUNET and Interroute when the sender ID is 9 and 54, respectively. In Fig. 7, ten unit multicast flows are utilized where each flow distributes 126 original blocks in the 1st phase. In Fig. 8, six unit multicast flows are utilized where each flow distributes 2 original blocks in the 1st phase. Then, these multicast flows are used in all phases until uncompleted recipients retrieve the entire file.

Finally, we demonstrate the necessity of our heuristics in the block allocation algorithms. Table 3 also shows how many times the deprioritized reallocation and the max-flow set resection (described in Sec. V-B and Sec. V-D) were performed to find a GBE-BA in the 1st phase.

Note that we did not provide any specific performance comparison of Coded-MPMC to the existing schemes such as QuickCast [26] or DaRTree [27] described in Sec. II. This is because those schemes focus on allocating network resources for multiple requests or demands for different one-to-many file transfers simultaneously happening, while Coded-MPMC focuses on a single dedicated request for a single one-to-many file transfer. Thus, it is obvious that Coded-MPMC outperforms those schemes in terms of...
minimizing each recipient’s RCT if there is only a single one-to-many file transfer with our assumption.

VII. PROTOTYPE IMPLEMENTATION

The prototype of Coded-MPMC is implemented on a sender, recipients, OpenFlow controller (OFC) system with Ryu framework [35], and OpenFlow switch (OFS) with Open vSwitch switch OS [36]. MDS coding at the sender and recipients is implemented by using the library [37] of a systematic Reed-Solomon coding. ①-② in Fig. 9 show the procedural order for the Coded-MPMC on an OpenFlow network.

Before starting a file transfer, the OFC floods Link Layer Discovery Protocol (LLDP) packets to OFSs to get information on the network topology. Then, the OFC queries the port information from OFSs and gets the bandwidth speed at the link-up of the port (③). Each recipient uses Membership Reports of Internet Group Management Protocol (IGMP) to join as a recipient, by which the OFC can know the locations of the recipients (④).

Next, a sender sends a request with the file size to the OFC (⑤), which triggers the OFC generates a schedule of the Coded-MPMC transfer from the file size and the topology information with the sender and the recipients (⑥). Flow entries for OFSs are created based on the generated schedule including the UDP transmission port number, the phase number, and the block number which identify each of multiple multicast flows. The OFC installs those flow entries into OFSs. Besides, the OFC notifies the sender and the recipients of the necessary information on schedule (⑦). Based on the schedule information, the sender generates a necessary number of the coded blocks to be transmitted (⑧). If the file size is not a multiple of the division number of the file, a zero-padding is applied to create equally-sized blocks.

After starting block transmissions, when a recipient has received a necessary number of blocks, the recipient sends the retrieval completion notification to the sender, and the sender knows which recipients have completed. Then, the recipient decodes their blocks and restores the entire file.

In this implementation, a simple packet retransmission mechanism is adopted against packet-loss during a file transfer (⑨). Each recipient detects that some packets in a block are lost by using the sequence number of received packets and a timer. When a recipient detects packet-loss, unicast retransmission is performed on the shortest path from the sender to the recipient in parallel to multicast transmissions of packets of the same or a different block.

In [38], we verified our prototype of Coded-MPMC on a global OpenFlow testbed network, consisting of our in-lab OpenFlow network and a wide-area OpenFlow testbed network “RISE” [39], which is extended from Fukuoka and Tokyo in Japan to Seattle in the U.S. As a result of the verifications, each recipient at globally distributed locations could receive all blocks to be retrieved in an RCT very close (less than 102%) to its lower-bound when packet-loss does not occur. Besides, we could check that the retransmission scheme worked correctly when some recipients detected packet-loss. These facts imply the basic correctness of the implementation.

VIII. DISCUSSION

A. LIMITATION ON NUMBER OF CODED BLOCKS

As shown in Table 3, our proposed block allocation algorithm (Alg. 2) can realize a GBE-BA for the 1st phase, but the increase of encoding and decoding times with the increase of a necessary number of coded blocks is not considered. Furthermore, if we use Reed-Solomon (RS) coding on GF(2^8), the total number of the original and coded blocks is limited to 255. Using RS coding on GF(2^{16}) or GF(2^{32}) may relax the limit, but requires longer encoding and decoding times.

To reduce the total number of the original and coded blocks, apart from an effort of block reuse in Alg. 1, it is essential to reduce the flow values of some recipients.
TABLE 4. Changes of some flow values to reduce the number of original blocks to equal or less than 127.

| Topology     | UUNET | Barabasi-Albert model |
|--------------|-------|-----------------------|
| Sender ID    |       |                       |
| Max-flow value | Before | After | Before | After |
| 12           | 1     | 1        | 1      | 1     |
| 10           | 2     | 2        | 1      | 1     |
| 9            | 1     | 1        | 1      | 1     |
| 8            | 2     | 2        | 2      | 2     |
| 7            | 2     | 2        | 1      | 1     |
| 6            | 3     | 9        | 1      | 1     |
| 5            | 2     | 4        | 4      | 4     |
| 4            | 1     | 5        | 4      | 4     |
| 3            | 1     | 9        | 9      | 9     |
| 2            | 1     | 6        | 6      | 6     |
| 1            | 1     | 14       | 14     | 14    |
| Number of original blocks | 1260 | 120 | 180 | 120 |
| Number of coded blocks | 771 | 87 | 180 | 89 |
| Ratio of average RCT in Coded-MPMC to that in ST | 0.508 | 0.509 | 0.369 | 0.370 |

IX. CONCLUSION

We focus on a single one-to-many file transfer with a single sender and multiple recipients for fully-controlled networks with full-duplex wired links. In particular, the coded Multipath Multicast (Coded-MPMC) is introduced in which a file is divided into appropriate number \(d\) of equally-sized original blocks and both the original blocks and the coded blocks generated from the original blocks are transmitted to the recipients using multiple multicast trees. In Coded-MPMC, each recipient can retrieve the entire file by receiving any of \(d\) different original or coded blocks.

What we have shown in this paper is how to design an optimal schedule of block-wise transmissions over multiple phases in Coded-MPMC so that every recipient can retrieve the file in its lower-bound time. For any optimal schedule, we need to find a set of multicast flows on which appropriate blocks are transmitted so as to realize the max-flow value from the sender to each recipient simultaneously in each phase. We call it a globally bandwidth-efficient block allocation (GBE-BA). We have developed a heuristic method to find a GBE-BA and verified it through extensive simulations with many large-scale real-world network topologies as well as different types of synthetic topologies randomly generated. Furthermore, we briefly reported an OpenFlow-based prototype implementation of Coded-MPMC that was verified on a wide-area OpenFlow testbed network.

Note that, the present one-to-many transfer scenario has a limitation of concurrent processing, i.e., only one sender is allowed in the network during the period of transmitting bulk data. However, we believe that Coded-MPMC can also contribute to many-to-many file transfers in a centrally-managed network where multiple tasks from different senders are scheduled concurrently. Therefore, in future work, we will try to extend the Coded-MPMC approach to deal with multiple one-to-many file transfers that happen at the same time or at different overlapped times, at the sacrifice of giving up the theoretical lower-bounds of file retrieval completion times.

APPENDIX

NECESSARY CONDITION CHECK FOR GBE-BA

In this Appendix, the explanation assumes all links (edges) in both directions have the unit capacity of 1 for simplicity. A block allocation (BA) in the 1st phase is BE for recipient \(r\) when \(r\) belongs to \(m(s, r)\) different multicast flows. Therefore, any set of cut edges from \(s\) to \(r\) should be finally allocated to at least \(m(s, r)\) different multicast flows. This fact suggests that before the block allocation for recipient \(r\) is performed, we can know that the current BA configuration will be non-BE at least for \(r\) by checking a necessary condition. For example, let \(E\) be a set of cut edges from \(s\) to \(r\). Suppose a...
Then, max-flow value $m^*$ from $s$ to $r'$ on the reduced network $D(V^*, A^*, C^*)$ is computed to find possible flows other than $F^*$ on $D(V, A, C)$. If $m^* + |F^*| < m$, there is no room to extend $(m-|F^*|)$ different multicast flows to $r'$ and the future BA cannot be BE for $r'$, thus Fail is returned. If $m^* + |F^*| = m$ for every subsequent recipient, Pass is returned.

After Fail is returned, as shown in Alg. 3, the current block allocation for $r$ is canceled and postponed. In Alg. 2, next time, a new max-flow set $P^*$ for $r$ different from the failed $P$ will be selected by finding a max-flow set on the graph on which a sufficiently large cost is assigned to the links used by the failed block allocation for $r$.

In the case of Fig. 10(b), after the block allocation for $r = 9$, we will check $r' = 14$ and see that the future block allocation is non-BE for 14 because $m^* = 3, |F^*| = 2$, and $m^* = 0$; therefore, this block allocation for 9 is canceled. Later, for the next trial, a sufficiently large cost is assigned to the links $(10 \rightarrow 9)$ and $(10 \rightarrow 14 \rightarrow 13 \rightarrow 9)$ to select a new max-flow set of 9 that is not harmful to 14 such as Fig. 10(c).

We recognize that it is still inefficient to examine the condition to every subsequent recipient after the block allocation for the current recipient; the total number of this check is proportional to $K^2$, where $K$ is the number of recipients. Therefore, we optionally use this check as Alg. 3 by inserting it to Alg. 2.

**REFERENCES**

[1] L. Luo, H. Yu, K.-T. Foerster, M. Noormohammadi, and S. Schmid, “Inter-datacenter bulk transfers: Trends and challenges,” *IEEE Netw.*, vol. 34, no. 5, pp. 240–246, Sep. 2020.

[2] K. Miller, K. Robertson, A. Tweedy, and M. White, *Starburst Multicast File Transfer Protocol (MFTP) Specification*, document draft-miller-mftp-spec-03.txt, Internet Draft, Apr. 1998, Accessed: Feb. 22, 2021. [Online]. Available: https://tools.ietf.org/html/draft-miller-mftp-spec-03

[3] S. Bhattacharyya, J. F. Kurose, D. Towsley, and R. Nagarajan, “Efficient rate-controlled bulk data transfer using multiple multicast groups,” *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 895–907, Dec. 2003.

[4] Y. Zhao, J. Wu, and C. Liu, “On peer-assisted data dissemination in data center networks: Analysis and implementation,” *Tsinghua Sci. Technol.*, vol. 19, no. 1, pp. 51–64, Feb. 2014.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, and J. Zolla, “B4: Experience with a globally-deployed software defined WAN,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013.

[6] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, “Calendaring for wide area networks,” in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, vol. 44, no. 4, pp. 515–526.

[7] M. Jimenez and H. Kwok. (May 2017). Building Express Backbone: Facebook’s New Long-Haul Network. Accessed: Feb. 22, 2021. [Online]. Available: https://engineering.fb.com/data-center-engineering/building-express-backbone-facebook-s-new-long-haul-network/

[8] S. Islam, N. Muslim, and J. W. Atwood, “A survey on multicasting in software-defined networking,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 355–387, 1st Quart., 2018.

[9] M. Noormohammadi and C. S. Raghavendra, “Datacenter traffic control: Understanding techniques and tradeoffs,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1492–1522, 2nd Quart., 2018.

[10] J. H. Wang, J. Wang, C. An, and Q. Zhang, “A survey on resource scheduling for data transfers in inter-datacenter WANs,” *Comput. Netw.*, vol. 161, pp. 115–137, Oct. 2019.

[11] S. K. Singh, T. Das, and A. Jukan, “A survey on Internet multipath routing and provisioning,” *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2157–2175, Jul. 2015.
A. Nagata, Y. Tsukiji, and M. Tsuru, “Delivering a file by multipath-multicast with gossiping,” in Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst., Sep. 2013, pp. 835–840.

N. Dukkipati and N. McKeown, “Why flow-completion time is the right metric for congestion control,” ACM SIGCOMM Comput. Commun. Rev., vol. 36, no. 1, pp. 59–62, Jan. 2006.

M. Kurata, K. Heira, M. Shibata, and M. Tsuru, “Minimizing one-to-many file transfer times using multipath-multicast with Reed-Solomon coding,” in Proc. 31st Int. Teletraffic Congr. (ITC), Aug. 2019, pp. 115–116.

S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet topology zoo,” IEEE J. Sel. Areas Commun., vol. 29, no. 9, pp. 1765–1775, Oct. 2011. Accessed: Feb. 22, 2021. [Online]. Available: http://www.topology-zoo.org/dataset.html

NetworkX—NetworkX Documentation. Accessed: Feb. 22, 2021. [Online]. Available: http://networkx.github.io/

A. Iyer, P. Kumar, and V. Mann, “Avalanche: Data center multicast using software defined networking,” in Proc. 6th Int. Conf. Commun. Syst. Netw. (COMSNETS), Jan. 2014, pp. 1–8.

M. Noormohammadpour, C. S. Raghavendra, S. Rao, and S. Kundula, “DCCast: Efficient point to multipoint transfers across datacenters,” in Proc. 9th USENIX Workshop Hot Topics Cloud Comput. (HotCloud), 2017, pp. 1–8.

S. Ji, S. Liu, and B. Li, “Deadline-aware scheduling and routing for inter-datacenter multicast transfers,” in Proc. IEEE Int. Conf. Cloud Eng. (IC2E), Apr. 2018, pp. 124–133.

M. Noormohammadpour, C. S. Raghavendra, S. Kundula, and S. Rao, “QuickCast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts,” in Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun., Apr. 2018, pp. 225–233.

L. Luo, K.-T. Foerster, S. Schmid, and H. Yu, “DaRTree: Deadline-aware multicast transfers in reconfigurable wide-area networks,” in Proc. IEEE/ACM 27th Int. Symp. Qual. Service, Jun. 2019, pp. 1–10.

O. Alhussein, P. T. Do, Q. Ye, J. Li, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, “A virtual network customization framework for multicast services in NFV-enabled core networks,” IEEE J. Sel. Areas Commun., vol. 38, no. 6, pp. 1025–1039, Jun. 2020.

K. Ogawa, T. Iwamoto, and M. Tsuru, “One-to-many file transfers using multipath-multicast with coding at source,” in Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun.; IEEE 14th Int. Conf. Smart City; IEEE 2nd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS), Dec. 2016, pp. 687–694.

K. Heira, K. Ogawa, and M. Tsuru, “One-to-many file transfer using multipath-multicast with gossiping,” in Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft), Jun. 2018, pp. 187–191.

P. Erdős and A. Rényi, “On random graphs I,” Publ. Math. Debrecen, no. 6, pp. 290–297, 1959.

B. M. Waxman, “Routing of multipoint connections,” IEEE J. Sel. Areas Commun., vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

A-L. Barabási and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

A. Elmokashfi, A. Kvalbein, and C. Dovrolis, “On the scalability of BGP: The role of topology growth,” IEEE J. Sel. Areas Commun., vol. 28, no. 8, pp. 1250–1261, Oct. 2010.

Ryu SDN Framework. Accessed: Feb. 22, 2021. [Online]. Available: https://ryu-sdn.org/