Deadline-guaranteed Point-to-Multipoint Bulk Transfers in Inter-Datacenter Networks

Long Luo, Hongfang Yu, Zilong Ye*
School of Information&Communication Engineering, Cyber Space Security Center,
University of Electronic Science and Technology of China, Chengdu, P. R. China
* California State University, Los Angeles, CA 90032, USA

***Abstract*—Many modern cloud services are operated across geographically distributed datacenters, and they are usually associated with a demand of transferring bulk data among datacenters for achieving a high performance and reliability. These transfers (*e.g.*, data replications and synchronizations) may require the inter-datacenter networks to deliver data from one Point (or datacenter) to Multiple Points (or datacenters) and impose a deadline on the transfer for providing a guaranteed service to end users. However, very little work has been done to manage an efficient point-to-multipoint bulk transfer while considering the deadline requirements. In this paper, we investigate the deadline-aware point-to-multipoint (P2MP) transfer problem and propose a centralized deAdline-Guaranteed transfEr (AGE) approach that can guarantee the deadline for P2MP transfers while efficiently utilizing the inter-datacenter bandwidth resources. For each arriving request, AGE jointly determines the transfer source selections and bandwidth allocations such that the number of deadline-guaranteed transfers can be maximized. Our simulation experiments show that AGE can accommodate up to 53.3% more transfers whose deadline requirements are met. In addition, AGE can achieve 20% higher network utilization than prior bulk transfer approaches.**

I. Introduction

Today, an increasing number of service providers operate their services in the cloud or datacenters at geographically distributed locations to serve their users worldwide [1, 2]. In order to improve the service performance, fault tolerance, data synchronization or replication, many cloud applications often perform Point-to-MultiPoint (P2MP) bulk transfers among multiple geographically distributed datacenters that run their tasks [3]–[5]. For example, the geo-distributed machine learning computing task that relies on the servers from multiple datacenters may regularly transfers bulk synchronization data and intermediate computation results across multiple working datacenters during the learning procedure.

In addition to the bulk size (*e.g.*, ranging from tens of terabytes to petabytes), one key characteristic of these P2MP transfers is that they are often associated with deadline requirements [6]–[8]. According to an interview survey at Microsoft [6], 100% of the customers claim that they have deadline requirements for all the inter-datacenter transfers and opt to use a network that can guarantee the transfer deadline for ensuring their application service quality. Actually, missing the deadline will greatly degrade the service performance or even violate the service level agreements (SLAs) contracted with the customers, which may result in a significant amount of penalty [8].

Managing an efficient bulk transfers while ensuring the deadline is not easy. Recently, there are many existing approaches on the inter-datacenter transfer problem [1, 2, 7]–[14]. Most of them focused on how to fully utilize the dynamic leftover bandwidth resources to schedule and deliver data for point-to-point (P2P) transfer requests. There are few works on investigating P2MP bulk transfers [12, 14]. Compared to the P2P transfer with a single destination, the P2MP transfer is associated with multiple required destinations, which is more challenging to be addressed. To accommodate P2MP transfers, one possible solution is to consider a P2MP transfer as multiple individual P2P transfers and use existing P2P transfer methods to schedule them separately [1, 2, 7]–[11]. This solution may finish some of these individual P2P transfers before deadlines, but does not guarantee the completion of the entire P2MP transfer till the deadline. While to optimize the application performance, we should provide deadline guarantees at the level of P2MP transfers rather than individual P2P transfers. More recently work [12, 14] considered to schedule a given P2MP transfer as a whole and proposed to construct forwarding trees to simultaneously deliver the required data from the source datacenter to all destinations at the same rate. It maximizes the data transferred before deadline for all required destinations, however, it has poor performance because the slowest destination throttles the completion time for all destinations. Hence, it is not efficient if we just simply apply the existing P2P transfer approaches or the state-of-the-art P2MP transfer approach to address the deadline-aware P2MP transfer problem.

In this paper, we propose a centralized deAdline-Guaranteed transfEr (AGE) approach, which can efficiently address the P2MP transfer problem such that the P2MP-level deadline can be guaranteed and network utilization can be optimized. Our key idea of AGE is that, in a P2MP transfer request, any destination that has completed the transfer before deadline can serve as a source and deliver the required data to other uncompleted destinations of this request. This gives us the opportunity to dynamically select the transfer data source for required destinations during the transfer procedure. More specifically, AGE jointly optimizes the transfer source selection

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61271171, 61401070 and 61701074, and the National Key Research and Development (R&D) Program under Grant No. 2016YFB0800105.

and bandwidth allocation, which enables many unused bandwidth resources to be available for delivering the required data, thus achieving an efficient utilization of the network resources while ensuring the deadline for transfer requests. AGE differs significantly from prior work since it performs flexible transfer source selections in addition to bandwidth allocations. In AGE, the computation of the optimal (*i.e.*, the maximum deadline guarantees) transfer source selection and bandwidth allocation for each request is formulated as an Mixed Integer Linear Program (MIP). For many small-scale transfer problems, we can obtain the optimal solutions by directly solving the MIPs built for them. However, it is time-consuming or sometimes intractable to apply MIPs to solve large-scale transfers (*i.e.*, those have deadline of several hours). In these cases, AGE divides the time into a series of small time intervals and accordingly builds a series of small MIPs, each of which is to maximize the data that can be transferred in a particular small time interval. AGE progressively solves these small MIPs until it finds a deadline guaranteed solution or exceeds the transfer deadline. Whatever the actual deadline is, AGE can always compute a deadline guaranteed schedule for every transfer request in a reasonable time.

The main contributions of this work are summarized as below:

- We address the deadline-constrained P2MP transfer problem by jointly optimizing the transfer source selection and the bandwidth allocation to guarantee deadlines and efficiently utilize the network resources. To the best of our knowledge, this is the first study that explores the deadline-guaranteed P2MP bulk transfer (Section III).
- We formalize the transfer problem as a MIP model, and we also present a basic algorithm to compute the optimal solution and an acceleration algorithm to speed up the computation through progressively solving a series of smaller MIPs (Section V).
- We evaluate AGE by simulating data transfer scenarios for network topologies observed from real inter-datacenter networks. Our results show that AGE provides deadline guarantees for up to 53.3% more transfer requests and achieves around 20% higher network utilization, compared to prior solutions (Section VII).

II. Related Work

III. Motivation Example

A. Point-to-MultiPoint (P2MP) transfers

Data transfer is very common in inter-datacenter networks. A key finding is that up to 77% cloud services run backup and replication among three or more geographically distributed datacenters [10]. Such geo-replication requires to transfer bulk data from one point to multiple points (P2MP) over the inter-datacenter networks. Examples of service providers that use geo-replication include Google [5], Microsoft [3], Amazon [4] and etc. [12]. The applications (*e.g.*, search, CDNs, streaming, etc.) they host may need the replication of search index, popular content, database, and etc., to improve the service reliability against failures.

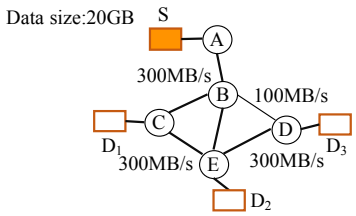
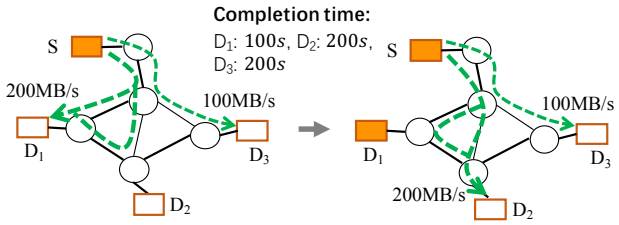
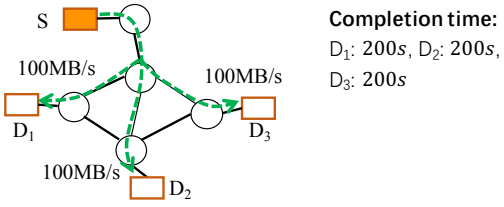


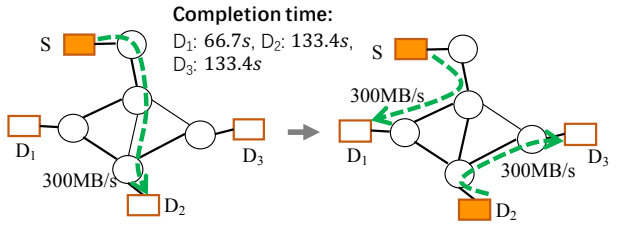
Fig. 1: A P2MP request example: 20GB data are required to be transferred from S to D_1, D_2 and D_3.



(a) Approach 1: Processing a P2MP transfer as multiple P2P transfers.



(b) Approach 2: Constructing a forwarding tree to transfer data.



(c) Approach 3: Guaranteeing deadline through flexibly source selection.

Fig. 2: A motivation example: comparison of different policies to the transfer problem in Fig. 1.

Deadline: Deadline is one of the most important SLAs imposed by many cloud applications to the inter-datacenter transfers. Many online service applications such as search and video streaming are time-sensitive, which naturally expect to complete transfers before a certain time, or namely deadline [1, 2, 7, 8, 15]. While, the inter-datacenter transfers are delay-tolerant and usually do not impose strict bandwidth guarantee during the transferring process, which allows cloud providers to dynamically schedule the bulk data transfer by fully utilizing the bandwidth resources on inter-datacenter links.

Although there exists many literatures on the bulk transfer problem, we unfortunately observe that none of them can efficiently handle the P2MP transfers, especially those are required to meet certain deadline by customers. Considering

an illustrative P2MP transfer request in Fig. 1, a volume of $20GB$ data needs to be replicated from datacenter S to three remote destination datacenters D_1, $D2$ and D_3 within 150 seconds (*i.e.*, the deadline is $150s$) after it is submitted. The link between B and D has a available capacity of $100MB/s$, while the rest network links have a capacity of $300MB/s$. One possible approach (*i.e.*, approach 1 in Fig. 2) to satisfy the transfer request is to handle it as three individual point-to-point (P2P) transfers and process them independently using current P2P transfer approaches [7]–[11]. Such an approach can split the original P2MP transfer three sub-transfers: R_1 : $(src = S, dest = D_1)$, R_2 : $(src = S, dest = D_2)$, and R_3 : $(src = S, dest = D_3)$ as shown in Fig. 2(a). Fig. 2(a) shows one of the possible bandwidth allocations using Approach 1: the bulk data at source A will be delivered to D_1 at a rate of $200MB/s$ and to D_3 at a rate of $100MB/s$. As a result, the sub-transfers R_1 and R_3 will be completed in $100s$ and $200s$, respectively. While for sub-transfer R_2, to fully utilize the leftover bandwidth resources, it will transfer the data at a rate of $200MB/s$ between $100s$ and $200s$. Anther approach is to handle the P2MP transfer as a whole [12], which constructs a forwarding tree rooted at the source that transfers data to all the destinations at the same rate (*e.g.*, using the maximum achievable rate of the bottleneck link). Fig. 2(b) illustrates one of the possible forwarding trees built by this approach (*i.e.*, approach 2 in Fig. 2): the bulk data will be transferred from S to three destinations at the same rate $100MB/s$, which is the maximum achievable rate of the bottleneck link B-D. Consequently, approach 2 would obtain the same completion time $200s$ as that of approach 1. Obviously, neither of these two approaches can finish the transfer before the required deadline $150s$.

Actually, there is an more efficient way to schedule this P2MP transfer (in Fig. 1) by flexibly determining the transfer source for required destinations, which can guarantee the completion of the transfer before the required deadline. Recall the transfer request in Fig. 1, one possible solution is to first arrange a sub-transfer from source S to destination D_2. Once this sub-transfer is completed, we can initiate two sub-transfers: one from S to D_1 and another one from D_2 to D_3 (see Fig. 2(c)). This approach can lead to the completion of the requested P2MP transfer no later than 133.4 seconds, which meets the deadline $150s$. This example gives us the motivation that jointly considering the transfer source selection and the bandwidth allocation can significantly reduce the transfer completion time for P2MP transfer requests. Accordingly, the chance of guaranteeing deadline can also be greatly improved. In addition, the network resource utilization can also be highly increased when the transfer source selection and the bandwidth allocation are jointly optimized.

IV. Overview of AGE

Based on the motivation, we propose AGE, which jointly optimizes the transfer source selection and the bandwidth allocation to maximize the probability of deadline guarantee for inter-datacenter P2MP transfers. AGE adopts a logically centralized *controller* to manage the transfer requests. The controller maintains the network wide states, bandwidth usages and traffic demands, and performs the admission control to the submitted transfer requests, as well as provisioning the transfer source selection and spatial-temporal bandwidth allocation for the accepted requests. In particular, AGE works as follows. When a P2MP transfer request arrives, AGE quickly determines whether the request can be admitted or not, taking into consideration the inputs such as available bandwidth resources, inter-datacenter tunneled paths and the requested data transfer. For the accepted requests, the controller enforces the joint optimization solution using available tools in inter-datacenter networks, such as SDN, to dynamically add rules to active forwarding paths between datacenters [8].

AGE performs the request admission control in an online fashion. It is based on a first-come first-served manner and does not allow preemption. In AGE, the previously made decisions are not allowed to be revoked, and each newly arriving transfer request needs to be scheduled by considering the residual bandwidth in the network. Note that one may achieve better solutions by rescheduling all the unfinished transfer requests to accommodate new transfer requests. However, such reschedule is time-consuming because it increases not only the problem complexity but also the problem scale, which is not practical. AGE can ensure the deadlines for many P2MP transfers requests through flexibly selecting the transfer source.

Similar to previous works on the inter-datacenter transfer problem [7, 8, 10, 12], AGE assumes a time slotted system where time is divided into a series of discrete timeslots at equal lengths. The allocated bandwidth is fixed within a timeslot but may vary across different timeslots. The main idea of our transfer schedule algorithm (see the details in Section V) is summarized as follows. When a P2MP transfer request arrives, AGE essentially tries to use residual network capacity to complete it before deadline by solving an optimization problem (Section V-B). For a P2MP transfer request, AGE quickly accepts it if all the required destinations can complete the data transfer before the deadline, otherwise AGE will reject it. Note that a rejected request can be resubmitted by the clients later time, as long as the deadline has not expired.

V. Problem description and algorithm design

A. Network model

We use a graph $G = (V, E)$ to model the inter-datacenter network, where set V includes all the datacenters and set E contains the inter-datacenter links. Each link $e \in E$ connects two datacenters and has a time-varying capacity $c_{e,t}$. We assume tunnel-based forwarding is used and there exists several pre-established tunnel paths between each pair of datacenters. Meanwhile, we use $P_{u,v} = (u, u_1, ..., v)$ to denote a tunnel path between datacenter u and v. As for a P2MP transfer request, we describe it by $(s, \{d^1, d^2, \ldots, d^m\}, f, t^1, t^2)$, where s, $\{d^1, d^2, \ldots, d^m\}$, f, t^1, t^2 respectively denote the source datacenter, the required destination datacenters, the to-be-transferred data size, the start time and the deadline.

B. Problem formulation

Our formulation is based on the following assumptions for every P2MP transfer request. First, we take the request arrival time τ^{arr} as the start time (t^1) of data transfer. Second, a destination datacenter can be as a potential new transfer source for other destinations when it has received all the required data. For the practical concerns such as the operational cost, we assume that each destination receives data from one datacenter. The proposed AGE approach consists of three main steps, which are (1) Determining the transfer sources, (2) allocating available bandwidth resources, and (3) guaranteeing deadlines for each transfer requests. The detailed designs of these three steps are as follows.

Determining the source for transfers: We use a binary h_k^t to denote whether a datacenter $k \in \{s, d^1, d^2, \ldots, d^m\}$ can serve as the transfer source at time $t \in [t^1, t^2]$. For the original transfer source datacenter s, we certainly have

$$\forall t : h_s^t = 1 \tag{1}$$

While for a transfer destination datacenter $d \in \{d^1, ..., d^m\}$, as mentioned before, it can become an available transfer source at time t if it has received the complete package of data f by the end of timeslot $t-1$:

$$\forall t, d : \sum_{\tau=t^1}^{t^1+t-1} \sum_{k,k\neq d} x_{k,d}^{\tau} \geq f h_d^t \tag{2}$$

where $x_{k,d}^{\tau} \geq 0$ represents the data volume received at destination d from datacenter k at time $\tau \in (t^1, t^1+t-1]$. Meanwhile, a datacenter k is able to transfer data to datacenter $d, d \neq k$ only if it is an available transfer source:

$$\forall t, d, k, k \neq d : x_{k,d}^t \leq f h_k^t \tag{3}$$

To be practical, each destination is restricted to set up transfer connection with a single source datacenter in the process of data delivery:

$$\forall d : \sum_{k,k\neq d} z_{k,d} = 1 \tag{4}$$

where binary $z_{k,d}$ denotes whether datacenter k is the transfer source of destination d. Clearly, there should be

$$\forall t, d, k, k \neq d : x_{k,d}^t \leq f z_{k,d} \tag{5}$$

$$\forall t, d, k, k \neq d : x_{k,d}^t \geq z_{k,d} \tag{6}$$

Allocating available bandwidth resources: Let $y_{k,d,P}^t$ denote the bandwidth resources that allocated along the routing path P from datacenter k to datacenter d at time t. The allocations are feasible if:

$$\forall t, d, k, k \neq d : \sum_{P\in\mathcal{P}_{k,d}} \alpha y_{k,d,P}^t \geq x_{k,d}^t \tag{7}$$

$$\forall e, t : \sum_k \sum_d \sum_{P\in\mathcal{P}_{k,d}} y_{k,d,P}^t I(e \in P) \leq c_{e,t} \tag{8}$$

where constant α denotes the length of each time slot and expression $\sum_k \sum_d \sum_{P\in\mathcal{P}_{k,d}} y_{k,d,P}^t I(e \in P)$ denotes the amount of traffic on link e at time t. Equation (7) states that the allocated routing and bandwidth resources should at least be capable of transferring the demanding data size; Equation (8) expresses the link load is restricted to not exceed the capacity to avoid link congestion.

Guaranteeing deadline for each transfer request: This needs us to ensure the data transfer can be completed before the deadline for every destination datacenter. Let binary w_d denote whether the destination $d \in \{d^1, ..., d^m\}$ has received all the data before deadline t^2. Guaranteeing the deadline for destination d requires the transfer to be completed no later than time $t^1 + t^2$, so we have

$$\forall d : \sum_{\tau=t^1}^{t^1+t^2} \sum_{k,k\neq d} x_{k,d}^{\tau} \geq f w_d \tag{9}$$

C. Basic algorithm

Maximizing the deadline-guaranteed destinations Our baseline algorithm for addressing the P2MP transfer problem is straightforward and is summarized in Algorithm 1. Given G and a deadline-constrained P2MP transfer request R, our goal is to maximize the number of destinations that meet the specified deadline through solving the corresponding MIP problem in Algorithm 1.

If the number $\sum_d w_d$ of completed destinations equals to the number of required destinations for a request, we accept this request and transfer the required data during the scheduled transfer period. Besides, for every link, the residual bandwidth resources will be updated as the available bandwidth (before the acceptation) minus the allocated resources ($\boldsymbol{y}$).

Algorithm 1 Basic Algorithm

1: **Input:** $R = (s, d^1, ..., d^m, f, t^1, t^2)$: a transfer request;
$\mathcal{P}_{u,v} = \{P_1, P_2, ..., P_k\}$: k-shortest paths between the datacenter u and v;
$c_{e,t}$: residual link bandwidth on link $e \in E$ at time $t \in (t^1, t^2]$.

2: **Output:** Return the completion status w_d in the solution of the following problem:

$$\max \sum_{d\in\{d^1,\ldots,d^m\}} w_d$$
$$s.t. \text{ constraints (1)-(9).}$$

D. Acceleration algorithm

Although our baseline algorithm can obtain the optimal transfer solutions for many requests, it may not work well for requests that last for a long time. The linear programming (in Algorithm 1) built for a P2MP transfer request has $\mathcal{O}(|N|^2|\mathcal{P}||T|)$ variables, where $|N|$, $|\mathcal{P}|$, $|T|$ denotes the number of datacenters, tunnel paths and time slots, respectively. $|N|$ and $|\mathcal{P}|$ are usually small because the inter-datacenter networks usually have limited number of datacenters and a small number of available tunnel paths between two datacenters. However, the

scale of the number of time slots $|T|$ may vary largely (*e.g.*, ranging from dozens to hundreds or even thousands of 5-minute time slots) because the transfer deadline usually ranges from hours to a few days [7]. As a result, when the deadline of transfers increases, the number of variables involved in the linear programming will grow dramatically, making the problem difficult to be solved in a reasonable time. For example, for a transfer request that involves 5 datacenters, 5 tunnel paths between each pair of datacenters and a completion deadline of 10 hours, the linear programming built for it includes at least $5^2 \times 5 \times \frac{10\times 60}{5} = 15000$ variables, if the time slot is set to be 5-minute. For such cases, it is computationally intractable to solve the linear programming problem and obtain the optimal solution through employing the commodity optimization solvers. Therefore, to overcome this limitation, we present an acceleration algorithm to efficiently address the transfer source selection and the bandwidth allocation when the scale of this transfer problem is large.

Algorithm 2 Acceleration Algorithm

Input: A P2MP transfer request $R = \{src = s, dest = \{d_1, \cdots, d_m\}, f_{size}, t^1, t^2\}\}$, tunneling paths $\mathcal{P}$ and link residual capacities $\{c_{e,t}, \forall e \in E, t \in [t^1, t^2]\}$.
Output: A transfer allocation solution.
1: Generate time series $\{T_\theta\}$ through $T_\theta = T_{\theta-1} + \frac{t^2-t^1}{\kappa}, \forall \theta = 2, ..., \kappa$, $T_1 = t^1$, $T_\kappa = t^2$,
2: $\theta = 1$;
3: $\overline{w}_d = 1, \forall d = 1, ..., m$;
4: **while** $\sum_d \overline{w}_d > 0$ **and** $\theta < \kappa$ **do**
5: Let $\overline{w}_d$ be given parameters and solve the linear programming with equations (1)-(8), (10)-(11) and objective of $\max \sum_d w_d$;
6: For all destinations d, update $\overline{w}_d$: $\overline{w}_d = \overline{w}_d - w_d$;
7: $\theta = \theta + 1$;
8: **end while**
9: **if** $\sum_d \overline{w}_d^\theta \leq 0$ **then**
10: **return** The transfer solutions $\boldsymbol{y}$.
11: **end if**
12: **return** Missing the deadline.

The main idea of our acceleration algorithm (Algorithm 2) is to divide the time line into several (*i.e.*, κ) number of short length-equal time intervals and solve a series of small optimization problems, each of which is built for a time interval. Since each divided time interval contains a few number of time slots, we can efficiently solve the small linear programming built for each time interval. In particular, if there exists data that remains to be transferred at the beginning of a given time interval, the algorithm generates an optimization problem that maximizes the data amount that can be transferred before the end of this time interval (Line 4-Line 22 in Algorithm 2). For the optimization problem built for the θ-th time interval $(T_\theta, T_{\theta+1}]$, the percentage w_d of the can-be-transferred data (in this time interval) over the proportion $\overline{w}_d$ of the data that remains to be transferred to destination d satisfies:

$$\forall d : \sum_{\tau=T_\theta}^{T_{\theta+1}} \sum_{k,k\neq d} x_{k,d}^\tau \geq f w_d \tag{10}$$

$$\forall d : 0 \leq w_d \leq \overline{w}_d \tag{11}$$

After obtaining the transfer allocations at each iteration, the algorithm fixes the allocation decisions and update the volume $\{\overline{w}_d, \forall d\}$ that remains to be transferred according to $\{w_d, \forall d\}$ results. The Algorithm 2 is stopped when one of the following conditions is satisfied: 1) the required data has been transferred to all destinations (or $\sum_d \overline{w}_d^\theta \leq 0$); 2) the number of iteration θ reaches the iteration limit κ.

VI. Quick transfer scheduling algorithm

By solving transfer problems with a small time span, the acceleration algorithm can improve the computation efficiency of the basic algorithm. However, the acceleration algorithm is not panacea for all the transfer cases.

In the above text, we assume these P2MP transfers are destined to a small fraction of datacenters. However, there exists cases that P2MP transfers are destined to a large fraction As revealed in recent work [16], the inter-DC transfer workload in Baidu [1] has the following key characteristic. A majority (e.g., 90%) of transfers are destined to at least 60% of the DCs, and 70% are destined to more than 80% of the DCs [16]. For these transfers that destinate to many datacenters, we show that the acceleration algorithm may be struggle to deal with.

As showed in the Line 5 of Algorithm 2, the acceleration algorithm iteratively solves linear programming with equations (1)-(8), (10)-(11) and objective to $\max \sum_d w_d$. As analyzed before, these optimization model has $\mathcal{O}(|N|^2|\mathcal{P}||T|)$ variables, where $|N|$, $|\mathcal{P}|$, $|T|$ denotes the number of destination datacenters, tunnel paths and time slots, respectively. We can see that $|N|$ dominates the complexity of the optimization problem. Even though both $|\mathcal{P}|$ and $|T|$ are small, the large $|N|$ can easily make this optimization problem intractable to solve in large-scale networks. Therefore, to be general, it is necessary to develop more efficient transfer algorithm that can handle transfer requests even with a large portion of destination datacenters.

We present Algorithm 3 to quickly obtain the transfer scheduling and bandwidth allocation solution for P2MP requests in large networks. As illustrated in our motivation example (see Fig. 2), flexibly determining the transfer source for destinations may improve the chance to satisfy the deadline requirement. So, our core idea here is to make as many destination nodes as possible available transfer sources as early as possible.

At the start, there is only one datacenter that can transfer data to destination datacenters, as shown in Algorithm 3. We want to find a destination that can receive full data within the minimum completion time. ***

[1]Baidu is a global-scale online service provider

Algorithm 3 QucikTransfer

Input: A P2MP transfer request $R = \{src = s, dest = \{d_1, \cdots, d_m\}, f_{size}, t^1, t^2\}\}$, tunneling paths $\mathcal{P}$ and link residual capacities $\{c_{e,t}, \forall e \in E, t \in [t^1, t^2]\}$.
Output: A transfer allocation solution.

$SrouceList \leftarrow R.src$
$DestList \leftarrow R.dest$;
$c'_{e,t} = c_{e,t}, \forall e \in E, t \in [t^1, t^2]$;
while $DestList \neq \emptyset$ **do**
  for $d \in DestList$ **do**
    for $s \in SrouceList$ **do**
      $t_{s,d} \leftarrow$ COMPUTETCT$(s, d, t^1, f_{size}, \mathcal{P}_{s,d}, \boldsymbol{c}')$;
    end for
  end for
  Let $t_{min} = \min\{t_{s,d}, s \in SourceList, d \in DestList\}$;
  if $t_{min} > t^2$ **then**
    break the while-loop;
    return Can't find a transfer allocation solution that satisfies the deadline.
  end if
  Let $s', d' = \arg\min_{s,d} f(t) := \{t | t \in \{t_{s,d}, s \in SourceList, d \in DestList\} : f(t) = t\}$
  Let s' be the transferring source of d';
  $t^{d'}_{TCT} = t_{min}$;
  Allocate resources for transferring data from s' to d';
  Update link residual capacities $c'_{e,t}$;
  $DestList.remove(d')$;
  $SourceList.append(d')$;
end while
$c_{e,t} = c'_{e,t}, \forall e \in E, t \in [t^1, t^2]$;
return Transfer solutions and $\{c_{e,t}, \forall e \in E, t \in [t^1, t^2]\}$.

function COMPUTETCT(s, d, t^{start}, f_{size}, $\mathcal{P}$, $\boldsymbol{c}$)
  Construct a directed graph G for s and d;
  Add nodes $\{v | v \in \mathcal{P}\}$ to G;
  Add path links (with direction) $\{l | l \in \mathcal{P}\}$ to G;
  Assign the capacity for links $l \in G$ according to $\boldsymbol{c}$;
  $t_{TCT} = t^{start}$, $f_{resize} = f_{size}$;
  while $f_{resize} > 0$ **do**
    $f_{deliver}, bw_{allocate} \leftarrow$ MAXFLOW(G, s, d, t_{TCT});
    $f_{resize} = \max\{f_{resize} - f_{deliver}, 0\}$;
    $t_{TCT} = t_{TCT} + span(timeslot)$;
  end while
  return t_{TCT}, $bw_{allocate}$
end function

TABLE I: Summary of topologies used in evaluations.

Name	Description
GScale [2]	Google's inter-datacenter wide-area network, which has 12 datacenters and 19 inter-datacenter links.
Equnix [17]	An inter-datacenter wide-area network from Equnix. It connects 20 datacenter nodes and has 141 inter-datacenter links.
IDN [1]	Microsoft's inter-datacenter network that includes 40 datacenters and each is connected to 2-16 other datacenters.
Cogent [14, 18]	A large backbone, customer and transit network that connects 197 datacenters across USA and Europe with 243 inter-datacenter links.

VII. PERFORMANCE EVALUATION

In this section, we conduct extensive experiments to evaluate the performance of AGE. We firstly ***, then ***, and finally **.

Dataset: 1) Topologies: We evaluate the performance of the above three approaches by running simulations over four realistic inter-datacenter networks: a) Google's GScale topology [2], which has 12 datacenters and 19 inter-datacenter links [1], b) Equnix topology, which has 20 datacenters and 141 inter-datacenter links, c) Microsoft's inter-datacenter network (IDN), which has 40 datacenters and each datacenter connect 2-16 other datacenters, and d) a large backbone, customer and transit network (Cogent) that has 197 datacenters and 243 links. 2) Transfer requests: We consider P2MP transfer requests with different number of required destinations and varying traffic load, respectively.

Approaches compared: We compare the proposed AGE approach with the following P2MP transfer approaches:

- MP2P: directly applies the state-of-the-art deadline-aware P2P transfer approach [8] to P2MP transfers. Specifically, MP2P considers each P2MP transfer request as multiple individual point-to-point (P2P) transfer requests and adopts the approach in [8] to schedule each P2P transfer and arrange a deadline-guaranteed allocation via solving a min-cost flow formulation;
- DCCast: schedules each P2MP transfer as a whole. However, in contrast to guaranteeing deadline for transfers as [8] and our work does, its objective is to minimize the transfer completion time [12].
- AGE: the approach proposed in this paper. In particular, we use AGE-Basic, AGE-Accelerate and AGE-Quick to distinguish the basic algorithm (Algorithm 1), the acceleration algorithm (Algorithm 2) and the quick transfer algorithm (Algorithm 3) presented in the above sections.

We implement all the simulations in Python and employed Mosek (https://www.mosek.com/) as our backend optimizer to solve the MIPs in simulated algorithms. We evaluate concerned transfer approaches via the following performance metrics:

- Deadline-met ratio: it is the ratio of the number $N_{deadline-met}$ of transfer requests that meet their deadlines to the total number N_{total} of P2MP transfer requests

submitted during the simulation runtime, and it can be calculated by $\frac{N_{deadline-met}}{N_{total}} \times 100\%$.
- Network utilization: it is the ratio of the data carried by a network to its capacity.
- Computation time: it is the time to obtain the final bandwidth allocation and transfer scheduling solution of transfer approaches.

A. Flexibly assigning transfer source datacenter is efficient

Figure 3 shows the ratios of deadline-met transfer requests with varying number of destinations (from 1 to 7). The results show that AGE outperforms both MP2P and DCCast in terms of the deadline-met ratio. This is because AGE considers the already completed destinations as selectable new transfer sources, and thus potentially exploits many unused bandwidth resources to speed up the completion time. As the number of the required destinations increases, we also find out that the deadline-met ratio of the MP2P approach decreases while AGE maintains a high standard. This is as expected since an increasing number of P2P transfer requests will be generated by MP2P for a given P2MP transfer as the number of destinations grows, which will resulting in consuming more bandwidth resources and a longer time to complete all the P2P transfer requests. In contrast, AGE considers the P2MP transfer request as a whole, which leads to a stable deadline-met ratio performance. From Fig. 3, we can also observe that DCCast performs the worst deadline-met ratio among three compared approaches. This is because DCCast greedily transfers data from the source to all required destinations at the maximum achievable rate of the bottleneck link per timeslot without considering the deadline requirement. As a result, many requests will miss their deadlines when DCCast is used.

In Fig. 4, we evaluate the impact of traffic load on the deadline-met ratios. In particular, we scale the actual transfer demand by a constant factor to simulate a wide range of traffic load. As the load factor increases, the deadline-met ratio decreases for all three algorithms in both simulated networks. However, MP2P and DCCast experience a more dramatic performance drop compared to AGE. The superior performance of AGE further proves that it can take the advantage of the resources scattered around the network to accelerate the data transfer thanks to its flexible transfer source selections.

Figure 5 plots the network utilization collected from simulations on GScale network. Note that we omit to present the results obtained from Equnix network because they are similar as those from GScale network. We use the metrics of total network utilization and effective utilization in [8] to measure the network utilization achieved by the three algorithms. Total network utilization refers to the network utilization of all (including both deadline-met and deadline-missed transfer data) transfers, and effective network utilization only refers to the transfers that meet their deadlines. From the plotted curves in Fig. 5, we can observe that the deadline-agnostic solution, DCCast, achieves a high total network utilization but a very low effective network utilization. This result is as expected because DCCast does not respect the deadline requirements when scheduling bulk transfers. In contrast, AGE and MP2P achieves much better effective network utilization, as they avoid wasting resources for deadline-missed requests by only admitting transfer requests whose deadlines can be guaranteed. In addition, we find AGE outperforms than MP2P solution in terms of effective network utilization, due to the higher deadline-guarantees achieved by AGE.

add simulations:

3) benefit of flexible source selection. 1) deadline-met ratio across different request arrival rate 2) algorithm computation time under different setting of parameters (deadline, destnum) **Scalability**: simulate the computation time of approaches;
In-depth analysis: 1) near-optimality of our approach; 2) choosing the values of key parameters;

VIII. Conclusion

In this paper, we investigate how to efficiently manage the inter-datacenter P2MP bulk transfers that are associated with deadline requirements. We propose a centralized deAdline-Guaranteed transfEr (AGE) approach, which performs on-line admission control and joint optimizes the transfer source selection and the bandwidth allocation for inter-datacenter P2MP transfer requests to guarantee the deadline while efficiently utilizing the network resources.

The transfer problem is formulated as a Mixed Integer Linear Program (MIP). The deadline-guaranteed schedule can be found by solving the MIP, which ensures that the maximum number of destinations can completely receive all the to-be-transferred data before deadlines. For a given transfer request, AGE accepts if all the destinations can complete the required data transfer before the deadline; otherwise AGE rejects it. A baseline algorithm is presented to solve the MIP and obtains the optimal results if the problem can be solved in a reasonable time. Furthermore, to solve the large-scale transfer problems, an accelerate algorithm is proposed that progressively splits the original transfer schedule into a series of small schedules and solves them efficiently. We conducted comprehensive simulations on real-world network topologies, and found that the proposed AGE approach can effectively accommodate 53.3% more transfer requests with deadline guarantees, while achieving around 20% higher network utilization, compared to state-of-the-art transfer solutions.

References

[1] C.-Y. Hong, S. Kandula *et al.*, "Achieving high utilization with software-driven wan," in *SIGCOMM*, 2013.

[2] S. Jain, A. Kumar, S. Mandal *et al.*, "B4: Experience with a globally-deployed software defined wan," in *SIGCOMM*, 2013.

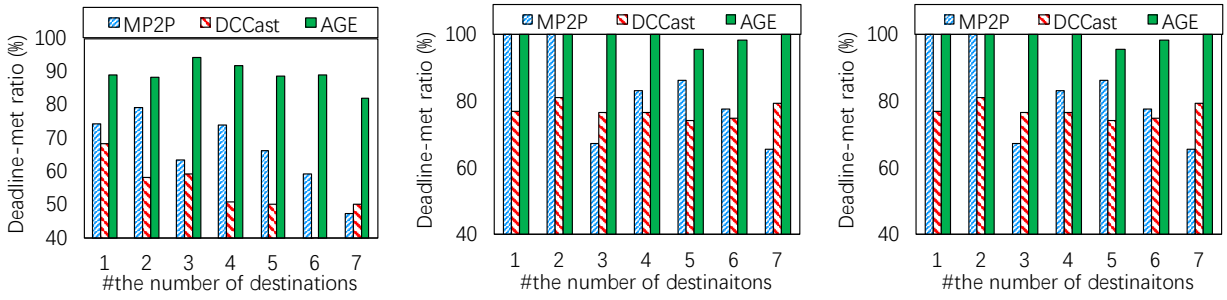
[3] "Microsoft azure: Cloud computing platform & services." https://azure.microsoft.com/.

[4] "Amazon web services (aws) - cloud computing services." https://aws.amazon.com/.

[5] "Compute engine - iaas - google cloud platform." https://cloud.google.com/compute/.

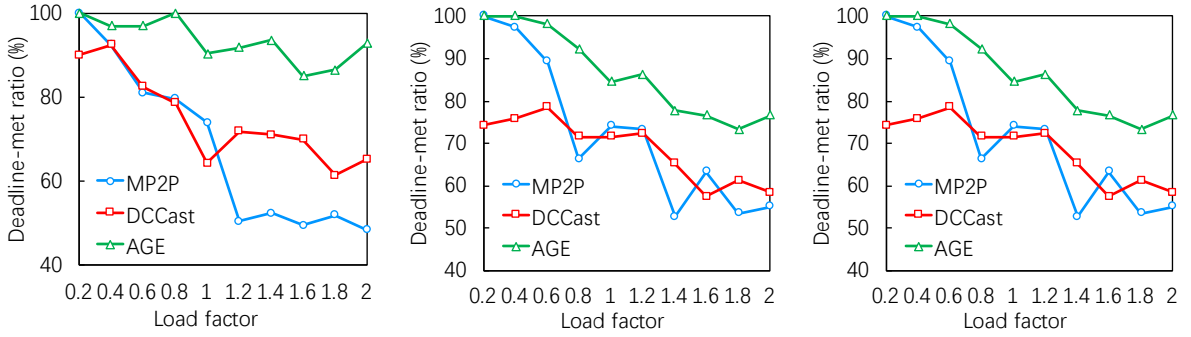
[6] V. Jalaparti *et al.*, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *SIGCOMM*, 2016.

[7] S. Kandula, I. Menache, R. Schwartz *et al.*, "Calendaring for wide area networks," in *SIGCOMM*, 2014.



(a) GScale (b) Equnix (c) IDN, **replace this figure with results simulated over IDN**

Fig. 3: The impact of the number of destinations. x-axis should be the percentage of destinations to the total node number



(a) GScale (b) Equnix (c) IDN, **replace this figure with results obtained in experiments simulated over IDN**

Fig. 4: The impact of the traffic load.

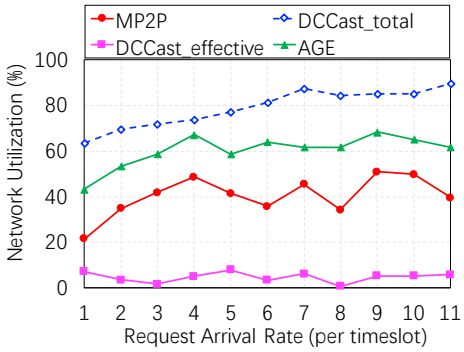


Fig. 5: AGE achieves the improved effective network utilization, compared to MP2P and DCCast [12].

[8] H. Zhang, K. Chen, W. Bai *et al.*, "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 1, pp. 579–595, 2017.

[9] X. Jin *et al.*, "Optimizing bulk transfers with software-defined optical wan," in *SIGCOMM*, 2016.

[10] N. Laoutaris, M. Sirivianos, X. Yang *et al.*, "Inter-datacenter bulk transfers with netstitcher," in *SIGCOMM*, 2011.

[11] Y. Feng, B. Li, and B. Li, "Postcard: Minimizing costs on inter-datacenter traffic with store-and-forward," in *ICDCSW*, 2012.

[12] M. Noormohammadpour *et al.*, "Dccast: Efficient point to multipoint transfers across datacenters," in *HotCloud*, 2017.

[13] L. Luo, H. Yu, Z. Ye, and X. Du, "Online deadline-aware bulk transfer over inter-datacenter wans," in *INFOCOM*, 2018.

[14] N. Mohammad, C. S. Raghavendra, K. Srikanth, and R. Sriram, "Quickcast: Fast and efficient inter-datacenter transfers using forwarding tree," in *INFOCOM*, 2018.

[15] K. Hsieh, A. Harlap, N. Vijaykumar *et al.*, "Gaia: Geo-distributed machine learning approaching lan speeds." in *NSDI*, 2017.

[16] Y. Zhang, J. Jiang, K. Xu *et al.*, "Bds: a centralized near-optimal overlay network for inter-datacenter data replication," in *EuroSys*, 2018.

[17] "Global data centers & colocation services." https://www.equinix.com/locations/.

[18] "The internet topology zoo (cogent)." http://www.topology-zoo.org/files/Cogentco.gml, visited on July 19, 2017.