Proactive Inter-Datacenter Multicast with Realtime and Bulk Transfers

Mahdi Dolati
University of Tehran
Tehran, Iran
University of Calgary
Calgary, Canada
mahdidolati@ut.ac.ir

Majid Ghaderi
University of Calgary
Calgary, Canada
mghaderi@ucalgary.ca

Ahmad Khonsari
University of Tehran
Tehran, Iran
IPM
Tehran, Iran
ak@ipm.ir

ABSTRACT

In content distribution networks, a key objective is the efficient utilization of the network that interconnects geographically distributed datacenters. This is a challenging problem due to vastly different characteristics and requirements of *bulk* and *realtime* transfers that share the interconnection network. Bulk transfers aim at delivering a copy of a usually large file to multiple datacenters before a deadline, while realtime transfers are absolutely delay-intolerant with unsteady and dynamic demands. In this paper, we consider the problem of multicasting deadline-critical bulk transfers in an inter-datacenter network in the presence of unknown and fluctuating demand by realtime transfers. Specifically, we develop a joint admission control and routing algorithm called PMDx, which anticipates future realtime demands and proactively reserves just the right amount of network resources in order to serve future realtime transfers without adversely affecting network utilization or bulk transfer deadlines. We show that the PMDx algorithm is a $2/\delta$-approximation with probability $1-\varepsilon$, and runs in polynomial time proportional to $\ln(1/\varepsilon)/(1-\delta)^2$, for $0 < \delta, \varepsilon < 1$. We also provide extensive model-driven simulation results to study the behaviour of our algorithms in real world network topologies. Our results confirm that PMDx is very close to the optimal, and improves the utilization of the network by 14% compared to a recently proposed algorithm.

CCS CONCEPTS

• **Networks → Network resources allocation**; *Network dynamics*; *Network management*.

KEYWORDS

inter-datacenter networks, bulk transfer, realtime transfer

ACM Reference Format:
Mahdi Dolati, Majid Ghaderi, and Ahmad Khonsari. 2019. Proactive Inter-Datacenter Multicast with Realtime and Bulk Transfers. In *IEEE/ACM International Symposium on Quality of Service (IWQoS '19), June 24–25, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3326285.3329059

IWQoS '19, June 24–25, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6778-3/19/06…$15.00
https://doi.org/10.1145/3326285.3329059

1 INTRODUCTION

1.1 Background and Motivation

Content distribution networks (CDNs) have emerged as the infrastructure of choice for global service delivery over the Internet. In order to offer low-latency and dependable delivery of services, CDN providers deploy several datacenters (DC) in different geographical locations. As such, the network that connects the geo-distributed DCs plays a critical role in the quality of services delivered via a CDN. The performance of many CDN functions such as content distribution, data synchronization, and remote back-up is directly affected by the performance of the underlying inter-DC network. Thus, it is vital for CDN providers to efficiently manage inter-DC network resources (*e.g.*, link bandwidth) in order to respond to ever-increasing demand for their services.

To manage network resources efficiently, it is important to consider the unique requirements and characteristics of data traffic in an inter-DC network (IDCN). Based on a survey by Microsoft [7], bulk data transfers that are *multicast* in nature (*e.g.*, database replication) constitute a sizable portion of traffic in IDCNs. While bulk transfers are delay tolerant, they are typically *deadline-critical*, *i.e.*, their transmission should be completed before a pre-specified deadline. A relatively small fraction of inter-DC traffic is constituted by realtime transfers (*e.g.*, video streams) that are delay sensitive. These transfers have priority over bulk transfers and should be scheduled and transmitted over the IDCN upon arrival.

Achieving high bandwidth utilization in an IDCN requires an efficient *transfer scheduling algorithm* to pack as many bulk transfers in the IDCN as possible, while ensuring their deadlines are satisfied. Designing such a scheduling algorithm is a challenging task due to *unknown* future realtime transfers, and entails addressing the joint problem of admission control and routing of bulk transfers. In an IDCN, the bandwidth demands of realtime transfers are known only at runtime upon the arrival of the realtime transfer requests. This means that, a scheduling algorithm has to schedule deadline-critical bulk transfers without prior knowledge of future realtime transfer demands. If the algorithm tightly schedules many bulk transfers, once a realtime transfer arrives, it may be forced to throttle bulk transfers in order to make room for the arriving realtime transfer, thus jeopardizing the deadlines of the bulk transfers. In general, any algorithm that *reactively* throttles the ongoing bulk transfers in response to changes in realtime transfer demands (*e.g.*, [7]) risks missing bulk transfers deadlines. Ideally, a scheduling algorithm has to anticipate future realtime demands and *proactively* reserve just the right amount of bandwidth for them in order to avoid the need for adjusting bulk transfers on-the-fly.

Most prior works on bulk transfer scheduling in IDCNs are focused on reactive algorithms. These works generally ignore bulk transfer deadlines (*e.g.*, [17]) in order to simplify the scheduling problem. There exist only a few works on proactive transfer scheduling algorithms, where a portion of the IDCN's bandwidth is reserved to accommodate future realtime transfers (*e.g.*, [22]). The challenge in these works is how to determine the optimal amount of reserved bandwidth to avoid over or under reservation. A common approach is to reserve sufficient amount of bandwidth to accommodate the worst-case demands of realtime transfers (*e.g.*, [22]). Clearly, this approach results in under-utilization of the IDCN links. Our objective in this paper is to systematically design a scheduling algorithm that proactively reserves just the right amount of bandwidth for future realtime transfers assuming that realtime demands conform to an *uncertainty set*. Specifically, we consider multicast bulk transfers with deadline and design exact and approximate scheduling algorithms that perform well under general realtime transfer demands (those consistent with the uncertainty set) without requiring frequent re-provisioning of bulk transfers.

1.2 Related Work

A review of existing works that are more related to our work is presented below.

Deadline-oblivious Unicasting. Instead of guaranteeing deadlines, some works consider a fair allocation of bandwidth to ensure an acceptable performance and completion time for bulk transfers. Google's inter-DC method [6], called B4, uses a greedy heuristic to ensure that competing transfers on each link receive a max-min fair share of the link bandwidth. To accommodate high priority flows, B4 uses a reactive mechanism in which the transmission rates of low priority flows are throttled at the application level. Microsoft's SWAN [5] is a reactive method that uses software-defined networking to frequently re-configure IDCN's data plane to match the current traffic demand. A two-stage transfer scheduling algorithm is proposed in [12], where in the first stage, it routes the high priority transfers using 90% of links bandwidth, and then in the second stage, packs the residual network with lower priority transfers that are further tagged for potential dropping during transient loss periods. These works, however, use unicast to transfer bulk data with no native support for multicast.

Deadline-oblivious Multicasting. Multicast is an important communication primitive for efficient data delivery in IDCNs. The most common technique for routing multicast traffic is to construct Steiner trees. Exact algorithms for computing Steiner trees in a network are presented in [11], while approximate algorithms are presented in [15, 21]. Under the assumption that link delays are fixed, a distributed heuristic algorithm is presented in [9] to compute a multicast tree with bounded delay from the source to every destination. The problem of bandwidth allocation when each transfer is allowed to use multiple Steiner trees is investigated in [17], however, only one class of traffic is considered. Considering bulk discount pricing, where the price of inter-DC transfers is reduced as the data volume increases, an ILP-based periodic (*e.g.*, every 5 minutes) traffic re-routing is proposed in [3] to minimize data transfer costs. Multicast routing with end-to-end delay guarantee and constrained link bandwidth is considered in [14], where the Lagrangian relaxation technique is applied to solve the problem. Load-balancing across multicast bulk transfers is considered in [19], where it is assumed that full knowledge about network resources (*e.g.*, available link bandwidth) is available. They compute minimum weight Steiner trees and then schedule each transfer request in a way that it can be finished as early as possible. Some of these works restrict the maximum delay between the multicast source and every destination, which is preferable for multimedia applications but not enough for inter-DC bulk transfers. Nevertheless, none of them provide a *guarantee* to finish a bulk transfer before a given deadline.

Deadline-aware Multicasting. Some works in this category ignore dynamic changes in available network resources (due to realtime transfers) [8, 16, 18]. The authors in [18] design a fast admission control for inter-DC unicast transfers that can guarantee deadlines based on the As-Late-As-Possible policy [13]. In [8], it is argued that it is possible to accommodate more transfers in IDCN by using multiple Steiner trees. Their solution is based on a log-based heuristic to find a sparse solution of the corresponding non-convex optimization problem to keep the number of Steiner trees reasonably low. A unicast-based linear program is developed in [16] to minimize the transfer completion time by allowing some transfer destinations to receive data sooner, and then act as new transmission sources for other destinations. A few works have considered dynamic demand fluctuations of realtime transfers, as it has a major effect on deadlines for bulk transfers [7, 10, 22]. In [7], the realtime traffic rates are estimated using historical usage data, which are then fed to a linear program (LP) that guarantees deadlines. However, because the LP does not consider future demand fluctuations, this approach has to reactively perform adjustments when it becomes necessary. To be able to absorb realtime traffic rate fluctuations, a proactive algorithm is presented in [10], which tries to under-allocate network bandwidth by spreading traffic across paths and time. This approach maximizes the minimal fraction of requests completed before their deadlines, but does not guarantee deadlines. To address estimation errors when estimating realtime traffic demands, the work [22] considers a fixed error margin (*e.g.*, between 5% to 15%) and then performs an adaptive rescheduling periodically in order to reduce the amount of unused (but, reserved) bandwidth.

1.3 Our Work

In this paper, we consider the problem of scheduling deadline-critical bulk multicast transfers in an IDCN, while considering demand fluctuations of realtime transfers. Specifically, a set of bulk multicast transfers are given as input, where serving each transfer provides a certain profit for the IDCN owner. The objective is to *admit* a subset of the given transfers and optimally *route* them such that the total profit is maximized, while the deadline of each admitted transfer is guaranteed regardless of realtime transfers demands. To avoid traffic duplication, we employ Steiner trees to route multicast transfers, and allow routing a data transfer on multiple trees.

We consider a general model for deadlines in which each transfer has two deadlines, namely a soft deadline and a hard deadline. If a transfer is completed before its soft deadline, then the total profit associated with that transfer is received. However, if the transfer

is completed after its hard deadline, the received profit is zero. If the transfer is completed after its soft deadline but before its hard deadline, the received profit decreases linearly from its maximum to zero.

To solve the bulk multicast problem, we develop exact and approximate algorithms that perform well under general service demands without requiring frequent re-provisioning. We assume that some knowledge about future realtime transfer demands is available (specifically, they belong to an uncertainty set) and formulate the problem as a robust optimization problem. A key feature of our algorithms is that they can achieve any desired trade-off between the algorithm proactivity and its profit. They can allocate bandwidth such that the profit is maximized but cause frequent re-provisioning at runtime. Alternatively, they can completely avoid runtime re-provisioning at the cost of decreased profit.

Our main contributions in this paper are summarized below:

- We formulate the proactive bulk multicast problem with deadline as a mixed-integer linear program (MILP), which can be solved for small problem instances.
- To tackle the computational complexity of the MILP formulation, we design an approximate algorithm called PMDx, which attains the approximation ratio $2/\delta$ with probability $1-\varepsilon$, and runs in polynomial time proportional to $\ln(1/\varepsilon)/(1-\delta)^2$, for $0 < \delta, \varepsilon < 1$.
- We present extensive model-driven simulation results to study the behaviour of our algorithms in real world network topologies and demonstrate their ability to achieve any desired proactivity-profit trade-off. We also compare our algorithms against a recently proposed algorithm called Amoeba [22].

1.4 Paper Organization

We start by discussing the system model and assumptions in Section 2. The derivation of our model and proactive algorithms is presented in Section 4. The evaluation results and their analyses are presented in Section 5. Finally, concluding remarks are provided in Section 6.

2 SYSTEM MODEL AND ASSUMPTIONS

2.1 Inter-DC Network

Consider a set of datacenters $\mathcal{S}$ that are connected through a wide-area network. Let $\mathcal{W}$ and $\mathcal{L}$, respectively, denote the set of switches and links in the network. Each link $\ell \in \mathcal{L}$ has bandwidth b_ℓ. Define the set $\mathcal{N} = \mathcal{S} \cup \mathcal{W}$ as the set of all nodes in the network (*i.e.*, datacenters and switches). The system works in a time-slotted fashion where routing and bandwidth allocation during each time slot is fixed and determined independent of other time slots. We assume that the traffic between DCs belongs to one of the two classes [22]: (1) bulk transfers, and (2) realtime transfers, as described below.

2.2 Bulk Transfers

Bulk transfers arrive to the system in batches, where each transfer r in a batch $\mathcal{R}$ is considered to be a multicast data transfer of volume Q_r from a source DC s_r to multiple destination DCs that are specified by the set $\mathcal{D}_r = \{d_r | d_r \in \mathcal{S} \backslash \{s_r\}\}$. Note that, this model supports unicast and broadcast transfer models as well. Each transfer r can start its data transmission after time slot τ_r, and has a soft deadline τ_r^s and a hard deadline τ_r^h (as in [5, 10, 22]). More specifically, when r finishes before, τ_r^s it yields μ_r units of profit, which can be interpreted as the money that the owner of r pays for transferring the data volume Q_r. However, when the completion of r is delayed beyond the time slot τ_r^h, the achieved profit would be 0, and from τ_r^s to τ_r^h the profit degrades linearly. Each bulk transfer in a batch should be admitted or rejected explicitly, where admission implies that the hard deadline of the transfer is guaranteed. The objective is to admit a subset of transfers that yield the maximum profit.

2.3 Realtime Transfers

Realtime transfers have higher priority than bulk transfers and are absolutely delay intolerant. We denote the set of realtime transfers in time slot t by $\mathcal{Y}(t)$, where each transfer $y \in \mathcal{Y}(t)$ is specified by its source s_y, destination d_y, and transmission rate (*i.e.*, bandwidth demand) q_y. In contrast to a bulk transfer whose transmission rate is controlled by the scheduling algorithm, the exact transmission rate of a realtime transfer is not known a priori and exhibits a significant level of fluctuation over time. Thus, it is only reasonable to assume that the actual value of q_y is known up to some estimation error (*e.g.*, within a range). Given this uncertainty about q_y's, our goal is to route realtime transfers and proactively reserve sufficient bandwidth for them to absorb their fluctuations. Note that, due to the higher priority of realtime transfers, they can preempt resources allocated to bulk transfers and prevent them from finishing before their hard deadlines (which is undesirable). On the other hand, it is necessary to minimize the amount of bandwidth that is reserved for realtime transfers to avoid wasting the inter-DC bandwidth.

3 PROBLEM FORMULATION

In this section, we formally formulate the bulk multicast problem introduced in Section 2. We call the problem the *Proactive Multicast with Deadline* (PMD) problem. The mathematical notation used in this section is summarized in Table 1. In Subsection 3.1, we present

Table 1: Important Mathematical Notations.

Inputs	
$[n]$	Set of integers 1 to n
$\mathcal{N}$	Union set of datacenters $\mathcal{S}$ and switches $\mathcal{W}$
$\mathcal{L}$	Set of inter-DC links
$\mathcal{R}$	Set of bulk transfers
$\mathcal{Y}(t)$	Set of realtime transfers at time t
μ_r	Maximum profit of bulk transfer r
b_ℓ	Bandwidth capacity of link $\ell \in \mathcal{L}$
Decision Variables	
a_r	Admittance status of bulk transfer r
f_r	Finish time of bulk transfer r
u_r	Profit gained from bulk transfer r
$p_r^k(t)$	Allocated bandwidth to bulk transfer r at time slot t on the k-th tree

the Linear Program (LP) formulation of the problem under the assumption of known realtime flow rates at the time of scheduling bulk transfers. In Subsection 3.2, we discuss re-routing as an important practical consideration. Then, in Subsection 3.3, we extend the proposed formulation to consider fluctuations in realtime flow rates.

3.1 LP Formulation

The formulation of the problem involves admitting bulk transfers, routing of all traffic in the network, enforcing the bandwidth capacity of IDCN links, and finally deriving the objective function in terms of the system profit. In the following, we elaborate on each part separately.

Routing Bulk Transfers. In this work, we adopt the standard bi-directed cut formulation of Steiner trees for multicast. In each time slot t, we allow each transfer r to split its traffic among K Steiner trees, where each tree spans all the terminal nodes $\{s_r\} \cup \mathcal{D}_r$ and any number of switches in $\mathcal{W}$. We can select K based on the characteristics of traffic to smoothly handle the effects of splitting traffic (*e.g.*, we can set $K = 1$ to avoid traffic splitting completely). We use binary variables $x_\ell^{r,k}(t)$ to indicate whether the link ℓ is in the k-th tree of transfer r at time t. Then, for each bulk transfer r, we compute all subsets $U \subseteq \mathcal{N}$ such that each subset contains at least one of r's destinations and $\mathcal{N} - U$ includes r's source. Let $\mathcal{U}_r$ denote the set of all such subsets:

$$\mathcal{U}_r = \{U | \, U \subseteq \mathcal{N} \backslash \{s_r\} : U \cap \mathcal{D}_r \neq \emptyset\} \ . \tag{1}$$

To compute the multicast trees, it is sufficient to have at least one incoming edge for each $U \in \mathcal{U}_r$ to ensure that the source node is connected to every multicast destination. That is,

$$\sum_{\ell \in \delta^-(U)} x_\ell^{r,k}(t) \geq 1, \quad k \in [K], \forall U \in \mathcal{U}_r, \tag{2}$$

where, $\delta^-(U)$ denotes the set of directed links (v, u) going into subset U, *i.e.*, $v \notin U \wedge u \in U$.

Routing Realtime Transfers. The realtime transfers are unicast flows between DCs, therefore we only enforce the flow conservation constraints to route each of them over a single path. To this end, we define binary variables $z_\ell^y(t)$ to show whether the link ℓ is used by realtime transfer y in time slot t. We have,

$$\sum_{(i,j) \in \mathcal{L}} z_{(i,j)}^y(t) - \sum_{(j,i) \in \mathcal{L}} z_{(j,i)}^y(t) = \begin{cases} 1, & i = s_y \\ -1, & i = d_y \\ 0, & \text{o.w.} \end{cases} \quad \forall t, y, i \tag{3}$$

Link Capacity Constraints. We only need to consider the admitted bulk transfers when enforcing the link capacity constraints. Let the binary variable a_r indicate whether transfer r is admitted. Furthermore, we define continuous variables $p_r^k(t)$ to specify the traffic rate of bulk transfer r on its k-th tree in time slot t. The capacity constraints of the IDCN's links are enforced as follows,

$$\breve{x}_\ell^{r,k}(t) = p_r^k(t) \times a_r \times x_\ell^{r,k}(t), \qquad \forall r, \ell, k, t \tag{4}$$

$$\sum_{\substack{k \in [K], \\ r \in \mathcal{R}}} \breve{x}_\ell^{r,k}(t) + \sum_{y \in \mathcal{Y}(t)} q_y \times z_\ell^y(t) \leq b_\ell, \qquad \forall t, \ell \tag{5}$$

where, $\breve{x}_\ell^{r,k}(t)$ is an auxiliary variable. Constraint (4) ensures that we only consider the bulk transfers that are admitted (*i.e.*, $a_r = 1$), while constraint (5) ensures that the total bandwidth usage on link ℓ is less than or equal to its capacity b_ℓ. Notice that, since a_r, $x_l^{r,k}(t)$, and $z_\ell^y(t)$ are *binary* variables, we can simply linearize the multiplications in constraint (4) to preserve the linearity of the formulation.

Deadline Guarantee. To guarantee that an admitted bulk transfer r can send its entire data (Q_r) before its hard deadline, the sum of its transferred data from its start time slot τ_r until before its hard deadline, *i.e.*, $\tau_r^h - 1$, should be equal to Q_r. Without loss of generality, we assume that the duration of each time slot is 1, thus the amount of data transferred for r in time slot t over the k-th tree is given by $p_r^k(t)$. Then, we have the following constraint on the total amount of data transferred by each bulk transfer:

$$a_r \times Q_r \leq \sum_{t=\tau_r}^{\tau_r^h - 1} \sum_{k \in [K]} p_r^k(t). \qquad \forall r \tag{6}$$

This constraint is also linearizable because a_r is binary.

System Profit. The profit gain of each bulk transfer r depends on its finish time, denoted by f_r. To compute f_r, we define binary variable $I^r(t)$ to indicate whether the transfer r sends data during time slot t, that is,

$$I^r(t) = \begin{cases} 1, & \sum_{k \in [K]} p_r^k(t) > 0, \\ 0, & \text{o.w.} \end{cases} \tag{7}$$

Then, the finish time is computed as the maximum time slot in which $I^r(t)$ is 1, that is,

$$f_r = \underset{t}{\text{argmax}} \left\{ I^r(t) \times t \right\}. \qquad \forall r \tag{8}$$

Finally, given the linearity of the profit from τ_r^s to τ_r^h, we can compute the actual profit gained from serving the bulk transfer r as follows:

$$u_r = \mu_r \Big(1 - \max \Big\{ \frac{f_r - \tau_r^s}{\tau_r^h - \tau_r^s}, 0 \Big\} \Big), \qquad \forall r \tag{9}$$

where, without loss of generality, we can assume that $\tau_r^s < \tau_r^h$. Note that, if the bulk transfer r is completed before its soft deadline (*i.e.*, $f_r \leq \tau_r^s$) the maximum profit is received (*i.e.*, $u_r = \mu_r$), and when the completion time exceeds the hard deadline the received profit is zero. Since, linearizing constraints (7), (8), and (9) is not trivial, we provide their linearized versions in Appendix A. Finally, the objective is to maximize the total profit gained from the set of admitted bulk transfers, which is expressed as:

$$\max_{a_r} \sum_{r \in \mathcal{R}} u_r \times a_r \ . \tag{10}$$

3.2 Traffic Re-routing

The formulation in Subsection 3.1 allows using a completely different set of K Steiner trees in each time slot. However, changing the routing configuration frequently can disturb the operation of the network (*e.g.*, due to packet re-ordering and delays involved when updating the internal memory of switches [20]). Therefore, we include a mechanism to limit the changes in the set of the Steiner trees used by each bulk transfer. Let the non-negative variable

$w^{r,k}(t)$ indicate if the k-th Steiner tree of bulk transfer r changes from time slot $t-1$ to t. We have the following constraint,

$$w^{r,k}(t) \geq x_\ell^{r,k}(t) - x_\ell^{r,k}(t-1). \quad \forall r,k,t,\ell \tag{11}$$

The value of $w^{r,k}(t)$ in time slot t is 1 if there is a new link in the k-th tree that was not used in the previous time slot. Then, by summing over the life-time of a bulk transfer, we can limit the number of re-routings to a pre-defined constant T, as follows,

$$\sum_{t=\tau}^{\tau_r^h-1} \sum_{k\in[K]} w^{r,k}(t) \leq T. \qquad \forall r \tag{12}$$

3.3 Proactive LP Formulation

An important assumption of the optimization program presented in Subsection 3.1 is that the actual transmission rates (*i.e.*, bandwidth demands) of realtime transfers are known at the time of scheduling bulk transfers. Previous studies, however, show that this is not a realistic assumption as the realtime transfers are bursty and hard to predict accurately [22]. Therefore, it is more realistic to model the transmission rates of realtime transfers (*i.e.*, q_y) as random variables instead of considering them as deterministic input values. If the distribution of transmission rates is fully known, we could re-write the LP of Subsection 3.1 as a stochastic optimization program and compute a transfer schedule that preserves the bulk transfer deadlines with some desired probability. However, assuming full knowledge of the distribution of transmission rates is generally infeasible in real-world applications. A practical alternative is to assume that we only have limited statistical information about the rates of realtime transfers. Specifically, we assume that the transmission rate of realtime transfer q_y belongs to an *uncertainty set*, which is represented by an interval, *i.e.*,

$$q_y \in [\bar{q}_y - \hat{q}_y, \bar{q}_y + \hat{q}_y],$$

where, $\bar{q}_y$ is the average rate and $\hat{q}_y$ is the maximum rate fluctuation. One can also state the value of $\hat{q}_y$ as the fraction of average rate, *i.e.*, $\hat{q}_y = \gamma \bar{q}_y$, where $\gamma \geq 0$. With no further knowledge about realization of q_y variables over time, the only option for guaranteeing the deadlines is to consider the worst-case scenario (*i.e.*, $q_y = (1+\gamma)\bar{q}_y$), which would result in significant underutilization of IDCN links. However, despite the fact that q_y's are independent random variables, it is unlikely that all realtime transfers deviate *maximally* from their average rates simultaneously. To this end, we assume that at most Γ of the realtime transfers can maximally deviate from their average rates, and re-write the constraint (5) as follows:

$$\sum_{\substack{k\in[K],\\ r\in\mathcal{R}}} \breve{x}_\ell^{r,k}(t) + \sum_{y\in\mathcal{Y}(t)} \bar{q}_y \times z_\ell^y(t) + \max_{\substack{\mathcal{V}(t)\subseteq\mathcal{Y}(t)\\ |\mathcal{V}(t)|\leq\Gamma}} \sum_{y\in\mathcal{V}(t)} \gamma\bar{q}_y \times z_\ell^y(t) \leq b_\ell, \quad \forall t,\ell \tag{13}$$

This constraint states that in each time slot t and on each link ℓ at most Γ realtime transfers can maximally deviate from their rates, and regardless of which Γ transfers deviate, the link capacity constraints are respected. With this approach, throttling bulk transfers to deal with a sudden surge in realtime traffic is completely avoided. Notice that, setting $\Gamma = 0$ implies that the estimates (*i.e.*, $\bar{q}_y$) are perfectly accurate, while $\Gamma = |\mathcal{Y}(t)|$ reduces this approach to the worst-case approach which allocates the maximum (*i.e.*, worst-case) bandwidth requirement to each transfer. Intermediate values of Γ provide a wide range of resource allocation options, where each option results in a different trade-off between guaranteeing the bulk deadlines and optimal utilization of IDCN links. Specifically, assume that the probability that a realtime transfer hits its maximum rate is π. Assume the goal is to guarantee, with probability of at least $1-\epsilon$, that a transfer schedule is feasible under any realtime demand fluctuation without throttling bulk transfers. Under the assumption of $\gamma \leq 1$, which is usually the case, the fluctuation interval would be symmetrical and the optimal value of Γ, denoted by Γ^*, can be determined as follows,

$$\Gamma^* = \arg\min_{\Gamma} \sum_{i=\Gamma+1}^{|\mathcal{Y}(t)|} e^{-\frac{\Gamma^2}{2i}} \times \binom{|\mathcal{Y}(t)|}{i} \pi^i (1-\pi)^{|\mathcal{Y}(t)|-i} \leq \epsilon \tag{14}$$

where the term $e^{-\frac{\Gamma^2}{2i}}$ is the probability that the link capacity constraint (13) is violated if $i > \Gamma$ realtime transfers deviate from their average rates [1].

The next step is to replace constraint (5) with constraint (13), which is non-linear due to the max operator. To preserve the linearity of the model, using the theory of robust optimization [1], we extract the non-linear term from (13) and re-write it as a separate linear program, as follows.

$$\max \sum_{y\in\mathcal{Y}(t)} \gamma q_y \times z_\ell^y(t) \times v_\ell^y(t), \tag{16}$$

$$\text{s.t.} \sum_{y\in\mathcal{Y}(t)} v_\ell^y(t) \leq \Gamma, \tag{16a}$$

$$0 \leq v_\ell^y(t) \leq 1, \tag{16b}$$

where, we introduce the new variable $v_\ell^y(t)$ to indicate if the realtime transfer y maximally deviates on link ℓ in time slot t. Then, constraint (16a) restricts the number of deviating transfers to Γ. The next step in the process of linearization involves computing the dual of linear program (16), which is also a linear program. Define the dual variables $\lambda_\ell(t)$ and $\phi_\ell^y(t)$, respectively, corresponding to the constraints (16a) and (16b). The dual of (16) is then expressed

Program 1 PMD: Proactive Multicasting with Deadline

$$\text{Max.} \quad \sum_{r\in\mathcal{R}} u_r \times a_r \tag{15}$$

$$\text{s.t.} \quad \sum_{\substack{k\in[K],\\ r\in\mathcal{R}}} \breve{x}_\ell^{r,k}(t) + \sum_{y\in\mathcal{Y}(t)} q_y \times z_\ell^y(t) + \lambda_\ell(t)\Gamma + \sum_{y\in\mathcal{Y}(t)} \phi_\ell^y(t) \leq b_\ell \tag{15a}$$

$$\phi_\ell^y(t) + \lambda_\ell(t) \geq \gamma q_y \times z_\ell^y(t) \tag{15b}$$

$$\lambda_\ell(t),\ \phi_\ell^y(t) \geq 0 \tag{15c}$$

(2), (3), (4), (6), (7), (8), (9), (11), (12)

as,

$$\min \lambda_\ell(t)\Gamma + \sum_{y \in \mathcal{Y}(t)} \phi_\ell^y(t), \tag{17}$$

$$\text{s.t. } \phi_\ell^y(t) + \lambda_\ell(t) \geq \gamma q_y \times z_\ell^y(t), \tag{17a}$$

$$\lambda_\ell(t), \phi_\ell^y(t) \geq 0\,. \tag{17b}$$

Finally, we can substitute the objective of (17) for the non-linear term in the constraint (13). The complete formulation of PMD is presented in Program 1.

4 POLYNOMIAL-TIME SOLUTION

The formulation in Program 1 is an Integer Linear Program (ILP), which is generally NP-hard. It means that computing the optimal solution even for a moderate-size instance of the problem using standard techniques such as Branch and Bound and Cutting Plane incurs a prohibitively long time. Our goal, is to design an approximate solution for the problem, which runs in polynomial time. Specifically, we develop an algorithm to solve the problem using the randomized rounding technique. This technique has two phases: The first phase (*i.e.*, relaxation) involves building a linear program (LP) by removing the integrality constraints of the integer linear program. In the second phase, a feasible solution to the original ILP is constructed by rounding, in a randomized fashion, the solution of the LP that is fractional but computable in polynomial time.

Another technical difficulty in designing a polynomial-time solution is the exponentially many constraints for computing the Steiner trees (*i.e.*, constraint (2)). Although we can incorporate existing LP approximation approaches (like [2]) in our solution, previous studies have demonstrated that it is possible to achieve good performance by using only a limited number of pre-computed trees to serve the bulk requests [8]. Thus, we assume that each bulk transfer request is restricted to be served with a pre-computed set of Steiner trees and present our LP-rounding-based algorithm called PMDx to approximate PMD.

4.1 Algorithm Description

The PMDx algorithm, presented in Algorithm 1, receives a PMD instance along with a performance-tuning parameters, *i.e.*, $\mathcal{K}$ which will be explained in section 4.3, and produces a transfer schedule as output. It starts by relaxing the integrality constraints on $z_\ell^y(t)$ and a_r. Notice that, if we find integer values for $z_\ell^y(t)$ and a_r that are feasible in the PMD model, it is possible to compute an integer value for the variable $I^r(t)$ based on the value of the variable $p^{r,k}(t)$ without affecting the feasibility of the solution. Thus, we do not consider $I^r(t)$ when rounding and fixing other variables. The rounding happens in two steps. In the first step, the routing of realtime transfers is computed (RouteRealTime in Algorithm 2). In the second step, we consider bulk transfers one-by-one and determine their admission and resource allocation decision variables (RouteBulk in Algorithm 3). The second step is repeated $\mathcal{K}$ times to find a provably good solution with high probability.

RouteRealTime. After relaxation, the fractional values of variables $z_\ell^y(t)$ are interpreted as the probability of using the link ℓ to construct the path of realtime transfer y at time t. Thus, we approximate the optimal path of realtime transfer y by the path

Algorithm 1 PMDx: Approximate PMD

Input: PMD model, $\mathcal{K}$ ▷ Described in Problem 1
Output: Approximate solution for PMD

1: $\widetilde{\text{PMD}} \leftarrow$ relax integer constraints of PMD
2: RouteRealTime($\widetilde{\text{PMD}}$)
3: Run RouteBulk($\widetilde{\text{PMD}}$) $\mathcal{K}$ times and return the best result

that maximizes the minimum probability associated with its links. Specifically, the path of transfer y at time t, denoted by $\rho_y(t)$, is given by:

$$\rho_y(t) = \underset{p \in \mathcal{P}_y}{\text{argmax}} \left\{ \min_{\ell \in p} z_\ell^y(t) \right\}, \tag{18}$$

where, $\mathcal{P}_y$ is the set of all paths from s_y to d_y. To this end, for each flow y at time t, we assign the weight $z_\ell^y(t)$ to link ℓ in the IDCN and use *modified* Dijkstra's algorithm (see lines 3 to 9) to find the path whose bottleneck link has the highest weight. The Dijkstra's algorithm starts from node s_y and uses a priority queue to find the path to d_y. Each element in the priority queue has three components: 1) a node id, 2) a path prefix that can connect s_y to that node, and 3) the minimum $z_\ell^y(t)$ along that path prefix. The priority queue selects the element with the highest third component, which consequently maximizes the minimum $z_\ell^y(t)$. Then, the variables $z_\ell^y(t)$ are fixed to be 1 for all links in $\rho_y(t)$.

Algorithm 2 RouteRealTime

Input: $\widetilde{\text{PMD}}$
Output: Routing of realtime transfers

1: Solve $\widetilde{\text{PMD}}$
2: **for** each realtime flow y **do**
3: q ← PriorityQueue((s_y, [], 1))
4: **while** $c \neq d_y$ **do**
5: c, e, u ← q.pop()
6: **for** $n \notin \rho_y$ neighbour of c **do**
7: q.push((n, e.append(c), min{u, $z^y_{(c,n)}(t)$}))
8: **end for**
9: **end while**
10: ρ_y ← e.append(c)
11: Fix $z_\ell^y(t) = 1$ for all $\ell \in \rho_y$
12: **end for**

RouteBulk. Next, we iteratively process bulk transfers one-by-one, where in each iteration a single bulk transfer is admitted or rejected. Notice that, the LP solver may admit many bulk requests partially (*i.e.*, $0 < a_r < 1$) in order to maximize the system profit, which means that it may violate some deadlines. Specifically, if $a_r < 1$, the bulk transfer r can only deliver a data volume of size $a_r Q_r$, see constraint (6), and hence can not finish before its corresponding deadline. Therefore, we choose some requests and explicitly reject them (*i.e.*, set $a_r = 0$) such that the available bandwidth becomes sufficient to serve other requests. This step is also performed iteratively because when we set $0 < a_r < 1$ to 0 for a transfer r, it may be possible to round $0 < a_{r'} < 1$ to 1 for some other transfer r'. To select a request to reject when the available bandwidth is not sufficient to serve all requests, we interpret a_r's as probabilities and reject a request with probability proportional to $1 - a_r$. To keep track of the requests $\mathcal{A}_r$, a set with all admission variables a_r is created (see line 1). A request is removed from $\mathcal{A}_r$

when it is admitted or rejected. The algorithm considers the transfers iteratively and rounds them using the randomized rounding technique. That is, in each iteration, the algorithm selects the transfer with the highest value of $|a_r - 0.5|$, namely m, and admits it with the probability of a_m, or reject it with probability $1 - a_m$ (see line 4 where `rand()` is a function that generates a random number in $[0, 1]$). Note that, a larger value of $|a_r - 0.5|$ means that a_m is either closer to 1 or 0, which means that we can admit or reject it more confidently (all transfers whose admission decision variable a_r is equal to 1 or 0 will be fixed first.). There is a possibility that there is not enough bandwidth capacity to serve a_m, which would make the model infeasible. Thus, we check the feasibility of the model and set $a_m = 0$ if the model becomes infeasible (see lines 5 to 9).

Algorithm 3 RouteBulk

Input: $\widetilde{\text{PMD}}$
Output: Routing of bulk transfers
1: $\mathcal{A}_r = \{a_r \mid r \in \mathcal{R}\}$
2: **while** $\mathcal{A}_r$ is not empty **do**
3: $a_m \leftarrow \text{argmax}_{a_r \in \mathcal{A}_r} |a_r - 0.5|$
4: **if** `rand()` $< a_m$ **then**
5: Set $a_m = 1$
6: Solve $\widetilde{\text{PMD}}$
7: **if** rPMDis infeasible **then**
8: Set $a_m = 0$
9: **end if**
10: **else**
11: Set $a_m = 0$
12: **end if**
13: Remove a_m from $\mathcal{A}_r$
14: Solve $\widetilde{\text{PMD}}$
15: **end while**

4.2 Runtime Analysis

Define $\widetilde{T}(n)$ to be the complexity of solving an LP with n decision variables. RouteRealTime solves the relaxed PMD once and then runs the Dijkstra's algorithm once for each transfer. The time it takes to solve the relaxed problem (which is LP) dominates the running time of the algorithm, thus the complexity of RouteRealTime is $O(\widetilde{T}(n))$. RouteBulk iterates over the set of bulk transfers and solves two LPs in each iteration resulting in time complexity $T(n, \mathcal{R}) = O(2\,|\mathcal{R}|\,\widetilde{T}(n))$. Therefore, the complexity of PMDx would be $O(\mathcal{K}\,T(n, \mathcal{R}))$. Note that the number of variables and constraints in our model is polynomial in the size of the problem (recall that Steiner trees are pre-computed) which means that $\widetilde{T}(n)$ is polynomial in n. For example, using the interior method, we get $\widetilde{T}(n) = O(n^{3.5})$. As will be shown in Section 5, in practice, PMDx runs very fast because in each iteration, it solves an LP which is slightly different from the one solved in the previous iteration.

4.3 Approximation Analysis

The PMDx algorithm computes an approximate solution for the original integer problem PMD. Our goal is to find the approximation factor of PMDx, to bound its performance deviation from the optimal integer solution. Let Φ_{OPT} and Φ_{PMDx}, denote the total profit achieved by the optimal integer program PMD and approximate algorithm PMDx, respectively. We say PMDx is a $(1/\delta)$-approximation algorithm if $\delta\Phi_{\text{OPT}} \le \Phi_{\text{PMDx}} \le \Phi_{\text{OPT}}$, for some $0 < \delta < 1$.

To this end, we first start by analyzing PMDx under the assumption that $\tau_r^s = \tau_r^h - 1$. After that, we analyze the effect of this assumption on the solution computed by PMDx.

Part I. Let $\widetilde{\Phi}$ denote the total profit achieved by the linear program that results from relaxing the integrality constraints in PMD (without rounding). We have,

$$\widetilde{\Phi} = \sum_{r \in \mathcal{R}} \mu_r \widetilde{a}_r \,. \tag{19}$$

Observation 1. *If the original integer problem is feasible, i.e., at least one of the bulk transfers in $\mathcal{R}$ can be routed, we have $\widetilde{\Phi} \ge \min_{r \in \mathcal{R}} \mu_r > 0$. In general, if the cut capacity of the IDCN is Q, then $\widetilde{\Phi} \ge \frac{Q}{\sum_r Q_r} \sum_r \mu_r$. By the definition, we also have $\widetilde{\Phi} \le \sum_r \mu_r$.*

Observation 2. *Any solution of the integer problem is also a solution of the relaxed problem, thus $\Phi_{\text{OPT}} \le \widetilde{\Phi}$.*

Consequently, in order to show that $\delta\Phi_{\text{OPT}} \le \Phi_{\text{PMDx}}$, it is sufficient to show that $\delta\widetilde{\Phi} \le \Phi_{\text{PMDx}}$. To this end, define the random variable Z_r (corresponding to bulk transfer $r \in \mathcal{R}$), as follows,

$$Z_r = \begin{cases} \mu_r, & \text{with probability } \widetilde{a}_r, \\ 0, & \text{with probability } 1 - \widetilde{a}_r \,. \end{cases}$$

We have, $\Phi_{\text{PMDx}} = \sum_{r \in \mathcal{R}} Z_r$. That is, Φ_{PMDx} is itself a random variable given by the sum of random variables Z_r. It then follows that,

$$\mathbb{E}\left[\Phi_{\text{PMDx}}\right] = \sum_{r \in \mathcal{R}} \mathbb{E}\left[Z_r\right] = \sum_{r \in \mathcal{R}} \mu_r \widetilde{a}_r \,. \tag{20}$$

Without loss of generality, we assume that profits μ_r are normalized so that $0 < Z_r < 1$. We then apply the Chernoff bound to random variable Φ_{PMDx}, which yields,

$$\mathbb{P}\left\{\delta\widetilde{\Phi} \le \Phi_{\text{PMDx}}\right\} \ge 1 - e^{-\frac{\widetilde{\Phi}(1-\delta)^2}{2}} \ge 1 - e^{-\frac{c_0(1-\delta)^2}{2}}, \tag{21}$$

where, $c_0 = \frac{Q}{\sum_r Q_r} \sum_r \mu_r > 0$ is a constant dependent on the structure of the IDCN and problem instance, as described in Observation 2. By running the algorithm multiple times with independent randomized rounding in each run, we can make the probability $\mathbb{P}\left\{\delta\widetilde{\Phi} \le \Phi_{\text{PMDx}}\right\}$ arbitrarily close to 1. Specifically, to achieve the approximation ratio $1/\delta$ with probability of $1 - \varepsilon$, for $0 < \varepsilon < 1$, the required number of runs is given by $\mathcal{K} = \frac{2\ln(1/\varepsilon)}{c_0(1-\delta)^2}$.

Part II. Next, we investigate the effect of assuming $\tau_r^s = \tau_r^h - 1$. Those transfers that finish before their soft deadlines or after their heard deadlines are not affected by this assumption. Thus, we focus on those transfers whose finish time falls within the interval $[\tau_r^s + 1, \tau_r^h - 1]$ and assume that actual finish time is uniformly distributed in this interval. A geometric representation of the problem is illustrated in Fig. 1. The shaded area in the figure shows the additional profit accumulated by Φ_{PMDx}. Clearly, this area is at most equal to the non-shaded area, which is the profit that would have been accumulated without the assumption $\tau_r^s = \tau_r^h - 1$. Therefore,

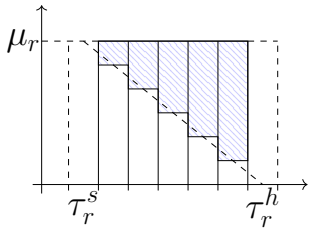


Figure 1: Additional profit due to the assumption $\tau_r^s = \tau_r^h - 1$.

the total profit considered in the analysis of PMDx (*i.e.*, Φ_{PMDx}) is overestimated by at most a factor of 2 due to this assumption.

Theorem 1. *The* PMDx *algorithm runs in* $O\big(\frac{2\ln(1/\varepsilon)}{c_0(1-\delta)^2} \cdot T(n,\mathcal{R})\big)$ *time and computes a* $\frac{2}{\delta}$*-approximate solution for the* PMD *problem with probability* $1-\varepsilon$*, for any* $0 < \varepsilon, \delta < 1$.

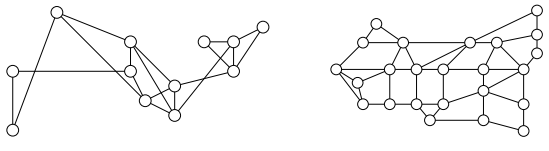
Proof. The proof follows from the analyzes presented in subsection 4.2 as well as Part I and Part II. □

5 PERFORMANCE EVALUATION

In this section, we use simulations to evaluate the performance of the PMDx algorithm in terms of profit, proactivity, and scalability. For comparison, we choose **Amoeba** [22], which is recently proposed and is comparable to our algorithm in the sense that it also considers hard and soft deadlines. However, Amoeba uses unicast to implement multicast which is not bandwidth efficient. To have a fair comparison, we modified Amoeba to use the same set of Steiner trees as those used in PMDx. We also present the optimal results obtained by solving the PMD model using Gurobi [4]. All algorithms are implemented in Python 2.7 and the computations are carried out on a computer with an Intel® Core™ i7-4790 processor at 3.60 GHz and 8 GB of RAM. Throughout the experiments, we set $\mathcal{K} = 1$, *i.e.*, we do not target any guaranteed δ/ε.

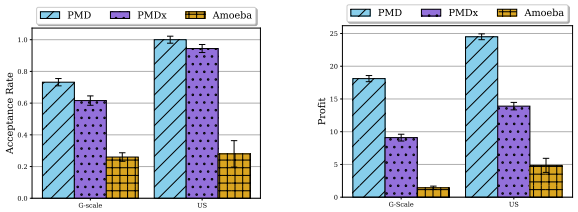
5.1 Simulation Parameters

We use two real-world network topologies in our evaluations: USA (24 nodes and 43 links) and G-Scale [6] (12 nodes and 19 links) (see Fig. 2). The link bandwidths are sampled from a uniform distribution in the range $[0.7, 1]$ Gbps. Each bulk transfer request desires to send 1 to 3 Gbits of data to 2 to 5 destinations and the profit of all requests are assumed to be 1 (*i.e.*, $\mu_r = 1$). The number of pre-computed Steiner trees for routing each request, *i.e.*, K, is set to 10. In each time slot, there are 50 realtime transfers in the IDCN, where their rates are randomly selected from the range $[1, 3]$ Mbps. The results reported in this section are obtained by averaging over 10 runs, where each run simulates 14 timeslots. The values of τ_r, τ_r^s, and τ_r^h are selected randomly from $[1, 14]$ with the constraint $\tau_r < \tau_r^s < \tau_r^h$.



(a) G-scale network [6]. (b) USA network.

Figure 2: Network topologies used for evaluation.



(a) **Acceptance Rate.** (b) **Profit.**

Figure 3: Comparison of acceptance rate and profit.

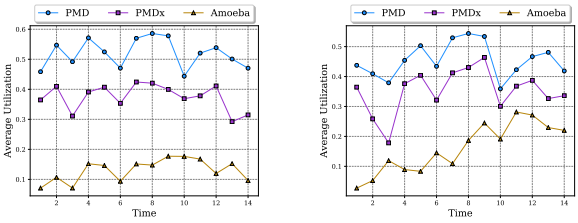
5.2 Performance Metrics

The following performance metrics are considered:

- **Acceptance Rate:** The fraction of bulk transfers that are admitted with a guaranteed deadline.
- **Profit:** The average profit over all bulk transfer batches that arrive to the system.
- **Average Utilization:** The average utilization of all links used to serve data transfers in a time slot.
- **Maximum Utilization:** The maximum utilization among all links used during the life-time of the system.
- **Finish Time:** The average completion time of all admitted transfers in a batch.
- **Runtime:** The average time it takes to schedule a batch.

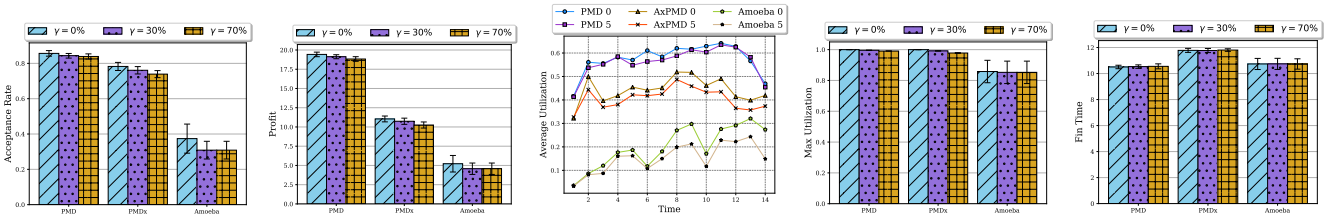
5.3 Full-knowledge Scenario

In this section, we assume that the bulk transfers are given to the algorithms in batches of size 25 and the full knowledge about the rate of realtime transfers is available to them. Figs. 3(a) and 3(b), respectively, show the acceptance rate and profit of different algorithms. We see that the capacity of the G-Scale network is not sufficient to serve all requests, and thus about 25% of the requests are rejected. The performance of PMDx is very close to PMD, where it only rejects about 3 more requests. However, Amoeba performs poorly and admits about 73% less requests compared to PMDx. The capacity of the US network is larger, where PMD and PMDx mange to admit almost all the requests, while Amoeba still achieves a low acceptance rate. We also observe that PMDx significantly outperforms Amoeba in terms of profit. Specifically, PMDx achieves 56% of the maximum profit obtained by PMD, while Amoeba can only achieve about 18% of that.



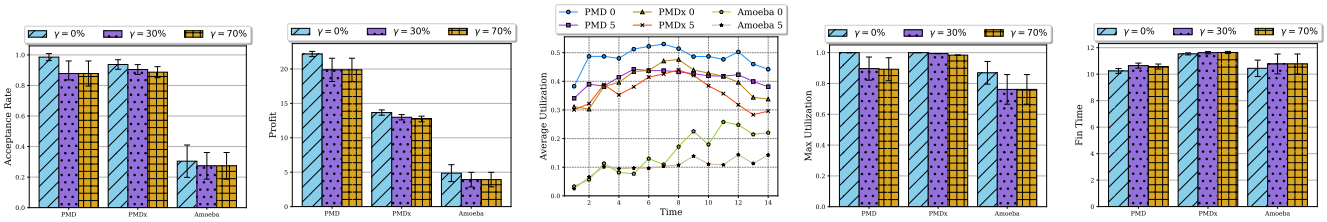
(a) **G-Scale network.** (b) **USA network.**

Figure 4: Average link utilization over time.



(a) **Acceptance Rate.** (b) **Profit.** (c) **Avg. link utilization.** (d) **Max. link utilization.** (e) **Finish time.**

Figure 5: G-Scale network: Comparison under partial-knowledge.



(a) **Acceptance Rate.** (b) **Profit.** (c) **Avg. link utilization.** (d) **Max. link utilization.** (e) **Finish time.**

Figure 6: USA network: Comparison under partial-knowledge.

Fig. 4 compares the average link utilization under each algorithm. We observe that PMDx closely follows the behavior of PMD, and achieves a utilization that is only 10% lower. Amoeba, on the other hand, is not able to efficiently schedule bulk transfers across time slots and thus rejects many requests until it can serve them in a deadline guaranteed manner. This leads to an under-utilized network with the average utilization of at most 30%.

5.4 Partial-knowledge Scenario

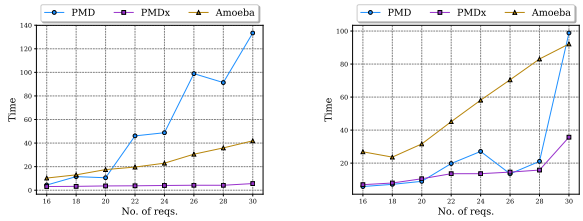
In this section, we evaluate the effect of proactive resource allocation on the performance of algorithms. We let the size of the transfer batches vary from 16 to 30. The rate of each realtime transfer is specified by its average, $\bar{q}_{y_t}$, and maximum deviation percentage, γ. We present the results for $\gamma = 30\%$ and $\gamma = 70\%$. Throughout the experiments, Γ is set to be 0 or 5. Amoeba, has a static proactive policy in which it assumes 5% error for the near future predictions and 15% error for the rest of traffic estimates. Our algorithms, on the other hand, compute a suitable headroom for each link based on the values of Γ and γ. The results obtained in G-Scale and USA networks are similar. As such, we present the results for both networks, but only discuss the results for the G-Scale network.

Fig. 5(a) shows that when the demand fluctuation increases, all algorithms admit fewer bulk requests to ensure the deadlines are met. Similarly, in Fig. 5(b), it can be seen that all algorithms achieve lower profit as they reserve more bandwidth to accommodate unforeseen fluctuations in realtime transmission rates. We observe that PMDx and PMD can tolerate high levels of estimation errors (up to 70%) for realtime rates, while only losing less than 7% of the total profit. Amoeba, however, aggressively reserves bandwidth, which leads to reduced acceptance rate and profit by 17% and 12%, respectively. The effect of proactive resource allocation is clearly observed in the average link utilization results depicted in Fig. 4. We see that Amoeba has the highest bandwidth waste for handling demand fluctuations of realtime transfers (about 23%). In comparison, PMDx only reserves about 8% of bandwidth to fully accommodate realtime transfers. The maximum link utilization is shown in Fig. 5(d). Notice that both PMD and PMDx achieve just under 100% link utilization even in the presence of demand fluctuations. In contrast, the maximum link utilization of Amoeba is about 86%. Finally, Fig. 5(e) presents the average finish time for each algorithm. PMD admits more transfers and has the smallest average finish time, which is the reason for its high overall profit. The finish time of Amoeba is smaller than that achieved by PMDx, however, it is mainly due to the fact that Amoeba accepts substantially lower number of bulk transfers.

5.5 Scalability

We measure the runtime of the algorithms when the size of request batches increases from 16 to 30. Fig. 7(a) shows that PMDx solves each batch, regardless of its size, in less than 6 seconds, which is significantly faster than other algorithms. PMD and Amoeba, not only take longer times to solve each batch, but also show a sharper increase in runtime as the size of the batch increases. This behavior implies that they are less scalable compared to PMDx. Specifically, Amoeba's runtime increases *linearly* from 10 to 41, while that of PMD exhibits an exponential growth (due to the NP-hardness of the problem) where it creases from 6 to 133 seconds.

An interesting behavior can be observed in Fig. 7(b), where Amoeba takes more time than PMD. The reason is that Amoeba creates a temporal-spatial graph which can become very large as the size of the network or the number of time slots increases. However, because the runtime of PMD is exponential, eventually PMD takes more time than Amoeba (*e.g.*, when handling batches of size 30).



(a) **G-Scale network.** (b) **USA network.**

Figure 7: Runtime for different batch sizes.

In all cases, PMDx is significantly faster than PMD and Amoeba by at least 50%.

6 CONCLUSION

In this paper, we considered the problem of joint admission control and routing of bulk transfers in an inter-DC network. We proposed a polynomial-time algorithm, called PMDx, to admit a set of bulk transfers that maximizes the cumulative profit of the network, while guaranteeing their deadlines in the presence of real-time transfers with unknown fluctuating demands. We analyzed the theoretical behaviour of PMDx and extensively evaluated its practical performance against the optimal and a recently proposed approach. An interesting direction for future research is to extend our multicast model to the scenario where each multicast destination that receives a full copy of the data can become a source for other receivers.

ACKNOWLEDGMENTS

This work was supported by Alberta Innovates and by Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Dimitris Bertsimas and Melvyn Sim. 2004. The Price of Robustness. *Oper. Res.* 52, 1 (2004), 35–53.
[2] Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoss, and Laura Sanità. 2013. Steiner Tree Approximation via Iterative Randomized Rounding. *J. ACM* 60, 1 (2013), 6:1–6:33.
[3] José L. Garcia-Dorado and Sanjay G. Rao. 2018. Cost-aware Multi Data-Center Bulk Transfers in the Cloud from a Customer-Side Perspective. *IEEE Trans. Cloud Comput.* (2018), 1–1.
[4] LLC Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. (2018). http://www.gurobi.com
[5] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, et al. 2013. Achieving High Utilization with Software-driven WAN. In *Proc. ACM SIGCOMM*. 15–26.
[6] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, et al. 2013. B4: Experience with a Globally-deployed Software Defined WAN. In *Proc. ACM SIGCOMM*. 3–14.
[7] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, et al. 2016. Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers. In *Proc. ACM SIGCOMM*.
[8] Siqi Ji, Shuhao Liu, and Baochun Li. 2018. Deadline-Aware Scheduling and Routing for Inter-Datacenter Multicast Transfers. In *Proc. IEEE IC2E*. 124–133.
[9] Xiaohua Jia. 1998. A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. *IEEE/ACM Trans. Netw.* 6, 6 (1998), 828–837.
[10] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana R. Babbula. 2014. Calendaring for Wide Area Networks. In *Proc. ACM SIGCOMM*. 515–526.
[11] Vachaspathi P. Kompella, Joseph C. Pasquale, and George C. Polyzos. 1993. Multicast routing for multimedia communication. *IEEE/ACM Trans. Netw.* 1, 3 (1993), 286–292.
[12] Alok Kumar, Sushant Jain, Uday Naik, Nikhil Kasinadhuni, et al. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proc. ACM SIGCOMM*.
[13] Ke Y. Li and Robert J. Willis. 1992. An iterative scheduling technique for resource-constrained project scheduling. *Eur. J. Oper. Res.* 56, 3 (1992), 370 – 379.
[14] Xiaohui Li, Xiaohui Li, Yu-Chu Tian, Gerard Ledwich, et al. 2018. Constrained Optimization of Multicast Routing for Wide Area Control of Smart Grid. *IEEE Trans. Smart Grid* (2018), 1–1.
[15] Dean H. Lorenz, Ariel Orda, Danny Raz, and Yuval Shavitt. 2006. Efficient QoS Partition and Routing of Unicast and Multicast. *IEEE/ACM Trans. Netw.* 14, 6 (2006), 1336–1347.
[16] Long Luo, Hongfang Yu, and Zilong Ye. 2018. Deadline-Guaranteed Point-to-Multipoint Bulk Transfers in Inter-Datacenter Networks. In *Proc. IEEE ICC*. 1–6.
[17] Mohammad Noormohammadpour, Cauligi S. Raghavendra, Srikanth Kandula, and Sriram Rao. 2018. QuickCast: Fast and Efficient Inter-Datacenter Transfers Using Forwarding Tree Cohorts. In *Proc. IEEE INFOCOM*. 225–233.
[18] Mohammad Noormohammadpour, Cauligi S. Raghavendra, and Sriram Rao. 2016. DCRoute: Speeding up Inter-Datacenter Traffic Allocation while Guaranteeing Deadlines. In *Proc. IEEE HiPC*. 82–90.
[19] Mohammad Noormohammadpour, Cauligi S. Raghavendra, Sriram Rao, and Srikanth Kandula. 2017. DCCast: Efficient Point to Multipoint Transfers Across Datacenters. In *Proc. USENIX HotCloud*.
[20] Mark Reitblatt, Nate Foster, Jennifer Rexford, and David Walker. 2011. Consistent Updates for Software-defined Networks: Change You Can Believe in!. In *Proc. ACM HotNets*. 7:1–7:6.
[21] Hussein F. Salama, Douglas S. Reeves, and Yannis Viniotis. 1997. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. *IEEE J. Sel. Areas Commun.* 15, 3 (1997), 332–345.
[22] Hong Zhang, Kai Chen, Wei Bai, Dongsu Han, et al. 2017. Guaranteeing Deadlines for Inter-Data Center Transfers. *IEEE/ACM Trans. Netw.* 25, 1 (2017), 579–595.

A LINEARIZATIONS

We use the Big-M method to linearize constraint (7) as:

$$I^r(t) \geq \frac{\sum_{k\in[K]} p^{r,k}(t)}{M},$$

where, M is a constant that satisfies $M \geq \sum_{k\in[K]} p^{r,k}(t)$, for all requests and trees in all time slots. When $\sum_{k\in[K]} p^{r,k}(t)$ is 0 the solver automatically sets $I^r(t)$ to be 0, if possible, to reduce the objective of the program, thus it is not necessary to enforce $I^r(t)$ to be 0 in such a situation.

To linearize constraint (8), we can write:

$$f_r \geq I^r(t) \times t, \tag{22}$$

where, f_r is greater than any time slot t that its corresponding $I^r(t)$ is 1. The multiplication is linearizable because $I^r(t)$ is binary. Again, because having lower finish times aligns with the objective of the optimization program, we do not need to enforce the equality in (22).

To linearize constraint (9), the max operator must be removed. However, without the max operator, when f_r is less than τ_r^s, the value of u_r becomes greater than μ_r. This problem is solved by adding additional (upper-bound) constraint $u_r \leq \mu_r$, for all requests. Then the linearized version of the constraint can be expressed as:

$$u_r \leq \mu_r\Big(1 - \frac{f_r - \tau_r^s}{\tau_r^h - \tau_r^s}\Big). \tag{23}$$

Once again, we do not need to enforce the equality because increasing u_r is aligned with the objective of the optimization program.