Cost-minimizing Bandwidth Guarantee for Inter-datacenter Traffic

Wenxin Li, Keqiu Li, Deke Guo, Geyong Min, Heng Qi, and Jianhui Zhang

Abstract—The emerging deployment of large-scale cloud applications incurs significant inter-datacenter traffic, which makes the scarce wide-area bandwidth across data centers become the performance bottleneck. To achieve the desirable network performance, *bandwidth guarantee* should be provided for the resulting inter-datacenter traffic. However, the existing bandwidth allocation methods mainly focus on intra-datacenter traffic, and do not take the bandwidth guarantee requirement and network cost thus cannot achieve the cost-minimizing *bandwidth guarantee* across datacenters. In this paper, we focus on the *bandwidth guarantee* problem for inter-datacenter traffic and present a novel bandwidth allocation model. Our model can ensure the *bandwidth guarantee*, minimize the resulting network cost, and efficiently avoid the potential traffic overload on low cost links. To solve the large-scale optimization problem in our model, we are motivated to develop a distributed algorithm by blending the advantages of alternating direction method of multipliers (ADMM) and the auxiliary variable method. Specifically, we efficiently decompose the optimization problem into many small sub-problems, which are allowed to be processed in a large-scale computing environment, where each server solves a few small sub-problems. We further present a theoretically proved globally, asymptotically stable algorithm to solve these sub-problems. Extensive evaluation results demonstrate that our bandwidth allocation method can effectively realize the *bandwidth guarantee* for inter-datacenter traffic with reduced network cost and outperforms the prior method PS-L. In particular, the total network cost is reduced by 59.57% on average.

Index Terms—Inter-datacenter Network, Bandwidth Guarantee, ADMM.

1 Introduction

Cloud computing is being a common paradigm in today's business. Due to abundant resource availability and reduced management costs, cloud providers can offer an economical choice for enterprises to create and run their respective applications independently in data centers [1, 2]. In the meanwhile, many cloud providers construct a large number of datacenters across the world [3], to provide low latency (e.g., <35ms RTT [4]) for connecting their services. This indirectly causes that a single application may even run on multiple datacenters such as Netflix [5, 6]. Consequently, such applications incur not only large volume of traffic inside a datacenter but also increasing amount of inter-datacenter traffic. As reported in [7], inter-datacenter traffic may result from client's request, periodic data back-up, and routine background computation. A recent survey further highlights that the amount of inter-datacenter traffic will double or triple in the next two to four years [8].

The enormous growing traffic across datacenters leads to two main consequences from the perspectives of applications and cloud providers. First, the wide area network across datacenters is usually the bottleneck resource, which is shared by a large number of flows. Consequently, such applications suffer variable and unpredictable network performance, if they are unaware of the allocated bandwidth [9]. Second, the distribution of traffic loads among inter-datacenter links is nonuniform and partial links experience extremely low bandwidth utilization. This severely restricts the scalability of deployed applications. Moreover, the lack of any performance guarantee makes service providers unwilling to deploy applications across multiple datacenters. Accordingly, it will in turn decrease the revenue of cloud providers.

Fortunately, *bandwidth guarantee* can enable the desirable network performance for applications across datacenters. Prior bandwidth allocation methods [10–13], however, mainly focus on intra-datacenter traffic and cannot be simply used to address inter-datacenter traffic for the following reasons. Firstly, the existing allocation methods do not provide *bandwidth guarantee* since they either provide *bandwidth over-guarantee* [10] or *bandwidth sub-guarantee* [11]. Secondly, they do not consider the network cost for cloud providers. In reality, charged by Internet Service Providers (ISPs), such inter-datacenter traffic incurs substantial network cost to a cloud provider. Since multiple ISPs are employed by cloud providers to interconnect their geographically distributed datacenters with varied pricing strategies [14], the usage costs of such inter-datacenter links are different from each other. Thus, by carefully choosing optimal routing paths and assigning flow rates for inter-datacenter traffic, it is feasible to minimize the network cost for cloud providers.

In this paper, we propose three design rationales of bandwidth allocation for inter-datacenter traffic and design a novel bandwidth allocation model. Our model can ensure the *bandwidth guarantee*, minimize the resulting network cost, and avoid potential traffic overload at low cost links.

- *W. Li, K. Li, H. Qi and J. Zhang are with the School of Computer Science and Technology, Dalian University of Technology, No 2, Linggong Road, Dalian 116023, China. E-mail: liwenxin@mail.dlut.edu.cn, {keqiu, hengqi}@dlut.edu.cn, forev176@gmail.com. K. Li is the corresponding author.*
- *D. Guo is with the College of Information System and Management, National University of Defense Technology, Changsha 410073, P.R. China. E-mail: guodeke@gmail.com.*
- *G. Min is with the College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, United Kingdom. E-mail: G.Min@exeter.ac.uk.*

More precisely, the proposed bandwidth allocation problem is formulated as an optimization problem, typically with a large number of variables and constraints in a production datacenter system. Consequently, the critical obstacle is the lack of an efficient problem-solving method. To address this challenge, we design a novel distributed method, with the auxiliary variable method and the *alternating direction method of multipliers (ADMM)* [15]. Specifically, we efficiently decompose the optimization problem into many sub-problems. Thus, our algorithm allows for processing such sub-problems in a large-scale computing environment, where each server solves a few small sub-problems by using a theoretically proved globally asymptotically stable algorithm. We further propose the solutions for these sub-problems, and tackle two critical implementation issues of our method. Comprehensive evaluation results demonstrate that the proposed allocation method is capable of guaranteeing bandwidth for inter-datacenter traffic with reduced network cost. It outperforms the prior method of proportional sharing at link-level (PS-L) [11], which allocates bandwidth along link in proportion to communication pairs of VMs. In particular, the total network cost is reduced by 59.57% on average.

The main contributions of this paper are as follows:

- We address the challenging *bandwidth guarantee* problem for inter-datacenter traffic with the minimum network cost. Moreover, the formulated large-scale optimization model can efficiently avoid the potential traffic overload at low cost links.
- Instead of directly applying the original ADMM that leads to a centralized solution, we design an efficient distributed method by blending the advantage of the auxiliary variable method and ADMM. Theoretical analysis proves that the optimization problem can be well-addressed after decomposition. We further tackle the implementation and message exchanging issues in a large-scale computing environment.
- We conduct extensive experiments to evaluate the performance of our bandwidth allocation method. The performance results demonstrate that our proposed method can effectively guarantee the bandwidth requirements with reduced network cost and outperforms the prior method PS-L.

The rest of this paper is organized as follows. Section 2 presents the motivation and problem formulation. In Section 3, we design an efficient distributed method for the proposed bandwidth allocation model. In Section 4, we analyze and evaluate the performance of the proposed method. Section 5 summarizes the related work. The conclusions are discussed in Section 6.

2 Motivation and Problem Formulation

We first present the motivation, and then formulate cost-minimizing bandwidth guarantee problem.

2.1 Motivation

It is well-known that cloud providers deploy many geographically distributed datacenters and usually rent bandwidth from multiple ISPs for their inter-datacenter traffic

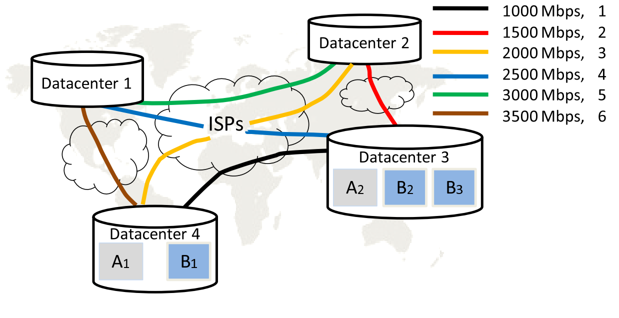


Fig. 1. An inter-datacenter network across 4 datacenters, where each inter-datacenter link is equipped with given bandwidth capacity and cost of per unit bandwidth.

[14]. Due to the competition, ISPs are evolving their business models and selling bandwidth to customers in many ways for retaining profits. Consequently, those inter-datacenter links differ in the usage cost. So, it is feasible to reduce or even minimize the cloud providers' costs by carefully designing the routing for inter-datacenter traffic. More preciously, inter-datacenter flows can be — and should be — split and transmitted along multiple-hop paths, each of which can be more cost-effective than the direct and short paths. In fact, such multi-paths can be technically implemented by applying MPTCP in the inter-datacenter network, which is an extension of TCP to handle multiple paths between endpoints [16]. The only problem is how to guarantee network performance for inter-datacenter traffic. We believe that bandwidth guarantee can enable desirable network performance, as it provides strong isolation for applications. Such bandwidth guarantee means that the allocated bandwidth to each inter-datacenter flow must be exactly equal to the corresponding bandwidth demand.

However, prior bandwidth allocation methods do not provide such bandwidth guarantee, because they provide either *bandwidth over-guarantee* or *bandwidth sub-guarantee*. To have a comprehensive understanding, we simply extending two existing allocation methods, i.e., PS-L [11] or Per-Flow [17], to the scenario of inter-datacenter traffic. The PS-L method allocates bandwidth to each flow between a VM-pair X-Y on the link based on the flow weight that is defined as $W_{XY} = \frac{W_X}{N_X} + \frac{W_Y}{N_Y}$, where W_X is the weight of VM X (similarly for W_Y), and N_X is the number of other VMs X is communicating with on this link (similarly for N_Y). The Per-Flow method fairly allocates bandwidth to each flow on the link. Figure 1 depicts an inter-datacenter network, where there are two applications. Application A deploys VM A_1 and VM A_2 on datacenters DC_4 and DC_3, respectively. Application B deploys VM B_1 on datacenter DC_4 and VM B_2, VM B_3 on datacenter DC_3. In this setting, A_1 communicates with A_2 while B_1 communicates with B_2 and B_3. By sending traffic through direct path, application A and B actually compete bandwidth on link $DC_4 \rightarrow DC_3$. Consider that each VM has one unit weight. Then, by applying the PS-L method, the allocated bandwidth per flow to applications A and B is $400Mbps$ and $300Mbps$, respectively. By applying the Per-Flow method, the allocated bandwidth per flow is $\frac{1000}{3}Mbps$ for both applications A and B. Given the bandwidth demand b_f, the allocation results of bandwidth face the following results:

- If $b_f{<}300Mbps$ for the above flows, the PS-L and Per-Flow methods provide *bandwidth over-guarantee* for applications A and B.
- If $300Mbps{<}b_f{<}\frac{1000}{3}Mbps$ for the above flows, the PS-L method provides *bandwidth over-guarantee* for application A while *bandwidth sub-guarantee* for application B. The Per-Flow method provides *bandwidth over-guarantee* for the two applications.
- If $\frac{1000}{3}Mbps{<}b_f{<}400Mbps$ for the above flows, the PS-L method provides *bandwidth over-guarantee* for application A while *bandwidth sub-guarantee* for application B. The Per-Flow method provides *bandwidth sub-guarantee* for both applications.
- If $b_f{>}400Mbps$ for the above flows, both PS-L and Per-Flow methods provide *bandwidth sub-guarantee* for the two applications.

These results imply that for any bandwidth demand, both PS-L and Per-Flow methods are insufficient to provide bandwidth guarantee for the inter-datacenter traffic.

2.2 Problem Formulation

We consider an inter-datacenter network across multiple geographically distributed datacenters, operated by a single cloud provider. Let a directed graph $\mathcal{G}{=}(\mathcal{N},\mathcal{M})$ represent such an inter-datacenter network, where $\mathcal{N}$ represents the set of vertexes corresponding to the set of datacenters and $\mathcal{M}$ is the set of inter-datacenter links. Let $u_{i,j}$ represent the bandwidth capacity on inter-datacenter link $(i,j){\in}\mathcal{M}$. Note that we focus on providing bandwidth guarantee to the inter-datacenter traffic. We therefore using a rate pricing, instead of using the volume pricing (i.e., the well-known pay-as-you go model) based on the number of bytes transferred, which is however insufficient to pricing such bandwidth guarantees [18]. The simplistic way to understand such rating pricing is that the traffic is charged based on the bandwidth allocated to it [19]. Hence, to characterize the cost diversity of inter-datacenter links, let $c_{i,j}$ be the cost of per unit bandwidth on link $(i,j){\in}\mathcal{M}$. Let $\mathcal{F}$ denote the set of inter-datacenter flows. For each flow $f{=}(s_f,d_f,b_f){\in}\mathcal{F}$, let s_f be the source datacenter, d_f be the destination datacenter, and b_f denote the bandwidth demand between them. Finally, $x^f_{i,j}$ is a fraction variable which means that $x^f_{i,j}{\times}b_f$ of the bandwidth capacity over link $(i,j){\in}\mathcal{M}$ is allocated to flow f. Clearly, $x^f_{i,j}$ should be limited from 0 to 1:

$$0{\le}x^f_{i,j}{\le}1, \forall(i,j){\in}\mathcal{M}, \forall f{\in}\mathcal{F}. \tag{1}$$

We consider three design rationales for allocating bandwidth to inter-datacenter traffic: bandwidth guarantee, minimizing the total network cost, and avoiding potential overload at low cost links.

Bandwidth guarantee: To provide the desirable network performance, *bandwidth guarantee*, rather than *over-guarantee* or *sub-guarantee* is essential for an effective bandwidth allocation method. The *bandwidth guarantee* is formally defined as follows:

Definition 1. *a) Bandwidth guarantee.* The bandwidth allocated to flow f is exactly equal to its demand b_f; *b) Bandwidth over-guarantee.* The bandwidth allocated to flow f is higher than its demand b_f; *c) Bandwidth sub-guarantee.* The bandwidth allocated to flow f is lower than its demand b_f.

It should be noted that inter-datacenter flows transmitted along multiple paths may obtain an amount of bandwidth on each path. In this case, the aggregate bandwidth allocated to a flow should exactly equal to its demand, such that bandwidth guarantee can be enforced for this flow. So, we formulate the requirement of *bandwidth guarantee* as

$$\sum_{\{j|(i,j)\in\mathcal{M}\}} x^f_{i,j} - \sum_{\{j|(j,i)\in\mathcal{M}\}} x^f_{j,i} = g^f_i = \begin{cases} 1 & i{=}s_f, \\ -1 & i{=}d_f, \forall i{\in}\mathcal{N}, \forall f{\in}\mathcal{F}. \\ 0 & \text{else}, \end{cases} \tag{2}$$

Equation (7) ensures flow conservation. Specifically, for each flow $f{\in}\mathcal{F}$, if datacenter i is an intermediate datacenter, the traffic volume entering into it equals to that leaving from it; if datacenter i is the source datacenter s_f, the traffic volume coming from it is exactly the demand of the flow; if datacenter i is the destination datacenter d_f, the traffic volume entering into it is exactly the demand of the flow.

Minimize the total cost of bandwidth guarantee: Inter-datacenter traffic incurs substantial network cost to cloud providers, due to the bandwidth rent from ISPs. As reported in [20], wide area transit bandwidth is expensive and costs more than the intra-network of a datacenter. It also reports that the cost of network resources amounts to around 15% of the total cost afforded by a cloud provider. The cost is usually similar to the power cost. Thus, the network cost caused by inter-datacenter traffic must be considered, when choosing the routing paths and assigning transmission rate for each flow. Figure 1 shows an illustrative example of the diverse cost on inter-datacenter links. If the bandwidth demand is $b_f{=}500Mbps$ for each flow among VM-pairs, the following allocation strategies can minimize the network cost as well as achieve *bandwidth guarantee* for each flow: $\frac{2}{3}{\times}500Mbps$ and $\frac{1}{3}{\times}500Mbps$ are allocated to each flow along path $\{DC_4{\to}DC_3\}$ and path $\{DC_4{\to}DC_2{\to}DC_3\}$, respectively. To this end, the total network cost of *bandwidth guarantee* is calculated as follow:

$$c(x){=}\sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} c_{i,j}b_f x^f_{i,j}. \tag{3}$$

where $c_{i,j}b_f x^f_{i,j}$ is the network cost on link (i,j) after allocating $b_f x^f_{i,j}$ bandwidth to flow f.

Avoiding potential overload at low cost links: The cost diversity of inter-datacenter links is likely to cause the potential overload at low cost links. For example, the bandwidth of link $\{DC_4{\to}DC_3\}$ in Figure 1 is fully utilized, and thus overload is possibly to happen on this link when many other flows are arriving one by one. To avoid potential overload at low cost links, we introduce a weight factor $w_{i,j}$ for each inter-datacenter link $(i,j){\in}\mathcal{M}$. The weight factor $w_{i,j}$ is defined to be inversely proportional to the square of the cost of per unit bandwidth on the corresponding link $(i,j)\in\mathcal{M}$:

$$w_{i,j}{=}\frac{(\frac{1}{c_{i,j}})^2}{\sum\limits_{(i,j)\in\mathcal{M}}(\frac{1}{c_{i,j}})^2}. \tag{4}$$

We can easily check that $\sum_{(i,j)\in\mathcal{M}} w_{i,j} = 1$. The rationale for this weight definition is that a link with a lower cost leads to a higher weight, hence should be relative less desirable compared with no weight considered when minimizing network cost. This means that the introduction of $w_{i,j}$ efficiently shifts some traffic from the links with low cost to the links with relative higher cost, thus results in a more balanced traffic distribution among inter-datacenter links. Now, we can re-formulate the total network cost associated with weight factors, called the total weighted network cost as follow:

$$wc(x)=\sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} w_{i,j}c_{i,j}b_f x_{i,j}^f, \tag{5}$$

where $w_{i,j}c_{i,j}b_f x_{i,j}^f$ is the weighted network cost on link (i,j) after allocating $b_f x_{i,j}^f$ bandwidth to flow f.

Given the three design rationales, we formulate the cost-minimizing bandwidth guarantee problem as follows:

Definition 2. *Given an inter-datacenter network $\mathcal{G}=(\mathcal{N},\mathcal{M})$, each inter-datacenter link is assigned the cost $c_{i,j}$, the weight $w_{i,j}$, and the bandwidth capacity $u_{i,j}$. A set of inter-datacenter flows $\mathcal{F}$ with source, destination and bandwidth demand for each flow, are injected into the network. We design a bandwidth allocation method such that the three rationales are all achieved. It can be formalized as:*

$$\begin{aligned}&\min_x \sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} w_{i,j}c_{i,j}b_f x_{i,j}^f\\ \text{s.t.}\quad &\mathit{Equations}\ (7),\ (7),\\ &\mathit{and}\ \sum_{f\in\mathcal{F}} b_f x_{i,j}^f \le u_{i,j}, \forall(i,j)\in\mathcal{M}.\end{aligned} \tag{6}$$

where the last constraint means that the bandwidth allocation on each link can not exceed the link capacity.

Note that this optimization problem has $|\mathcal{M}|\times|\mathcal{F}|$ variables. Thus, the computational complexity significantly increases as the number of traffic flows and inter-datacenter links grows. As reported in [7, 21], the number of datacenters is around $O(10^2)$, and the number of inter-datacenter flows is around $O(10^5)$ in some production clouds. To solve the large-scale optimization problem, we present a scalable and practical distributed method, by blending both advantages of the auxiliary variable method and the alternating direction method of multipliers (ADMM [15]).

3 Distributed Bandwidth Allocation Method

This section starts with presenting the problem decomposition. We then focus on a distributed method for the optimization problem and address two implementation issues.

3.1 Problem decomposition

The primer on ADMM can be found in [15], which has extensively studied the performance of such ADMM optimization technique in solving large scale convex problems. However, the original ADMM can not be simply applied to solve our problem in Equation (7), due to the following two facts. First, problems in the original ADMM are usually decomposable while in our problem, all variables are tightly coupled by Equation (7) and the bandwidth capacity constraint. Second, the original ADMM has high complexity and low scalability since it can lead to a centralized solution, while an efficient problem-solving method for such a large-scale optimization problem should aim to reduce the computational complexity as far as possible.

To tackle these challenges, we re-formulate the optimization problem by introducing a new set of auxiliary variables y, which satisfy the following constraints:

$$\min_x \sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} w_{i,j}c_{i,j}b_f x_{i,j}^f$$

$$0\le x_{i,j}^f\le 1, \forall(i,j)\in\mathcal{M}, \forall f\in\mathcal{F}.$$

$$\sum_{\{j|(i,j)\in\mathcal{M}\}} x_{i,j}^f - \sum_{\{j|(j,i)\in\mathcal{M}\}} x_{j,i}^f = g_i^f = \begin{cases}1 & i=s_f,\\ -1 & i=d_f, \forall i\in\mathcal{N}, \forall f\in\mathcal{F}.\\ 0 & \text{else,}\end{cases}$$

$$\sum_{f\in\mathcal{F}} b_f y_{i,j}^f \le u_{i,j}, \forall(i,j)\in\mathcal{M}, \tag{7}$$

$$0\le y_{i,j}^f\le 1, \forall(i,j)\in\mathcal{M}, \forall f\in\mathcal{F}, \tag{8}$$

$$x_{i,j}^f = y_{i,j}^f, \forall(i,j)\in\mathcal{M}, \forall f\in\mathcal{F}. \tag{9}$$

Then, we have

$$\begin{aligned}&\min_x \sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} w_{i,j}c_{i,j}b_f x_{i,j}^f\\ &\text{s.t.}\quad \mathit{Equations}\ (7),\ (7),\ (7),\ (8),\ (9).\end{aligned} \tag{10}$$

Clearly, this optimization problem and Equation (7) have the same optimal solution. The new formulation is now separable over the two sets of variables x and y. This is the key step towards decomposing the problem. The augmented Lagrangian associated with Equation (10) is defined as:

$$\begin{aligned}L_\rho(x,y,\lambda)=&\sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} w_{i,j}c_{i,j}b_f x_{i,j}^f +\\ &\sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}} \lambda_{i,j}^f\left(x_{i,j}^f - y_{i,j}^f\right) + \frac{\rho}{2}\left(x_{i,j}^f - y_{i,j}^f\right)^2,\end{aligned} \tag{11}$$

where $\lambda_{i,j}^f$ is the lagrangian multiplier and ρ is the penalty parameter. With augmented Lagrangian, we obtain:

$$\min\ L_\rho(x,y,\lambda)\ \text{s.t.}\quad \mathit{Equations}\ (7),\ (7),\ (7),\ (8). \tag{12}$$

For any feasible x and y, the penalty term added to the objective is zero. Thus, this problem is clearly equivalent to Equation (10). The benefit of introducing the penalty term is that $L_\rho(x,y,\lambda)$ is strictly convex, since the objective function is the sum of a set of convex and linear functions. The penalty term is also called a regularization term and will help to substantially improve the convergence through the following iterations:

$$x^{k+1} = \arg\min_x L_\rho(x, y^k, \lambda^k), \tag{13}$$

$$y^{k+1} = \arg\min_y L_\rho(x^{k+1}, y, \lambda^k), \tag{14}$$

$$\lambda^{k+1} = \lambda^k + \rho\left(x^{k+1} - y^{k+1}\right). \tag{15}$$

We observe that this method operates recursively. In each iteration, it minimizes the Augmented Lagrangian over x

while keeping y fixed, minimizes it over y while keeping x fixed, and then carries out a multiplier update. Thus, x and y are updated in an alternating or sequential manner, which accounts for the term *alternating direction multiplier method* for this type of algorithm.

According to Equation (13), the method first solves the following problem at the $(k{+}1)$-th iteration:

$$\begin{aligned}&\min_{x}\sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}}\left(\frac{\rho}{2}\left((y^f_{i,j})^k\right)^2-(\lambda^f_{i,j})^k(y^f_{i,j})^k\right)+\\&\sum_{f\in\mathcal{F}}\sum_{(i,j)\in\mathcal{M}}\frac{\rho}{2}(x^f_{i,j})^2+x^f_{i,j}\left(w_{i,j}c_{i,j}b_f+(\lambda^f_{i,j})^k-\rho(y^f_{i,j})^k\right)\\&\text{s.t.}\quad Equations\ (7),\ (7).\end{aligned} \tag{16}$$

Equation (16) is now decomposable over flows, since the objective function and constraints are separable over flow f. The following per-flow sub-problem needs to be independently solved.

$$\begin{aligned}&\min_{x^f}\quad L_\rho(x^f,(y^f)^k,(\lambda^f)^k)=\\&\sum_{(i,j)\in\mathcal{M}}\left(\frac{\rho}{2}\left((y^f_{i,j})^k\right)^2-(\lambda^f_{i,j})^k(y^f_{i,j})^k\right)+\\&\sum_{(i,j)\in\mathcal{M}}\left(\frac{\rho}{2}(x^f_{i,j})^2+x^f_{i,j}\left(w_{i,j}c_{i,j}b_f+(\lambda^f_{i,j})^k-\rho(y^f_{i,j})^k\right)\right)\\&\text{s.t.}\quad\sum_{\{j|(i,j)\in\mathcal{M}\}}x^f_{i,j}-\sum_{\{j|(j,i)\in\mathcal{M}\}}x^f_{j,i}=g^f_i,\forall i\in\mathcal{N},\\&\qquad 0\le x^f_{i,j}\le 1,\forall(i,j)\in\mathcal{M}.\end{aligned} \tag{17}$$

Equation (17) is of a much smaller scale than Equation (16), with $|\mathcal{M}|$ variables and $|\mathcal{M}|+|\mathcal{N}|$ constraints. Thus it can be solved efficiently as proved in Theorem 1.

Theorem 1. At the $(k{+}1)$-th iteration, the optimal solution to the per-flow sub-problem in Equation (17) for a given flow f is as follow:

$$x^f_{i,j}=\frac{\rho(y^f_{i,j})^k-w_{i,j}c_{i,j}b_f-(\lambda^f_{i,j})^k-\alpha_i+\alpha_j+\beta_{i,j}-\gamma_{i,j}}{\rho},$$

where $\alpha_i\neq 0$, $\beta_{i,j}\ge 0$ and $\gamma_{i,j}\ge 0$ are Lagrangian multipliers.

Proof: Please refer to the supplementary file. □

By decomposing Equation (16) into $|\mathcal{F}|$ per-flow sub-problems in Equation (17), the x-minimization step can then be solved distributively across multiple servers of the large-scale computing environment, as to be shown in Section 3.2. Moreover, once x^{k+1} is obtained, the problem in Equation (14) can also be solved in a similar way. According to this equation, the y-minimization step consists of solving the following problem:

$$\begin{aligned}&\min_{y}\sum_{(i,j)\in\mathcal{M}}\sum_{f\in\mathcal{F}}\frac{\rho}{2}(y^f_{i,j})^2-y^f_{i,j}\left((\lambda^f_{i,j})^k+\rho(x^f_{i,j})^{k+1}\right)+\\&\sum_{(i,j)\in\mathcal{M}}\sum_{f\in\mathcal{F}}\frac{\rho}{2}((x^f_{i,j})^{k+1})^2+(x^f_{i,j})^{k+1}(w_{i,j}c_{i,j}b_f+(\lambda^f_{i,j})^k)\\&\text{s.t.}\quad Equations\ (7),\ (8).\end{aligned} \tag{18}$$

This problem can also be decomposable over the set of inter-datacenter links $\mathcal{M}$ into $|\mathcal{M}|$ sub-problems. Specifically, the following per-link sub-problem needs to be independently solved:

$$\begin{aligned}&\min_{y_{i,j}}\quad L_\rho(x^{k+1}_{i,j},y_{i,j},\lambda^k_{i,j})=\\&\sum_{f\in\mathcal{F}}\frac{\rho}{2}(y^f_{i,j})^2-y^f_{i,j}\left((\lambda^f_{i,j})^k+\rho(x^f_{i,j})^{k+1}\right)+\\&\sum_{f\in\mathcal{F}}\frac{\rho}{2}((x^f_{i,j})^{k+1})^2+(x^f_{i,j})^{k+1}(w_{i,j}c_{i,j}b_f+(\lambda^f_{i,j})^k)\\&\text{s.t.}\quad\sum_{f\in\mathcal{F}}b_f y^f_{i,j}\le u_{i,j}\text{ and }0\le y^f_{i,j}\le 1,\forall f\in\mathcal{F}.\end{aligned} \tag{19}$$

Theorem 2 also presents the optimal solution to Equation (19).

Theorem 2. At the $(k+1)$-th iteration, the optimal solution to the per-link sub-problem in Equation (19) for a given inter-datacenter link $(i,j)\in\mathcal{M}$ is as follow:

$$y^f_{i,j}=\frac{\rho(x^f_{i,j})^{k+1}+(\lambda^f_{i,j})^k-\delta b_f+\mu_f-\nu_f}{\rho},$$

where $\delta\ge 0$, $\mu_f\ge 0$ and $\nu_f\ge 0$ are Lagrangian multipliers.

Proof: Please refer to the supplementary file. □

These sub-problems are all convex problems, and can be solved by using a primal-dual approach derived from the primal-dual approach to internet congestion control [22]. As a result, Equation (17) can be solved as follows:

$$\dot{x}^f_{i,j}=\frac{\partial(-L_\rho(x^f,(y^f)^k,(\lambda^f)^k))}{\partial x^f_{i,j}}-q^f_{i,j}+\beta_{i,j}-\gamma_{i,j}, \tag{20}$$

$$\dot{\alpha}^f_i=z^f_i-g^f_i, \tag{21}$$

$$\dot{\beta}_{i,j}=-x^f_{i,j}, \tag{22}$$

$$\dot{\gamma}_{i,j}=x^f_{i,j}-1, \tag{23}$$

where

$$\begin{aligned}q^f_{i,j}&=\alpha_i-\alpha_j,\\z^f_i&=\sum_{\{j|(i,j)\}\in\mathcal{M}}x^f_{i,j}-\sum_{\{j|(j,i)\in\mathcal{M}\}}x^f_{j,i}.\end{aligned}$$

Similarly, we solve Equation (19) as follows:

$$\dot{y}^f_{i,j}=\frac{\partial(-L_\rho(x^{k+1}_{i,j},y_{i,j},\lambda^k_{i,j}))}{\partial y^f_{i,j}}-\delta b_f+\mu_f-\nu_f, \tag{24}$$

$$\dot{\delta}=\sum_{f\in\mathcal{F}}b_f x^f_{i,j}-u_{i,j}, \tag{25}$$

$$\dot{\mu}_f=-y^f_{i,j}, \tag{26}$$

$$\dot{\nu}_f=y^f_{i,j}-1. \tag{27}$$

The algorithms specified by those equations are both globally, asymptotically stable. The proof process is as follow.

Theorem 3. The algorithms specified by Equations (20) - (23) and Equations (24) - (27) are both globally, asymptotically stable.

Proof: Please refer to the supplementary file. □

In the following, we will present our distributed algorithm.

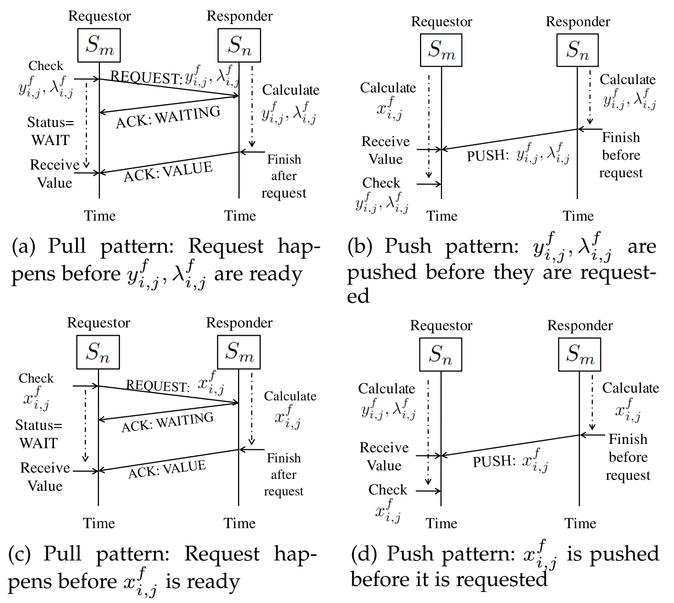
Algorithm 1 Distributed Algorithm

```
Input:
    Flow specification: $<s_f, d_f, b_f>, \forall f \in \mathcal{F}$;
    Bandwidth capacity: $u_{i,j}, \forall (i,j) \in \mathcal{M}$;
    Cost of per unit bandwidth: $c_{i,j}, \forall (i,j) \in \mathcal{M}$;
    Weighted cost of per unit bandwidth: $w_{i,j}, \forall (i,j) \in \mathcal{M}$;
    Parameter setting: $\rho$, $\xi$;
    The rounds of iteration: $K$;
Output:
    Fraction variable: $x^f_{i,j}, \forall (i,j) \in \mathcal{M}, \forall f \in \mathcal{F}$;
 1: Initialize $x^0=0, y^0=0, \lambda^0=0$;
 2: while $k<K$ do
 3:     for each flow $f \in \mathcal{F}$ do
 4:         Given auxiliary variables $(y^f)^k$ and dual variables
            $(\lambda^f)^k$, resolve the per-flow sub-problem in Equa-
            tion (17) by applying the primal-dual algorithm
            specified by Equations (20) - (23);
 5:     end for
 6:     for each link $(i,j) \in \mathcal{M}$ do
 7:         Given $x^{k+1}_{i,j}$ and $\lambda^k_{i,j}$, resolve the per-link sub-
            problem in Equation (19) through the primal-dual
            algorithm specified by Equations (24) - (27);
 8:     end for
 9:     Update $\lambda^{k+1}=\lambda^k+\rho(x^{k+1}-y^{k+1})$ (Equation (15));
10:     $k$++;
11: end while
```

3.2 Distributed algorithm

The key idea is to select the cost-effective paths, so as to reduce the cost of inter-datacenter traffic. Such cost-effective paths may turn out to be multi-hop paths, which may lead to relatively long end-to-end delays compared to the direct and short paths. Fortunately, we observe that the inter-datacenter traffic typically have different time-sensitivities [7]. Examples include interactive traffic that is most sensitive to delay, larger data transfers which require delivery within several hours, and background traffic without strict time requirements [3, 23]. So, we mainly focus on the delay-tolerant traffic (e.g., large transfers and background traffic). In the following, we design a distributed algorithm to achieve this goal.

The distributed algorithm is shown in **Algorithm** 1. Given a solution y^k computed at the previous iteration, it first optimizes x for the problem given y^k and λ^k. It then optimizes y for the problem, given λ^k and a previously computed mapping solution x^{k+1}. Finally, λ is updated to ensure that x and y converge to the same solution. The distributed nature of this algorithm allows for simultaneously solving those sub-problems, across multiple servers in a large-scale computing environment. Two parameters ρ and ξ can be configured across all servers at the beginning of the algorithm. As the prerequisites, $u_{i,j}$, b_f, $c_{i,j}$ and $w_{i,j}$ need to be gathered. In reality, such prerequisites become feasible, due to the introduction of software-defined networks. For example, Google successfully implemented B4 [3], a globally-deployed software defined WAN, to interconnect its distributed datacenters. To pertain such an algorithm in a large-scale computing environment, we discuss two critical implementation issues.



(a) Pull pattern: Request happens before $y^f_{i,j}, \lambda^f_{i,j}$ are ready

(b) Push pattern: $y^f_{i,j}, \lambda^f_{i,j}$ are pushed before they are requested

(c) Pull pattern: Request happens before $x^f_{i,j}$ is ready

(d) Push pattern: $x^f_{i,j}$ is pushed before it is requested

Fig. 2. Messages exchange between server S_m solving the per-flow sub-problem and server S_n solving the per-link sub-problem.

Parallel implementation issues. First, in Step 3-5 of each iteration, all per-flow sub-problems can be solved in a parallel fashion on each server in the computing environment. Recall that in some production systems [7], the number of inter-datacenter delay-sensitive flows is around $O(10^5)$. A computing environment typically has $O(10^4)$-$O(10^5)$ servers [20]. Thus, each server only needs to solve $O(10)$-$O(1)$ per-flow sub-problems at each iteration. The computational complexity of our algorithm is low since the per-flow sub-problem in Equation (17) is a small-scale convex optimization.

Second, in Step 6-8, we solve the per-link sub problems, which can be solved in a similar parallel manner. Only $|\mathcal{M}|$ servers are employed from the computing environment to solve these per-link sub-problems. It can even be allocated to the same servers, which host the tasks for solving the per-flow sub-problems. Those dual variables $\lambda^{k+1}_{i,j}=[(\lambda^1_{i,j})^{k+1}, (\lambda^2_{i,j})^{k+1}, \cdots]$ are also be updated in these $|\mathcal{M}|$ servers.

Finally, according to $x^f_{i,j}$, the output of Algorithm 1, the bandwidth allocation process can be really implemented by forwarding the bandwidth allocation information to the openflow controllers in the inter-datacenter Software Defined Network [3].

Message exchanging issues. Consider that the distributed algorithm needs to utilize a number of $|\mathcal{M}|+|\mathcal{F}|$ servers in the computing environment. $|\mathcal{F}|$ servers are responsible for solving all per-flow sub-problems, each of which is denoted by S_m. The other $|\mathcal{M}|$ servers solve all per-link sub-problems, each of which is denoted by S_n. All the data transmission involves in the running time of our algorithm is that x, y, and λ should be exchanged between the servers that solve per-flow sub-problems and the servers that solve per-link sub-problems. More precisely, to complete the per-flow sub-problem, S_m needs variables $y^f=[y^f_{1,2}, y^f_{1,3}, \cdots]$ and $\lambda^f=[\lambda^f_{1,2}, \lambda^f_{1,3}, \cdots]$, which are produced by the servers solving per-link sub-problems. Thus, $y^f_{i,j}$ and $\lambda^f_{i,j}$ should be

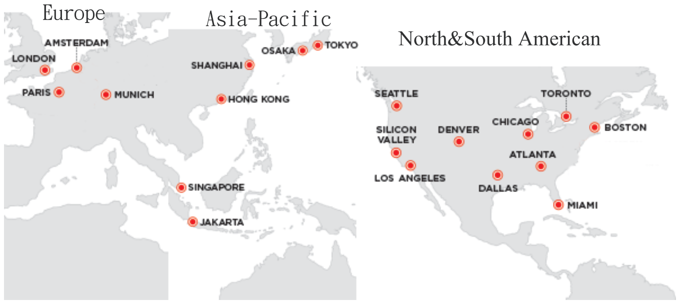


Fig. 3. An example of inter-datacenter network consists of 20 datacenters.

transmitted from S_n to S_m on the granular level.

The message exchanging can be processed in two patterns: pull pattern and push pattern. Each server in the communication plays two different roles: requestor and responder. Figure 2 shows the messages exchange between server S_m and S_n. We first study the exchange of $y^f_{i,j}$ and dual variable $\lambda^f_{i,j}$. As shown in Figure 2(a), when a requestor server S_m requests variables $y^f_{i,j}$ and $\lambda^f_{i,j}$ from responder server S_n, it first checks whether it has already received these variables or not. If yes, it will continue the iteration. Otherwise, it sends a REQUEST message to the corresponding server S_n. It then waits until it receives an ACK with VALUE of the requested variables.

Once receiving a request, the response server S_n will send back an ACK of WAITING if two variables $y^f_{i,j}$ and $\lambda^f_{i,j}$ are not ready. Once the response server completes the calculation, it will try to send an ACK with the values of $y^f_{i,j}$ and $\lambda^f_{i,j}$ back to the request server S_m. Otherwise, as show in Figure 2(b), it will push these variables to the corresponding server S_m for future use.

On the other hand, with the local variables $\lambda_{i,j}$, S_n only needs variables $x_{i,j}$ to complete the per-link sub-problem. More precisely, variable $x^f_{i,j}$ should be transmitted from S_m to S_n, which can also be exchanged in both pull (Figure 2(c)) and push patterns (Figure 2(d)). The difference is that S_m becomes response server and S_n is request server.

4 Performance Evaluation

In this section, we investigate the performance of our bandwidth allocation method from four aspects, including the *bandwidth guarantee*, total cost, the performance on avoiding the potential overload, and convergence.

4.1 Experiments settings

The inter-datacenter network in our experiments is built based on Equinix [24]. As shown in Figure 3, the simulated inter-datacenter network consists of three modules: 1) 10 datacenters locate in North&South American; 2) 6 datacenters locate in Asia-Pacific; 3) 4 datacenters locate in Europe. Each datacenter is linked with all other datacenters inside the same module. Each pair of the module has 3 remote links. Those links between modules are called as module-pair links. The bandwidth capacity on each link is set to be uniformly random within the range $[1, 10]Gbps$, which is a commonly WAN bandwidth in many companies [8]. The cost on each link inside the module is set as a random value in the range $[1, 50]$. In the meanwhile, the cost on each link between modules is uniformly set to be a random value within $[50, 100]$. The rationale for the cost setting refers to literature [19], which reports that the cost of intercontinental traffic is higher than that of intracontinental traffic.

In our experiments, we consider 10 applications, each of which is deployed across multiple datacenters. Each application deploys one VM on each of the 20 datacenters. For each application k, its VMs communicate with other VMs in the following way: $\mathcal{T}^i_k \leftrightarrow \mathcal{T}^{i+k}_k (i{=}1, 2, \cdots, 20{-}k)$, where $\mathcal{T}^i_k$ denotes the i-th VM of application k. Such communication pattern incurs many communication pairs of VMs for even one application. For each pair of VMs, it incurs an inter-datacenter flow, which has a bandwidth demand of $500Mbps$.

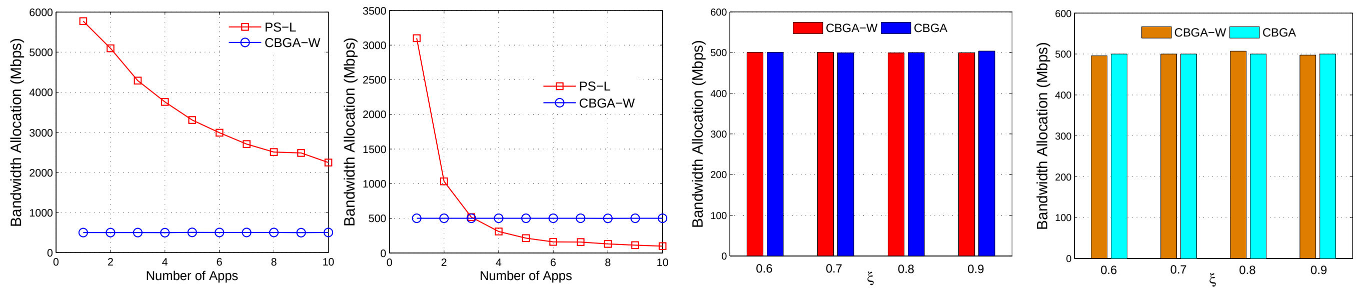
For the purpose of comparison, we also evaluate a representative allocation method *PS-L* [11], which allocates bandwidth based on the weight of communication between a pair of VMs. Note that this weight refers to the weight of communication between VMs, and is different from our weight factor $w_{i,j}$ on $(i, j) \in \mathcal{M}$. Each VM is uniformly assigned with one unit weight. We set the penalty parameter $\rho{=}1$ in all our simulation experiments. For the clarity of presentation, we use *CBGA* (cost-minimizing bandwidth guaranteed allocation method) to denote our bandwidth allocation method. Specifically, *CBGA-W* represents our bandwidth allocation method with weight factors being considered.

Not that we set a ratio ξ on each link, which means that the bandwidth capacity on each link is $\xi u_{i,j}$. Such that we are able to investigate the impact of different bandwidth capacity on the performance of our bandwidth allocation method. The performance evaluation focuses on verifying whether our bandwidth allocation method can satisfy the proposed three design rationales, under different numbers of applications and values of ξ. Moreover, the evaluation aims to quantify the convergence of our distributed algorithm.

4.2 Evaluation results and analysis

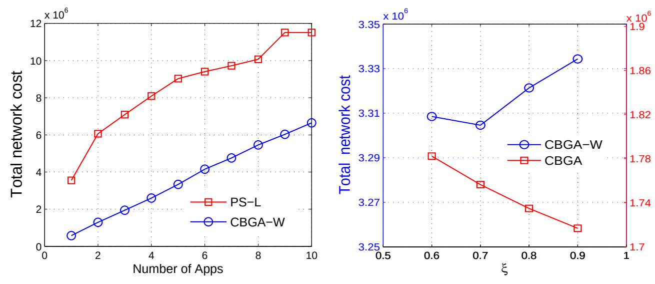
4.2.1 *Bandwidth guarantee*

Bandwidth guarantee is the fundamental requirement of our bandwidth allocation method. Since module-pair links are most likely to become congested links, the network-wide allocation is insufficient to show the network performance on these module-pair links. As a result, we evaluate the performance of *bandwidth guarantee* at both network-wide level (contains all inter-datacenter links) and module-pair link level (contains those module-pair links). Figure 4(a) first presents the bandwidth allocation per flow at the network-wide level as the number of applications increases. As the similar results are achieved under other settings of workload threshold ξ, we only plot *CBGA-W* in the case that $\xi{=}0.9$. It can be seen that the allocated bandwidth per flow by our method is always retained around $500Mbps$. That is, *CBGA-W* can provide *bandwidth guarantee* for each flow, irrespective of the applications. We can see from Figure 4(a)



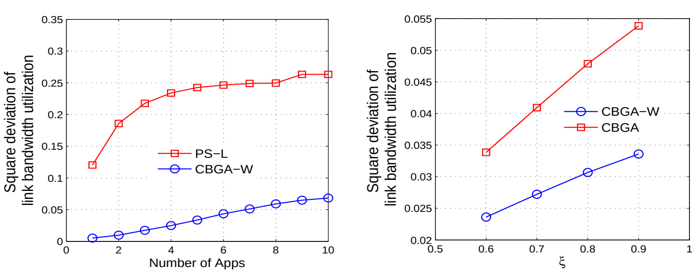
(a) Network-wide level (b) Module-pair link level (c) Network-wide level (d) Module-pair link level

Fig. 4. The performance of bandwidth guarantee. (a) The bandwidth allocation per flow at the network-wide level vs. number of applications: $\xi{=}0.9$; (b) The bandwidth allocation per flow at the module-pair link level vs. number of applications: $\xi{=}0.9$; (c) The bandwidth allocation per flow at the network-wide level vs. ξ; (d) The bandwidth allocation per flow at the module-pair link level vs. ξ.



(a) Varying Apps (b) Varying ξ

Fig. 5. Total network cost. (a) Total network cost vs. number of applications: $\xi{=}0.9$; (b)Total network cost vs. ξ.



(a) Varying Apps (b) Varying ξ

Fig. 6. Overloading. (a) Square deviation of link bandwidth utilization vs. number of applications: $\xi{=}0.9$; (b) Square deviation of link bandwidth utilization vs. ξ.

that *PS-L* provides bandwidth *over-guarantee*. More precisely, the allocated bandwidth per flow is more than at least 4 times of the bandwidth demand ($500Mbps$).

To understand the performance of *bandwidth guarantee* at the module-pair link level, we conducted related evaluations and plot Figure 4(b) to indicate the bandwidth allocation per flow on module-pair links. To ease the presentation, we only plot the bandwidth allocation per flow in the setting of $\xi = 0.9$. We find that *CBGA-W* enables the bandwidth allocation per flow to maintain around $500Mbps$, irrespective of the number of applications. On the contrary, the allocated bandwidth per flow significantly decreases as the number of applications goes up for *PS-L* method. Note that, *PS-L* always provides *bandwidth over-guarantee* when the number of applications is less than 3, and delivers *bandwidth sub-guarantee* after the number of applications exceeds 3. For the latter case, traffic congestion will appear and become more serious when the number of applications continues to increase.

Specifically, we study the impact of varying workload threshold ξ on the performance of *bandwidth guarantee*. Figure 4(c) plots the bandwidth allocation per flow at network-wide level, with variable workload threshold ξ and a fixed number of 5 applications. We can find that both *CBGA* and *CBGA-W* methods maintain the bandwidth allocation per flow closely around $500Mbps$, which is just equal to the bandwidth demand of each flow. Thus, it is clear that our *CBGA* method can provide *bandwidth guarantee*, irrespective of the workload threshold ξ. We also plot the allocated bandwidth per flow on those module-pair links in Figure 4(d). Clearly, both *CBGA* and *CBGA-W* can provide closely around $500Mbps$ bandwidth per flow, with varying workload threshold ξ. Thus, it is clear that our method can provide *bandwidth guarantee* for the flows on those module-pair links.

In a summary, our bandwidth allocation method can provide *bandwidth guarantee* for each inter-datacenter flow at both network-wide level and module-pair level, irrespective of the number of applications and parameter ξ.

4.2.2 **Total cost**

In this section, we measure the total network cost of *bandwidth guarantee* with different numbers of applications and workload threshold ξ, after using our bandwidth allocation method. Figure 5(a) presents the total network cost with both *PS-L* and our *CBGA-W* methods, under variable number of applications, where the workload threshold $\xi = 0.9$ in *CBGA-W*. It is obvious that the total network cost significantly increases as the number of applications grows. This implies that those application providers, typically tenants in cloud, have to pay more in turn for the usage of inter-datacenter links. We can further find that *PS-L* results in the higher network cost than *CBGA-W*. More precisely, *CBGA-W* achieves a total network cost reduction of 59.57% on average; hence, our bandwidth allocation method is more cost-effective than prior *PS-L* method.

Figure 5(b) reports the total network cost caused by five applications, when the workload threshold ξ ranges from 0.6 to 0.9. It can be seen that the total network cost significantly descends as ξ grows under our *CBGA* method. The root cause is that a larger workload threshold ξ leads to more flows scheduled on low cost links. Note that the weighted cost on link $(i,j) \in \mathcal{M}$ for *CBGA-W* is $w_{i,j}c_{i,j}=\frac{1/c_{i,j}}{\sum_{(i,j)\in\mathcal{M}} 1/(c_{i,j})^2}$, which first decreases along with the increasing $c_{i,j}$ and then increases after a threshold. This implies that a lower $c_{i,j}$ may not necessarily leads to a lower $w_{i,j}c_{i,j}$. The introduction of $w_{i,j}c_{i,j}$ can efficiently shift some traffic from those low cost links to other high cost links. Besides, the larger workload threshold ξ causes more traffic on all inter-datacenter links. This is the reason why the total network cost under our *CBGA-W* method always decreases until ξ exceeds 0.7. We can further find that the total network cost under our *CBGA-W* method is larger than that under our *CBGA* method, but is always lower than that under the prior *PS-L* method.

4.2.3 *The performance on avoiding potential overload*

To efficiently avoid the potential overload at low cost links, the proposed method in this paper introduces a weight factor $w_{i,j}$ on each link, and shifts some workload from low cost links to a little higher links, so as to achieve a more balanced traffic distribution. Hence, we simply use square deviation of link bandwidth utilization as the performance metric since a more balanced traffic distribution has a lower square deviation of link bandwidth utilization. To evaluate the effectiveness, we investigate the behaviors of the proposed allocation method under different numbers of applications and workload threshold ξ. Figure 6(a) presents the changing trend of the square deviation of link bandwidth utilization along with the increasing number of applications for both *PS-L* and *CBGA-W* where ξ=0.9. It is obvious that the square deviation of link bandwidth utilization increases significantly with the increasing number of applications for both *PS-L* and *CBGA-W*. This further verifies that the overloading problem definitely appears along with the increasing bandwidth demand of inter-datacenter traffic. Additionally, our *CBGA-W* method achieves the better effects when avoiding the overloading problem compared to *PS-L*. The root cause is that the square deviation of link bandwidth utilization in *CBGA-W* is lower than that in *PS-L*. Moreover, the square deviation for *CBGA-W* with 10 applications is even lower than that for *PS-L* with one application.

To understand the impact of ξ and weight on overloading, we also demonstrate the square deviation of the link bandwidth utilization with different ξ for both *CBGA* and *CBGA-W* methods in Figure 6(b). The square deviation of link bandwidth utilization significantly increases along with the increase of workload threshold ξ for *CBGA*. This is because a higher workload threshold ξ leads to more bandwidth demand on low cost links. Consequently, the overloading problem occurs since a large amount of traffic causes congestion on these links. Nevertheless, if weight $w_{i,j}$ is considered and the workload threshold ξ is low, the overloading problem can be efficiently avoided. This is why the line *CBGA-W* is always lower than that of *CBGA*.

4.2.4 *Convergence*

In this section, we investigate the convergence rate of our proposed algorithm. Due to the similar results achieved

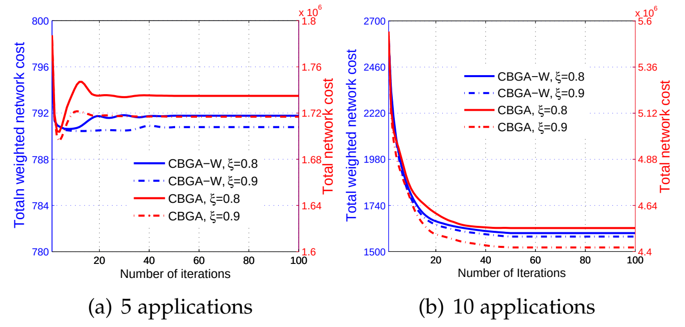


Fig. 7. Convergence rate.

under different settings, we only plot the convergence rate for some settings in Figure 7. We can find that the convergence rate changes quickly at the beginning of iterations. The root cause is that the variables x and y are set to 0 at the beginning of our implementation. As shown in Figure 7(a), the convergence rate fluctuates around the stable value along with the increase of iterations, and finally converges at a stable value when considering five applications. The convergence rate also quickly converges to the stable value for ten applications, as shown in Figure 7(b). It is clear that our algorithm takes at most 50 iterations before converging at a stable value for all cases in Figure 7.

4.2.5 *Impact of different settings of link cost*

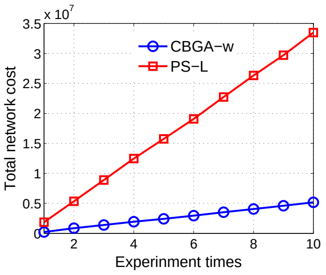


Fig. 8. The impact of different settings of link cost on the total network cost.

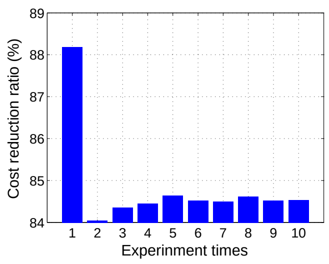


Fig. 9. The reduction ratio of the total network cost by comparing *CBGA* with *PS-L*.

In order to evaluate the impact of link cost on the total network cost, we conduct our experiments 10 times. Each time simulates different per unit bandwidth cost on inter-datacenter links. Specifically, for each experiment time t $(=1, 2\cdots, 10)$, the per unit bandwidth cost on each link is set as a random value within the range $[(t-1)\times 10, t\times 10]$. It should be noted that in the running process of each experiment time, the total network cost is recorded as the total bandwidth cost for transmitting a same set of inter-datacenter flows. Fig. 8 first plots the total network cost for both CBGA and PS-L in different experiments. Clearly, the total network cost of both *CBGA* and *PS-L* increase as the growth of per unit bandwidth cost on links. Additionally, *CBGA* can reduce the total network cost at each experiment time. To quantitatively characterize such cost reduction, we also plot the reduction ratio on the total network cost across the 10 experiment times in Fig. 9, by comparing *CBGA* with *PS-L*. We observe that despite some minor fluctuations, the cost reduction ratio is always larger than 84%, across all experiment times. To be more precise, the maximum, minimum and average cost reduction over the 10 experiment times is 88.18%, 84.84%, 84.05%. These results imply that *CBGA* is always more cost-effective than *PS-L*, irrespective of the change of per unit bandwidth cost on inter-datacenter links. The root reason is that *CBGA* can find more cost-effective paths than *PS-L*.

5 Related Work

5.1 Intra-datacenter traffic

Towards achieving predictable network performance for intra-datacenter traffic, researchers have proposed numbers of methods to share bandwidth in datacenter networks. The proposed methods so far can be generally classified into two categories: bandwidth reservation during the VM-s placement process and bandwidth allocation after VMs placement.

5.1.1 Bandwidth reservation

The proposed methods on bandwidth reservation mainly includes reservations, time-varying reservations, minimum bandwidth reservations. For example, SecondNet [25] proposes VDC (virtual data center) as the abstraction for resource allocation and distributes bandwidth reservation state at the hypervisors of servers, thus provides VM-to-VM bandwidth guarantee. Lee et al. present CloudMirror, a solution that provides bandwidth guarantees to cloud applications based on a new network abstraction TAG (tenant application graph) and workload placement algorithm which can meet bandwidth requirements specified by TAGs [26]. Oktopus [27] presents a VOC model (virtual oversubscribed cluster) and uses VM placement to provide bandwidth guarantees. It enforces static rate limits to reserve bandwidth for VMs with homogeneous bandwidth demand. While Oktopus can provide predictable performance for VMs, it ignores the highly variable bandwidth demand of VMs. Zhu et al. [28] focus on the problem of VM allocation under the consideration of providing bandwidth guarantees with both homogeneous and heterogeneous bandwidth demand considered, while they ignore the dynamic feature of datacenter traffic. Xie et al. [29] proposes TIVC, which makes time-varying bandwidth reservations based on the requirements of specific Mapreduce applications. EyeQ [30] makes minimum bandwidth reservations for each endpoint at provision time. Gatekeeper [31] reserves link bandwidth, and provides minimum bandwidth guarantee for VMs of tenants by using a distributed mechanism based on hypervisor's rate limit and feedback.

5.1.2 Bandwidth allocation

The other idea for sharing datacenter network is to allocate bandwidth for VMs after their palcement. Faircloud presents the basic bandwidth requirements of bandwidth allocation problem and proposes three kinds of sharing methods, i.e., PS-L, which allocates bandwidth on congested links based on the weights of the communication between the pairs of VMs [11]. Faircloud focuses on achieving the VM-pair level fairness. Guo et al. propose an allocation strategy based on game theory, which achieves the Nash bargaining solution for sharing datacenter network and the bargaining solution guarantees minimum bandwidth, while keeping fairness among VMs [10]. Seawall establishes hypervisor-to-hypervisor tunnels among physical servers, then realizes the communication between VMs by using these tunnels, and finally achieves the per-source fair sharing of congested links [12]. NetShare [13] provides tenant-level fairness on congested links and achieves proportional bandwidth sharing by using weighted fair queues. Chen et al. focus on application-level fairness and they introduce a rigorous definition of performance-centric fairness with the guiding principle that the performance of data parallel applications should be proportional to their weights [32]. ElasticSwitch is an efficient and practical method for providing bandwidth guarantees since it utilizes the spare bandwidth and it can be fully implemented in hypervisors, but it has fluctuations under bursty traffic when the rate limit is beyond the guarantee [33]. Guo et al. design a novel distributed bandwidth allocation algorithm based on the Logistic Model, to cope with highly dynamic traffic in the datacenter network [34].

5.2 Inter-datacenter traffic

Although many efforts have been made on the bandwidth allocation for intra-datacenter traffic, researchers only pay few attentions to the inter-datacenter traffic. For example, Feng et al. present Jetway, which minimizes the cost on inter-datacenter video traffic [35]. Liu et al. [36] present SD3, where a datacenter jointly considers update rate and visit rate to select user data for replication, making sure that a replica always reduces inter-datacenter communication. Laoutaris et al. [37] present the NetStitcher to exploit the unutilized bandwidth to transfer bulk traffic among datacenters, it gathers information about leftover resources, uses a store-and-forward algorithm to schedule data transfers, and adopts to resource fluctuations. Sushant et al. [3] propose B4, which leverages the common ownship (by Google) of all the applications, servers and data center networks all the way up to the edge of the backbone. For the bandwidth allocation, B4 delivers the max-min fair allocation to applications.

6 Conclusion

This paper presents a novel bandwidth allocation model for inter-datacenter traffic, which can ensure the bandwidth guarantee, minimize the network cost, and efficiently avoid potential traffic overload. We model the design of bandwidth allocation model under three rationales as an optimization problem. To solve the large scale optimization problem, we incorporate the advantages of both auxiliary variables and the original ADMM to design a distributed method. Theoretical proof shows that the optimization problem can be well-addressed after decomposition into many sub-problems, which can be addressed simultaneously. We further tackle the implementation of our model in the large-scale computing environment. The evaluation results demonstrate that the proposed method can effectively guarantee the bandwidth requirements of inter-datacenter traffic with reduced network cost; hence, it outperforms the prior method PS-L.

Acknowledgment

The authors would also like to sincerely thank the editors and anonymous reviewers for their thoughtful suggestions and constructive comments, which have greatly helped and inspired us to improve the quality of this paper. This work is supported by the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61225010); NSFC under Grant nos. of 61173160, 61173161, 61173162, 61173165, 61272417, 61300187, 61300189, 61370198 and 61370199; Project funded by China Postdoctoral Science Foundation (Grant No. 2013M530916), the Fundamental Research Funds for the Central Universities (Grant No.DUT13ZD101 and DUT13JS04).

References

[1] G. Motta, N. Sfondrini, and D. Sacco, "Cloud computing: An architectural and technological overview," in *Service Sciences (IJCSS), 2012 International Joint Conference on*. IEEE, 2012, pp. 23–27.

[2] N. Sfondrini, G. Motta, and L. You, "Service level agreement (sla) in public cloud environments: A survey on the current enterprises adoption," in *Proceedings of IEEE ICIST*, 2015.

[3] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of ACM SIGCOMM*, Hong Kong, 2013.

[4] "The world's first 100% hydroelectric colocation data center," Available at http://www.hydro66.com/assets/pdf/Introduction%20Data%20Sheet.pdf.

[5] F. Chen, K. Guo, J. Lin, and T. La Porta, "Intra-cloud lightning: Building CDNs in the cloud," in *Proceedings of IEEE INFOCOM*, Orlando, America, 2012.

[6] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *Proceedings of IEEE INFOCOM*, Orlando, America, 2012.

[7] Y. Chen, S. Jain, V. K. Adhikari, Z.-L. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via yahoo! datasets," in *Proceedings of IEEE INFOCOM*, Shanghai, China, 2011.

[8] "Forrester research," http://info.infineta.com/1/5622/2011-01-27/Y26.

[9] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Proceedings of IEEE/ACM CCGrid*, Newport Beach, America, 2011.

[10] J. Guo, F. Liu, D. Zeng, J. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proceedings of IEEE INFOCOM*, Turin, Italy, 2013.

[11] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: sharing the network in cloud computing," in *Proceedings of ACM SIGCOMM*, Helsinki, Finland, 2012.

[12] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proceedings of USENIX NSDI*, Boston, America, 2011.

[13] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese, "Netshare: Virtualizing data center networks across services," University of California, San Diego, Tech. Rep., 2010.

[14] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing cost and performance in online service provider networks," in *Proceedings of USENIX NSDI*, San Jose, America, 2010.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] A. Ford, C. Raiciu, M. Handley, O. Bonaventure *et al.*, "Tcp extensions for multipath operation with multiple addresses," *Internet-draft, IETF*, 2011.

[17] B. Briscoe, "Flow rate fairness: Dismantling a religion," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 63–74, 2007.

[18] D. Niu, C. Feng, and B. Li, "A theory of cloud bandwidth pricing for video-on-demand providers," in *Proceedings of IEEE INFOCOM*, Orlando, Florida, 2012.

[19] V. Valancius, C. Lumezanu, N. Feamster, R. Johari, and V. V. Vazirani, "How many tiers?: pricing in the internet transit market," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 194–205, 2011.

[20] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[21] S. Narayana, J. W. Jiang, J. Rexford, and M. Chiang, "To coordinate or not to coordinate? wide-area traffic management for data centers," Princeton University, Tech. Rep., 2012.

[22] R. Srikant, *The mathematics of Internet congestion control*. Springer, 2004.

[23] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-datacenter transfers," in *Proceedings of the Tenth European Conference on Computer Systems*, 2015.

[24] "Equinix dataceter map," http://www.equinix.com/data-center-locations/map.

[25] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the ACM CoNEXT*, Philadelphia, PA, USA, 2010.

[26] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma, "Application-driven bandwidth guarantees in datacenters," in *Proceedings of ACM SIGCOM*, Chicago, USA, 2014.

[27] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proceedings of ACM SIGCOMM*, Toronto, ON, Canada, 2011.

[28] J. Zhu, D. Li, J. Wu, H. Liu, Y. Zhang, and J. Zhang, "Towards bandwidth guarantee in multi-tenancy cloud computing networks," in *Proceedings of IEEE ICNP*, Austin, TX, USA, 2012.

[29] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," in *Proceedings of ACM SIGCOMM*, Helsinki, Finland, 2012.

[30] V. Jeyakumar, M. Alizadeh, D. Mazieres, B. Prabhakar, C. Kim, and A. Greenberg, "Eyeq: Practical network performance isolation at the edge," in *Proceedings of USENIX NSDI*, Lombard, IL, USA, 2013.

[31] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *USENIX WIOV*, Portland, OR, USA, 2011.

[32] L. Chen, Y. Feng, B. Li, and B. Li, "Towards performance-centric fairness in datacenter networks," in *Proceedings of IEEE INFOCOM*, Toronto, Canada, 2014.

[33] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: practical work-conserving bandwidth guarantees for cloud computing," in *Proceedings of ACM SIGCOMM*, Hong Kong, China, 2013.

[34] J. Guo, F. Liu, X. Huang, J. C. S. Lui, M. Hu, Q. Gao, and H. Jin, "On efficient bandwidth allocation for traffic variability in datacenters," in *Proceedings of IEEE INFOCOM*, Toronto, Canada, 2014.

[35] Y. Feng, B. Li, and B. Li, "Jetway: minimizing costs on inter-datacenter video traffic," in *Proceedings of ACM Multimedia*, Nara, Japan, 2012.

[36] G. Liu, H. Shen, and H. Chandler, "Selective data replication for online social networks with distributed datacenters," in *Proceedings of IEEE ICNP*, Goettingen, 2013.

[37] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *Proceedings of ACM SIGCOMM*, Toronto, Canada, 2011.

Wenxin Li received the B.E. degree from the School of Computer Science and Technology, Dalian University of Technology, China, in 2012. Currently, he is a Ph.D. candidate in the School of Computer Science and Technology, Dalian University of Technology, China. His research interests include datacenter networks and cloud computing.

Keqiu Li received the bachelors and masters degrees from the Department of Applied Mathematics at the Dalian University of Technology in 1994 and 1997, respectively. He received the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology in 2005. He also has two-year postdoctoral experience in the University of Tokyo, Japan. He is currently a professor in the School of Computer Science and Technology, Dalian University of Technology, China. He has published more than 100 technical papers, such as IEEE TPDS, ACM TOIT, and ACM TOMCCAP. He is an Associate Editor of IEEE TPDS and IEEE TC. He is a senior member of IEEE. His research interests include internet technology, data center networks, cloud computing and wireless networks.

Deke Guo received the B.S. degree in industry engineering from Beijing University of Aeronautic and Astronautic, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from National University of Defense Technology, Changsha, China, in 2008. He is an Associate Professor with the College of Information System and Management, National University of Defense Technology, Changsha, China. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks.

Geyong Min is the Chair Professor and Director of High Performance Computing and Networking (HPCN) Research Group at the University of Exeter, UK. He received the PhD degree in Computing Science from the University of Glasgow, UK, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. He joined the University of Bradford as a Lecturer in 2002, became a Senior Lecturer in 2005 and a Reader in 2007, and was promoted to a Professor in Computer Science in 2012. His main research interests include Next-Generation Internet, Wireless Networks, Mobile Computing, Cloud Computing, Big Data, Multimedia Systems, Information Security, System Modelling and Performance Optimization.

Heng Qi was a Lecture at the School of Computer Science and Technology, Dalian University of Technology, China. He got bachelor's degree from Hunan University in 2004 and master's degree from Dalian University of Technology in 2006. He servered as a software engineer in GlobalLogic-3CIS from 2006 to 2008. Then he got his doctorate degree from Dalian University of Technology in 2012. His research interests include computer network, multimedia computing, and mobile cloud computing. He has published more than 20 technical papers in international journals and conferences, including ACM Transactions on Multimedia Computing, Communications and Applications (ACM TOMCCAP) and Pattern Recognition (PR).

Jianhui Zhang received the B.E. degree from the School of Computer Science and Technology, Harbin Engineering University, Harbin, China, in 2009. Currently, he is pursuing the Ph.D. degree in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His research interests include datacenter networks, network protocols and cloud computing.

APPENDIX A
PROOF OF THEOREM 1

Let $\alpha_i \neq 0, \forall i \in \mathcal{N}$, $\beta_{i,j} \geq 0, \forall (i,j) \in \mathcal{M}$, $\gamma_{i,j} \geq 0, \forall (i,j) \in \mathcal{M}$ denote the Lagrange multipliers. Then, the Lagrangian of Equation (17) is shown as follow:

$$\psi(x^f,\alpha,\beta,\gamma)=L_\rho(x^f,(y^f)^k,(\lambda^f)^k)-\sum_{(i,j)\in\mathcal{M}}\beta_{i,j}x^f_{i,j}+\sum_{(i,j)\in\mathcal{M}}\gamma_{i,j}(x^f_{i,j}-1)+\sum_{i\in\mathcal{N}}\alpha_i(\sum_{\{j|(i,j)\in\mathcal{M}\}}x^f_{i,j}-\sum_{\{j|(j,i)\in\mathcal{M}\}}x^f_{j,i}-g^f_i)$$

The Karush-Kuhn-Tucker (KKT) conditions for Equation (17) are:

$$\begin{cases}\frac{\partial\psi(x^f,\alpha,\beta,\gamma)}{\partial x^f}=0,\\ \alpha_i\neq 0, \sum\limits_{\{j|(i,j)\in\mathcal{M}\}}x^f_{i,j}-\sum\limits_{\{j|(j,i)\in\mathcal{M}\}}x^f_{j,i}=g^f_i, \forall i\in\mathcal{N},\\ \beta_{i,j}x^f_{i,j}=0,\ \beta_{i,j}\geq 0, \forall (i,j)\in\mathcal{M},\\ \gamma_{i,j}(1-x^f_{i,j})=0,\ \gamma_{i,j}\geq 0, \forall (i,j)\in\mathcal{M}.\end{cases} \tag{28}$$

Then we can obtain $x^f_{i,j}$ by solving Equation (28).

APPENDIX B
PROOF OF THEOREM 2

The Lagrangian of Equation (19) is:

$$\psi(y_{i,j},\delta,\mu,\nu)=L_\rho(x^{k+1}_{i,j},y_{i,j},\lambda^k_{i,j})+\delta(\sum_{f\in\mathcal{F}}b_f y^f_{i,j}-u_{i,j}) -\sum_{f\in\mathcal{F}}\mu_f y^f_{i,j}+\sum_{f\in\mathcal{F}}\nu_f(y^f_{i,j}-1),$$

where $\delta \geq 0$, $\mu_f \geq 0$, and $\nu_f \geq 0$ are Lagrange multipliers. The Karush-Kuhn-Tucker (KKT) conditions are:

$$\begin{cases}\frac{\partial\psi(y_{i,j},\delta,\mu,\nu)}{\partial y_{i,j}}=0,\\ \delta(u_{i,j}-\sum_f bfy^f_{i,j})=0, \quad \delta\geq 0,\\ \mu_f y^f_{i,j}=0,\ \mu_f\geq 0, \quad \forall\ f\in\mathcal{F},\\ \nu_f(1-y^f_{i,j})=0,\ \nu_f\geq 0, \quad \forall\ f\in\mathcal{F}.\end{cases} \tag{29}$$

Then we can obtain $y^f_{i,j}$ by solving Equation (29).

APPENDIX C
PROOF OF THEOREM 3

We prove the stability of the primal-dual algorithm by using the theory of Lyapunov stability. The function $\Psi_\rho(x^f,y^k,\lambda^k)$ in the equivalent per-flow sub-problem in Equation (17) is strictly concave, so there exists an optimal solution which has been proved in Theorem 1. Suppose that $(\hat{x^f},\hat{\alpha},\hat{\beta},\hat{\gamma})$ is an equilibrium point of the primal-dual algorithm for problem in Equation (17). Now, we prove that this point is globally, asymptotically stable. Let the Lyapunov function be defined as follow:

$$V(x^f,\alpha,\beta,\gamma)=\sum_{(i,j)\in\mathcal{M}}\int_{\hat{x^f_{i,j}}}^{x^f_{i,j}}\sigma-\hat{x^f_{i,j}}\,d\sigma+\sum_{i\in\mathcal{N}}\int_{\hat{\alpha^f_i}}^{\alpha^f_i}\epsilon-\hat{\alpha^f_i}\,d\epsilon +\sum_{(i,j)\in\mathcal{M}}\int_{\hat{\beta_{i,j}}}^{\beta_{i,j}}\theta-\hat{\beta_{i,j}}\,d\theta+\sum_{(i,j)\in\mathcal{M}}\int_{\hat{\gamma_{i,j}}}^{\gamma_{i,j}}\eta-\hat{\gamma_{i,j}}\,d\eta.$$

Note that $V(\hat{x^f},\hat{\alpha},\hat{\beta},\hat{\gamma})=0$. if $x^f_{i,j}\neq\hat{x^f_{i,j}}$, we have

$$\int_{\hat{x^f_{i,j}}}^{x^f_{i,j}}\sigma-\hat{x^f_{i,j}}\,d\sigma=\frac{1}{2}(x^f_{i,j}-\hat{x^f_{i,j}})^2>0.$$

This argument can be extended to the other terms as well. Thus, whenever $(x^f,\alpha,\beta,\gamma)\neq(\hat{x^f},\hat{\alpha},\hat{\beta},\hat{\gamma})$, we have $V(x^f,\alpha,\beta,\gamma)>0$. Now we get the differential of Lyapunov function as follow:

$$\begin{aligned}\dot{V}(x^f,\alpha,\beta,\gamma)=&\sum_{(i,j)\in\mathcal{M}}(x^f_{i,j}-\hat{x^f_{i,j}})\dot{x^f_{i,j}}+(\beta_{i,j}-\hat{\beta_{i,j}})\dot{\beta_{i,j}}\\ &+\sum_{(i,j)\in\mathcal{M}}(\gamma_{i,j}-\hat{\gamma_{i,j}})\dot{\gamma_{i,j}}+\sum_{i\in\mathcal{N}}(\alpha^f_i-\hat{\alpha^f_i})\dot{\alpha^f_i}\\ &=\left(\frac{\partial\Psi_\rho(x^f,y^k,\lambda^k)}{\partial x^f}-\frac{\partial\Psi_\rho(\hat{x^f},y^k,\lambda^k)}{\partial\hat{x}}\right)(x^f-\hat{x^f})\\ &+(\alpha^f-\hat{\alpha^f})(z^f-\hat{z^f})-(q^f-\hat{q^f})(x^f-\hat{x^f})\\ &-\sum_{(i,j)\in\mathcal{M}}\left(\beta_{i,j}\hat{x^f_{i,j}}+\gamma_{i,j}(1-\hat{x^f_{i,j}}))\right),\end{aligned}$$

where the last line follows from the Karush-Kuhn-Tucker conditions in Equation (28) and $(\alpha^f-\hat{\alpha^f})(z^f-\hat{z^f})-(q^f-\hat{q^f})(x^f-\hat{x^f})=0$ follows the fact that

$$\begin{aligned}\alpha^f z^f&=\sum_{i\in\mathcal{N}}\alpha^f_i(\sum_{j:(i,j)\in\mathcal{M}}x^f_{i,j}-\sum_{j:(j,i)\in\mathcal{M}}x^f_{j,i})\\ &=\sum_{(i,j)\in\mathcal{M}}x^f_{i,j}(\alpha^f_i-\alpha^f_j)=q^f x^f.\end{aligned}$$

Also, we get

$$\left(\frac{\partial\Psi_\rho(x^f,y^k,\lambda^k)}{\partial x^f}-\frac{\partial\Psi_\rho(\hat{x^f},y^k,\lambda^k)}{\partial\hat{x}}\right)(x^f-\hat{x^f}) =\rho(\hat{x^f}-x^f)(x^f-\hat{x^f})\leq 0$$

by substituting the differential. This result can also be verified by the strictly concavity of $\Psi_\rho(x^f,y^k,\lambda^k)$. As a whole, we have $\dot{V}(x^f,\alpha,\beta,\gamma)\leq 0$.

Similarly, suppose that $(\hat{y_{i,j}},\hat{\delta},\hat{\mu},\hat{\nu})$ is the optimal solution to equivalent per-link sub-problem in Equation (19). The Lyapunov function is:

$$V(y_{i,j},\delta,\mu,\nu)=\int_{\hat{\delta}}^{\delta}\kappa-\hat{\delta}\,d\kappa+\sum_{f\in\mathbb{F}}\int_{\hat{y^f_{i,j}}}^{y^f_{i,j}}\iota-\hat{y^f_{i,j}}\,d\iota+ \sum_{f\in\mathbb{F}}\int_{\hat{\mu_f}}^{\mu_f}\pi-\hat{\mu_f}\,d\pi+\sum_{f\in\mathbb{F}}\int_{\hat{\nu_f}}^{\nu_f}\tau-\hat{\nu_f}\,d\tau.$$

Obviously, whenever $(y_{i,j},\delta,\mu,\nu)\neq(\hat{y_{i,j}},\hat{\delta},\hat{\mu},\hat{\nu})$, we have

$V(y_{i,j}, \delta, \mu, \nu) > 0$. The differential is as follow:

$$\dot{V}(y_{i,j}, \delta, \mu, \nu) = \sum_{f \in \mathbb{F}} (y_{i,j} - \hat{y_{i,j}})\dot{y_{i,j}} + (\delta - \hat{\delta})\dot{\delta} +$$

$$\sum_{f \in \mathbb{F}} (\mu_f - \hat{\mu^f})\dot{\mu_f} + \sum_{f \in \mathcal{F}} (\nu_f - \hat{\nu_f})\dot{\nu_f}$$

$$= \left(\frac{\partial \Psi_\rho(x^{k+1}, y_{i,j}, \lambda^k)}{\partial y_{i,j}} - \frac{\partial \Psi_\rho(x^{k+1}, \hat{y_{i,j}}, \lambda^k)}{\partial \hat{y_{i,j}}} \right) (y_{i,j} - \hat{y_{i,j}})$$

$$- \delta \left(u_{i,j} - \sum_{f \in \mathcal{F}} b_f \hat{y_{i,j}^f} \right)$$

(according to Equation (29))

$$= \rho(\hat{y_{i,j}} - y_{i,j})(y_{i,j} - \hat{y_{i,j}}) \leq 0$$

Thus, it follows by the theory of Lyapunov stability that the algorithm is indeed globally, asymptotically stable.