Jetway: Minimizing Costs on Inter-Datacenter Video Traffic

Yuan Feng, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Bo Li
Department of Computer Science
Hong Kong University of Science and Technology

ABSTRACT

It is typical for video streaming service providers (such as NetFlix) to rely on services from cloud providers (such as Amazon), in order to build a scalable video streaming platform with high availability. The trend is largely driven by the fact that cloud providers deploy a number of datacenters inter-connected by high-capacity links, spanning different geographical regions. Video traffic across datacenters, such as video replication and transit server-to-customer video serving, constitutes a large portion of a cloud provider's inter-datacenter traffic. Charged by ISPs, such inter-datacenter video traffic incurs substantial operational costs to a cloud provider. In this paper, we argue that costs incurred by such inter-datacenter video traffic can be reduced or even minimized by carefully choosing paths, and by assigning flow rates on each inter-datacenter link along every path. We present *Jetway*, a new set of algorithms designed to minimize cloud providers' operational costs on inter-datacenter video traffic, by optimally routing video flows in an online fashion. Algorithms in *Jetway* are designed by following a methodical approach based on an in-depth theoretical analysis. As a highlight of this paper, we have built a real-world system framework to implement and deploy *Jetway* in the Amazon EC2 datacenters. With both simulations and real-world experiments using our implementation, we show that *Jetway* effectively helps transmitting videos across datacenters with reduced costs to cloud providers and satisfactory real-world performance.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Inter-Datacenter Traffic, Flow Optimization

1. INTRODUCTION

Due to abundant resource availability and reduced management costs, it is an emerging trend for video streaming service providers, such as Netflix, to resort to the services of cloud providers, such as Amazon Web Services (AWS). As an example, Netflix is using the Amazon Simple Storage Service (S3) for storing all of its video masters, which are further transcoded to a number of video formats, and are then distributed to Content Distribution Networks (CDNs), ready to be served to end users [1, 2].

On the other hand, Amazon Web Services as cloud providers have recently introduced *CloudFront*, a CDN service with edge servers around the world, designed to meet the needs of video streaming providers. CloudFront seamlessly integrates with Amazon S3 and the Amazon Elastic Compute Cloud (EC2), so that videos hosted in S3 or EC2 can be streamed using the Real Time Messaging Protocol (RTMP) to end users, from one of the edge servers with geographical proximity and low network latencies [3]. In addition, in order to accommodate high-definition videos, Amazon S3 has substantially raised its limit on object sizes (from 5 GB to 5 TB) in December 2010 [4]. Given such a win-win situation as video streaming providers are going "100% cloud" [1], it will be a near-term certainty that large volumes of video traffic will flow from cloud datacenters (*e.g.,* S3), which host video masters, to CDN edge servers (*e.g.,* CloudFront), which serve end users.

The tenet of providing cloud services is to maximize the sharing of resources, while keeping tenants (*e.g.,* Netflix) satisfied. To provide cloud services with better availability and scalability, it is customary for cloud providers to deploy a number of datacenters across different geographical regions. These datacenters are typically inter-connected with high-capacity links leased from ISPs. With substantial upfront investments to construct these datacenters, it is certainly to the advantage of cloud providers to minimize operational costs. Recent research reveals that traffic costs amount to around 15% of operational costs incurred to a cloud provider, a percentage that is similar to energy costs [5].

As large quantities of high-definition videos are being hosted in these datacenters, a substantial amount of inter-datacenter traffic will be incurred by replicating these videos and by serving these videos to CDN edge servers, in order to provide a highly available and scalable streaming service. Such inter-datacenter video traffic constitutes a large portion of a cloud provider's inter-datacenter traffic. Since most cloud providers today rely on multiple Internet Service Providers (ISPs) to connect their geographically dispersed datacenters [6], operational costs can be effectively reduced, if costs charged by these ISPs on inter-datacenter video traffic can be minimized.

Given dominant percentile-based charging models currently in use by most ISPs [7], *e.g.,* the 95-th percentile charging model, it is feasible to reduce or even minimize cloud providers' costs by designing optimal routing and flow assignment algorithms for inter-datacenter video traffic. In other words, video flows across inter-datacenter links can be — and should be — *split and trans-*

MM'12, October 29–November 2, 2012, Nara, Japan.

mitted along multiple multi-hop paths, each of which can be optimally and dynamically computed over time. The rationale is that, the cost of transmitting the same amount of videos varies significantly across different inter-datacenter links, due to regional pricing and peering relationships among ISPs [8]. For example, domestic video flows are substantially cheaper than flows to global destinations, and video flows within the backbone network built by cloud providers themselves incur very low costs.

Further, spatial and temporal characteristics of inter-datacenter video traffic also motivates the design of new routing and flow assignment algorithms. *Temporally*, a portion of the video flows across datacenters may be more delay-tolerant than others, if they represent video replication and backups. To reduce costs, we may re-route these delay-tolerant video flows by using intermediate datacenters as relays, flowing over multi-hop paths and splitting into multiple paths [9]. *Spatially*, datacenters located in different time zones experience peak video traffic at different times, providing more opportunities of resource multiplexing.

In this paper, we present *Jetway*, a new set of algorithms designed to minimize operational costs on inter-datacenter video traffic in an efficient and simple way. To guide the design of our algorithms in *Jetway*, we present a methodical and in-depth analytical study on how inter-datacenter video traffic costs are to be minimized by routing video flows via multiple multi-hop paths in an optimized fashion. With *Jetway*, we take advantage of different traffic costs on inter-datacenter links, usually charged by a multitude of ISPs with the percentile-based charging model, taking into account practical constraints of limited link capacities, as well as different *desired transmission rates* of videos, representing their delay tolerance. Our study leads to new combinatorial algorithms that are simple yet efficient enough for cloud providers to implement in practice: video flows are split and routed in an on-the-fly fashion by solving the classic *minimum-cost multicommodity flow* and *maximum concurrent flow* problems. An illustrative example of routing inter-datacenter video flows is shown in Fig. 1, in which the width of each flow denotes the flow rate on a particular link.

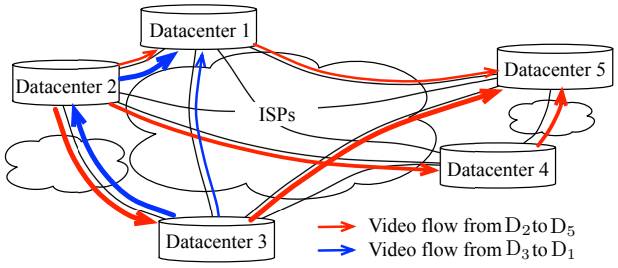


Figure 1: Selecting the best paths for inter-datacenter video traffic in a cloud with 5 datacenters. Based on differing traffic costs on inter-datacenter links and varying transmission rates of videos, the video from Datacenter 2 (D_2) to Datacenter 5 (D_5) is best routed along path $\{D_2 \rightarrow D_1 \rightarrow D_5\}$, $\{D_2 \rightarrow D_3 \rightarrow D_5\}$, and $\{D_2 \rightarrow D_4 \rightarrow D_5\}$, represented by red (light gray) flows; the video from Datacenter 3 (D_3) to Datacenter 1 (D_1) is best routed along path $\{D_3 \rightarrow D_2 \rightarrow D_1\}$ and $\{D_3 \rightarrow D_1\}$, represented by blue (dark gray) flows.

We evaluate the performance of *Jetway* in minimizing costs on inter-datacenter video traffic with our real-world implementation in the Amazon EC2 cloud, as well as extensive simulations. Our *Jetway* implementation has been developed based on a flexible and reusable video streaming framework: a highly-optimized packet forwarder is hosted on each datacenter in the cloud, referred to as a *Jet*. We have developed the *Jet* from scratch using the asynchronous networking interface in the OS kernel, taking advantage of the Asynchronous I/O (`asio`) framework in the Boost C++ library. The *Jet* is optimized to support concurrent video flows, each with its own routing paths and flow assignments. Both our real-world experimental results and simulations have shown that, by routing video flows optimally, *Jetway* is capable of reducing costs on inter-datacenter video traffic.

The remainder of this paper is organized as follows. In Sec. 2, we discuss the challenges of minimizing operational costs on inter-datacenter video traffic to a cloud provider, and formulate the optimal routing and flow assignment problem in an online fashion. In Sec. 3, we propose algorithms in *Jetway* that seek to minimize costs by splitting and routing video flows optimally. In Sec. 4, we present our real-world implementation of *Jetway* in detail, and evaluate its performance with both real-world experiments in the Amazon EC2 cloud and simulations. We discuss related work and conclude the paper in Sec. 5 and Sec. 6, respectively.

2. *JETWAY*: RATIONALE, CHALLENGES, AND PROBLEM FORMUATION

We first present the rationale and challenges that motivate the design of *Jetway*, with an objective of minimizing operational costs on inter-datacenter video traffic.

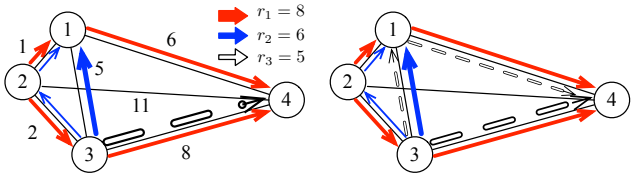
2.1 Rationale and Challenges

A cloud provider is usually charged by ISPs for its inter-datacenter traffic. The operational costs incurred are typically based on the amount of traffic the cloud provider generates. The percentile-based charging model, which is also called the q-th percentile charging model, is predominantly used by ISPs today. With this charging model, an ISP records the traffic volume a cloud provider generates during each 5-minute interval and sort them in a descending order. At the end of a complete charging period, the q-th percentile of all 5-minute traffic volumes is considered as the charging volume x, which will be used to derive the cost by a piece-wise linear non-decreasing function $c(x)$ [10]. For example, if the 95-th percentile charging model is in use and the charging period is one year, then the charging volume x of the cost function corresponds to the traffic volume sent during the 99864-th sorted interval ($95\% \times 365 \times 24 \times 60/5 = 99864$).

Besides the fact that traffic costs on each inter-datacenter link differ from one another, in that a relay path might incur much lower costs than a direct path, the percentile-based charging model provides further opportunities to reduce operational costs. With a percentile-based charging model, if some video traffic is already generated on one link, idling or transmitting less video traffic in subsequent time intervals within the same charging period will be a waste of capital investment, as these time intervals will be charged based on the already generated traffic volume anyway. As such, a feasible way to reduce costs is to carefully design routing paths and flow assignments for each pair of inter-datacenter video flow — a key idea in the design of *Jetway* — such that the idling and under-utilized time intervals are eliminated as much as possible, and the q-th percentile of video volumes over all time intervals is minimized.

The following example intuitively explains the rationale of optimal routing with the percentile-based charging model. Shown in Fig. 2, 3 videos are to be transferred in an inter-datacenter network with 4 datacenters. Assume that Video 1 and 2 are to be streamed from datacenter D_2 to D_4 with rate 8 and from D_3 to D_1 with rate 6 within the first time interval, respectively; and Video 3 is to be

streamed from D_3 to D_4 with rate 5 in the second time interval. For the sake of simplicity, we assume that the 100-th percentile charging model with a linear cost function is in use in this example, which implies that the cost incurred between each datacenter pair is the maximum video volume sent during these time intervals, multiplied by a flat cost per traffic unit shown on the link. The link capacity are assumed to be 5 for all the links.



(a) Optimal routing considering link costs only

(b) Optimal routing considering link costs and the charging model

Figure 2: How traffic costs can be reduced with optimal routing: a motivating example.

If the difference of costs per unit of traffic on each inter-datacenter link is considered, cheaper paths are preferred by video flows to reduce traffic costs. Shown in Fig. 2 (a), the optimal routing and flow assignment in this scenario is: Video 1 will take paths $D_2 \rightarrow D_1 \rightarrow D_4$ and $D_2 \rightarrow D_3 \rightarrow D_4$, each with a flow of 4 and Video 2 will take paths $D_3 \rightarrow D_2 \rightarrow D_1$ and $D_3 \rightarrow D_1$ with flow 1 and 5, respectively in the first time interval; and Video 3 will take the direct path at the second time interval, leading to a total cost of 104 per time interval.

If we further incorporate the consideration of the 100-percentile charging model in this example, we can see that a part of Video 3 can be routed along the more expensive path $D_3 \rightarrow D_1 \rightarrow D_4$, taking advantage of the already generated traffic volume on this path in past time intervals, when carrying the flows of Video 1 and 2. Shown in Fig. 2 (b), the optimal routing and flow assignment in this scenario is to route Video 3 through path $D_3 \rightarrow D_1 \rightarrow D_4$ and $D_3 \rightarrow D_4$ in the second time interval, with rates 1 and 4, respectively. By doing so, costs on inter-datacenter video traffic per time interval can be reduced to 96, as Video 3 is carried for free with the percentile-based charging model.

Unfortunately, applying the basic concept of multi-hop routing in each video flow presents formidable challenges when it comes to more general cases, involving multiple video flows with different source-destination datacenters. Due to the consideration of the percentile-based charging model, the dimension of *time* has to be taken into account when computing incurred costs on a link, which increases the complexity of the problem significantly. The cost of inter-datacenter video traffic in one time interval is affected by the traffic volume in time intervals before and after that time interval within the same charging period. If we wish to optimize the cost globally, we will need to estimate future traffic demand within the entire charging period (say, a month or a year), yet inter-datacenter traffic may not be accurately predictable beyond much finer time scales (such as a few seconds) [11]. In order to design algorithms to minimize costs incurred by inter-datacenter video traffic, it is our objective to formulate the problem such that it is practically solvable, yet sufficiently efficient.

2.2 Network Model

Before formulating the problem of minimizing operational costs on inter-datacenter traffic formally, we first introduce our network model in this paper. Important notations used throughout this paper are listed in Table 1.

Table 1: Notations and Definitions

Notation	Definition
$\mathcal{V}$	the set of datacenters operated by a single cloud provider
$\mathcal{E}$	the set of directed links connecting datacenters in $\mathcal{V}$
$\mathcal{K}(t)$	the set of video flows initiated in the time interval t
(s_k, d_k)	source and destination datacenters of video flow k
r_k	the desired transmission rate of video flow k
c_{ij}	the available capacity on link $\{i,j\}$
a_{ij}	the cost per traffic unit on link $\{i,j\}$
f_{ij}^k	the flow assigned on link $\{i,j\}$ for video flow k
$\bar{t}$	the duration of one time interval
I	the number of time intervals in a charging period
$\max_t f_{ij}$	the maximum aggregate flow rate on link $\{i,j\}$ up to interval t
$d_{ij}(t)$	the rate of other inter-datacenter traffic on link $\{i,j\}$ in the time interval t
$\text{cost}_{ij}(t)$	the operational costs on link $\{i,j\}$ up to interval t
$\mathcal{K}_f(t)$	the already paid set of video flows during interval t
$\mathcal{K}_c(t)$	the set of video flows with additional cost during interval t

We consider a cloud with multiple geographically distributed datacenters operated by a single cloud provider. Every datacenter in the cloud is connected to all other datacenters. We use a complete directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent the inter-datacenter network, where $\mathcal{V}$ indicates the set of datacenters, and $\mathcal{E}$ indicates the set of directed links inter-connecting datacenters. For each link $\{i,j\} \in \mathcal{E}$, we use a non-negative real-value function a_{ij} to denote the cost per traffic unit from datacenter i to j; a positive function $c_{ij}(t)$ to denote the available link capacity at time t, which is the maximum available rate of transmission from datacenter i to j.

Let $\mathcal{K}(t)$ be the set of videos to be transmitted at time t, all of which are represented by source-destination video flows. During its transmission to the destination datacenter, each flow can be split and relayed by other datacenters, *e.g.*, it can be routed over multiple paths and multiple hops within a path. For each video flow $k \in \mathcal{K}(t)$, we use a specification (s_k, d_k, r_k) to describe it. Here, vertices s_k and d_k indicate the source and the destination datacenter from and to which flow k is being routed, and r_k is the desired transmission rate for video flow k. In the interest of minimizing traffic costs incurred on inter-datacenter links, the desired transmission rate for each video flow that is being replicated to other datacenters is obtained by its size divided by its corresponding maximum tolerable transfer time; and is the minimum rate for an enjoyable video playback for transit server-to-customer video flows.

Since the time dimension has to be considered in the percentile-based charging model, we use a time-slotted model to incorporate multiple time intervals in a charging period. Let I be the number of time intervals in a charging period, with t as indices. The duration of one time interval is assumed to be 5 minutes, which is denoted by $\bar{t}$. To focus on the essence of the problem, we assume that the 100-th percentile charging model is in use, *i.e.*, a cloud provider is charged based on the *maximum* of traffic volumes generated over all time intervals in a charging period. Our results can be extended to a cloud environment with any percentile-based charging model, which will be a topic of discussion forthcoming in this paper.

For simplicity, we assume that the cost function $c(x)$ is a linear function $c(x) = a \cdot x$, where x is the traffic volume to be charged. To be exact, if we use $f_{ij}(t)$ to denote the aggregate flow rate on link $\{i,j\}$ in the time interval t, cost on inter-datacenter traffic in

one charging period is the dot product:

$$\text{cost} = \sum_{\{i,j\}\in\mathcal{E}} a_{ij}(\max_I f_{ij})\bar{t}I,$$

where $\max_I f_{ij}$ is the maximum aggregate flow rate on link $\{i,j\}$ of all $f_{ij}(t)$, from $1 \le t \le I$. Note that the aggregate flow rate $f_{ij}(t)$ may also contain other inter-datacenter traffic such as backups and propagation of large updates. If we use $d_{ij}(t)$ to denote the rate of this part of flow on link $\{i,j\}$ in the time interval t, then the aggregate flow rate on link $\{i,j\}$ can be represented by

$$f_{ij}(t) = d_{ij}(t) + \sum_{k\in\mathcal{K}(t)} f_{ij}^k(t),$$

where $f_{ij}^k(t)$ indicates the amount of flow assigned on link $\{i,j\}$ for video flow k in the time interval t.

To capture the fact that each video flow k may be initiated at any time in a charging period, and will also be terminated after a period of time, we let $r_k(t)$, the desired transmission rate of video flow k, depend on the time index t, indicating whether or not flow k is in transmission. More precisely:

$$r_k(t) = \begin{cases} r_k & t \in \text{time intervals video flow } k \text{ is in transmission,} \\ 0 & \text{otherwise.} \end{cases}$$

To simplify the model, we assume that video flows are always initiated at the beginning of a time interval, and terminated at the end of a time interval.

2.3 Formulating the Online Problem

In the design of *Jetway*, the problem we are trying to solve is: what is the optimal routing and flow assignment strategy we can apply to inter-datacenter video flows initiated in the current time interval to minimize operational costs, with the assumption that *past* (historical) information is known? In other words, we seek to find optimal routing paths and flow assignments for each video flow initiated in the current time interval t, so that the operational costs to the cloud provider are minimized till the end of time interval t, given all routing paths and flow assignments for video flows initiated from time interval 1 to $t-1$.

If we use $\text{cost}_{ij}(t)$ to denote the operational costs on link $\{i,j\}$ *up to* time interval t, the optimal routing and flow assignment problem in *Jetway* can be formulated as the following optimization problem, with the assumption that all datacenters in the cloud possess information about all video flows:

$$\min_{f_{ij}^k(t)} \quad \sum_{\{i,j\}\in\mathcal{E}} \text{cost}_{ij}(t) \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}(t)} f_{ij}^k(t) + d_{ij}(t) \le c_{ij}(t), \forall\{i,j\}\in\mathcal{E} \tag{2}$$

$$\sum_{j\in\mathcal{V}} f_{ij}^k(t) = \sum_{j\in\mathcal{V}} f_{ji}^k(t), \forall k\in\mathcal{K}(t), \forall i\in\mathcal{V}/\{s_k,d_k\}$$

$$\sum_{j\in\mathcal{V}} f_{s_kj}^k(t) - \sum_{j\in\mathcal{V}} f_{js_k}^k(t) = r_k(t), \forall k\in\mathcal{K}(t) \tag{3}$$

$$f_{ij}^k(t) \ge 0, \forall k\in\mathcal{K}(t), \forall\{i,j\}\in\mathcal{E}, \tag{4}$$

where $\text{cost}_{ij}(t)$ equals the product of the maximum aggregate flow rate on each link over t time intervals and the duration of a time interval with the 100-th percentile charging model, *i.e.*,

$$\text{cost}_{ij}(t) = a_{ij}(\max_t f_{ij})\bar{t}I. \tag{5}$$

The optimization variable of problem (1) is $f_{ij}^k(t)$, which indicates the flow assigned on link $\{i,j\}$ for video flow k in the time interval t. The assumption is that flows assigned on every link for each video flow in the past time intervals, and current flow rates incurred by other applications are known *a priori*, *i.e.*, $f_{ij}^k(1)$ up to $f_{ij}^k(t-1)$ and $d_{ij}(t)$ are known. Inequality (2) stands for the link capacity constraint, which ensures that the total flow assigned on one link will not exceed its current capacity. Inequalities (3) represent the flow conservation constraint. For each video flow, the flows going into any intermediate datacenter should equal to flows going out of that datacenter; while the flows coming from the source datacenter should be exactly the same as its desired transmission rate. Inequality (4) ensures that all flows are non-negative.

With the 100-th percentile charging model, the cost function (5) can be rewritten as:

$$\text{cost}_{ij}(t) = \begin{cases} \text{cost}_{ij}(t-1) & f_{ij}(t) \le \max_{t-1} f_{ij} \\ a_{ij}f_{ij}(t)\bar{t}I & \text{otherwise.} \end{cases}$$

Based on historical information, $\max_{t-1} f_{ij}\bar{t}$, the charging volume up to time interval $t-1$, is known. The cost function on link $\{i,j\}$ up to time interval t is equivalent to:

$$\text{cost}_{ij}(t) = \text{cost}_{ij}(t-1) + a_{ij}(t)\left(f_{ij}(t) - \max_{t-1} f_{ij}\right)\bar{t}I, \tag{6}$$

where $a_{ij}(t)$ is a step-function of $f_{ij}(t)$ that has the form of:

$$a_{ij}(t) = \begin{cases} 0 & f_{ij}(t) \le \max_{t-1} f_{ij} \\ a_{ij} & \text{otherwise.} \end{cases}$$

By substituting Eqn. (6) into the objective function, we get the following optimization problem that is equivalent to problem (1):

$$\min_{f_{ij}^k(t)} \sum_{\{i,j\}\in\mathcal{E}} a_{ij}(t)f_{ij}(t)\bar{t}I. \tag{7}$$

If we focus on decisions in the time interval t, and drop the time indices in the problem expressions, we can see that the optimization problem (7) is in the form of the classic minimum-cost multicommodity flow problem.

However, none of the algorithms solving the minimum-cost multicommodity flow problem can be applied to solve this problem. The reason is that $a_{ij}(t)$ is now a function of $f_{ij}(t)$, which results in a non-linear cost function. Since our objective is to find a simple yet efficient algorithm that can be readily implemented, we would like to study alternative and more tractable formulations of this problem.

3. SPLITTING AND ROUTING FLOWS OPTIMALLY TO MINIMIZE COSTS

From Eqn. (6), we find that the cost on a link up to time interval t equals the sum of the cost on that link up to time interval $t-1$, and the additional cost incurred by the possible overflow in the time interval t. This implies that, if the traffic volume on a link during one time interval is less than the charging volume on that link *up to* the previous time interval, which is the maximum aggregate flow rate on that link over $t-1$ time intervals times the duration of one time interval, no additional costs will be incurred. In other words, the traffic volume in interval t is carried on the link *for free*. However, if the traffic on that link exceeds the previous charging volume, the cloud provider will be charged extra for the overflow.

Fig. 3 illustrates the operational costs on link $\{i,j\}$. Having the charging volume up to time interval $t-1$, the same amount of traffic is already paid for in the time interval t, no matter if it is used up or not. The blue net area in the figure represents the already paid portion of the traffic volume, and the red diagonal area indicates

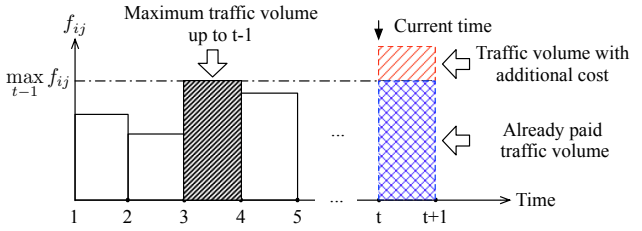


Figure 3: An illustration of the operational costs on link $\{i, j\}$.

the potential traffic volume that will incur additional costs. This implies that, if the video flows on each link are assigned in a fashion that, *the already paid portion of the traffic volume is utilized as much as possible, and the traffic volume with additional costs is minimized*, the operational costs on inter-datacenter traffic up to time interval t will be minimized as a result. Based on this idea, the design of *Jetway* is based on decoupling problem (1) into two sequential optimization problems.

3.1 Fully Utilizing the Already Paid Portion of Traffic Volume

Having $\max_{t-1} f_{ij}\bar{t}$ as the charging traffic volume on link $\{i, j\}$ up to time interval $t-1$, the amount of already paid traffic flow in the time interval t can be obtained as $\sum_{\{i,j\}\in\mathcal{E}} \min(\max_{t-1} f_{ij}, c_{ij}(t) - d_{ij}(t))\bar{t}$, in which the flow on each link is determined by its charging traffic volume up to time interval $t-1$ and its available link capacity in the time interval t. Choosing from video flows initiated in the time interval t, we can obtain a subset $\mathcal{K}_f(t)$, in which the sum of the desired transmission rates are the *maximum* possible amount that is no larger than the already paid traffic volume divided by the duration of a time interval in interval t. $\mathcal{K}_f(t)$ is the already paid set of video flows to be carried during time interval t. In the ideal case, all video flows in $\mathcal{K}_f(t)$ should be carried without incurring additional costs.

As a consequence, for video flows in $\mathcal{K}_f(t)$, our objective is to find their *feasible flow assignments*, under the joint link capacity constraint, the flow conservation constraint, and the non-negative flow assignment constraint. The optimization presentation of this problem is to find the maximum fraction z, such that up to z fraction of each flow's desired transmission rate is assigned on links in the time interval t [12]. Referred to as the *maximum concurrent flow problem*, it has the following form:

$$\max_{f_{ij}^k(t)} \quad z \tag{8}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}_f(t)} f_{ij}^k(t) + d_{ij}(t) \le \min(\max_{t-1} f_{ij}, c_{ij}(t)), \forall\{i,j\}$$

$$\sum_{j\in\mathcal{V}} f_{ij}^k(t) = \sum_{j\in\mathcal{V}} f_{ji}^k(t), \forall k \in \mathcal{K}_f(t), \forall i \in \mathcal{V}/\{s_k, d_k\}$$

$$\sum_{j\in\mathcal{V}} f_{s_k j}^k(t) - \sum_{j\in\mathcal{V}} f_{j s_k}^k(t) = z r_k(t), \forall k \in \mathcal{K}_f(t)$$

$$f_{ij}^k(t) \ge 0, \forall k \in \mathcal{K}_f(t), \forall\{i,j\} \in \mathcal{E}$$

$$0 \le z \le 1.$$

Note that compared to the general maximum concurrent flow problem, optimization problem (8) has its additional constraints resulted from our objective to minimize costs on inter-datacenter video traffic. Instead of being restricted by the available link capacity only, the capacity constraint in problem (8) is further restricted by the already paid portion of video traffic on each link, which ensures that no potential cost is incurred.

There exists a fast combinatorial algorithm to approach the ϵ-optimal solution of this problem. A flow assignment is said to be ϵ-optimal if it overflows the link capacities by at most $1+\epsilon$ factor and has a cost that is within $1+\epsilon$ of the optimum. Since ϵ can be defined as small as possible, the ϵ-optimal solution can approach the optimum value as close as possible. It has been proved that the ϵ-optimal flow can be computed deterministically in $O(\epsilon^{-2}knm \log K \log^3 n)$ time, where K is the number of commodities (video flows), m is the number of links, and n is the number of datacenters in the cloud [13].

The general idea of this algorithm is to find the minimum dual variable λ that indicates the congestion of flows. The procedure is as follows. Routing each video flow separately, an initial flow assignment $\mathbf{f^k}$ that satisfies the desired transmission rate and obeys the link capacity constraint to the extent of λ_k is obtained for all $k \in \mathcal{K}_f(t)$. Define a cost function b_k to each flow assignment $\mathbf{f^k}$ with respect to a non-zero, non-negative length function. Repeatedly examine all flows in $\mathcal{K}_f(t)$ in a round-robin fashion. If a flow is found with a "bad" flow assignment by solving its corresponding minimum-cost flow problem, its flow assignment is updated. Note that the minimum-cost flow problem at each iteration can be solved by successive approximation,which has a running time of $O(\min(nm \log n, n^{5/3}m^{2/3}, n^3) \log(nC))$ [14].

The combinatorial maximum concurrent flow algorithm is described in **Algorithm 1**, where a potential function ϕ is used to guide the algorithm [13].

Algorithm 1 *The combinatorial maximum concurrent flow algorithm.*

1: Obtain the initial solution $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{f}(\mathbf{t})|})$.
2: Set $\alpha = 3(1+\epsilon)\bar{\lambda}^{-1} \ln m\epsilon^{-1}$; $\sigma = (2\epsilon)/(\alpha\bar{\lambda}^{-1})$.
3: **while** $\lambda_f > (1-\frac{\epsilon}{2})\bar{\lambda}$ & $\Delta\phi > \epsilon^2\phi^{(k)}$ **do**
4: **for** Each $k \in \mathcal{K}_f(t)$ **do**
5: Find the minimum cost flow assignment $\mathbf{f^{k*}}$ over $\mathbf{P^k}$.
6: **if** $b_k - b_k^* \ge \epsilon b_k$ **then**
7: Update $\mathbf{f^k} = \mathbf{f^k} + \sigma(\mathbf{f^{k*}} - \mathbf{f^k})$.
8: **end if**
9: **end for**
10: **end while**
11: Return $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{f}(\mathbf{t})|})$.

3.2 Minimizing the Additional Cost

The optimal fraction z^* obtained by solving problem (8) indicates that at most z^* fraction of flows from $\mathcal{K}_f(t)$ can be carried using the already paid traffic volume. If $z^* = 1$, all flows in $\mathcal{K}_f(t)$ can be carried without additional cost. If $z^* < 1$, only z^*F_k of each video flow k incurs no extra cost, and the transmission of the remaining part of each flow does incur an additional cost. In this case, we use the same indicator k to denote the leftover part of the original video flow k, which has a transmission rate of $(1-z^*)r_k$.

Let $\mathcal{K}_c(t)$ denote the set of video flows that incur additional costs in the time interval t, including both flows in $\mathcal{K}(t) - \mathcal{K}_f(t)$ and the remaining partial flows in $\mathcal{K}(t)$ after solving optimization problem (8). Since the traffic of carrying video flows in $\mathcal{K}_c(t)$ is bound to incur additional costs, the cost function remains a linear function with a flat per-unit cost. As a result, the optimal flow assignment for these flows can be found by solving a minimum-cost multicommodity flow problem with a linear cost function in the time interval

t, which is exactly in the form of *minimum-cost multicommodity flow problem*.

The conventional way of solving this problem is to express it as a linear program, and then to solve it with a polynomial-time linear program solver [12]. Similar to the maximum concurrent flow problem, there exists a fast combinatorial algorithm to approach the ϵ-optimal solution to this problem, and the running time of this combinatorial algorithm is proved to be $\tilde{O}(\epsilon^{-3}Kmn)$ [15].

The algorithm solves the problem as follows. Represent the flow assignment as $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|}) \in (\mathbf{P^1}, \mathbf{P^2}, ..., \mathbf{P}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|})$, where $\mathbf{f^k}$ is the $|\mathcal{E}| \times 1$ flow assignment vector for flow k, and $\mathbf{f^k} \in \mathbf{P^k}$. The polytope $\mathbf{P^k}$ corresponds to the feasible flow assignments of flow k, *i.e.*, flow assignments that obey the flow conservation constraint and the individual link capacity constraint, disregarding the rest of the video flows. At each iteration, a flow k is randomly chosen for any $k \in \mathcal{K}_c(t)$. Compute the minimum-cost flow assignment $\mathbf{f^{k^*}}$ over $\mathbf{P^k}$, and update its flow assignment $\mathbf{f^k}$ to $(1-\sigma)\mathbf{f^k} + \sigma\mathbf{f^{k^*}}$ if it causes a decrease in the potential function ϕ, which is used to guide the algorithm [15]. Similarly, the minimum-cost flow problem at each iteration can be solved by successive approximation [14]. The combinatorial minimum-cost multicommodity flow algorithm is described in **Algorithm 2**.

Algorithm 2 *The combinatorial minimum-cost multicommodity flow algorithm.*

Obtain the initial solution $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|})$.
Set $\alpha = (1/\epsilon)\ln(3|\mathcal{E}|)$; $\sigma = \Theta(1/\alpha^3)$.
while $\phi(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|}) > 3|\mathcal{E}|$ **do**
  Randomly choose $k \in \mathcal{K}_c(t)$.
  Find the minimum cost flow assignment $\mathbf{f^{k^*}}$ over $\mathbf{P^k}$.
  if $\phi(\mathbf{f^1}, ..., \mathbf{f^{k^*}}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|}) < \phi(\mathbf{f^1}, ..., \mathbf{f^k}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|})$ **then**
    Update $\mathbf{f^k} = (1-\sigma)\mathbf{f^k} + \sigma\mathbf{f^{k^*}}$.
  end if
end while
Return $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|})$.

Our complete routing and flow assignment strategy in *Jetway* in the time interval t is presented in **Algorithm 3**. As indicated by our problem formulation, the routing and flow assignment for video flows in *Jetway* is updated in a periodic fashion, which can be adjusted flexibly by cloud providers.

Implementation Issues. A key assumption in **Algorithm 3** is that every datacenter is required to possess information about all video flows. If we need a distributed heuristic, there are both good and bad news. The good news is that there exist distributed algorithms to approach the ϵ-optimal solution of the maximum concurrent flow problem [16], which we use to maximize the utilization of the already paid portion of the traffic volume. The bad news is that the design of efficient distributed solutions to the minimum-cost multicommodity flow problem — which we use to minimize additional traffic costs — remains an open and elusive problem in theoretical computer science [17]. Fortunately, since all datacenters are operated by the same cloud provider, we believe that, partly due to the small number of datacenters in use, it is technically straightforward to devise a centralized controller to obtain and access information about all the video flows, which makes the algorithm design in *Jetway* much less dependent on the availability of distributed algorithms.

Although the 100-percentile charging model is used as an example in this paper, our algorithm is not only limited to this specific charging model. Instead, it can be applied to any percentile-based

Algorithm 3 *Routing and flow assignment strategy in the time interval t.*

Compute the already paid traffic volume in the time interval t.
Obtain the set of video flows $\mathcal{K}_f(t)$.
For all flows $k \in \mathcal{K}_f(t)$, find the optimal flow assignment $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{f}(\mathbf{t})|})^*_f$ by solving the maximum concurrent flow problem using **Algorithm 1**.
Obtain the set of video flows $\mathcal{K}_c(t)$.
For all flows $k \in \mathcal{K}_c(t)$, find the optimal flow assignment $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}_\mathbf{c}(\mathbf{t})|})^*_c$ by solving the minimum-cost multicommodity flow problem using **Algorithm 2**.
for Each flow $k \in \mathcal{K}(t)$ **do**
  if $k \in \mathcal{K}_f(t)$ & $k \in \mathcal{K}_c(t)$ **then**
    $\mathbf{f^{k^*}} = \mathbf{f^{k^*}_f} + \mathbf{f^{k^*}_c}$.
  else
    if $k \in \mathcal{K}_f(t)$ **then**
      $\mathbf{f^{k^*}} = \mathbf{f^{k^*}_f}$.
    else
      $\mathbf{f^{k^*}} = \mathbf{f^{k^*}_c}$.
    end if
  end if
end for
Return $(\mathbf{f^1}, \mathbf{f^2}, ..., \mathbf{f}^{|\mathcal{K}(\mathbf{t})|})^*$.

charging model as a feasible approach to reduce traffic cost when no workload prediction is considered. With a general q-th percentile charging model, the charging volume will be the $q*I$-th largest traffic volume over all time intervals in a charging period, which implies that traffic volumes from the $q*I+1$-th to the largest time intervals can be as large as possible since they will not generate additional costs. Again, we seek to find out the optimal routing and flow assignment strategy we can apply at the current time interval, with the objective of minimizing operational costs with a q-th percentile charging model.

To be specific, if we sort the aggregate flow rate on link $\{i,j\}$ from time interval 1 to the last time interval I in a decreasing order, and use $\text{qt}(f_{ij}, q)_I$ to denote the $q*I$-th value in this time series of the aggregate flow rates, the already paid traffic volume in the time interval t can be represented by a general form of $\sum_{\{i,j\}\in\mathcal{E}} \min(\text{qt}(f_{ij}, q)_I, c_{ij}(t) - d_{ij}(t))\bar{t}$. Then **Algorithm 3** can be applied to make the routing and flow assignment strategy with any given q-th percentile charging model.

4. *JETWAY:* PERFORMANCE EVALUATION

We believe that the best way to evaluate *Jetway* is to implement it, and in this section, we evaluate how *Jetway* performs in the Amazon EC2 inter-datacenter network. Our real-world experimental results have validated that, by optimally finding routing paths and assigning flows for videos flows at each time interval, *Jetway* reduces costs on inter-datacenter video traffic by a substantial margin.

4.1 Implementation

Our *Jetway* implementation contains more than 9000 lines of C++ code, and is developed from scratch. To evaluate our proposed algorithms in the Amazon EC2 inter-datacenter network, we need to send actual video traffic between source-destination pairs, over either single-hop or multi-hop paths, with possibilities of splitting flows. For this reason, we have implemented a daemon process that is able to send, receive, and relay video traffic, referred to as a *Jet*. By using asynchronous event-driven networking provided by the Boost `asio` C++ framework, it is designed with performance and

Table 2: Link Capacities and Costs per Traffic Unit in the Amazon EC2 Inter-Datacenter Network

Link Capacity (Mbps)/Cost	North California	Oregon	Virginia	Sao Paulo	Ireland	Singapore	Tokyo
North California	—	520.40/1	252.67/2	116.75/15	98.67/20	103.69/27	173.06/30
Oregon	545.06/1	—	215.84/3	81.18/17	104.22/15	81.99/25	152.75/27
Virginia	240.78/2	210.64/3	—	139.41/10	221.55/10	81.44/15	110.10/17
Sao Paulo	40.98/15	60.84/17	11.85/10	—	22.41/25	9.59/18	62.42/22
Ireland	106.45/20	135.02/15	215.85/10	90.12/25	—	77.89/23	76.10/20
Singapore	124.40/27	110.95/25	84.54/15	57.31/18	80.92/23	—	242.80/5
Tokyo	178.36/30	143.44/27	99.40/17	61.99/22	43.61/20	116.33/5	—

scalability in mind. Since it is supposed to be running in cloud VMs, the *Jet* supports major UNIX variants and Windows. As compared to the traditional thread pool concurrency model, our implementation incurs less memory and CPU overhead, even at high packet processing rates.

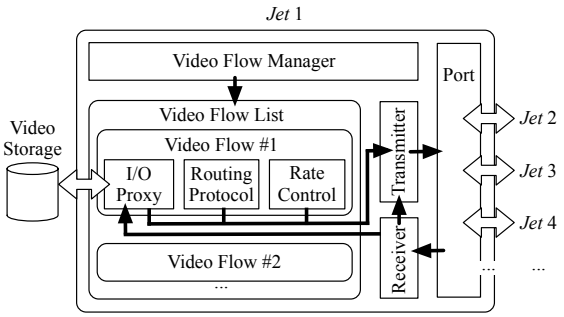


Figure 4: *Jet*: architectural components and design.

Fig. 4 has shown the architecture overview of a *Jet*, comprising of a number of key components.

Video Flow Manager. This component manages all ongoing video flows within each *Jet*, where all video flows are included in the Video Flow List. When the source-destination pair corresponding to a new video flow is initiated, the Video Flow Manager will add the corresponding flow in the video flow list, and create an I/O proxy for transmission between the stored video and the *Jet* at the source datacenter. The *Jet* at the source is responsible for determining the routing paths and their corresponding transmission rates for each video flow, based on our *Jetway* algorithms. All data packets are then passed on to the Transmitter.

Transmitter. For data packets within a video flow, the Transmitter retrieves its route from the Video Flow List, and send them to the corresponding next-hop *Jet*, which is either the destination *Jet* over a direct path, or a relay *Jet* in a multi-hop path with routing information embedded. Further, the flow rate on each path is carefully controlled by the rate control algorithm, so that they conform to the decisions made by the *Jetway* algorithms.

Receiver. As an incoming data packet arrives, the *Receiver* identifies which destination datacenter that packet is addressed to. If it is addressed to the datacenter this *Jet* is located at, the packet will be saved via the I/O proxy. Otherwise, if it belongs to a multi-hop relay path, the packet will be forwarded by the Transmitter to the next-hop jet according to the routing path carried in the header.

Port. This component maintains active connections to all *Jets* for video flows, each of which is located in a different datacenter. The Port is also able to probe link capacities and detect connection failures, such that the routing and flow assignment for each flow can be adjusted adaptively as the network status evolves. The *Jet* is designed to support a large number of concurrent video transmissions. Both higher-level Transmitter and Receiver and the lower-level network Port support multiplexed inter-datacenter connections to reduce additional overhead consumed by each video flow.

4.2 Experiments in the Amazon EC2 Cloud

We have conducted our *Jetway* experiments in the Amazon EC2 Cloud, one of the dominant Infrastructure as a Service (IaaS) cloud providers. Fig. 5 shows the inter-datacenter network topology in the Amazon EC2 cloud that we have used in our experiments. We have launched 7 standard on-demand medium instances with 2 compute unit and 1.7 GB memory in each of the datacenters, installed with our *Jetway* implementation. We log all statistics on every link for each video flow every 30 seconds, including the actual transmission rate, receiving rate, and the end-to-end delay.

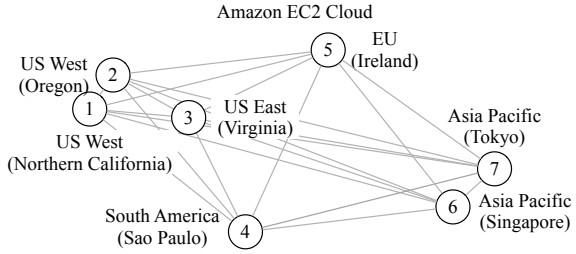


Figure 5: The network topology used in our experiments.

We have first obtained the link capacities in the inter-datacenter network through saturating the outgoing link of each *Jet*. The results are listed in Table 2, showing the average link capacity between each datacenter pair that we observe in 3 minutes. Since costs charged by ISPs on inter-datacenter links in the Amazon cloud are not revealed, we assume a certain cost per Mbps per time interval (5 minutes) on each link, reflecting different costs on transmitting the same amount of video. Costs we used in our experiments are also listed in Table 2.

We conduct our experiments for one hour, which has 12 time intervals in total. We consider the scenario that 10 videos are to be replicated to the datacenter located at Singapore at the beginning of our experiments, with 3 of them being Standard Definition (SD) videos with sizes uniformly random between $[500, 800]$ MB, and 7 of them being High Definition (HD) videos with sizes uniformly random between $[2, 4]$ GB. Source datacenters of these videos are randomly selected from the remaining 6 datacenters in the EC2 cloud. If we assume all video replications have to be finished within 30 minutes, we get the desired transmission rate of each video, ranging from $[2.22, 17.78]$ Mbps. We further assume that there are 3 inter-datacenter video flows satisfying requests from CDN edge servers at the beginning of each time interval, with the source and destination datacenters randomly selected from all 7 datacenters in the cloud. The desired transmission rate for these videos are assumed to be uniformly random between $[2.5, 8]$ Mbps, which are standard rate requirements for today's video streaming services.

For fair comparisons, we have also implemented an alternative routing solution that performs better than the straightforward approach in terms of reducing traffic costs. Referred to as the *time-slotted minimum cost algorithm* (TSMC), it considers different costs

per traffic unit on each link only, without considering important aspects of the time dimension. In particular, TSMC solves the multi-commodity minimum cost problem using **Algorithm 2** at each time interval, and obtains the routing paths and flow assignments for the set of videos to be transmitted, based on current available link capacities. To study the effects of a more realistic percentile-based charging model, we have implemented *Jetway* and TSMC, with both the 100-th percentile charging model and the 95-th percentile charging model, and compared their performance with different charging models.

We first present the main performance metric, the total cost per traffic unit over all links in the inter-datacenter network over time. Fig. 6 shows the total cost per traffic unit in the Amazon EC2 inter-datacenter network over one hour by using *Jetway* and TSMC, respectively, with the 100-th percentile charging model. As we can observe, starting from minute 15, *Jetway* is substantially more cost-effective than TSMC. The benefits of *Jetway* is becoming increasingly visible as time elapses, and achieves a cost reduction of 19% after one hour with the 100-th percentile charging model.

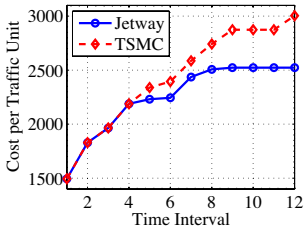


Figure 6: Cost per traffic unit with the 100-th percentile charging model.

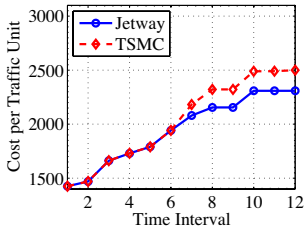


Figure 7: Cost per traffic unit with the 95-th percentile charging model.

Note that both algorithms incurred almost the same cost over the first few time intervals in our experiment. The rationale is that, due to the limited number of video flows over a large number of inter-datacenter links, the already-paid traffic volume can hardly be utilized to the benefit of cost reduction. For each video flow, both algorithms lead to similar optimal routing and flow assignments, by solving the same multi-commodity minimum cost problem. Possibilities of utilizing the already-paid traffic volume increase as time elapses, which results in better performance with *Jetway*. Take the transit server-to-customer video flow (Video 20) from the Ireland datacenter (5) to the Oregon datacenter (2) as an example. Fig. 8 shows the routing and flow assignment for this video flow in *Jetway*. We can see that, initiated at minute 20, the flows of this video takes both cheaper paths $5 \rightarrow 3 \rightarrow 2$ and $5 \rightarrow 3 \rightarrow 1 \rightarrow 2$, and expensive paths $5 \rightarrow 6 \rightarrow 3 \rightarrow 2$ and $5 \rightarrow 6 \rightarrow 3 \rightarrow 1 \rightarrow 2$, taking advantage of links $5 \rightarrow 6$ and $6 \rightarrow 3$ that are already used for video replication at the first time interval.


```
{
  "Protocol": "TCP",
  "PacketSize": 4096,
  "AverageRate": 3,
  "NodeID": 5,
  "RoutingList": [ {  "PathList": ["3,1,2" ], "Weight": 0.334967  },
                   {  "PathList": ["3,2"], "Weight": 0.334967  },
                   {  "PathList": ["6,3,1,2"], "Weight": 1.8986165  },
                   {  "PathList": ["6,3,2"], "Weight": 0.4314495  }
                 ],
  "VideoID": 20,
  "StartTime": 1200
}
```


Figure 8: Routing and flow assignment for Video 20 in *Jetway*.

We also present the total cost per traffic unit using both *Jetway* and TSMC with the 95-th percentile charging model in Fig. 7, where we can observe a similar trend. With smaller absolute values in costs, the reduction on the total cost per traffic unit with *Jetway* is up to 8% in this case. Our results have confirmed that *Jetway* can reduce costs on inter-datacenter video traffic substantially, even with the (more realistic) 95-th percentile charging model.

To investigate how our *Jetway* algorithms affect the performance of video streaming services, we record the receiving rates for each video flow every 30 seconds in our experiment. We define the normalized receiving rate to be the actual receiving rate observed at the destination datacenter divided by the desired transmission rate of this video flow. Fig. 9 shows the normalized receiving rates for 5 video flows from the 2nd time interval (minute 5) to the 7th time interval (minute 30). Each of them starts its transmission at the beginning of every time interval, respectively. As we can see, the normalized receiving rates for 3 video flows are 1 most of the time, with minor fluctuations; and the normalized receiving rates for Video 15 and Video 17 exhibit obvious fluctuations, which is resulted by the unstable network status on their common link $3 \rightarrow 6$.

We also plot the CDF of the normalized receiving rates for all video flows using *Jetway* in our experiment. Shown in Fig. 10, over 91.5% video flows using *Jetway* exhibit receiving rates that are more than 90% of the desired transmission rates at their destination datacenters. The mean value of the normalized receiving rate is 0.97 by using *Jetway*, which shows that the performance of video streaming regarding the receiving rates is satisfactory by using *Jetway*.

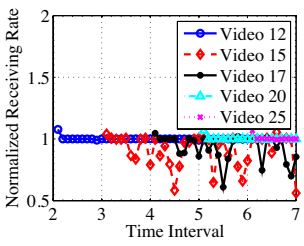


Figure 9: Normalized receiving rates for 5 video flows using *Jetway*.

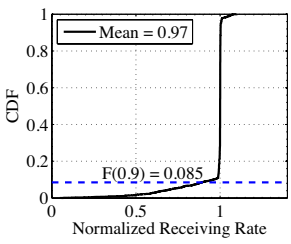


Figure 10: CDF of the normalized receiving rates for all video flows using *Jetway*.

Another important performance metric in video streaming service is the end-to-end delay experienced by each video flow. Since *Jetway* allows multi-path multi-hop transmission, we are interested in investigating whether it will affect the end-to-end delay when transmitting videos. Fig. 11 shows the CDF of the end-to-end delays experienced by all video flows using *Jetway* in our experiment. We can observe that 90% of the video flows experienced end-to-end delays that are smaller than 587.3 ms, and the mean value of end-to-end delays for all videos is 349.09 ms. We believe that these values are reasonable, considering the nature of long-distance inter continental inter-datacenter video transmissions.

To further investigate the effect of *Jetway* on end-to-end delays for video streaming, we take the Singapore datacenter (6) as an example, and compute the average end-to-end delays experienced by video flows from every other datacenter to this one, together with their 95% confidence intervals. For comparisons, we also compute the average end-to-end delays in each direct link, which are obtained by the measured round trip time (RTT) on each link divided by 2, with their corresponding 95% confidence intervals as well. The results are shown in Fig. 12. As we can see, video

flows in *Jetway* have experienced almost the same end-to-end delays as compared to those by sending through direct paths. Our results prove that the multi-path multi-hop routing and flow assignment algorithms in *Jetway* will not affect the performance of video streaming services significantly, yet the deployment of *Jetway* leads to substantially reduced operational costs on inter-datacenter video traffic.

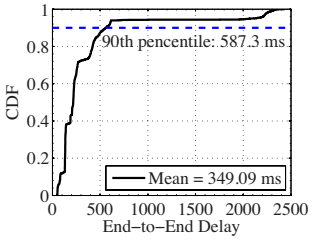


Figure 11: CDF of the end-to-end delays for all video flows using *Jetway*.

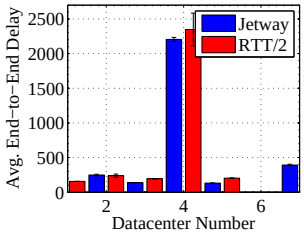


Figure 12: Average end-to-end delays to the Singapore datacenter (6).

4.3 Simulation Results

To investigate the scalability and stability of *Jetway*, we further evaluated it in a time-slotted simulator. The simulated inter-datacenter network has 20 datacenters, forming a complete graph. The capacity on each link is assumed to be 1000 units, and the cost per traffic unit on each link is set to be uniformly random within $[1, 100]$. In each time interval, the number of video flows to be transmitted is uniformly random between $[1, 200]$, each with a desired transmission rate uniformly random between $[1, 10]$ units. The source and destination datacenters of each video flow are also chosen uniformly random from the datacenter set. We conduct our simulations 20 times, each lasts for 100 time slots.

Again, the main performance metric we are interested in is the reduction in the total cost per traffic unit over time by using *Jetway*, as a percentage of the normalized cost reduction ratios compared to TSMC and their 95% confidence intervals. Fig. 13 and Fig. 14 show the average percentage of reduction in the total cost per traffic unit on video traffic with both the 100-th percentile charging model and the 95-th percentile charging model, respectively. Consistent with our experimental results shown in Fig. 6 and Fig. 7, the normalized cost reduction ratio by applying *Jetway* is increasing as time elapses, and reaches up to an average of 13% after 100 time intervals with the 100-th percentile charging scheme. The similar trend is observed with the 95-th percentile charging model, with even more cost reduction of up to 20% in the long term. Our results further confirmed that *Jetway* can successfully reduce costs on inter-datacenter video traffic, even in inter-datacenter networks of larger scales and many video flows.

To further study the effects of the 95-th percentile charging model on *Jetway*, we show the portion of video flows carried by utilizing the already paid portion of traffic volume in *Jetway* with both the 100-th percentile and the 95-th percentile charging models. Shown in Fig. 15 and Fig. 16, around 75% of video flows are carried by utilizing the already paid portion of traffic volume with both charging models. These results have provided further solid evidence that *Jetway* works effectively when it comes to utilizing the *"free of charge"* bandwidth that is available during later time intervals in the same charging period, made possible by any percentile-based charging models typically used by ISPs.

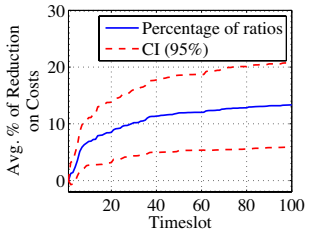


Figure 13: Average percentage of cost reduction with the 100-th percentile charging model.

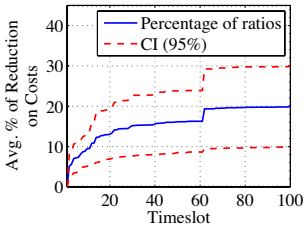


Figure 14: Average percentage of cost reduction with the 95-th percentile charging model.

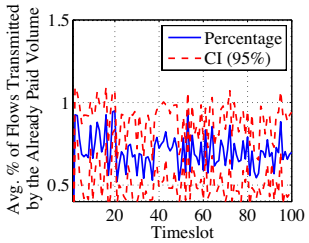


Figure 15: Average percentage of flows carried by utilizing the already paid portion of traffic volume with the 100-th percentile charging model.

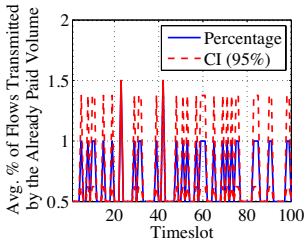


Figure 16: Average percentage of flows carried by utilizing the already paid portion of traffic volume with the 95-th percentile charging model.

5. RELATED WORK

Geographically dispersed datacenters have attracted recent research attention, as they have brought new opportunities and challenges. Most recent works focus on dynamic load distribution across datacenters, with the objective of minimizing cloud providers' operational costs on energy or minimizing cloud users' performance penalty. Wu *et al.* proposed to migrate social media applications into geo-distributed clouds operated by one or more cloud providers, and designed an online content migration and request distribution algorithm [18]. Liu *et al.* sought to reduce the total energy use of a cloud by geographical load balancing [19]. In terms of improving the operational flexibilities on inter-datacenter communication for cloud providers, GRIPhoN is proposed to offer cost-effective restoration capabilities from the carrier's perspective [20].

Besides analytical results, there are also a few measurement studies regarding the traffic characteristics across multiple datacenters. [21] and [22] shed some light on good designs of caching and load balancing strategies through measuring traffic dynamics in the Google cloud. Chen *et al.* presented a first study of inter-datacenter traffic characteristics via five Yahoo! datacenters [23]. Their measurement results motivate our study of reducing costs on inter-datacenter video traffic.

To our knowledge, there exist two recent papers that considered cloud providers' operational costs on traffic. Zhang *et al.* designed a routing algorithm to optimize the costs on datacenter-to-client traffic [6]. However, with no consideration of the time dimension, their problem is substantially simplified. Laoutaris *et al.* proposed NetStitcher, which takes advantage of the already paid traffic volumes at night to reduce costs on inter-datacenter bulk traffic, such as backups [9]. Our work differs in that, instead of conservatively utilizing leftover bandwidth only for bulk transfers, we argue that such costs can — and should — be minimized by globally optimizing the routing strategies for all inter-datacenter video traffic. In

addition, *Jetway* has considered the co-existence of multiple video flows in its problem formulation, which is far more realistic and complicated than the transfer of a single file in NetStitcher.

6. CONCLUDING REMARKS

To stream videos to end users, it is now the norm for video streaming service providers, such as Netflix, to use services of cloud providers, such as Amazon Web Services. To provide a better performance, the cloud provider typically deploys multiple datacenters across different geographical regions, and connect them with an inter-datacenter network. Inter-datacenter traffic is typically charged by ISPs based on a percentile-based charging model, with which cloud providers pay based on the q-th percentile of traffic volumes measured in a short time interval, over a number of such intervals in a charging period. Such a model implies that, if traffic has already been generated during one time interval, up to the same volume of traffic may be carried *free of charge* in subsequent time intervals.

In this paper, we have presented *Jetway*, designed to minimize costs on inter-datacenter video traffic by splitting and routing video flows over multiple multi-hop paths. *Jetway* takes full advantage of our key observation that some of the traffic volumes can be transferred free of charge, while the desired transmission rates of video flows remain satisfied. In order to design *Jetway*, we have formulated the problem of minimizing costs with the intent of maintaining its tractability and the practicality of our solutions. We have evaluated the performance of *Jetway* using our real-world implementation, with actual traffic flowing across seven Amazon EC2 datacenters around the world. We have shown that *Jetway* is capable of reducing traffic costs, while maintaining satisfactory performance with respect to both throughput and end-to-end delay.

7. ACKNOWLEDGMENTS

This research was supported in part by the SAVI Strategic Network (NSERC), and by a grant from NSFC/RGC under the contract N_HKUST610/11, by grants from HKUST under contracts RPC11EG29, SRFI11EG17-C, and SBI09/10.EG01-C, and by a grant from ChinaCache Int. Corp. under the contract CCNT12EG01.

8. REFERENCES

[1] A. Cockcroft, "Netflix in the Cloud," 2011. [Online]. Available: http://velocityconf.com/velocity2011/public/schedule/detail/17785

[2] 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 *Proc. IEEE INFOCOM*, 2012.

[3] Amazon, "Amazon CloudFront," 2012. [Online]. Available: http://aws.amazon.com/cloudfront/

[4] ——, "Amazon S3: Object Size Limit Now 5 TB," December 2010. [Online]. Available: http://aws.typepad.com/aws/2010/12/amazon-s3-object-size-limit.html

[5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[6] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing Cost and Performance in Online Service Provider Networks," in *Proc. 7th Conference on Networked Systems Design and Implementation (NSDI)*, 2010, pp. 1–15.

[7] I. RouteScience Technologies, "Route Optimization for Ebusiness Applications," *White Paper*, 2003.

[8] V. Valancius, C. Lumezanu, N. Feamster, R. Johari, and V. V. Vazirani, "How Many Tiers? Pricing in the Internet Transit Market," in *Proc. ACM SIGCOMM*, 2011, pp. 194–205.

[9] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-Datecenter Bulk Transfers with NetStitcher," in *Proc. ACM SIGCOMM*, 2011.

[10] D. K. Goldberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang, "Optimizing Cost and Performance for Multihoming," in *Proc. ACM SIGCOMM*, 2004, pp. 79–92.

[11] T. Benson, A. Anand, A. Akella, and M. Zhang, "The Case for Fine-Grained Traffic Engineering in Data Centers," in *Proc. ACM SIGCOMM Workshop: Research on Enterprise Networking (WREN)*, 2010.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

[13] T. Radzik, "Fast Deterministic Approximation for the Multicommodity Flow Problem," in *Proc. 6th Annual ACM Symposium on Discrete Algorithms (SODA)*, 1995, pp. 486–492.

[14] A. Goldberg and R. Tarjan, "Solving Minimum-Cost Flow Problems by Successive Approximation," in *Proc. 9th ACM Symposium on Theory of Computing (SOTC)*, 1987, pp. 7–18.

[15] D. Karger and S. Plotkin, "Adding Multiple Cost Constraints to Combinatorial Optimization Problems, with Applications to Multicommodity Flows," in *Proc. 27th ACM Symposium on Theory of Computing (SOTC)*, 1995, pp. 18–25.

[16] B. Awerbuch and R. Khandekar, "Greedy Distributed Optimization of Multi-Commodity Flows," *Distributed Computing*, vol. 21, pp. 317–329, 2009.

[17] B. Awerbuch, R. Khandekar, and S. Rao, "Distributed Algorithms for Multicommodity Flow Problems via Approximate Steepest Descent Framework," in *Proc. 18th Annual ACM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 949–957.

[18] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. Lau, "Scalling Social Media Appliations into Geo-Distributed Clouds," in *Proc. IEEE INFOCOM*, 2012.

[19] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening Geographical Load Balancing," *SIGMETRICS Perform. Eval. Rev.*, vol. 39, pp. 193–204, 2011.

[20] A. Mahimkar, A. Chiu, R. Doverspike, M. D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S. L. Woodward, and J. Yates, "Bandwidth on Demand for Inter-Data Center Communication," in *Proc. 10th ACM Workshop on Hot Topics in Networks (HotNets)*, 2011, pp. 1–6.

[21] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving Beyond End-to-End Path Information to Optimize CDN Performance," in *Proc. 9th Internet Measurement Conference (IMC)*, 2009, pp. 190–201.

[22] V. K. Adhikari, S. Jain, and Z.-L. Zhang, "YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective," in *Proc. 10th Internet Measurement Conference (IMC)*, 2010, pp. 431–443.

[23] Y. Chen, S. Jain, V. Adhikari, Z.-L. Zhang, and K. Xu, "A First Look at Inter-Data Center Traffic Characteristics via Yahoo! Datasets," in *Proc. IEEE INFOCOM*, 2011, pp. 1620 –1628.