EcoFlow: An Economical and Deadline-Driven Inter-Datacenter Video Flow Scheduling System

Yuhua Lin, Haiying Shen and Liuhua Chen
Department of Electrical and Computer Engineering, Clemson University
Clemson, South Carolina 29634
{yuhual, shenh, liuhuac}@clemson.edu

ABSTRACT

Cloud providers need to transfer video contents between different datacenters in order to provide scalable and consistent service to users across different geographical regions. These inter-datacenter transfers are charged by ISPs under the dominant percentile-based charging models. In order to minimize the payment costs, existing works aim to keep the traffic on each link under the charging volume. However, these methods cannot fully utilize each link's available bandwidth capacity, and may increase the charging volumes. To further reduce the bandwidth payment cost by fully utilizing link bandwidth, we propose an economical and deadline-driven video flow scheduling system, called EcoFlow. Considering different video flows have different transmission deadlines, EcoFlow transmits videos in the order of their deadline tightness and postpones the deliveries of later-deadline videos to later time slots so that the charging volume at current time interval will not increase. The flows that are expected to miss their deadlines are divided into subflows to be rerouted to other under-utilized links in order to meet their deadlines without increasing charging volumes. Experimental results on EC2 show that compared to existing methods, EcoFlow achieves the least bandwidth costs for cloud providers.

Categories and Subject Descriptors

I.6 [**Simulation And Modeling**]: Applications, Model Validation and Analysis

Keywords

Video streaming; Bandwidth cost; Inter-datacenter traffic; Percentile-based charging models

1. INTRODUCTION

Cloud providers (e.g., Amazon) offer various pay-as-you-use cloud based services (e.g., Amazon Web Services) to cloud customers (e.g., Netflix) [1, 2]. Cloud has proved to be an effective infrastructure to host video streaming services with many benefits [3]. In order to enhance service availability and scalability, cloud providers generally deploy a number of datacenters across different geographical regions, which are inter-connected by high-capacity links leased from internet service providers (ISPs). Both server-to-customer video dissemination and video replication lead to a substantial amount of inter-datacenter traffic. Cloud providers purchase transit bandwidth from ISPs based on certain pricing schemes, such as the 95th percentile charging model adopted by most ISPs [4]. In the 95th percentile charging model, the bandwidth cost is charged based on the 95th percentile value in all traffic volumes recorded in every 5-minute interval generated within a charging period (e.g., 1 month [4]). We refer the 95th percentile traffic volume from the beginning of the charging period up to current time as **charging volume**. Many previous studies [5, 6, 4, 7] focus on minimizing the bandwidth payment cost on inter-datacenter video traffic to ISPs.

The *store-and-forward* methods [5, 6] take advantage of the spatial and temporal features of the inter-datacenter video traffic. The spatial feature means that at a specific time, datacenters in different geographic areas exhibit different traffic loads and available bandwidth capacities. The temporal feature means that the traffic loads on a datacenter exhibit strong diurnal patterns that are correlated with the local time [8]. These methods predefine peak and off-peak hours for each datacenter based on its local time and geographic area, and then utilize the leftover traffic volume (which is the charging volume minus the actual traffic volume) during off-peak hours to transfer delay-tolerant data flows. However, the *store-and-forward* methods fail to fully utilize the available bandwidth capacities of the light-traffic links during the peak hours.

The optimal routing path [4, 7] optimize the routing paths for video flows to minimize the charging volume on each link. When the transmission of a video is expected to exceed the charging volume on a link, the video will be transferred over an alternating path to maximize the utilization of other links without increasing their charging volumes. However, these methods transmit each video immediately when the video transmission request arrives at the source datacenter regardless of their deadlines. So these methods can easily increase the charging volumes of some links when a large number of video transfer requests arrive simultaneously.

To handle the problems in previous methods, we propose EcoFlow: an economical and deadline-driven video flow scheduling system. As different applications from cloud cus-

MM'15, October 26–30, 2015, Brisbane, Australia.

DOI: http://dx.doi.org/10.1145/2733373.2806403.

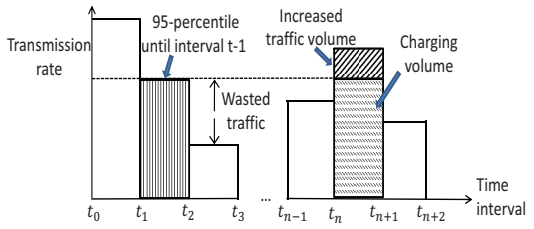


Figure 1: Bandwidth cost of inter-datacenter video traffic.

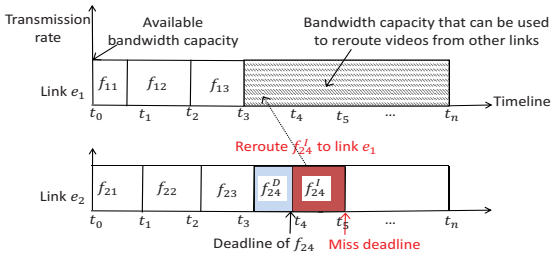


Figure 2: An overview of EcoFlow.

tomers have different service-level agreements (SLAs) that specify data Get/Put bounded latency [9] or a service probability [10] by ensuring a certain number of replicas in different locations [11]. The idea of EcoFlow is to postpone the transfers of delay-tolerant videos while still ensure their transmission within deadlines if the transmission of these videos will increase the current charging volume.

2. OVERVIEW OF ECOFLOW

Figure 1 shows an example of an inter-datacenter link's bandwidth cost under the 95th percentile charging model. The traffic volume in time interval $[t_1, t_2)$ is the 95th percentile value until time t_n, and is marked as the charging volume at t_n. When a larger traffic volume v_{ij} comes up in time interval $[t_n, t_{n+1})$, it becomes the new charging volume at time t_{n+1}. Then, from time t_0 to t_{n+1}, the unused bandwidth below v_{ij} is wasted, that is, the cloud provider does not fully utilized the charging volume. Given this observation, a feasible way to reduce bandwidth cost is to maximize the utilization of the charging volume at different time intervals. For example, the increased traffic volume in time interval $[t_n, t_{n+1})$ can be postponed to time interval $[t_{n+1}, t_{n+2})$.

Three steps of EcoFlow. We first briefly introduce the three steps using an example in Figure 2.

Step 1: **available bandwidth capacity estimation.** We use T_p to denote the time window used to estimate the available bandwidth capacity on each link, and use T_r (T_r<T_p) to denote the time window to record traffic volume in current charging model. Based on historical data, we estimate the total volume of video traffic needed to be transmitted on each link during time interval $[t_0, t_n)$, $t_n - t_0 = T_p$, denoted by $\tilde{v}(t_0, t_n)$. Assume link e_1's charging volume at time t_0 is $\hat{v}_1(t_0)$, it then can transfer a volume of $\hat{v}_1(t_0) \times T_p/T_r$ video during time interval $[t_0, t_n)$. We define **a link's available bandwidth capacity** as the maximum transmission rate that can be used to transfer videos without increasing the current charging volume during a certain time interval. We then calculate the available bandwidth capacity $\Delta c_1(t_0, t_n)$ on link e_1 during time interval $[t_0, t_n)$.

Step 2: **deadline-driven flow scheduling.** On each link, the pending video flows are scheduled on an earliest-deadline-first base. On link e_2, f_{24}'s expected transmission time is at t_5, which is later than its deadline. We divide f_{24} into two subflows: f_{24}^D and f_{24}^I. On link e_1, all pending videos are scheduled to finish transmission before t_3, its available capacity during $[t_3, t_n)$ is not utilized (highlighted in dashed fill). We call the available bandwidth capacity that are not utilized during $[t_3, t_n)$ **extra bandwidth capacity** ($\delta c_1(t_3, t_n)$), $\delta c_1(t_3, t_n)$=$\Delta c_1(t_0, t_n)$. We define the links with extra bandwidth capacity as the under-utilized links. The extra bandwidth capacity on link e_1 can be utilized to reroute subflow f_{24}^I from e_2 by its deadline.

Step 3: **routing path identification.** For the video rerouting, we identify an alternating path that has extra bandwidth capacity to transmit the video by its deadline.

3. SYSTEM DESIGN

3.1 Available Bandwidth Capacity Estimation

We first use Exponentially weighted moving average (EWMA) [12] to estimate the traffic volume during $[t_i, t_i + T_p]$ on link e_{ij} (denoted by $\tilde{v}_{ij}(t_i, t_i+T_p)$) based on the actual historical traffic volume. During time interval $[t_i, t_i + T_p)$, a total volume of $\hat{v}_{ij}(t_i) \times T_p/T_r$ video traffic can be transferred under the current bandwidth cost. Given estimated traffic volume $\tilde{v}_{ij}(t_i, t_i + T_p)$, we can calculate the available bandwidth capacity in time interval $[t_i, t_i + T_p)$:

$$\Delta c_{ij}(t_i, t_i + T_p) = \min\{c_{ij}, \hat{v}_{ij}(t_i)/T_r - \tilde{v}_{ij}(t_i, t_i + T_p)/T_p\}. \quad (1)$$

When $\Delta c_{ij}(t_i, t_i + T_p) > 0$, the current charging volume on link e_{ij} is larger than the expected traffic volume, and the available bandwidth capacity can be used to reroute video flows from other links.

3.2 Deadline-driven Flow Scheduling

Like existing work [7] that assumes the existence of a centralized server connecting to all datacenters that schedules the video flows in all datacenters, we first introduce EcoFlow in a centralized manner. The network scheduler maintains a sending queue $Q(t_i)$=($< f_1, d_1, s_1 >$,$< f_2, d_2, s_2 >$,... $< f_m, d_m, s_m >$) to store all pending flows on each link e_{ij} at time t_i, which are ordered based on their deadlines. Note that flows on link e_{ij} includes all flows that are transmitted bidirectionally between datacenter i to j. Each triple $< f_k, d_k, s_k >$ in $Q(t_i)$ contains the flow information of f_k, where d_k and s_k are the deadline and size of f_k, respectively.

All pending videos in $Q(t_i)$ are sent out sequentially, and the flow transmission time for flow f_k can be computed as:

$$T_k = \mathcal{A}/\Delta c_{ij}(t_i, t_i + T_p). \quad (2)$$

$\mathcal{A} = \sum_{f_p \in F_{<k}} s_p$, where $F_{<k}$ is a subset of flows in $Q(t_i)$ that have earlier deadlines than flow f_k (including f_k), and $\Delta c_{ij}(t_i, t_i + T_p)$ is the estimated available bandwidth capacity on link e_{ij} in time interval $[t_i, t_i + T_p)$. The flow completion time for f_k is t_k^{end}:

$$t_k^{end} = \begin{cases} t_i + T_k & \text{If k=1} \\ t_{k-1}^{end} + T_k & \text{otherwise} \end{cases} \quad (3)$$

The flow start time for f_1 is t_i. For flow f_k ($k > 1$), the flow start time is the completion time of the previous flow,

that is, $t_k^{start} = t_{k-1}^{end}$. d_k is the deadline of video f_k. When $t_k^{end} \leq d_k$, flow f_k is expected to finish transmission before its deadline, and we call it a **Direct Flow** (DF). When $t_k^{end} > d_k$, flow f_k is likely to miss the transmission deadline under the expected bandwidth capacity. Then, f_k can be split to two subflows: f_k^D and f_k^I. f_k^D is the volume with size s_k^D that can be transmitted directly on link e_{ij} before its deadline; while f_k^I is the residual volume with size s_k^I (s_k^I=$s_k - s_k^D$), which should be rerouted in an alternating path. We call f_k^I an **Indirect Flow** (IF).

$$s_k^D = max(0, d_k \times \Delta c_{ij}(t_i, t_i + T_p) - \mathcal{A}). \quad (4)$$

Using the alternating routing path identification method in Section 3.3, we identify alternating paths for each indirect flow f_k^I which can transmit f_k^I before its deadline.

Algorithm 1 Pseudocode for identifying alternating path for flow f_k^I.

```
1: Input: G = (V, E); s_k, δc_ij(t_i, t_i + T_p), ∀ij ∈ E;
2: Output: alternating path P from source i to destination j
3:   for each vertex u in V:
4:     rate[u] := 0   //Maximum transmission rate on each path
5:     pre[u] := null   //Record last hop on the path
6:     t_k^end[u] := t_i   //Transmission completion time
7:     Q+ = j   //Q is a temporal set
8:   end for
9:   rate[i] := infinity
10:  while Q is not empty do:
11:     u := vertex in Q with max rate[u]
12:     remove u from Q
13:     for each neighbor v of u with δc_uv(t_i, t_i + T_p) > 0 do:
14:       t_k^end[v]:=s_k^I/min{rate[u], δc_uv(t_i, t_i + T_p)} + t_i
15:       alt :=max{rate[v], δc_uv(t_i, t_i + t_k^end[v])}
16:       if alt ≥ rate[v]:  //A path with higher transmission rate
17:           rate[v] := alt
18:           pre[v] := u
19:       end if
20:     end for
21:  end while
22:  P := empty sequence
23:  while pre[j] is defined do:   //Construct the alternating path
24:      insert j at the beginning of P
25:      δc_{j,pre[j]}(t_i, t_i + t_k^end[j]) := 0
26:      j := pre[j]   // Traverse from destination to source
27:  end while
28:  if t^end[v]<d_k: Return P
29:  if t^end[v]>d_k:   //P cannot transmit f_k^I before its deadline
30:    split f_k^I into f_k1, f_k2
31:    Return P, f_k1, f_k2
32:  end if
```

3.3 Alternating Routing Path Identification

$F^I(t_i)$ denotes the set of all IFs in the network at time t_i, which are sorted by their deadlines in ascending order. Assume f_k^I is an IF from datacenter i to datacenter j, in this section, we describe how the scheduler identifies an alternating routing path P for f_k^I, $P = (v_1, v_2, \ldots, v_p)$.

When f_k^I is transmitted on path P, its transmission rate is the minimum extra bandwidth capacity on all P's constituent edges, that is $\min_{\forall i \in (1,p-1)}\{\delta c_{i,i+1}(t_i, t_i+T_p)\}$. f_k^I's transmission completion time t_k^{end} is calculated by:

$$t_k^{end} = s_k^I / \min_{\forall i \in (1,p-1)}\{\delta c_{i,i+1}(t_i, t_i + T_p)\} + t_i. \quad (5)$$

s_k^I denotes the flow size of f_k^I and $\delta c_{i,i+1}(t_i, t_i + T_p)$ denotes the extra bandwidth capacity on link $e_{i,i+1}$. We then express the requirement that the transmission of flow f_k^I on path P would be finished before its deadline by: $t_k^{end} \leq d_k$.

We develop a modified Dijkstra's algorithm shown in Algorithm 1 to identify an alternating routing path for f_k^I. In this algorithm, we input the flow information including its size, deadline, source datacenter and destination datacenter, together with network information including all link's extra bandwidth capacity. Algorithm 1 will return an alternating path P for f_k^I, and splits f_k^I into two parts if P cannot finish transmission before its deadline. If the identified path can transmit f_k^I before its deadline, alternating path P is returned (Line 23); otherwise f_k^I is split into f_{k1} and f_{k2} (Line 24-26). The simplest implementation of the Dijkstra's algorithm requires a running time of $O(|E| + |V|^2) = O(|V|^2)$.

If P cannot finish f_k^I's transmission before its deadline and f_k^I is split into f_{k1} and f_{k2}, we will use Algorithm 1 to identify an alternating path for f_{k2}. If the new identified alternating path cannot transmit f_{k2} before its deadline, f_{k2} is further split into two parts: f_{k2}^1 and f_{k2}^2. f_{k2}^1 is transmitted on the new alternating path and f_{k2}^2 is transmitted on e_{ij} by increasing the charging volume on e_{ij}. The new charging volume $\hat{v}_{ij}(t_i)$ at time t_i is calculated by:

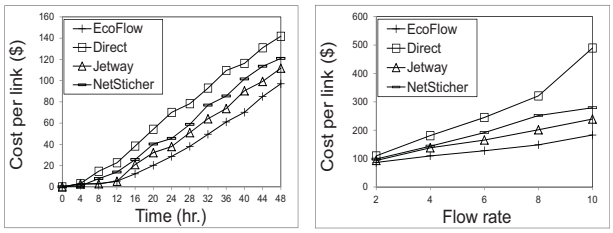
$$\hat{v}_{ij}(t_i) = (\mathcal{A} - s_k + s(f_{k2}^2) \times T_r/(d_k - t_i). \quad (6)$$

Where $s(f_{k2}^2)$ is the size of f_{k2}^2. The flow start time and completion time of all flows on link e_{ij} will then be updated based on Equation (2), and the schedule table $S(t_i) =< f_k, S, D, t_k^{start} >$ will also be updated.

4. PERFORMANCE EVALUATION

We conducted experiments on Amazon EC2 platform [13], which has a total of 7 datacenters, the capacity and cost per traffic unit of each link are set according to the studies in [7]. We compare EcoFlow with three datacenter traffic scheduling strategies: 1) Direct transfer (denoted as Direct), which directly transfers video flows to the destination whenever the video transfer requests are initiated by the cloud provider without considering each link's charging volume; 2) JetWay [7] and 3) NetSticher [6]. We defined two types of videos: Standard Definition (SD) videos with sizes randomly selected in [500, 800] MB, and High Definition (HD) videos with sizes randomly selected in [2, 4] GB [7]. For simplicity, we assumed that 10-12am and 6pm-12am of a node's local time are peak hours. A datacenter transfers x and y videos per hour to all other datacenters during its peak hours and off-peak hours, respectively, where x and y were randomly selected from [2, 5] and [0, 1], respectively. The transfer request of each video is initiated at a random time during the selected hours, and its deadline is chosen in [30, 120] minutes after the transfer request's initiated time. We set $T_p = 1$ hour and $T_r = 5$ minutes. We set a 48 hour period as an independent charging period and calculated the bandwidth cost on each link at the end of the experiment. In EcoFlow, we had a 48 hour warmup period and used the traffic records to predict the traffic volume during the charging period.

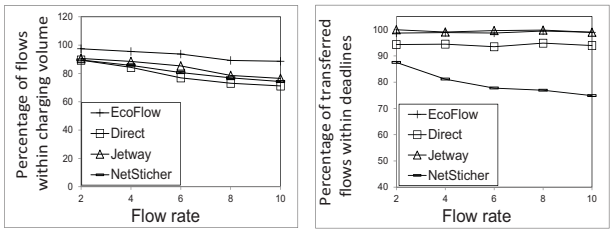
We defined a metric of *bandwidth cost per link* as the sum of bandwidth payment cost on all links divided by the total number of links in the network. Figure 3(a) shows the average cost per link at different time intervals for the 95th percentile charging models. We see that the per link cost follows: EcoFlow<JetWay<NetSticher<Direct. Direct results in the highest bandwidth cost as it immediately transfers the video by using only the direct link between two datacenters without considering the current charging volume on



(a) Results at different time intervals.

(b) Results at at different flow rates.

Figure 3: Average bandwidth cost.



(a) Average percentage of flows transmitted within the charging volume.

(b) Percentage of transferred flows within deadlines.

Figure 4: Results at at different flow rates.

the link. NetSticher postpones the transmission of delay-tolerant videos until both source datacenter and destination datacenter are during off-peak hours, so that the traffic load during peak hours is alleviated. However, as the available bandwidth capacity is not fully utilized during peak hours, there is still room for NetSticher to further reduce the bandwidth cost. JetWay is able to incur less bandwidth cost than NetSticher by controlling the transmissions of current videos within the charging volume. EcoFlow generates the least bandwidth cost among all comparison methods as it transmits each video with the link's available bandwidth capacity. Next, we changed the flow arrival rates during a link's peak hours from 2 to 10 flows per hour on each link. Figure 3(b) shows the average bandwidth cost per link at the end of the 48-hour charging period under different flow rates. The relative performance of different methods concurs that in Figure 3(a) due to the same reason.

Figure 4(a) shows the average percentage of flows transmitted within the charging volume at different flow rates. We see that performance of different methods with respect to average percentage of flows transmitted within the charging volume follows: EcoFlow>JetWay>NetSticher>Direct. Figure 4(b) show the percentage of video flows that are transferred within their deadlines at different flow rates. We see that the result follows: JetWay>EcoFlow>Direct> NetSticher. EcoFlow and JetWay generate comparably high percentage of transferred videos within the deadlines, as they both use the available bandwidth capacities from all links to finish the video's transmission before its deadline.

5. CONCLUSIONS

To provide video streaming services to users across different regions, cloud providers need to transfer video contents between different datacenters. Such inter-datacenter transfers are charged by ISPs under the percentile-based charging models. We take advantage of the particular characteristic of these models and propose EcoFlow to minimize cloud providers' payment costs on inter-datacenter traffics. It first calculates the under-utilized traffic volume on each link, then schedules video flows with the objective that these flows do not incur additional charges on the link and guaranteing that each video flow meets its transmission deadline. Finally, the under-utilized links with low traffic burden are used to build alternating paths for video flows that are estimated to miss their deadlines. Experimental results on EC2 show the effectiveness of EcoFlow in reducing bandwidth costs for inter-datacenter video transfers.

6. ACKNOWLEDGMENTS

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, and Microsoft Research Faculty Fellowship 8300751.

7. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[2] S. Rajani and T. Rajender. Literature review: Cloud computing-security issues, solution and technologie. *International Journal of Engineering Research*, 3(4):221–225, 2014.

[3] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *Proc. of INFOCOM*, 2012.

[4] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *Proc. of SIGCOMM*, 2004.

[5] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram. Delay tolerant bulk data transfers on the internet. In *Proc. of SIGMETRICS*, 2009.

[6] L. Nikolaos, S. Michael, X. Yang, and R. Pablo. Inter-datacenter bulk transfers with netstitcher. In *Proc. of SIGCOMM*, 2011.

[7] Y. Feng, B. Li, and B. Li. Jetway: Minimizing costs on inter-datacenter video traffic. In *Proc. of Multimedia*, 2012.

[8] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, 2009.

[9] A. Hussam, P. Lonnie, and W. Hakim. Racs: A case for cloud storage diversity. In *Proc. of SoCC*, 2010.

[10] Service Level Agreements. http://azure.microsoft.com/en-us/support/legal/sla/, [Accessed in Apr, 2015].

[11] Amazon S3. http://aws.amazon.com/s3/, [Accessed in Apr. 2015].

[12] J. Lucas and M. Saccucci. Exponentially weighted moving average control schemes: Properties and enhancements. *Technometrics*, 32(1):1–29, 1990.

[13] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/, [Accessed in Apr. 2015].