Journal of Engineering Sciences
Vol 10, Issue 9, Sept /2019
ISSN NO:0377-9254

Minimizing Costs On Scheduling Inter-Datacenter Video Flows

[1]K.Samson Paul, [2]Dr.C.Gulzar , [3]A.Emmanuel Raju
[1]Assistant Professor ,[2]Associate Professor , [3]Assistant Professor
DEPARTMENT OF CSE
Dr. K.V. SUBBA REDDY INSTITUTE OF TECHNOLOGY,KURNOOL

Abstract

As video streaming applications are deployed on the cloud, cloud providers are charged by ISPs for inter-datacenter transfers 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. As a solution, we propose an economical and deadline-driven video flow scheduling system, called EcoFlow. Considering that 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. The flows that are expected to miss their deadlines are divided into sub flows to be rerouted to other under-utilized links. We also propose setting each link's initial charging volume to reduce the scheduling latency at the beginning of the charging period and discuss how to deal with issues such as the prediction errors of link available bandwidth and the lack of charging volume's prior knowledge. Furthermore, we designed implementation strategies for using EcoFlow in both centralized and distributed situations.

I. INTRODUCTION

Cloud providers (e.g., Amazon) offer various pay-as-youuse cloud based services (e.g., Amazon Web Services) to cloud customers (e.g., Netflix) [1], [2]. The Cloud has proved to be an effective infrastructure to host video streaming services with many benefits [3]–[5]: First, cloud based infrastructures provide easy to use scalablity for hosting additional users and content by allowing cloud customers to buy additional resources from the cloud provider [6]. Second, distribution is similarly scalable and in this context is provided by a large number of globally distributed servers and streaming service providers (VSSPs) that allow for services to be provided in a vast array of different areas [7]. Lastly, cloud based infrastructures provide data-center management for their cloud customers, and as such are low cost way for VSSPs to deliver their streaming services [8]– [10]. As a result, a progressively higher number of VSSPs, such as Netflix, are deploying the web applications on the cloud. In order to enhance service availability and scalability, cloud providers generally deploy a number of datacenters across different geographical regions, which are interconnected by high-capacity links leased from internet service providers (ISPs). Newly published videos and their replicas are allocated to these distributed datacenters to serve users from different regions. Specifically, when a new video is uploaded to a datacenter, the datacenter disseminates it to other datacenters to serve users. When the number of video replicas is insufficient to allow for a streaming service that is scalable, accessible, and widely available, more video replications occur between datacenters. Video dissemination and video replication leads to a substantial amount of inter-datacenter traffic [11]. It is important to note, however, that this traffic does not include datacenter-to-customer video traffic. ISPs charge cloud providers for transit bandwidth using a wide array of pricing schemes. The most common of these, and the model adopted by most ISPs, is the 95th percentile charging model [12]. In this model the bandwidth cost is charged based on the 95th percentile value in all traffic volumes (data sizes) recorded in 5-minute intervals generated within a charging period (e.g., 1 month [12]). In the 95th percentile charging model, "Charging Volume" is defined as the volume of traffic from the beginning of the charging period up to the current time. Numerous previous studies focused on controlling new traffic volumes to ensure that they stayed below the charging volume [11]–[12] in order to minimize bandwidth payment costs on inter-datacenter video traffic to ISPs. These previous studies can primarily be classified into to groups: store-and-forward and optimal routing path.

The store-and-forward methods take advantage of unique spatial and temporal attributes of inter-datacenter video traffic. Spatial attributes refer to datacenters in different geographic areas and their traffic loads and available bandwidth capacities, which are highly dependent upon the user demands within those different areas. Temporal attributes relate to the loads placed on a datacenter during a given time period, and more specifically regard the strong diurnal patterns that correlate with a given local time [18]. Store-and-forward methods predefine peak and off-peak hours for each datacenter based

primarily on these two aspects - local time and geographic area - and subsequently utilize leftover traffic volume (charging volume minus actual traffic volume) during off-peak hours to transfer delay tolerant data flows. A hypothetical datacenter, i, on the East Coast of the United States (EST/GMT-5), along with datacenters j, in Chile (EST+1/GMT-4), and k, in Belgium (EST+6/GMT+1), each with peak hours of 9-12pm local time and off-peak hours of 3-6am local time, will serve as an example. Suppose that a number of delay-tolerant data flows need to be sent from datacenter i to datacenter j, however, it happens to be 9pm EST meaning that both i and j are in peak hours. It can subsequently be inferred that no overlap exists between their respective peak and off-peak hours. That is, the peak hours of i do not align with the off-peak hours of j and that the reverse is also true. The delay-tolerant data flows will then need to be sent to an intermediate datacenter, in this case datacenter k, which does have off-peak hours that overlap with the peak hours of datacenter i. These data flows will then be temporarily stored in datacenter k which will forward them to datacenter j when both k and j are in off-peak hours. The optimal routing path methods [11], [12] optimize the routing paths for video flows to minimize the charging volume on each link. The bandwidth costs of transmitting the same amount of videos vary across different inter-datacenter links. Accordingly, if the transmission of a video is expected to exceed the charging volume on a link, the video can be transferred to an alternative 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 its deadline. As a result, these methods can easily reach the charging volume of a current link and create a large number of reroute requests when a large number of video transfer requests arrive simultaneously. This may also increase the charging volumes of some links. An additional consequence of using optimal routing path methods concerns the difficulty in fully utilizing a link's available bandwidth capacity.

That is, a links available bandwidth capacity is not fully utilized when the cumulative transmission rate of all the currently transmitted videos is less than the link's available bandwidth capacity. We propose Eco Flow, an economical and deadline-drive video flow scheduling system, to address the problems of previous scheduling methods. It is based on the fact that different video flows have different deadlines. Different applications from cloud customers have different service level agreements (SLAs) that specify data Get/Put bounded latency [19] or a service probability by ensuring a certain number of replicas in different locations. Thus, cloud providers would like to assign shorter transmission deadlines (deadline) to videos in applications with more stringent SLAs in order to minimize the SLA violation penalty to maximize their profits [11]. Different videos in one application also have different deadlines. For example, the flows for new video dissemination to a datacenter to serve user requests should have more stringent deadlines than the flows for video replication backups to boost availability. Based on the different deadlines of video flows, the key idea of Eco Flow is to postpone the transfers of some delay-tolerant videos while still ensuring their transmissions within deadlines if the transmission of these videos will increase the current charging volume. This is the novelty of Eco Flow – to significantly reduce the payment cost of the previous flow scheduling methods. The Eco Flow system includes three key steps. Step 1: Available Bandwidth Capacity Estimation. We estimate the available bandwidth capacity on each link by comparing the charging volume and the expected traffic volume. This provides the maximum transmission rate that a link can provide in the next time interval without increasing the current bandwidth cost. Step 2: Deadline-Driven Flow Scheduling. We sort the flows on each link based on their deadline tightness. In the sorted flow queue, we generally give videos with .earlier deadlines higher priority in finishing transmission, and postpone the transfers of some delay-tolerant videos while still ensuring their transmissions occur within deadlines if the transmission of these videos will increase the current charging volume. Flows that are expected to miss their deadlines are split into sub flows, which will be rerouted to other underutilized links in order to meet their deadlines. Step 3: Alternating Routing Path Identification. In order to deliver these sub flows by their deadlines, we rely on Dijkstra's algorithm [23] to find the shortest path between the source and the destination datacenters in the inter datacenter network that guarantees the successful transmission by flow deadlines. In summary, existing flow scheduling methods fail to fully utilize each link's available bandwidth capacity, and may increase the charging volumes, and thus cannot reduce bandwidth costs maximally for cloud providers. Compared to existing methods, the advantage of Eco Flow lies in 1) the reduction in overall bandwidth cost as much as possible, and 2) video flows can finish transmission before their deadlines through fine-grained (i.e., hourly) estimation of the traffic load on each link and taking advantage of various flow deadlines in flow scheduling. Note that Eco Flow is a heuristic rather

than an optimal solution and it outperforms existing methods.

II. RELATED WORK

Many methods have been proposed to schedule the inter datacenter traffic in order to minimize the ISP bandwidth costs of cloud providers, which can be classified in two groups: store-and-forward and optimal routing path. Store-and-forward. The methods in this group postpone the transmissions of the delay-tolerant data flow from peak hours to off-peak hours, so as to utilize the leftover traffic during off-peak hours. Laoutaris et al. [13] proposed to employ a number of storage servers to collect delay-tolerant traffics and perform data transmission only when the destination datacenter is in predefined off-peak hours, so that the charging volume will not increase during peak hours. Net Sticher [15] performs transmissions of delay-tolerant data between two datacenters only when both datacenters are in off-peak hours. If there are no common off-peak hours between both datacenters, it uses an intermediate datacenter that has an overlap in off-peak hours with the destination datacenter as a relay datacenter to store the delay-tolerant data temporarily. Such store-and-forward transfer systems predefine offpeak hours of each datacenter. Delaying the transmission of delay-tolerant videos from peak hours to off-peak hours is a coarse-grained scheduling strategy. It does not fully utilize the link's available bandwidth capacity when actual traffic load is light during peak hours. Additionally, the transmission of a large number of non-delay-tolerant videos during the peak hours will increase the link's charging volume. Instead, EcoFlow is a fine-grained video flow scheduler which estimates the available bandwidth capacity on each link during a short time interval (i.e., 1 hour), and schedules the pending flows using a link's available bandwidth capacity in an earliest-deadline-first manner. The flows expected to miss their deadlines are rerouted to other under-utilized links to avoid increases in the current charging volume

Existing System

In the existing system, EcoFlow adopts some existing techniques such as sub flow and rerouting. It also adopts the earliest-deadline-first flow scheduling strategy. While these techniques have previously been applied with goals such as minimizing mean flow completion time or meeting flow deadlines, they do not inherently consider the payment costs for flow transfers between datacenters.

III. OVERVIEW OF ECOFLOW:

Problem Statement In order to examine the need for EcoFlow consider a cloud with multiple geographically distributed datacenters, operated by a single cloud provider, where ever datacenter in the cloud is connected to every other datacenter. Important notations used in this paper are listed in Table 1, however, a few critically important notations to begin this discussion include using a complete direct graph G = (V, E) to represent the inter-datacenter network, where V is the set of datacenters and E denotes the set of direct links connecting datacenters. We use ej,k ∈ E to denote a link from datacenter j to datacenter k. For each link ej,k, we use a positive value a to denote the cost per traffic unit, a nonnegative value c to denote the maximum link capacity (the maximum available transmission rate), and vˆ to denote the charging volume. We use Tr to denote the time window to record traffic volume (i.e., 5-minute interval) for calculating the charging volume, and use P c j,k(ti) to denote the bandwidth cost on link ej,k at time ti . We let vˆ(ti) represent the charging volume on link e at time ti and let v(ti−1, ti) present the total actual traffic during time interval [ti−1, ti). Assuming the charging period begins at time t0, the bandwidth cost on link e at time ti can be calculated by:

$$P^c_{j,k}(t_i) = \begin{cases} a\frac{\hat{v}(t_{i-1})}{T_r}(t_i - t_{i-1}) & \text{If } v(t_{i-1}, t_i) < \hat{v}(t_{i-1}) \\ a\frac{\hat{v}(t_i)}{T_r}(t_i - t_{i-1}) & \text{otherwise} \end{cases} \quad (1)$$

As shown in Equation (1), when the actual traffic volume during [ti−1, ti) is smaller than the charging volume at time ti−1, the bandwidth cost at time ti is calculated by applying the cost function to the charging volume at time ti−1; otherwise, the new bandwidth cost at time ti is calculated by applying the cost function to the new charging volume at time ti . Suppose that there are M video flows f1, ..., fm, ..., fM to be scheduled, where the starting time and the completion time of each fm are represented by t start m and t end m , respectively. We use u src m and u dst m to denote the source and the destination of fm, and use sm to denote the size of fm. Each flow is allowed to be partitioned into several subflows and to be delivered via multiple paths. We use x j,k m,i to represent the size of fm's subflow that is allocated to link ej,k at time [ti−1, ti). Then, the following conditions must be satisfied: For each flow fm, C1: The total data flowing out of the source u src m after t start m equals to sm: P i≥t start m P j∈V \usrc m x u src m ,j m,i = sm. C2: The total data flowing into the destination u dst m by time t end m equals to sm: P i≤t end m P j∈V \udst m x j,udst m m,i = sm. For each flow fm and each datacenter j that is neither source nor destination of fm, C3: The total data in fm flowing into j equals to the data flowing out of j for the whole time span: P i P l∈V \j x l,j m,i = P i P k∈V \j x j,k m,i C4: At each time tτ , the total data in fm flowing into j should be no smaller than the data flowing out of

j: P i≤τ P l∈V \j x l,j m,i ≥ P i≤τ P k∈V \j x j,k m,i.

Journal of Engineering Sciences
Vol 10, Issue 9, Sept /2019
ISSN NO:0377-9254

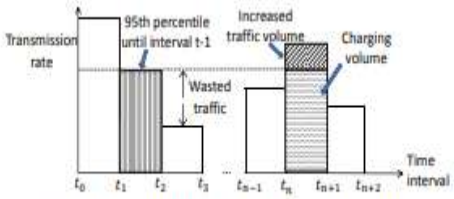


Fig. 1: Example: inter-datacenter video traffic bandwidth cost.

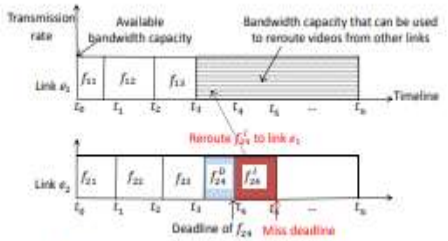


Fig. 2: An overview of EcoFlow.

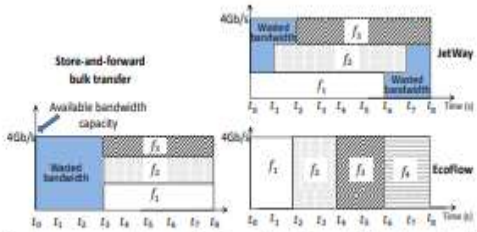


Fig. 3: A comparison of bandwidth utilization between different methods.

IV. SYSTEM DESIGN:

Proposed System

In this paper, we propose EcoFlow, an economical and deadline-drive video flow scheduling system, to address the problems of previous scheduling methods. It is based on the fact that different video flows have different deadlines. Different applications from cloud customers have different service level agreements (SLAs) that specify data Get/Put bounded latenc or a service probability by ensuring a certain number of replicas in different locations. Thus, cloud providers would like to assign shorter transmission deadlines (deadline) to videos in applications with more stringent SLAs in order to minimize the SLA violation penalty to maximize their profits.

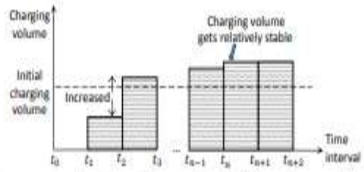


Fig. 4: An example of setting an initial charging volume at the beginning of a charging period.

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

Input: $G = (V, E)$; s_k, $\delta c(t_i, t_i + T_p), \forall e \in E$;
Output: alternating path P from source i to destination j
  for each vertex u in V
    $rate[u] := 0$ //The maximum transmission rate on each path
    $pre[u] :=$ null //Record last hop on the path
    $t_k^{end}[u] := t_i$ //Transmission completion time
    $Q+ = j$ //Q is a temporal set
  end for
  $rate[i] :=$ infinity
  while Q is not empty **do**
    u := vertex in Q with max rate[u]
    remove u from Q
    for each neighbor v of u with $\delta c_{uv}(t_i, t_i + T_p) > 0$ **do**
      $t_k^{end}[v] := s_k^I / \min\{rate[u], \delta c_{uv}(t_i, t_i + T_p)\} + t_i$
      alt :=max$\{rate[v], \delta c_{uv}(t_i, t_i + t_k^{end}[v])\}$
      if alt $\geq rate[v]$: // A path with higher transmission rate
        $rate[v] :=$ alt
        $pre[v] :=$ u
      end if
  $P :=$ empty sequence
  while $pre[j]$ is defined **do** //Construct the alternating path
    insert j at the beginning of P
    $\delta c_{j,pre[j]}(t_i, t_i + t_k^{end}[j]) := 0$
    $j := pre[j]$ // Traverse from destination to source
  if $t^{end}[v] < d_k$ **Return** P
  if $t^{end}[v] > d_k$ //P cannot transmit f_k^I before its deadline
    split f_k^I into f_{k1}, f_{k2}
    Return P, f_{k1}, f_{k2}
  end if

Distributed Implementation of EcoFlow

In the centralized implementation of EcoFlow, when the centralized scheduler fails, the scheduling function comes to a halt. In order to prevent the single point of failure problem, we propose a distributed implementation of EcoFlow. Thus, when one scheduler fails, the scheduling system can still function by utilizing other normal schedulers. As mentioned before, each datacenter i has a scheduler Ci . For flow scheduling on link e, we select a scheduler on datacenter i or j as a master scheduler, denoted by scheduler C, and the other schedule then becomes the slave scheduler. Scheduler C is responsible for scheduling transmission of flows on e, calculating the available bandwidth capacity (Δc) on link e using the same technique, and broadcasts this information to all schedulers in the network for alternating path identification. At each time interval Tr, scheduler Ci and scheduler Cj report information of their pending flows on link e (including flow ID, size and deadline) to scheduler C. Scheduler C then orders the flows by their deadlines in ascending order, and calculates start time and completion time for each flow using the same. All flows on link e are divided into DFs (F D(ti)) and IFs (F I (ti)). Scheduler C builds a schedule table S(ti) =< fk, S, D, tstart k > for flows in F D(ti) and forwards it to both scheduler Ci and scheduler Cj . As DFs can be

transmitted directly through e, scheduler Ci and scheduler Cj transfer flows in DFs according to the schedule table. Scheduler C also needs to find the alternating paths for F I (ti) and notifies scheduler Ci and scheduler Cj the alternating paths. Assume f I k is an IF flow from datacenter i to datacenter j, we need to identify an alternating path that can transferDistributed Implementation of EcoFlow In the centralized implementation of EcoFlow, when the centralized scheduler fails, the scheduling function comes to a halt. In order to prevent the single point of failure problem, we propose a distributed implementation of EcoFlow. Thus, when one scheduler fails, the scheduling system can still function by utilizing other normal schedulers. As mentioned before, each datacenter i has a scheduler Ci . For flow scheduling on link e, we select a scheduler on datacenter i or j as a master scheduler, denoted by scheduler C, and the other schedule then becomes the slave scheduler. Scheduler C is responsible for scheduling transmission of flows on e, calculating the available bandwidth capacity (Δc) on link e using the same technique and broadcasts this information to all schedulers in the network for alternating path identification. At each time interval Tr, scheduler Ci and scheduler Cj report information of their pending flows on link e (including flow ID, size and deadline) to scheduler C. Scheduler C then orders the flows by their deadlines in ascending order, and calculates start time and completion time for each flow using the same technique in Section 4.8. All flows on link e are divided into DFs (F D(ti)) and IFs (F I (ti)). Scheduler C builds a schedule table S(ti) =< fk, S, D, tstart k > for flows in F D(ti) and forwards it to both scheduler Ci and scheduler Cj . As DFs can be transmitted directly through e, scheduler Ci and scheduler Cj transfer flows in DFs according to the schedule table. Scheduler C also needs to find the alternating paths for F I (ti) and notifies scheduler Ci and scheduler Cj the alternating paths. Assume f I k is an IF flow from datacenter i to datacenter j, we need to identify an alternating path that can transfer f I k before its deadline. Due to the lack of a centralized scheduler, the challenge of the distributed identification method lies in finding an alternating path through the cooperation of multiple schedulers.

Under this scenario, identifying a path with the minimum transmission time f I k is complicated, so we aim to find a path that can transfer f I k before its deadline. As each datacenter has direct links connected to all other datacenters, a sufficient number of datacenters can be chosen as relay datacenters in the alternating paths. With a high probability, we can find a relay datacenter and build a 2-hop alternating path that can transfer f I k before its deadline. While identifying a multi-hop (more than 2-hop) alternating path requires cooperation of multiple schedulers and is not efficient in time complexity, we aim to simplify the implementation, and achieve algorithm time efficiency, we identify a 2-hop alternating path P = (i, h, j) for f I k in our proposed method, that is, find an intermediate datacenter h and transfer f I k on path P = (i, h, j). Multiple candidate datacenters might be able to relay and transfer f I k before its deadline, we then contact each datacenter's scheduler and randomly choose a datacenter h who are able to relay f I k as the intermediate datacenter. Note that this routing path identification method can be extended to identify alternating paths with more than two hops.

Algorithm 2 Pseudocode for scheduling video flows on link e.

Input: $G = (V, E)$; $Q(t_i)$;
Output: schedule table $S(t_i)$ for all flows in $Q(t_i)$
  if t_i is the beginning of a charging period
    set initial charging volume $\tilde{v}(t_0)$
  end if
  calculate available bandwidth capacity $\tilde{v}(t_i, t_i + T_p)$
  for each $f_k \in Q(t_i)$
    calculate expected transmission time t_k^{end}
    if t_k^{end} is smaller than deadline d_k
      f_k is transmitted directly on link e
    else
      split f_k into subflows: f_k^D and f_k^I
      find alternating path P for f_k^I using Algorithm 1
      update available bandwidth capacity on each link of P
      if P cannot transmit whole volume of f_k^I
        increase $\hat{v}(t_i)$ on e according to Equation (14)
      end if
    end if
  end for
  update scheduling table $S(t_i)$ for f_k and subflows

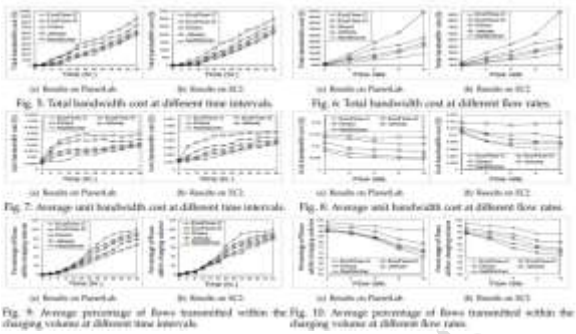
Algorithm 3 Pseudocode of finding intermediate datacenter h.

Input: $\delta c(t_i, t_i + T_p)$, $\forall ij \in E$;
Output: intermediate datacenter h between source i and dest. j
  for each vertex q in $V \backslash j$:
    $t_k^{end} := s_k / \min\{\Delta c_{iq}, \Delta c_{qj}\} + t_i$ //Calculate completion time
    if $t_k^{end} < d_k$: //Guarantee the transmission deadline
      scheduler C_i contacts scheduler C_q
      if $\delta c_{iq}(t_i, t_i + t_k^{end}) > 0$ and $\delta c_{qj}(t_i, t_i + t_k^{end}) > 0$:
        $h := q$
      end if
    end if
  end for

PERFORMANCE EVALUATION:

We conducted experiments on the PlanetLab real-world testbed and Amazon EC2 platform [40] to evaluate the performance of EcoFlow in comparison with other systems. For EcoFlow, We tested EcoFlow as both a centralized scheduler (denoted as EcoFlow-C) and as a distributed scheduler (denoted as EcoFlow-D). We compare the performance of EcoFlow with three datacenter traffic scheduling strategies: 1) Direct transfer (denoted as Direct): This transfer type directly transfers video flows to the destination whenever the video transfer requests are initated by

the cloud provider without considering each link's charging volume. We use it to represent optimal routing path methods to be compared wtih EcoFlow in our evaluation. 2) JetWay [11]: Jetway transfers video flows whenever the video transfer requests are initiated by the cloud provider, at a rate calculated by its size divided by its corresponding maximum tolerable transfer time. When a video flow is expected to increase a link's current charging volume, it splits the video flow into two sub-flows, and the subflows are transmitted along alternating paths to utilize the available bandwidth capacity of each link. 3) NetSticher [15]: NetSticher is a store-and-forward method that transfers delay-tolerant data between two datacenters only when both datacenters are in off-peak hours. When there are no common off-peak hours between both datacenters, an intermediate datacenter is used to store the data temporarily and then forward it to the destination datacenter. In both PlanetLab and EC2 experiments, 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 [11]. We assumed that the traffic load for each datacenter displays a periodic diurnal pattern [15]. For simplicity, we further 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 (including both SD and HD videos) 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 assumed a video with maximum tolerable transfer time longer than 60 minutes to be a delay-tolerant. We set Tp = 1 hour and Tr = 5 minutes. We simulated an inter-datacenter network running for 48 hours for all methods. We set this 48 hour period as an independent charging period and calculated the bandwidth cost on each link at the end of the experiment. In EcoFlow-C and EcoFlow-D, we had a 48 hour warmup period and used the traffic records in this period to predict the traffic volume on each link during the charging period.



V. CONCLUSIONS

To provide video streaming services to users across different regions, cloud providers need to transfer video contents between different datacenters. These inter-datacenter transfers are charged by ISPs under percentile-based charging models. We take advantage of this particular characteristic of these models and propose EcoFlow to minimize cloud providers' payment costs on inter-datacenter traffics. EcoFlow is an economical and deadline-driven video transfer strategy. It first estimates the total volume of video traffic needed to be transmitted between any two datacenters within a time period, compares it with the charging volume and calculates the under-utilized traffic volume on each link. EcoFlow then schedules video flows with the objective that these flows do not incur additional charges on the link, guaranteeing 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. To enhance EcoFlow, we also propose setting each link's initial charging volume to reduce the scheduling latency at the beginning of the charging period. We further discuss how to deal with link available bandwidth prediction errors and lack of prior knowledge of the charging volume. Moreover, we design the implementation of EcoFlow in both centralized and distributed manner. Experimental results on PlanetLab and EC2 show the effectiveness of EcoFlow in reducing bandwidth costs while guaranteeing that each video flow meets its transmission deadline for inter-datacenter video transfers. We will study a more tractable formulation of this bandwidth cost optimization problem in the future. In addition, considering the uncertainty of video flows over time, we will apply more sophisticated approaches like stochastic optimization, Lyapunov, or online algorithms to further improve the performance of our design. Also, we will study how to automatically generate the deadline that satisfies users when it is not indicated.

REFERENCES

[1] A. Khanafer, M. Kodialam, and K. Puttaswamy. To rent or to buy in the presence of statistical

information: The constrained skirental problem. TON, 23(4):1067–1077, 2014.
[2] S. Rajani and T. Rajender. Literature review: Cloud computingsecurity issues, solution and technologie. International Journal of Engineering Research, 3(4):221–225, 2014.
[3] V. Adhikari, Y. Guo, F. Hao, V. Hilt, Z. Zhang, M. Varvello, and M. Steiner. Measurement study of netflix, hulu, and a tale of three cdns. TON, 1(99):1–10, 2014.
[4] X. Liao, L. Lin, G. Tan, H. Jin, X. Yang, W. Zhang, and B. Li. Liverender: A cloud gaming system based on compressed graphics streaming. TON, 1(99):1–10, 2015.
[5] 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.
[6] Auto scaling in the amazon cloud, http://techblog.netflix.com/2012/01/auto-scaling-in-amazoncloud.html, [accessed in sep. 2015].
[7] Q. Zhu and G. Agrawal. Resource provisioning with budget constraints for adaptive applications in cloud environments. In Proc. of HPDC, 2010.
[8] F. Wang, J. Liu, M. Chen, and H. Wang. Migration towards cloudassisted live media streaming. TON, 1(99):1–10, 2014.
[9] Y. Wu, C. Wu, B. Li, and L. Zhang. Scaling social media applications into geo-distributed clouds. TON, 23(3):689–702, 2015.
[10] H. Xu and B. Li. Joint request mapping and response routing for geo-distributed cloud services. In Proc. of INFOCOM, 2013.
[11] Y. Feng, B. Li, and B. Li. Jetway: Minimizing costs on interdatacenter video traffic. In Proc. of Multimedia, 2012.
[12] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In Proc. of SIGCOMM, 2004.
[13] N. Laoutaris and P. Rodriguez. Good things come to those who (can) wait or how to handle delay tolerant traffic and make peace on the internet. In Proc. of HotNets-VII, 2008.
[14] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram. Delay tolerant bulk data transfers on the internet. In Proc. of SIGMETRICS, 2009.