Carbon-Efficient Internet Video Streaming

Zichen Zhu
Rutgers University
Piscataway, NJ, USA
zichen.zhu@rutgers.edu

Tian Guo
Worcester Polytechnic Institute
Worcester, MA, USA
tian@wpi.edu

Sheng Wei
Rutgers University
Piscataway, NJ, USA
sheng.wei@rutgers.edu

***Abstract*—Carbon-efficient computing has drawn significant attention recently aiming to achieve environmental sustainability. Several computation intensive applications, such as deep learning, have been studied in the community for carbon optimizations. However, video streaming, one of the most popular and resource-consuming Internet applications, is under-explored for carbon efficiency. In this paper, we for the first time investigate the carbon efficiency of video streaming systems with the goal of reducing carbon emissions caused by hosting the video streaming service. We develop a dynamic workload migration mechanism utilizing real-time carbon intensity data to select the hosting data center. Furthermore, to minimize the impact on the end-user experience, we consider the migration frequency to maintain service stability when making the migration decisions. The evaluation results indicate significant carbon reductions and acceptable stream switching overhead.**

***Index Terms*—Carbon efficiency, video streaming**

I. Introduction

Carbon-efficient computing has recently drawn significant attention with the goal of achieving the environmental sustainability of computer systems in the era of renewable energy. The community has been actively working on reducing carbon emissions [1]–[3] and achieving net-zero emissions by the next decade [4]. In particular, researchers have established the vision and roadmap of building a sustainable cloud and edge infrastructure [1], [3] for hosting Internet applications. Concurrently, several computation-intensive applications have been studied for carbon efficiency, such as deep learning training [5]–[7] and inference [8], [9], cloud management [3], [10], networking [11]–[14], and hardware devices [15].

Despite the existing efforts, many computation-intensive and thus carbon-emitting applications have not been explored for sustainability. Notably, Internet video streaming, such as video-on-demand or live streaming services provided by YouTube, Netflix, or Twitch, and real-time video streaming used by teleconferencing services like Zoom and WebEx, is one of the most popular Internet applications under-explored for carbon efficiency. Although many research efforts target energy-efficient video streaming designs [16], [17], the existing research mainly focuses on low-power hardware and software system designs without considering the significant carbon emissions at the streaming servers, which often involve a large amount of data center resources processing and serving the video streams at scale [18], [19].

In this paper, we investigate the carbon efficiency of video streaming systems with the goal of designing a dynamic workload migration mechanism to reduce carbon emissions. While an optimal carbon-efficient migration strategy can be devised based on the past and predicted carbon traces of the data centers, it poses a critical challenge specific to the video streaming system in that the user's quality of experience (QoE) may be compromised by the workload migrations. In particular, the impacts on the QoE are two-fold: (1) Frequent service migrations may result in frequent changes in the video quality; and (2) the time of interruption during each migration could create noticeable artifacts in the displayed video stream.

We conduct a pilot study on a carbon-efficient video streaming mechanism to address the aforementioned challenges with the potential of future real-world applications. The proposed system employs dynamic data center selection strategies, utilizing real-time carbon intensity data to optimize relocation decisions. The strategies prioritize the optimization of carbon savings while considering user experience, the frequency of migrations, and the maintenance of service stability. We employ a video stream switching mechanism that achieves smooth transitions between data centers, thereby maintaining the uninterrupted flow and high quality of the video streams. Our evaluation results show a significant improvement of 19.59% in carbon saving per switch (CSR) compared to Strategy *L*. Our investigation shows the potential of reducing carbon emissions with even a simple dynamic video stream switching mechanism. In the future work, we will explore other mechanisms that consider the spatial and temporal variations of carbon footprints.

II. Carbon-Efficient Video Streaming

A. System Overview

Fig. 1 shows a system overview of the proposed carbon-efficient video streaming solution. Traditionally, video streaming system designs have primarily focused on streaming quality metrics, such as re-buffering frequency and visual glitches, when selecting a backend data center. Carbon emissions, however, have not been factored into this decision-making process. Consequently, even when two data centers offer comparable streaming quality, the more environmentally friendly option may not be selected. Our key design principle is to dynamically migrate video streaming workloads to carbon-efficient data centers based on real-time and/or predicted carbon intensity traces (e.g., from WattTime [20]). The migration is managed by a *coordination server* that employs various data center selection strategies to determine the optimal time

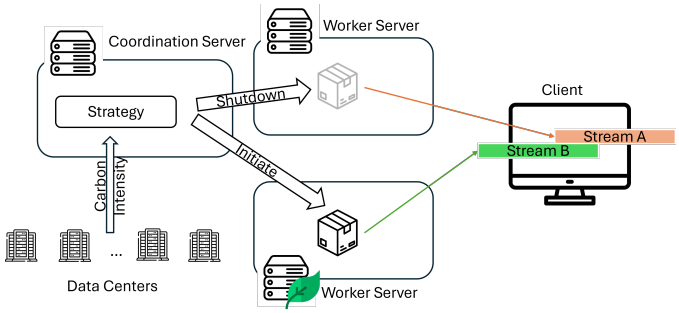

Fig. 1: Overview of the proposed carbon-efficient video streaming system.

and location to relocate the video stream. Considering that video content resources are deployed globally (e.g., through content distribution networks), these resources do not need to be moved. The primary goal is to achieve significant carbon reductions while maintaining high video quality. Once a migration is decided, the system initiates a stream switching procedure to achieve a seamless switch from the source stream to the destination stream, served by different *worker servers* with minimal interruption to the streaming workflow.

B. Carbon-Efficient Data Center Selection

A fundamental challenge in the proposed stream migration solution is to determine which data centers to utilize and when to migrate the video streaming workload. While it is often desirable to utilize the lowest carbon-emitting server, it is worth noting that the user experience would be affected during each migration of the stream from one server to another, mainly due to the interruption of the original stream and the time needed to establish the new stream. For example, in our feasibility study of stream migration, we observe that the carbon intensity status of the data centers is highly dynamic. Oftentimes, the least carbon-emitting server can only hold its greenest position for a brief period of time before another server becomes the most carbon-efficient choice. This situation may result in frequent back-and-forth stream migrations between data centers, negatively affecting the user experience.

Therefore, we design migration strategies to ensure that the candidate data center would be the lowest carbon-emitting one for an adequate period of time (i.e., *setup time*) prior to planned migration and ensure the candidate runs exclusively for a necessary period of time (i.e., *hold time*), to avoid the situation of two short-term migrations. In general, we consider the following 4 migration strategies:

- **Strategy *L*** (a.k.a., *always the **L**owest*) selects the lowest carbon-emitting data center to switch to instantly. Therefore, the video streaming workload is located on the most carbon-efficient server at all times, achieving the lowest carbon emission but at the expense of frequent migrations.
- **Strategy *S*** (a.k.a., *always the lowest with **S**tatic setup and hold time*) aims to address the frequent migration issue in Strategy *L*. Before migration, the strategy observes the server that emits the least carbon for a period of time (i.e., *setup time*) to check if it stays as the greenest data center across all candidates before a migration can be initiated. Then, after migration, the strategy maintains the workload on the new server for a period of time (i.e., *hold time*) before considering another migration. This strategy trades off some carbon savings for a reduced number of migrations to preserve the user experience. In this strategy, both *setup time* and *hold time* are predetermined constant values, hence the term "static" in the strategy name.
- **Strategy *D*** (a.k.a., *always the lowest with **D**ynamic setup and hold time*) aims to improve the flexibility of *hold time* in *Strategy S*. During a stream migration, both the old and new streams would co-exist for a certain period of time, which causes an overhead in carbon emissions. Presumably, this overhead is compensated by the lower carbon emission on the new server if a proper *hold time* is applied. Therefore, Strategy *D* introduces the dynamic hold time: $Hold_d = C_{new} \cdot t_{co}/(C_{old} - C_{new})$, where t_{co} is the time that both the old and the new streams are alive during the migration, C_{new} is the carbon intensity of the new server, and C_{old} is the carbon intensity of the current server. Combining the above two, we apply the dynamic *hold time* as follows: $Hold_{dynamic} = \min(Hold_d, Hold_{static})$. The *hold time* in this strategy is determined by the carbon traces of the two data centers before and after the migration, hence the term "dynamic" in the strategy name.
- **Strategy *P*** (a.k.a., *always the lowest with dynamic setup and hold time with **P**rediction*), aims to further improve the accuracy of *hold time* in Strategy *D* with the prediction of future carbon traces. Since we hold the workload on the new server for the duration of *hold time* after a migration, the actual carbon savings are based on the carbon intensity during *hold time*, which is not known at the observation stage before migration. The worst case scenario is that the new server drops from the most carbon-efficient to the least carbon-efficient one right after migration. To address this issue, Strategy *P* uses the average of predicted and latest carbon traces to calculate the hold time in Strategy *D*. The accuracy of carbon emission prediction is actively studied in the community to better reduce carbon emissions [21]. In the case of low prediction accuracy, it would merely downgrade the strategy from *P* to *D*, as it is akin to having zero knowledge about the future carbon trace, which is the case for Strategy *D*.

As a case study, Fig. 2 shows a comparison between Strategy *L* and Strategy *S* (with both setup time and hold time set to 3). Strategy *S* does not switch to data center B at time 1 like in Strategy *L* but switches to it at time 4 after 3 time units of observation. At time 6, data center B is no longer the lowest carbon intensity data center, and Strategy *L* switches back to data center A immediately, while Strategy *S* is still in hold time and must use data center B exclusively until time 8 to make the next decision.

In addition to the migration strategies discussed above, we also take the network communication between the client

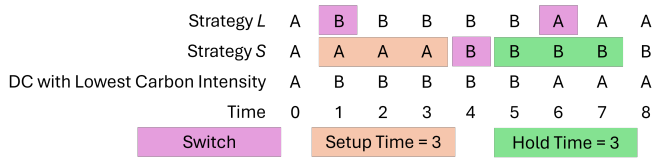

Fig. 2: Demonstration of switching schedule using Strategy *S* and Strategy *L*.

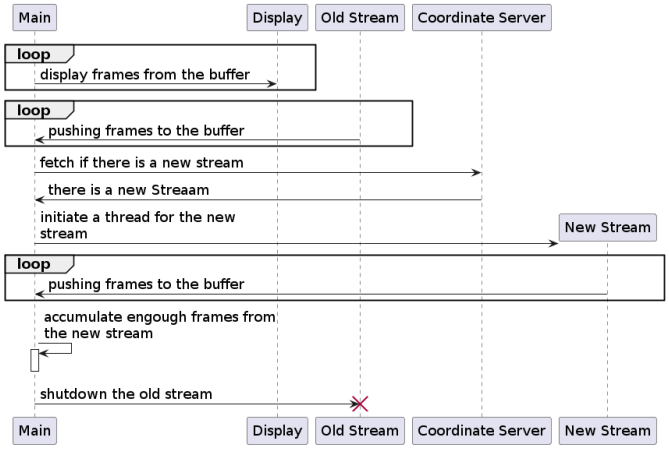

Fig. 3: Workflow of the stream switching process.

and various data centers into consideration. We aim to avoid switching to a distant data center where the carbon savings may be negated by the additional carbon emission caused by transmitting the video over long distance. Therefore, we explicitly incorporate the carbon impact caused by the network/distance in addition to the data center's carbon intensity. In our current implementation, the carbon emission due to network transmission is modeled as having a linear relationship with distance (e.g., 0.2 gCO_2eq per mile). However, the linear relationship can be changed to any distance-carbon function without affecting our proposed strategies, as the distance-related carbon emissions are evaluated every time the new carbon intensity data is collected.

C. Video Stream Switching

In order to migrate the stream between data centers with a seamless user experience, we establish a new stream with the new server, which will co-exist with the old stream for a period of time and eventually replace it. The challenge in implementing stream switching is to line up the frames between the two streams during migration. Considering that the two streams starting at different times may have different Presentation Time Stamp (PTS) marked on each frame, which is not suitable for synchronizing the two streams at the client side, we propose adding timestamps based on a real-world clock. As PTS is intended for inter-stream synchronization, we employ the real-world clock for intra-stream synchronization, which also requires all the servers and the client to calibrate their clocks reliably using techniques such as the NTP protocol [22]. When all parties in the system share one clock, the server can safely label the frame with real-world timestamp, indicating when this frame should be displayed just before the frame leaves the server. In addition, the client can easily sort frames from different streams based on chronological order. We design this stream switching as a 4-step procedure.

- **Step 1: Initialization**. The strategies discussed in Section II-B are utilized to determine if a migration is needed based on the real-time carbon intensity data.
- **Step 2: Preparation**. Once a migration is decided in the *Initialization* step, a coordination program is started to communicate with the new data center and deploy the stream for migration. When the new stream is initialized, it will co-exist with the old stream for a certain time duration t_{co}, during which the total carbon emission of the system is higher than normal, and the extra cost will be compensated by the new greener data center we switch to. After the new stream is ready and starts sending out frames the same as the old stream, the coordination program triggers the client to switch in the next step.
- **Step 3: Switching**. As illustrated in Fig. 3, the client obtains the new stream address from the coordination program and establishes the connection with the new server. Then, it begins to collect frames from the new stream into the buffer mixed with the frames from the old stream for display. An indicator in the client keeps track of which stream to display. After accumulating enough new frames, the client swaps the indicator from the old stream to the new stream. It then discards all the frames from the old stream and reports to the coordination program to start the next step.
- **Step 4: Cleanup**. No more frames are used from the old stream, and the coordination program communicates with the old data center to shut down the old stream. The stream migration process is then concluded.

III. System Implementation

Coordination Server. The coordination server carries the coordination program as described in Section II-C to collect carbon intensity data from all data centers, make migration decisions, execute the migrations, and trigger the client for stream switching. To accomplish these tasks, the coordination server is designed as a web server with a crawler function. The server accepts HTTP POST requests for passive carbon intensity collection and utilizes the crawler function for active collection. Every time when the latest data is collected, the migration strategy is triggered to determine if a stream migration is needed in a separate thread to prevent blocking the web server. When a migration is needed, the strategy thread continues with new data center preparation and keeps reading from the stream in the new data center until it receives valid outputs. When the new stream is ready, its address is placed in the corresponding location of the old stream for the client to poll. After the client fetches the address using HTTP GET, the server removes the address to prevent repeated stream switching on the client side and initiates a thread to clean up the old data center.

Worker Server. The worker server is responsible for hosting the stream and accepting control commands from the coordination server. Docker is utilized to manage the payload, encompassing its downloading, storage, versioning, and runtime environment, as well as exposing a control interface. To facilitate secure, password-less SSH accesses, each worker server's address and secret key are preloaded onto the coordination server. Upon completion of payload preparation, the worker server notifies the coordination server and initiates output monitoring. Additionally, the worker server implements a pruning mechanism for unused payloads to optimize storage.

Frames originating from the payload are assigned real-world timestamps upon transmission from the worker server (Section II-C) to facilitate synchronization. However, if these timestamps solely reflect the departure time, clients frequently receive outdated frames due to network latency. To address this issue, an ahead-offset is introduced, causing frames to appear as if they originate from a future time point. Given that the latency remains less than the offset, clients receive valid future frames instead of obsolete ones. This offset is implemented as a client-side configuration to mitigate latency effects without impacting server-side carbon emissions.

Client. The client is responsible for displaying the stream from the payload and switching to another payload seamlessly. To accomplish these two tasks, the client uses FFmpeg [23] as a low-level library. The client first continues to poll the coordination server to check whether a new data center is ready. When an address is obtained from the coordination server for a new data center, the client starts a new thread using FFmpeg to receive frames from the worker server and add labels to distinguish the frames from the old stream. The frames are stored in a buffer located in the client's main thread for display. After the buffer has enough frames from the new stream, the main thread notifies the display thread to switch streams. The display thread also uses FFmpeg to display frames on the screen. It reads from the buffer in the main thread and checks the timestamp of the frame, which includes 4 cases: 1) the frame is for the future, and the display thread blocks until the time of the frame's timestamp to display it; 2) the frame is just on time, and the display thread displays it immediately; 3) the frame's timestamp is behind the current time but still in a preset range, and the display thread displays it with a reduced frame interval to catch up the time; and 4) the frame is too late for the time, and the display thread discards it and fetches the next frame in the buffer. When the display thread receives the notification to switch streams, it also flushes all the frames of the old stream from the buffer.

IV. Evaluations

A. Experimental Setup

1) Carbon Trace-based Simulation Setup: To evaluate carbon savings, we utilize WattTime's dataset [20] covering 100 U.S. locations from January 1–15, 2025, with 5-minute granularity, for both real-time and predicted carbon traces. To account for distance effects, we assess 5 data centers, assuming a carbon cost of 0.2 gCO_2eq/mile. Distances range from 30 to 2,504 miles (averaging 1,204 miles). We simulate 5 data centers operating simultaneously with 120 seconds of co-existence, which is sufficient to ensure the wake-up of an additional payload within our experimental setup. To mitigate static best-performer bias, we select the top 100 sets from 1,000,000 runs with random combinations, prioritizing distant locations with similar carbon patterns. We assume that the 5-minute carbon intensity reflects a typical streaming server and that our server payload consumes 1 Watt. This assumption affects only absolute emissions, not relative savings.

The dataset undergoes active collection on the coordination server, as detailed in Section III. This method ensures the synchronization of carbon emission data despite varying collection times. All records are aligned to the beginning of their corresponding time slot; for instance, data collected at 10:02 is aligned to 10:00. When data is unavailable for a 5-minute slot, the data of the previous slot data is used. This approach prevents timing discrepancies that could trigger suboptimal data center migrations due to premature arrival of lower emission data from one data center.

Given that our objective extends beyond mere carbon reduction to encompass a balance between carbon emissions and user experience, we propose a new metric: the carbon-to-switch ratio (CSR). This ratio represents the amount of carbon emission saved per switch compared to using a fixed single data center that emits the lowest total amount of carbon ($carbon_{DC}$) without any switches. A higher CSR is preferable, as it indicates greater carbon savings with fewer switches, thus minimizing the impact on user experience.

$$CSR = (carbon_{DC} - carbon\_emission)/switches \quad (1)$$

2) Video Streaming System Setup: To evaluate the switching mechanism, we deploy a video streaming system in a local area network comprising one coordination server, two worker servers, and one client. All servers operate on separate Virtual Machines (VMs) with identical specifications: 8 vCPUs, 16 GiB memory, 128 GiB storage, and a 1 Gbps network connection. The client runs on a workstation equipped with two E5-2623 CPUs, 32 GiB memory, and a 1 Gbps connection. Servers are internally bridged and connected to the client via network cables. For testing purposes, the video source is initially placed on the coordination server but can be easily relocated. The video content is generated using FFmpeg's *testsrc* at 1920×1080 resolution and 30 FPS. The system utilizes the Secure Reliable Transport (SRT) protocol [24] for transmission, bypassing transcoding to eliminate payload effects. FFmpeg employs default settings with the codec specified via command line. A timestamp is attached to each frame with a 10-second offset to mitigate latency (Section II-C). For performance logging, the system records the current time, frame timestamp, and Presentation Time Stamp (PTS) in the console.

B. Carbon Savings

Across all the 100 combinations selected as described in Section IV-A, our proposed 4 strategies predominantly outper-

form the fixed data center with the lowest carbon emissions. Specifically, Strategies *L*, *S*, *D*, and *P* demonstrate lower carbon emissions in 99, 97, 97, and 95 out of 100 combinations, respectively, when compared to the fixed data center with the lowest carbon emissions. Further analysis reveals that Strategies *S* and *D* yield identical results. This similarity can be attributed to our selection of combinations, where the carbon emission profiles of the data centers are sufficiently close to each other, thereby increasing switching opportunities. Consequently, the benefits derived from switching are typically insufficient, though not negligible, to offset the co-existence overhead within the specified hold time. Despite the identical performance of Strategies *S* and *D*, Strategy *D* remains valuable as it serves as the foundation for Strategy *P*, which further enhances the CSR results. From the perspective of CSR, 92% of the combinations demonstrate improvements when transitioning from Strategy *L* to Strategies *S* and *D*, with an average increase of 15.44%. Furthermore, 57% of the combinations exhibit enhancements when moving from Strategy *L* to Strategy *P*, yielding an average improvement of 19.59%.

Table I shows a case study (1 of the 100 combinations) on the relative carbon savings when adopting the 4 proposed migration strategies, compared to using a fixed data center without switches. This analysis focuses on the scenario where Strategy *P* demonstrates the most significant improvements over Strategy *L*, with all strategies selecting configurations that yield the lowest carbon emissions. In the case of fixed data center, the carbon emission varies from 1257.43 gCO_2eq (ISONE_CT) to 1670.56 gCO_2eq (SPP_ND), with an average emission of 1461.41 gCO_2eq. When using the Strategy *L*, carbon emissions can be significantly reduced to 1076.79 gCO_2eq, with average savings of 26.32%. As aforementioned, Strategy *L* has a high overhead on the number of switches, which is 831 times in our experiments. Strategies *S* and *D* reduce the number of switches to 362, with average savings of 21.60%. To further reduce overhead while improving carbon savings, Strategy *P* uses prediction to achieve 55 times of switches, with carbon savings of 18.64% on average.

TABLE I: Case study on carbon emissions and savings of the 4 migration strategies compared to the fixed data center approach without switching (Unit: gCO_2eq).

Data Center	Use Fixed Data Center	Strtg.L	Strtg.S	Strtg.D	Strtg.P
		1076.79	1145.77	1145.77	1188.95
ISONE_CT	1257.43	14.37%	8.88%	8.88%	5.45%
NYISO_CAPITAL	1275.43	15.57%	10.17%	10.17%	6.78%
PJM_CHICAGO	1500.47	28.24%	23.64%	23.64%	20.76%
PJM_EASTERN_OH	1603.15	32.83%	28.53%	28.53%	25.84%
SPP_ND	1670.56	35.54%	31.41%	31.41%	28.83%
Average Savings		26.32%	21.60%	21.60%	18.64%
Switches		831	362	362	55
CSR		0.22	0.31	0.31	1.25

Although Strategy *L* achieves the lowest carbon emissions, it has the highest number of switches, which introduces the most impact on user experience. The other three strategies aim to reduce the impact of user experience and create a better carbon-experience balance compared to Strategy *L*. Trading off carbon savings for a better user experience is the reason why all data centers show limitations in carbon savings when using Strategies *S* and *D* compared to using Strategy *L*, but they gain 56.44% reductions in the number of switches. While the setup / hold time mechanism aims to reduce the number of switches, it also prevents the stream from quickly migrating to the lowest-carbon-emission data center during the setup time. Strategy *P* further improves for this limitation, achieving savings compared to all data centers and only 10.42% more carbon emissions compared to Strategy *L*, by taking advantage of the prediction with a switch reduction of 93.38%. In summary, our proposed Strategy *P* can achieve the best carbon-experience balance across the 4 strategies.

Fig. 4 illustrates the switching history of all 4 strategies with the 5 data centers. The top sub-figure represents the carbon traces of the 5 data centers. The following 4 sub-figures represent each strategy as the highlighted segment is the active data center at that time. The average time between two switches is 25.96 minutes using Strategy *L* and 392.00 minutes using Strategy *P*. By using our proposed Strategy *P*, we can extend the time for users to watch the stream without interruption given 55 switches, which significantly enhances the user experience.

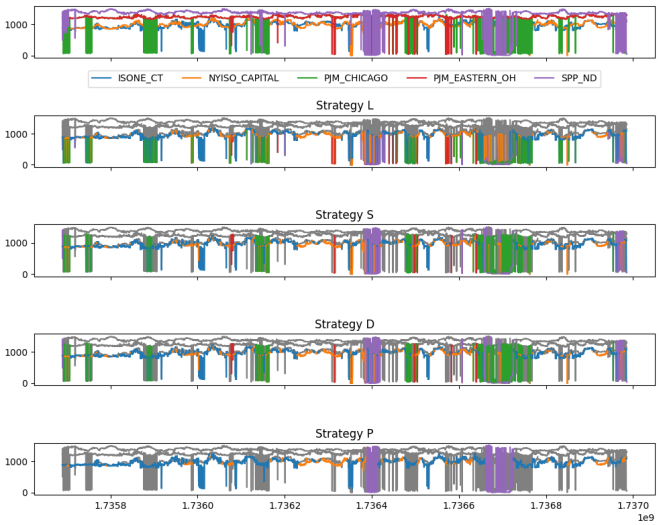

Fig. 4: The switching patterns of 4 strategies and the carbon trace of 5 data centers shown in Table I, where the colored trace is the active data center at that time.

Based on average savings compared to using a fixed data center without switching, the CSR of Strategy *P* is 1.25, significantly higher than 0.22 of Strategy *L* and 0.31 of Strategies *S* and *D*. In general, our proposed setup/hold mechanisms (i.e., in Strategies *S*, *D*, and *P*) can achieve a higher CSR than always switching to the lowest carbon-emitting data center (i.e., Strategy *L*).

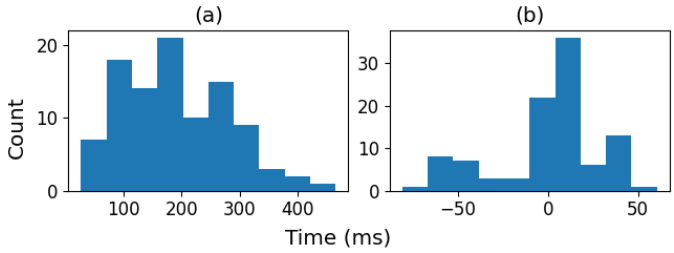

Fig. 5: Histograms of (a) the time taken for each switch, and (b) time difference between streams before and after switching.

C. Stream Switching Overhead

The stream switching introduces an overhead to the user experience due to preparing the new stream on the client. To evaluate the actual impact on the user experience, we measure the delay between the last frame from the old stream and the first frame from the new stream as the switching time. The average switching time of our proposed system is 189.75 ms and distributed as shown in Fig. 5(a). We observe that the switching time is independent of any previous switches and only correlated with the two streams of switching, and it does not accumulate over time to compromise the user experience. To further investigate the differences between streams, we first mark the same frame in two streams with the same Presentation Time Stamp (PTS) and then record the timestamp marked by the payload to the frame. To evaluate the impact on the user experience, we calculate the difference of the average offsets from the previous switch to the current switch for the old stream, and from the current switch to the next switch for the new stream. The average difference of the average offsets is 20.66 ms and distributed as shown in Fig. 5(b). This indicates that the two streams are well aligned, thereby maintaining the quality of user experience with a difference corresponding to only 1 or 2 frames at a frame rate of 30 FPS.

V. Conclusion

We have developed a carbon-efficient video streaming system to address the environmental sustainability of the popular Internet application with minimal impact on the user experience. The proposed system explores a series of dynamic data center selection strategies, as well as a seamless video stream switching mechanism, to achieve a balance between carbon efficiency and user experience. Our evaluations using carbon trace simulation and end-to-end video streaming system demonstrate significant carbon reductions and acceptable stream switching overhead. The repository of the project is at https://github.com/hwsel/CEVS.

Acknowledgment

This work was supported in part by the National Science Foundation award #2105564 and a VMware grant. We also thank WattTime for the support of carbon intensity dataset.

References

[1] N. Bashir, T. Guo, M. Hajiesmaili, D. Irwin, P. Shenoy, R. Sitaraman, A. Souza, and A. Wierman, "Enabling sustainable clouds: The case for virtualizing the energy system," in *ACM Symposium on Cloud Computing (SOCC)*, 2021, pp. 350–358.

[2] A. Souza, N. Bashir, J. Murillo, W. Hanafy, Q. Liang, D. Irwin, and P. Shenoy, "Ecovisor: A virtual energy system for carbon-efficient applications," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023, pp. 252–265.

[3] T. Anderson, A. Belay, M. Chowdhury, A. Cidon, and I. Zhang, "Treehouse: A case for carbon-aware datacenter software," *ACM SIGENERGY Energy Informatics Review*, vol. 3, no. 3, pp. 64–70, 2023.

[4] (2025) Innovating sustainable ideas. https://datacenters.google/operating-sustainably/.

[5] J. You, J.-W. Chung, and M. Chowdhury, "Zeus: Understanding and optimizing GPU energy consumption of DNN training," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023, pp. 119–139.

[6] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv:1906.02243*, 2019.

[7] Y. Zhao and T. Guo, "Carbon-efficient neural architecture search," in *Workshop on Sustainable Computer Systems (HotCarbon)*, 2023.

[8] A. A. Chien, L. Lin, H. Nguyen, V. Rao, T. Sharma, and R. Wijayawardana, "Reducing the carbon impact of generative AI inference (today and in 2035)," in *Workshop on Sustainable Computer Systems (HotCarbon)*, 2023.

[9] B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "Clover: Toward sustainable AI with carbon-aware machine learning inference service," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–15.

[10] B. Acun, B. Lee, F. Kazhamiaka, A. Sundarrajan, K. Maeng, M. Chakkaravarthy, D. Brooks, and C.-J. Wu, "Carbon dependencies in datacenter design and management," *ACM SIGENERGY Energy Informatics Review*, vol. 3, no. 3, pp. 21–26, 2023.

[11] N. Zilberman, E. M. Schooler, U. Cummings, R. Manohar, D. Nafus, R. Soulé, and R. Taylor, "Toward carbon-aware networking," *ACM SIGENERGY Energy Informatics Review*, vol. 3, no. 3, pp. 15–20, 2023.

[12] R. Jacob and L. Vanbever, "The Internet of tomorrow must sleep more and grow old," *ACM SIGENERGY Energy Informatics Review*, vol. 3, no. 3, pp. 27–32, 2023.

[13] A. Gupta, I. Jain, and D. Bharadia, "Multiple smaller base stations are greener than a single powerful one: Densification of wireless cellular networks," in *Workshop on Sustainable Computer Systems (HotCarbon)*, 2022.

[14] S. Zhang, W. Y. B. Lim, W. C. Ng, Z. Xiong, D. Niyato, X. S. Shen, and C. Miao, "Towards green metaverse networking: Technologies, advancements and future directions," *IEEE Network*, pp. 1–10, 2023.

[15] W. Wang, V. A. L. Sobral, M. F. R. M. Billah, N. Saoda, N. Nasir, and B. Campbell, "Low power but high energy: The looming costs of billions of smart devices," *ACM SIGENERGY Energy Informatics Review*, vol. 3, no. 3, pp. 10–14, 2023.

[16] N. Jiang, Y. Liu, T. Guo, W. Xu, V. Swaminathan, L. Xu, and S. Wei, "QuRate: Power-efficient mobile immersive video streaming," in *ACM Multimedia Systems Conference (MMSys)*, 2020, pp. 99–111.

[17] S. Liu, X. Li, Z. Zhou, B. Guo, M. Zhang, H. Shen, and Z. Yu, "Adaenlight: Energy-aware low-light video stream enhancement on mobile devices," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, vol. 6, no. 4, pp. 1–26, 2023.

[18] (2022) Back to basics: HTTP video streaming. https://aws.amazon.com/blogs/media/back-to-basics-http-video-streaming/.

[19] (2022) Status of video streaming solutions in data centers: 3 use cases. Https://antmedia.io/status-of-video-streaming-solutions-in-data-centers/.

[20] (2024) WattTime. https://watttime.org/.

[21] Y. Jin, A. Sharifi, Z. Li, S. Chen, S. Zeng, and S. Zhao, "Carbon emission prediction models: A review," *Science of The Total Environment*, vol. 927, p. 172319, 2024.

[22] *Network Time Protocol Version 4: Protocol and Algorithms Specification*, Std. RFC 5905, 2010.

[23] (2020) FFmpeg. http://ffmpeg.org/.

[24] (2023) Secure Reliable Transport (SRT) Protocol. https://github.com/Haivision/srt.