Reducing Bandwidth Demand for Video Streaming to Kbps

Jingkun Cao†, Tong Li†*, Bowen Hu†, Duling Xu†, Kezhi Wang‡, and Bo Wu§
Renmin University of China†, Brunel University of London‡, Tencent§

***Abstract*—This demonstration reports the design of a Real-Time Communication (RTC) system, Cactus, which achieves ultra-low bandwidth demand for high-definition video streaming with good video quality. When abstracting the data transmission paradigms, this demonstration validates exchanging computing resources for communication resources.**

I. Introduction

Video streaming has revolutionized our lives by providing instant access to entertainment, education, and communication, breaking geographical barriers, and transforming how we consume and share content. Traditional video streaming prioritizes transmitting *original* video and audio data, often resulting in high bandwidth consumption (e.g., 8 Mbps for a 720p video [1]). Under poor network conditions, such as in high-speed trains or underground parking lots, this legacy paradigm results in lag, rebuffering, and degraded quality, leading to unsatisfactory user experiences.

Fortunately, with the advent of User-Generated Content (UGC), Professionally Generated Content (PGC), and now AI-Generated Content (AIGC), the way data is generated has changed dramatically. This paper proposes an AIGC-based video streaming paradigm that reduces bandwidth demand for video streaming from Mbps to Kbps while ensuring high-quality video and audio transmission, even in poor network conditions. Specifically, we report the design of **Cactus**, a Real-Time Communication (RTC) system that demonstrates the new AIGC-based video streaming paradigm. The design rationale is that video streaming, aided by AIGC at the endpoints, can transmit only essential visuals and bandwidth-efficient audio, optimizing resource usage.

As shown in Figure 1, Cactus differs from the legacy video streaming system by transmitting pictures and audio instead of video and audio. Specifically, Cactus introduces a *Picture Picker* module on the sender side, which monitors real-time bandwidth conditions and video quality, dynamically picking essential visual pictures from video. Specifically, the Picture Picker module dynamically adjusts the time interval (i.e., picture interval) for sending pictures based on bandwidth constraints and quality constraints (see Section II for details). Finally, the video is recreated via a video generation model (see Section III for details) using the pictures and audio received on the receiver side.

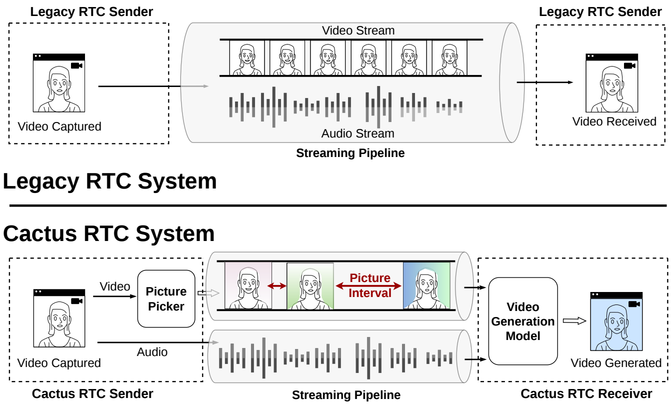


Fig. 1: The difference between Cactus and the legacy RTC system.

It is worth noting that, unlike existing state-of-the-art AIGC-based methods such as Txt2Vid [2], which rely on text-based audio reconstruction and synthesized speech generation, Cactus directly transmits audio in its original form. This design choice addresses several critical limitations: 1) Audio synthesis incurs high computational costs for model training and inference [3]; 2) Real-time tone reconstruction requires extensive user recordings (10–30 minutes) [4], which is impractical for many applications; 3) Text-based audio reconstruction raises privacy concerns due to potential exposure of sensitive biometric information. By preserving the original voice, Cactus ensures natural and secure communication.

We implement a prototype of Cactus[1] on two hosts equipped with Intel Xeon Gold 6330 CPUs, running Ubuntu Server 22.04 LTS (64-bit). We then evaluate its performance through real-time video transmission experiments with 480p video streams under a bandwidth limit of 1000 Kbps. Results demonstrate that Cactus reduces bandwidth demand by over 78% compared to baseline methods while maintaining competitive video quality.

II. The Design of Picture Picker

The core of Cactus is to deploy a Picture Picker module that picks and transmits a picture in a dynamically determined *picture interval* (denoted by I) from the source video. Next, we report the detailed design of the Picture Picker by answering two questions below.

This work is supported by the National Natural Science Foundation of China Projects (No.62202473, No.62441230) and the Tencent Basic Platform Technology Rhino-Bird Focused Research Program. Tong Li is the corresponding author (tong.li@ruc.edu.cn).

[1] The open-source implementation is maintained at https://github.com/litonglab/Cactus

Why dynamic picture interval? A straightforward approach is to capture pictures at static time intervals, such as every $I = 50$ ms. However, this method may fall short under dynamically changing network conditions and varying user experience demands. For example, if the picture interval is too large, there will be noticeable differences between consecutive pictures, causing the video generated by the model to diverge significantly from the original, which negatively impacts user experience. Conversely, if the picture interval is too small, the picture transmission essentially becomes raw video transmission, resulting in high bandwidth requirements.

How to dynamically set picture interval? When deciding whether to pick a picture, the Picture Picker should adhere to two constraints: bandwidth constraint and quality constraint.

- **Bandwidth constraint:** Given a bandwidth budget (BW, several Kbps), the average picture interval $\bar{I}$ should meet:

$$\bar{I} \geq \frac{|x|}{(BW - A)} \tag{1}$$

where $|x|$ is the frame size, and A is the audio bitrate.

- **Quality constraint:** We use the famous SSIM (Structural Similarity Index Measure) to represent video quality. Given a quality tolerance ($\alpha, 0 \leq \alpha \leq 1$), for each picture P_i, compute $SSIM(P_0, P_i)$, where P_0 is the last transmitted picture. The Picture Picker transmits a picture when it meets:

$$SSIM(P_0, P_i) < \alpha \tag{2}$$

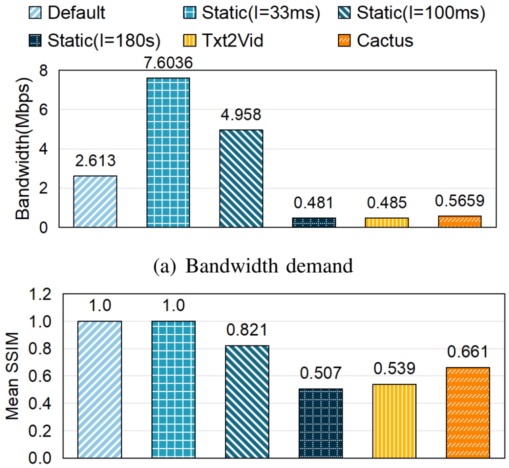
By simultaneously taking into account Equations (1) and (2), Cactus determines the picture interval. Note that both BW and α are user-customizable. They determine the trade-off between bandwidth requirement and user-perceived video quality, respectively. Generally, a low BW leads to poor video quality while a low α significantly reduces bandwidth demand.

III. Demonstration

A demo video is at https://youtu.be/OiMeYsJYeLU. We implement Cactus on a local computing platform. As illustrated in Figure 1, the video from the sender is first processed by the Picture Picker module. This module dynamically selects essential visuals and transmits them to the receiver. Meanwhile, the audio from the sender is directly forwarded to the receiver without being converted. On the receiver side, the received frames and audio are passed into the video generation module, which utilizes the open-sourced Wav2Lip [5] model to generate the final synchronized video.

Setup. The experiments were conducted on two hosts with the Intel Xeon Gold 6330 CPU. The system runs Ubuntu Server 22.04 LTS (64-bit). The video resolution is set to 480p. $\alpha = 0.7$, and $BW = 1000$ Kbps. Each test lasts 5 minutes.

Schemes. We conduct a comparative performance evaluation of Cactus with 5 transmission schemes to demonstrate its effectiveness. First, "*Default*" refers to the legacy way of encoded video streaming (with H.264 codec at 30 fps). "*Static ($I = 33$ ms)*", "*Static ($I = 100$ ms)*", and "*Static ($I = 180$ s)*" refers to the transmission schemes where the static picture interval (I) is set to 33ms, 100ms, and 180s, respectively. Notably, the scheme with $I = 33$ ms is equivalent to transmitting raw video frames without a video codec. We also include the state-of-the-art representative, *Txt2Vid*, that transmits only one picture at the beginning, which is equivalent to transferring pictures at an infinite interval (i.e., $I = \infty$).



(a) Bandwidth demand

(b) Video quality

Fig. 2: (a) Bandwidth demand for RTC application. (b) Average SSIM of the original and generated videos.

Results. We evaluate the performance of Cactus against several baseline methods in terms of bandwidth consumption and video quality (measured by mean SSIM). As shown in Fig 2(a), Cactus significantly reduces bandwidth usage, requiring only 566 Kbps, which reduces over 78% of bandwidth demand compared to the legacy way of RTC systems "*Default*" (2.613 Mbps). Meanwhile, Cactus has an average SSIM value of 0.661 (see Fig 2(b)), which is an improvement over the state-of-the-art approach *Txt2Vid* (0.539). These results show that Cactus effectively balances bandwidth efficiency and video quality, making it a superior solution for resource-constrained situations.

IV. Conclusion

This paper presents Cactus that dynamically transmits carefully picked pictures at the application layer, supported by a host-side AIGC computation model. Since the dynamic picture interval is closely related to the underlying network conditions, future work will explore cross-layer optimization, incorporating the transport layer (e.g., congestion control) and even the physical layer (e.g., channel coding [6]).

References

[1] Cisco, "Cisco visual networking index: forecast and trends, 2018-2023," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html, 2024.

[2] P. Tandon, S. Chandak, P. Pataranutaporn *et al.*, "Txt2vid: Ultra-low bitrate compression of talking-head videos via text," *IEEE JSAC*, 2023.

[3] J. Shen, R. Pang, Y. Zhang, and Y. Xu, "Tacotron 2: Generating human-like speech from text," *arXiv preprint arXiv:1703.07076*, 2018.

[4] Google, "Text-to-speech documentation," https://cloud.google.com/text-to-speech/docs, 2023.

[5] R. Mukhopadhyay, A. Jha, R. Abhinav, and V. P. Namboodiri, "A lip sync expert is all you need for speech to lip generation in the wild," *arXiv preprint arXiv:2008.10010*, 2020.

[6] P. Jiang, C.-K. Wen, S. Jin, and G. Y. Li, "Wireless semantic communications for video conferencing," *IEEE JSAC*, 2022.