LiFteR: Unleash Learned Codecs in Video Streaming with Loose Frame Referencing

Bo Chen[1], Zhisheng Yan[2], Yinjie Zhang[1], Zhe Yang[1], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University
Contact: boc2@illinois.edu (https://bochen.info/)

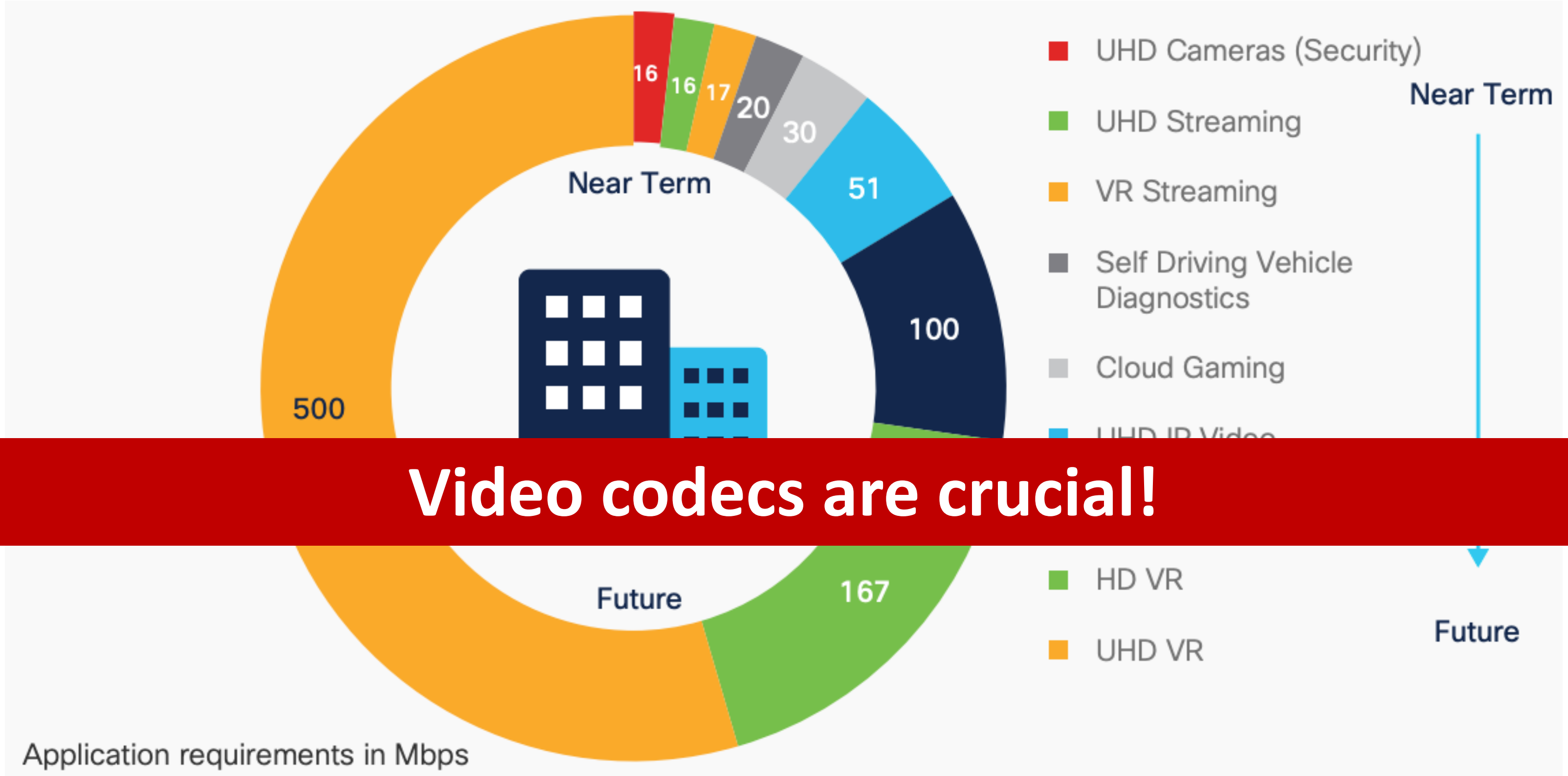
Video streaming

Video streaming is popular

- **One-third** of all online activity is spent watching videos.
- Americans spend **3h and 9min** a day streaming digital media.

Bandwidth demand

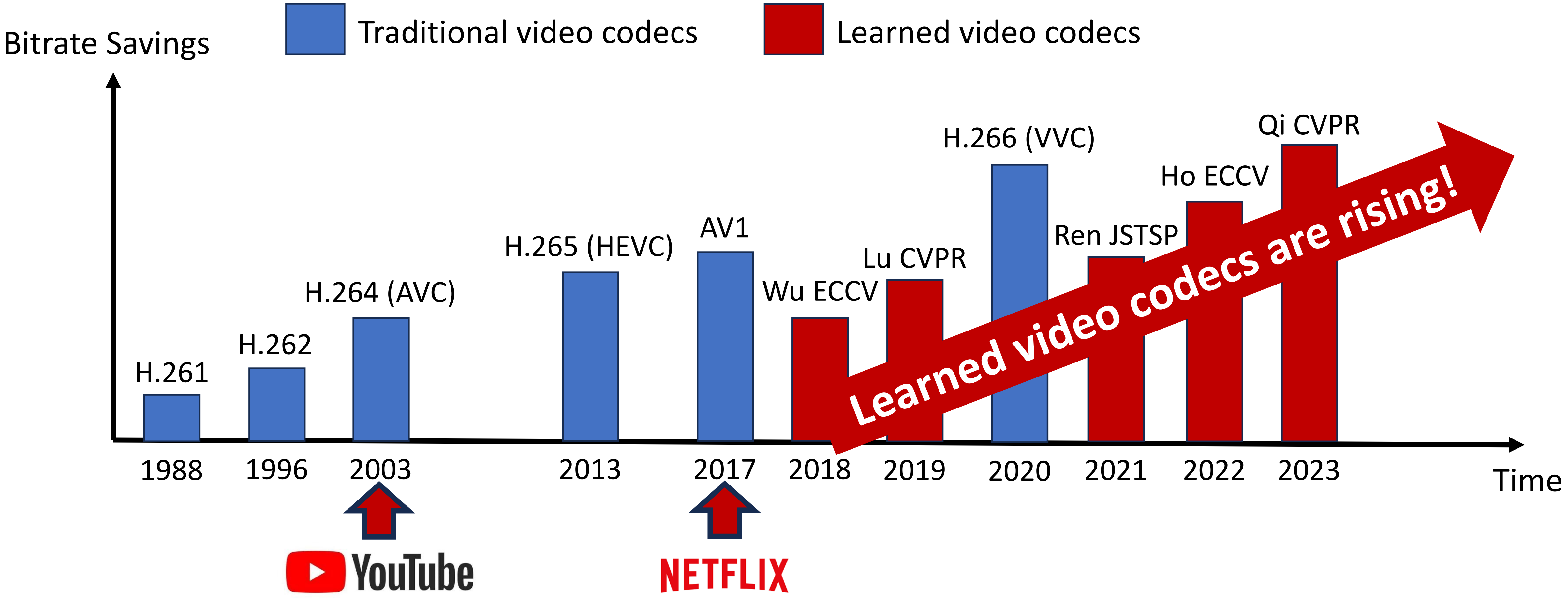
Source: Cisco Annual Internet Report, 2018–2023



Video codecs are crucial!

- Significant demand for bandwidth of video.
- Bandwidth needs grow exponentially.

History of video codecs



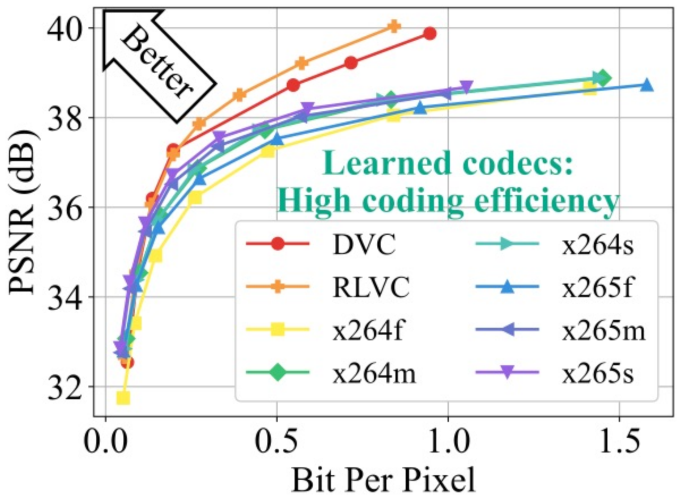
Can learned codecs replace traditional ones?

Not quite.

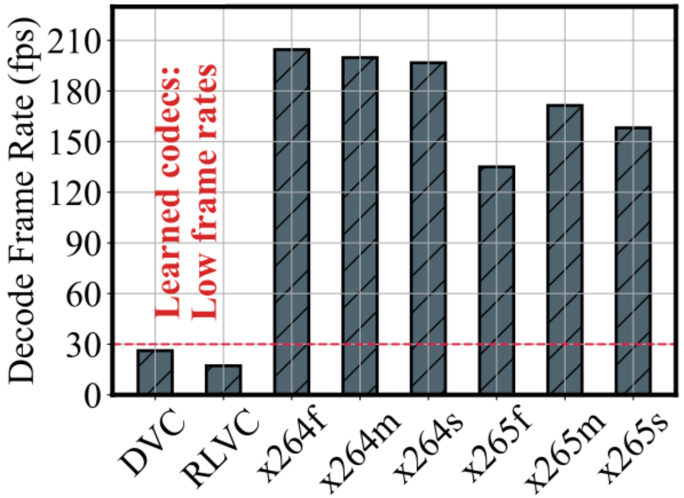
Pilot study

- Learned video codecs: DVC [1] and RLVC [2].
- Traditional video codecs: H.264 and H.265, with presets of very fast (f)/medium (m)/very slow (s).
- Hardware: Intel Core i9-8950HK CPU and NVIDIA GTX 1080 Ti GPU.

Coding Efficiency



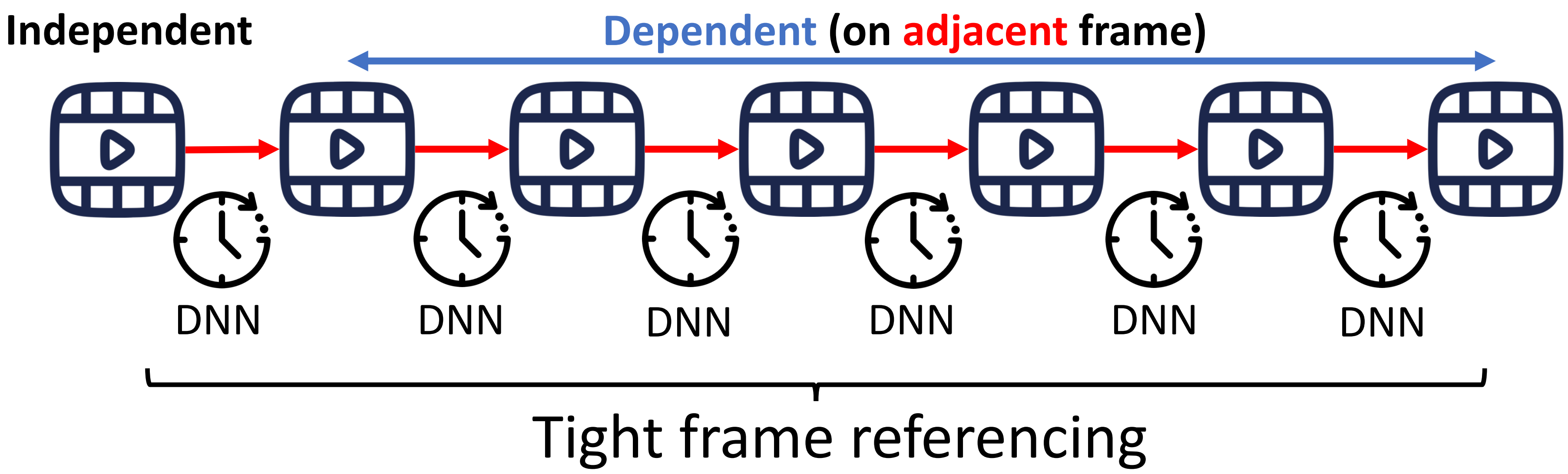
Decoding Speed



[1] Lu, Guo, et al. "Dvc: An end-to-end deep video compression framework." *CVPR*. 2019.
[2] Yang, Ren, et al. "Learning for video compression with recurrent auto-encoder and recurrent probability model." *IEEE JSTSP*. 2020.

Cause of slow decoding

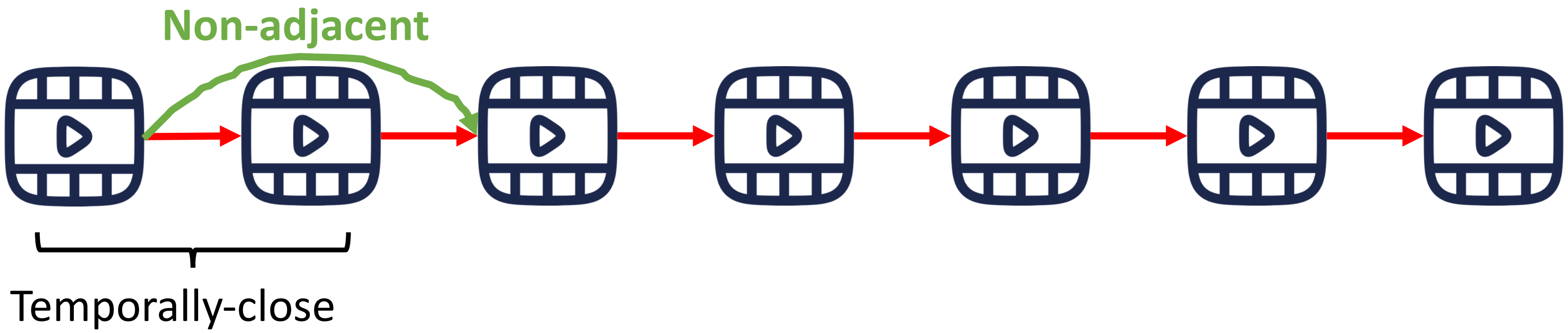
Coding pipeline adopted by most learned video codecs.



- Problems
  - Gap: slow deep neural network (DNN) v.s. real-time frame rate.

Intuition

Why not using non-adjacent frames as reference?



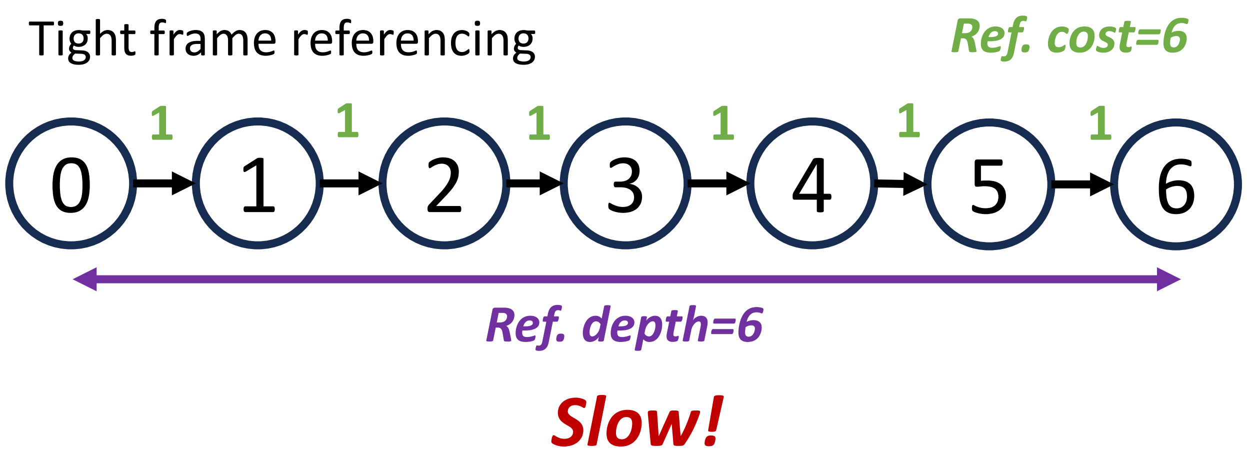
- Loose frame referencing (LFR)
  - **Coding efficiency**: temporally-close frames are similar.
  - **Decoding speed**: loose dependency allows parallelism.

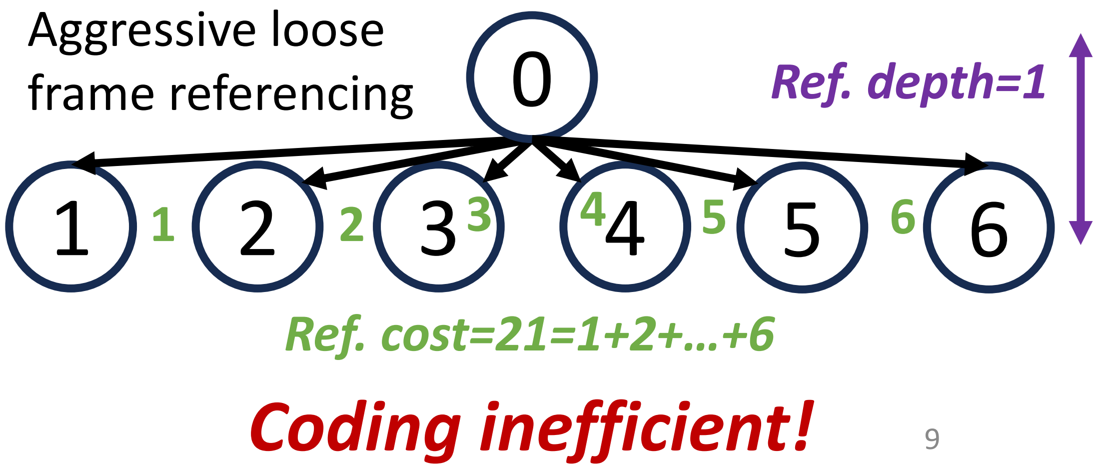
Challenges

- How do we schedule frame processing?
- How do we design learned codec?
- How do we adapt streaming infrastructure?

How should frames be processed?

	Frame processing	Graph traversal
Input	Video frame	**Vertex** ***Indexed by 0, 1,…***
Design space	Reference relation	**Directed edge** $\boldsymbol{cost = \Delta\text{index}}$
Objectives	Coding efficiency	**Reference cost** $\boldsymbol{\sum cost}$
	Decoding speed	**Reference depth** $\boldsymbol{\#iterations}$

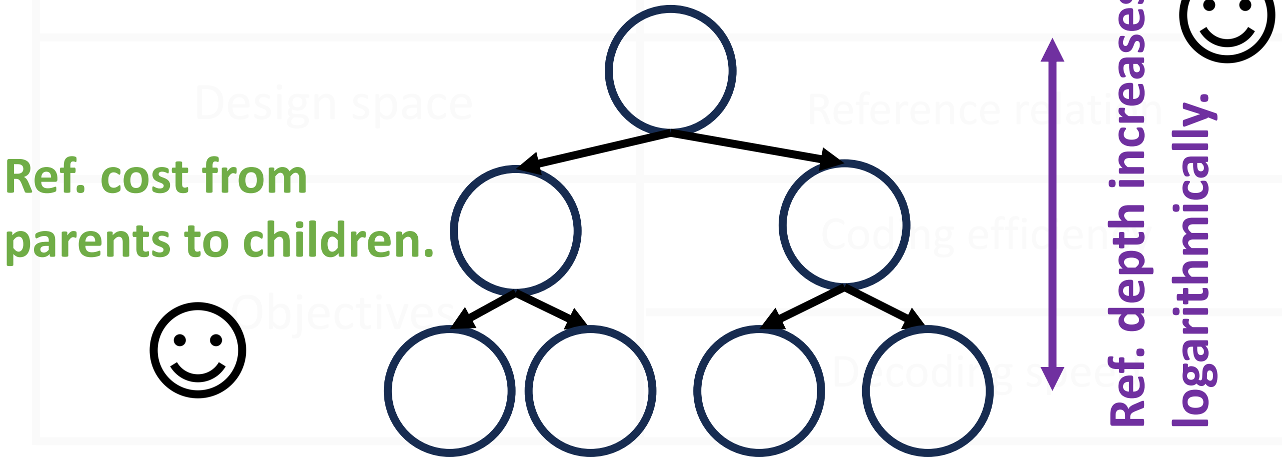




How should frames be processed?

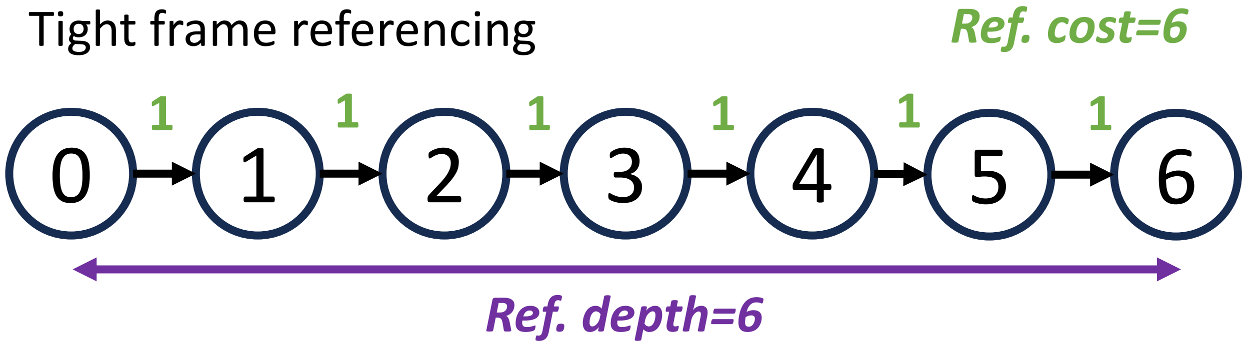
How about something in between?

A binary tree!



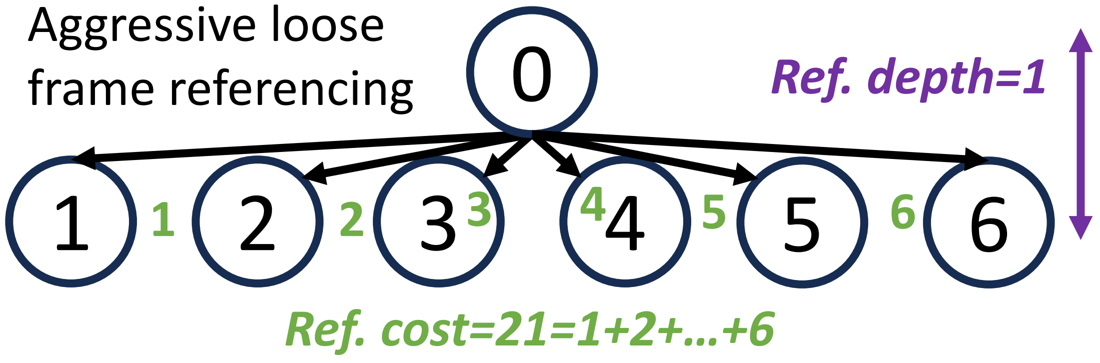
Graph traversal
Vertex ***Indexed by 0, 1,...***
Directed edge $cost = \Delta \mathrm{index}$
Reference cost $\sum cost$
Reference depth $\#iterations$

Tight frame referencing



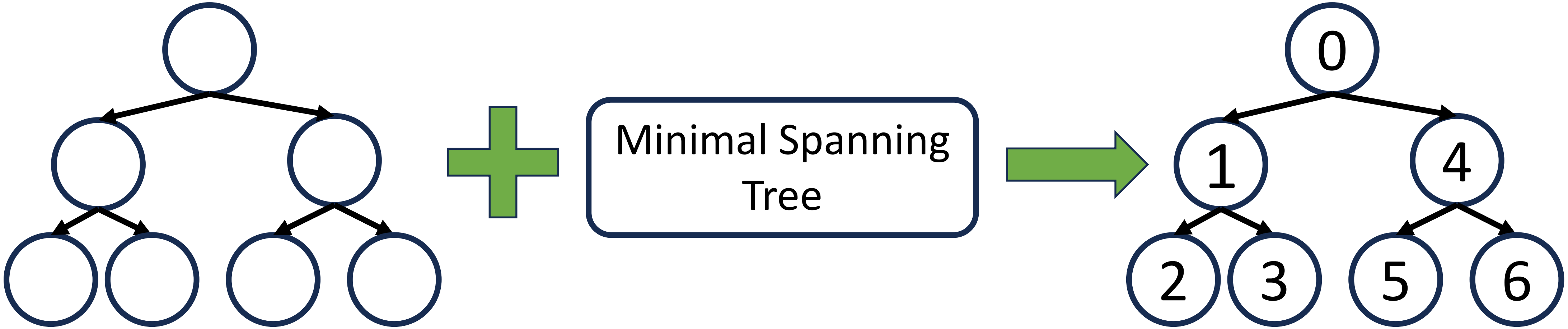
Slow!

Aggressive loose frame referencing



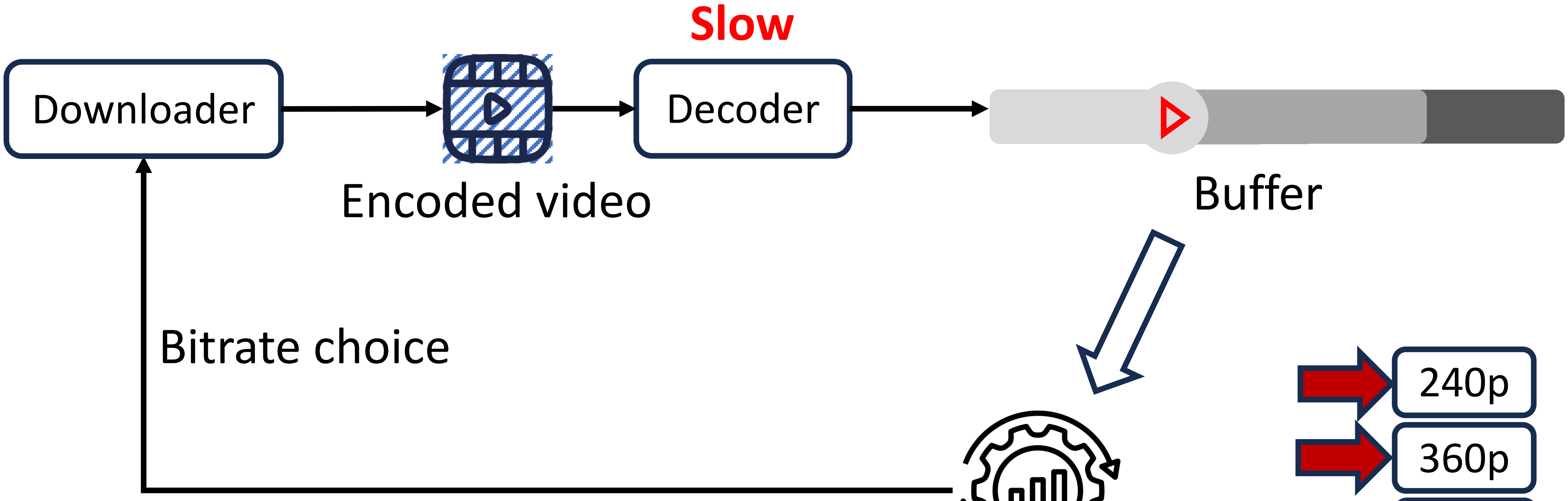
Coding inefficient!

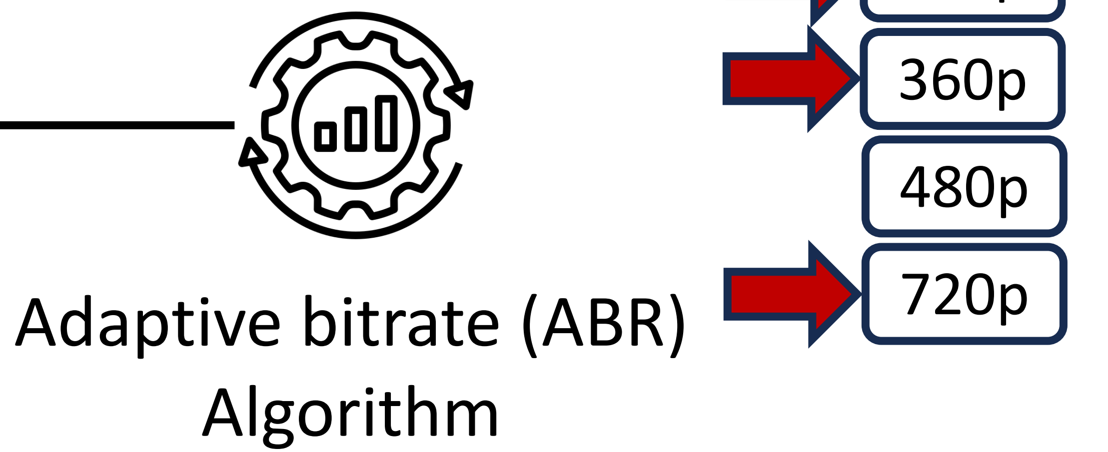
How should learned codecs be designed?



- **Opportunity**: **Inter-frame** correlation.
- **What we do**: Codec adaptation with the **self-attention** mechanism

How should streaming infra be adapted?

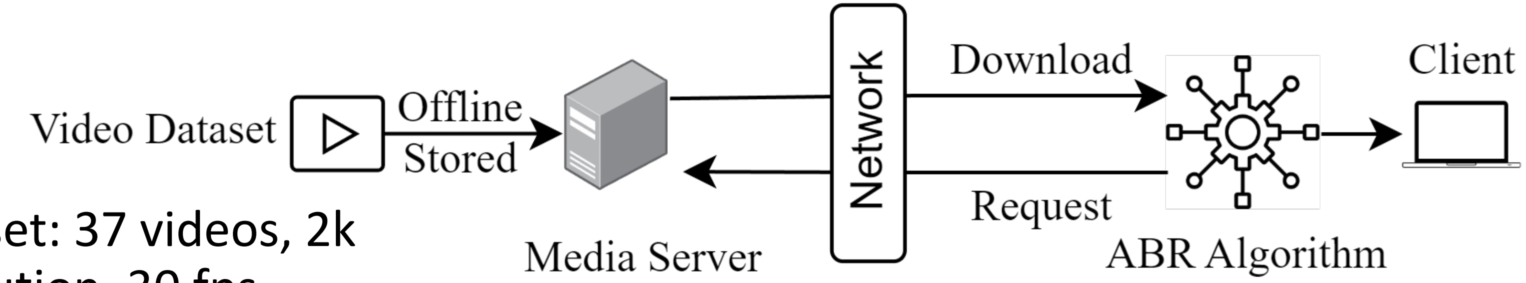




- **Opportunity:** ABR algorithm
- **Virtual buffer**: captures content
  - ✓In buffer (existing ABR algorithm)
  - ✓Not decoded yet

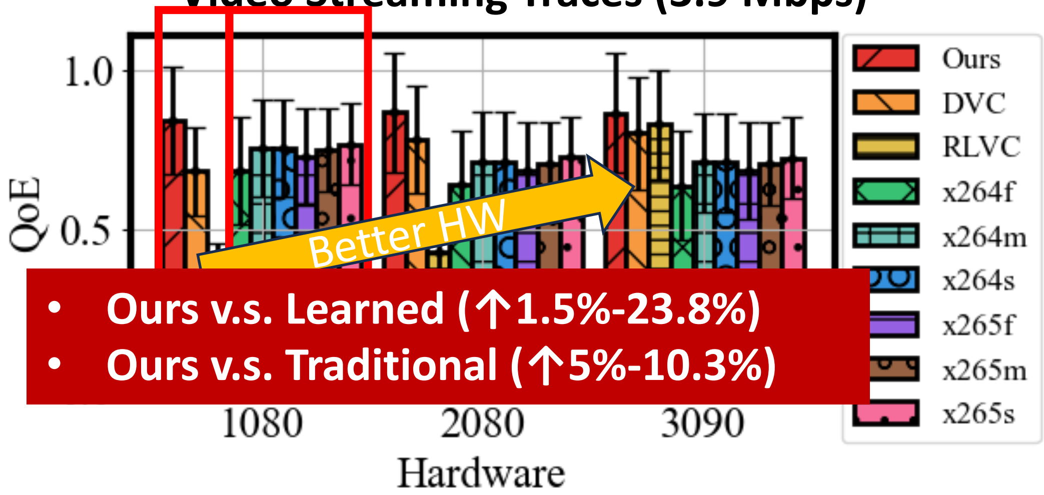
Evaluation

- ABR: BOLA [1]; all baselines with learned codecs adopt **virtual buffer**.



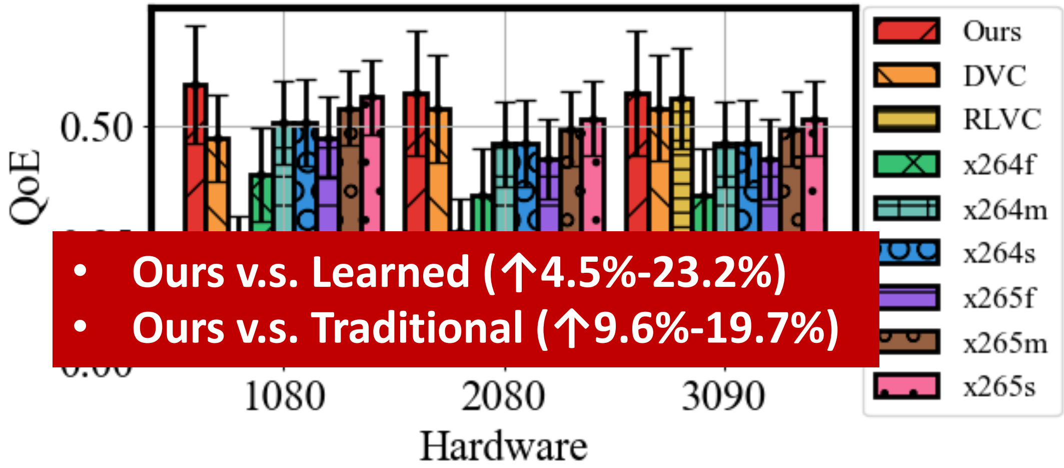
- Client hardware: Linux desktop with NVIDIA GeForce GPUs.
- Dataset: 37 videos, 2k resolution, 30 fps.
- Network: 1000 traces collected under two network conditions.
- QoE metric: weighted sum of video quality and negation of rebuffer rate.

Video Streaming Traces (3.9 Mbps)



- **Ours v.s. Learned (↑1.5%-23.8%)**
- **Ours v.s. Traditional (↑5%-10.3%)**

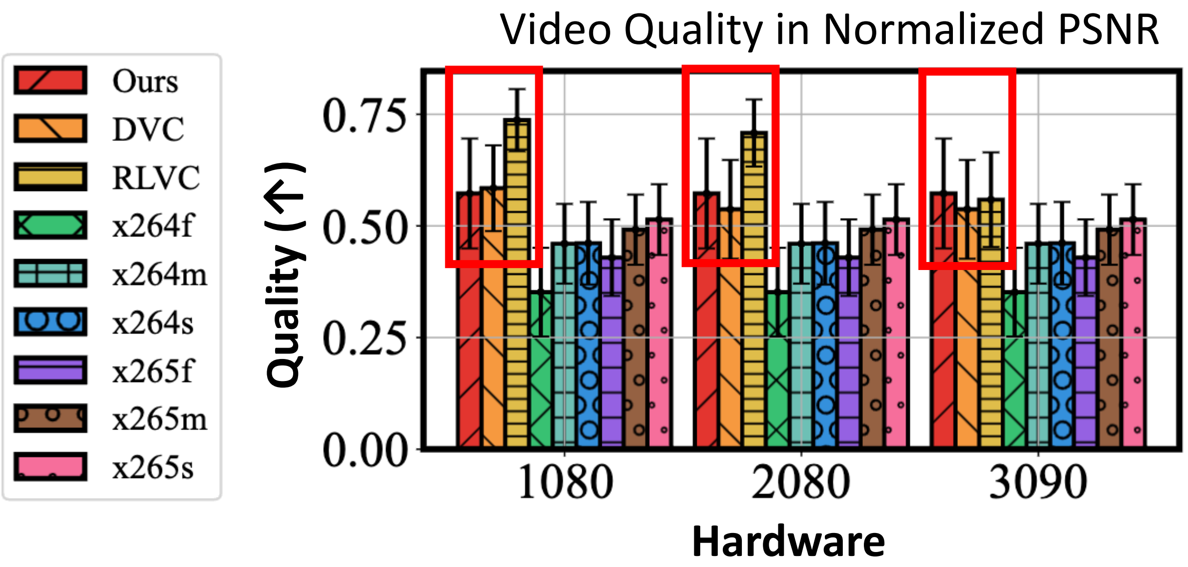
HTTP Get Traces (15.8 Mbps)

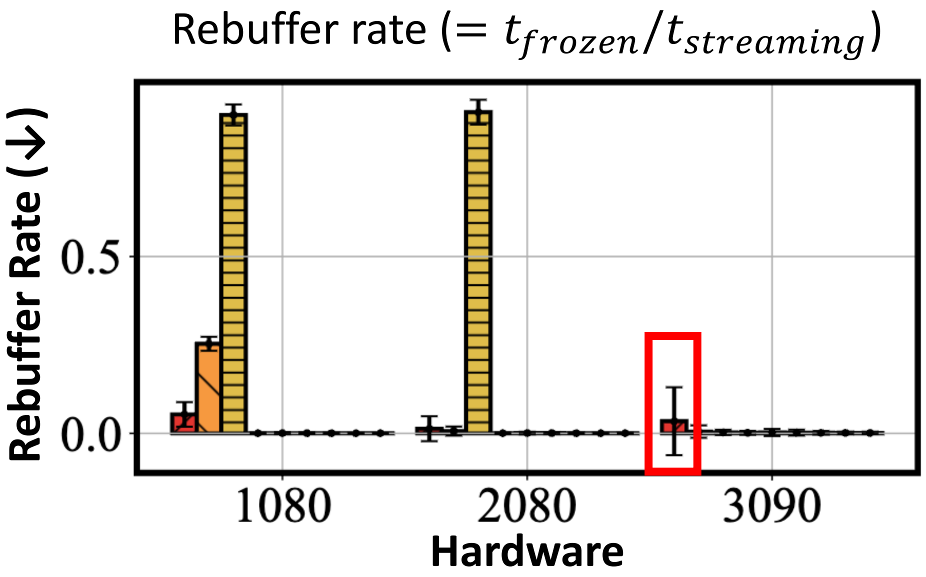


- **Ours v.s. Learned (↑4.5%-23.2%)**
- **Ours v.s. Traditional (↑9.6%-19.7%)**

[1] Spiteri, Kevin, Rahul Urgaonkar, and Ramesh K. Sitaraman. "BOLA: Near-optimal bitrate adaptation for online videos." IEEE/ACM TON (2020).

QoE breakdown





- LiFteR does not always achieve the highest quality.
- Virtual buffer: Slower decoding=>higher quality.

- LiFteR improves rebuffer rates.
- LiFteR's rebuffer rate becomes unstable: a downside from parallel processing.

Lessons learned

1. Tight frame referencing may not be necessary.
2. Codec should be co-designed with the frame processing pipeline.
3. There is room to improve existing infrastructure for learned codecs.

Thanks for listening!

Bo Chen[1], Zhisheng Yan[2], Yinjie Zhang[1], Zhe Yang[1], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University
Contact: boc2@illinois.edu (https://bochen.info/)