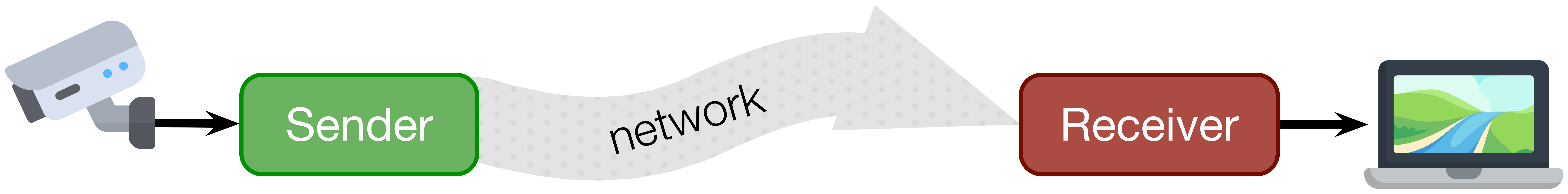
Salsify:
Low-Latency Network Video Through Tighter Integration Between a Video Codec and a Transport Protocol

Sadjad Fouladi
Stanford University

Joint work with: John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, Keith Winstein

What is *real-time* video?



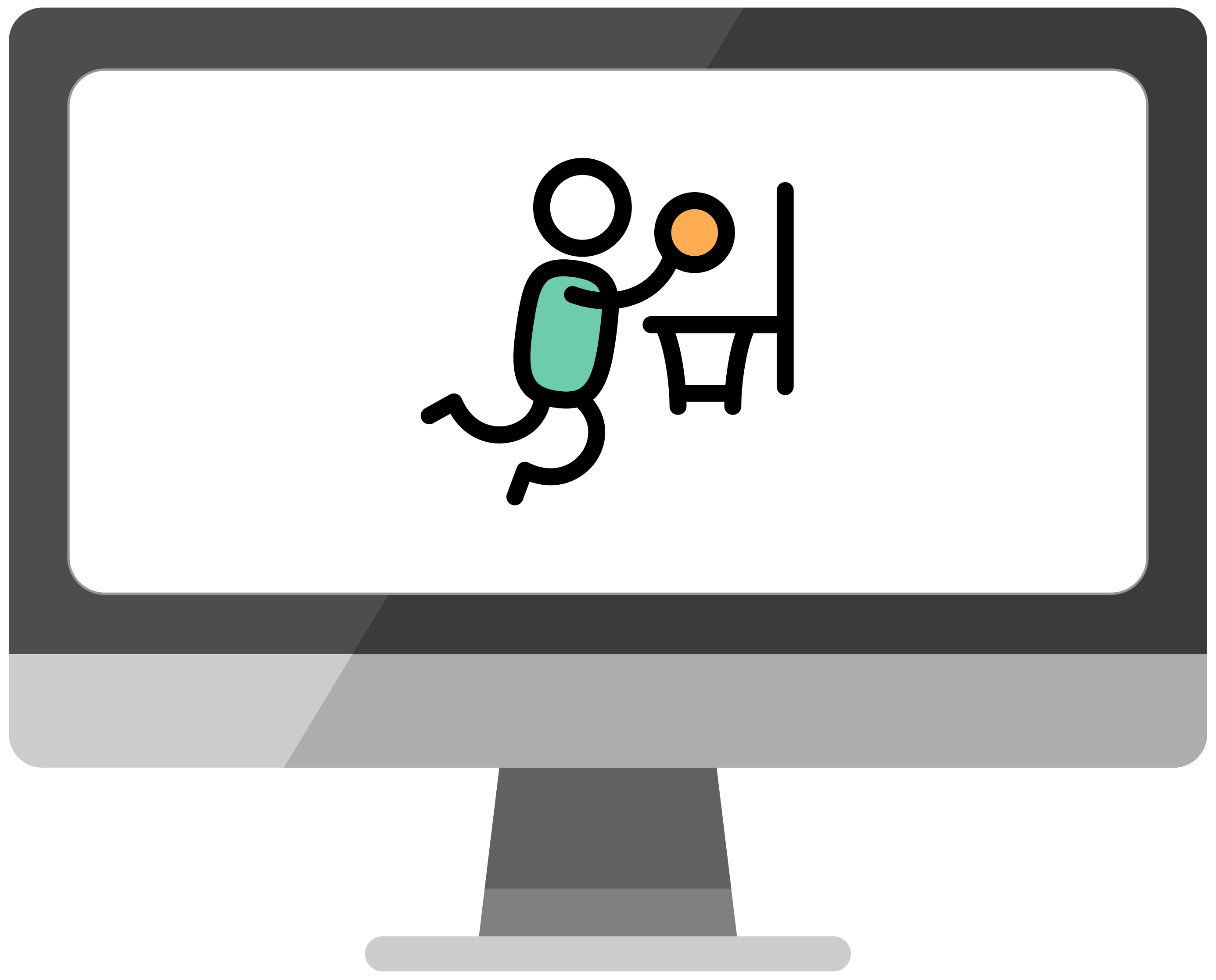
Real-time video latency target: *tens of milliseconds*

the amount of time between when

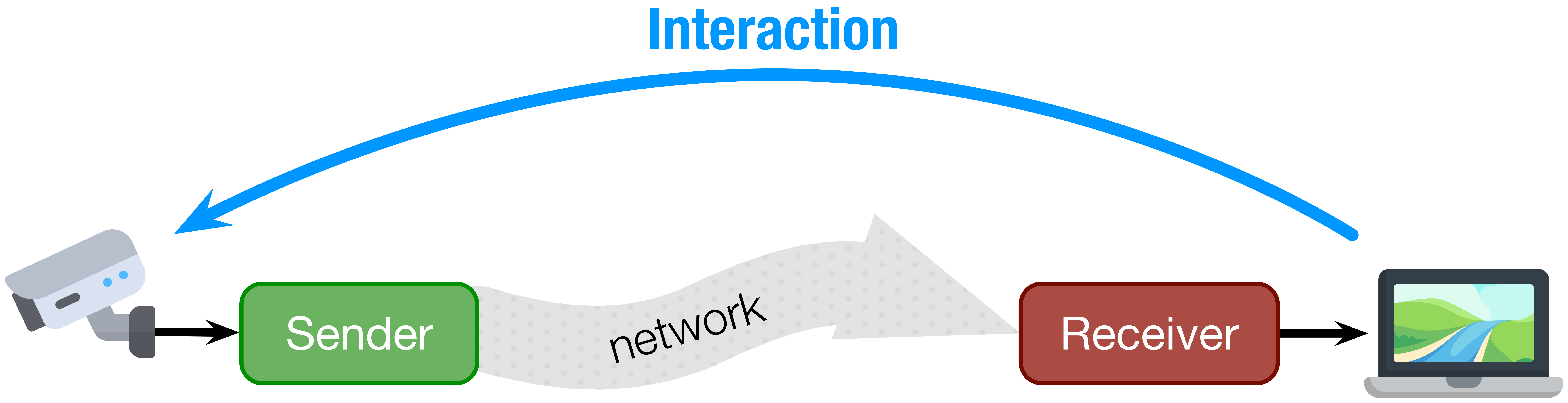
something happens

and

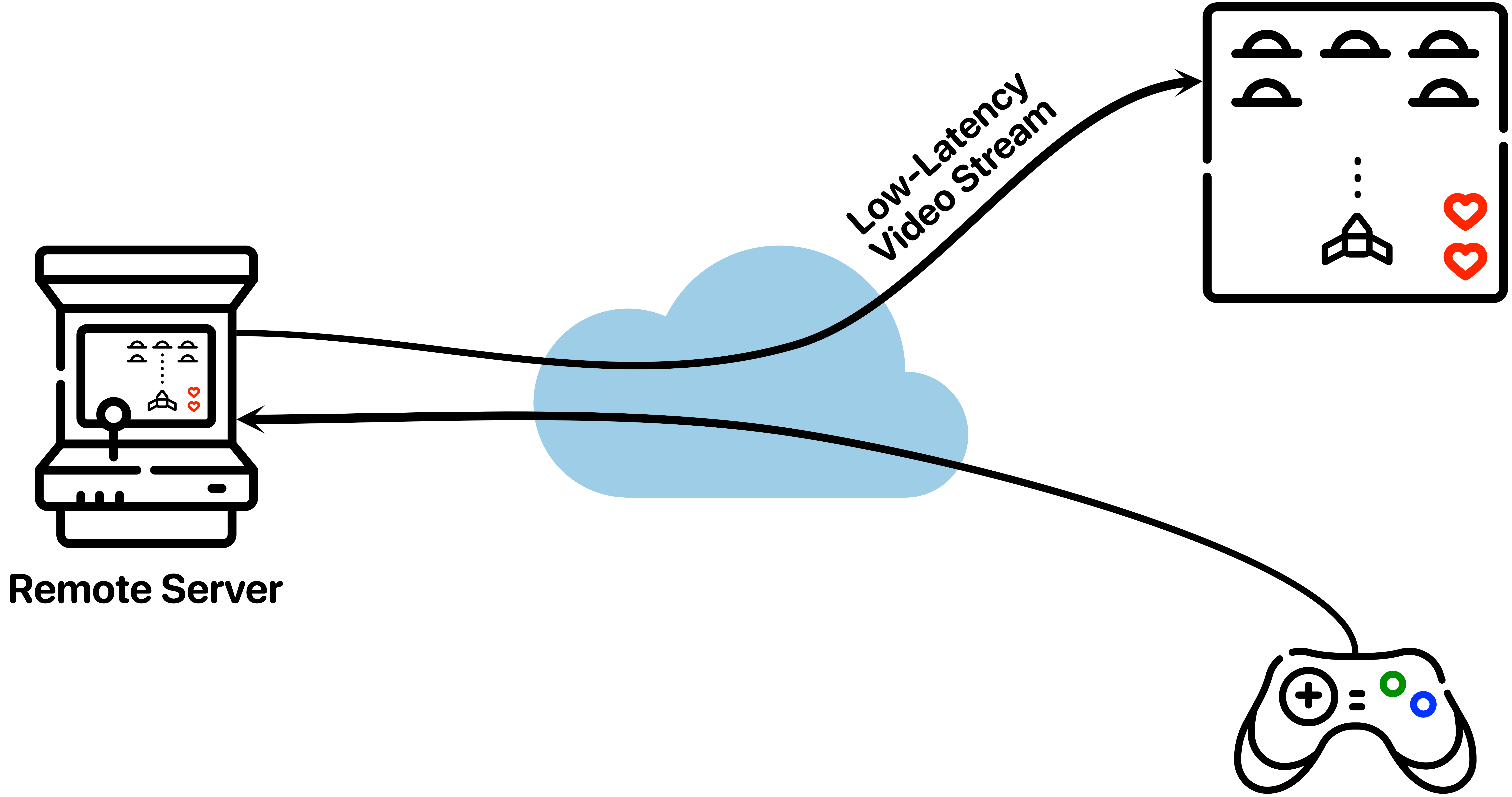
you see it



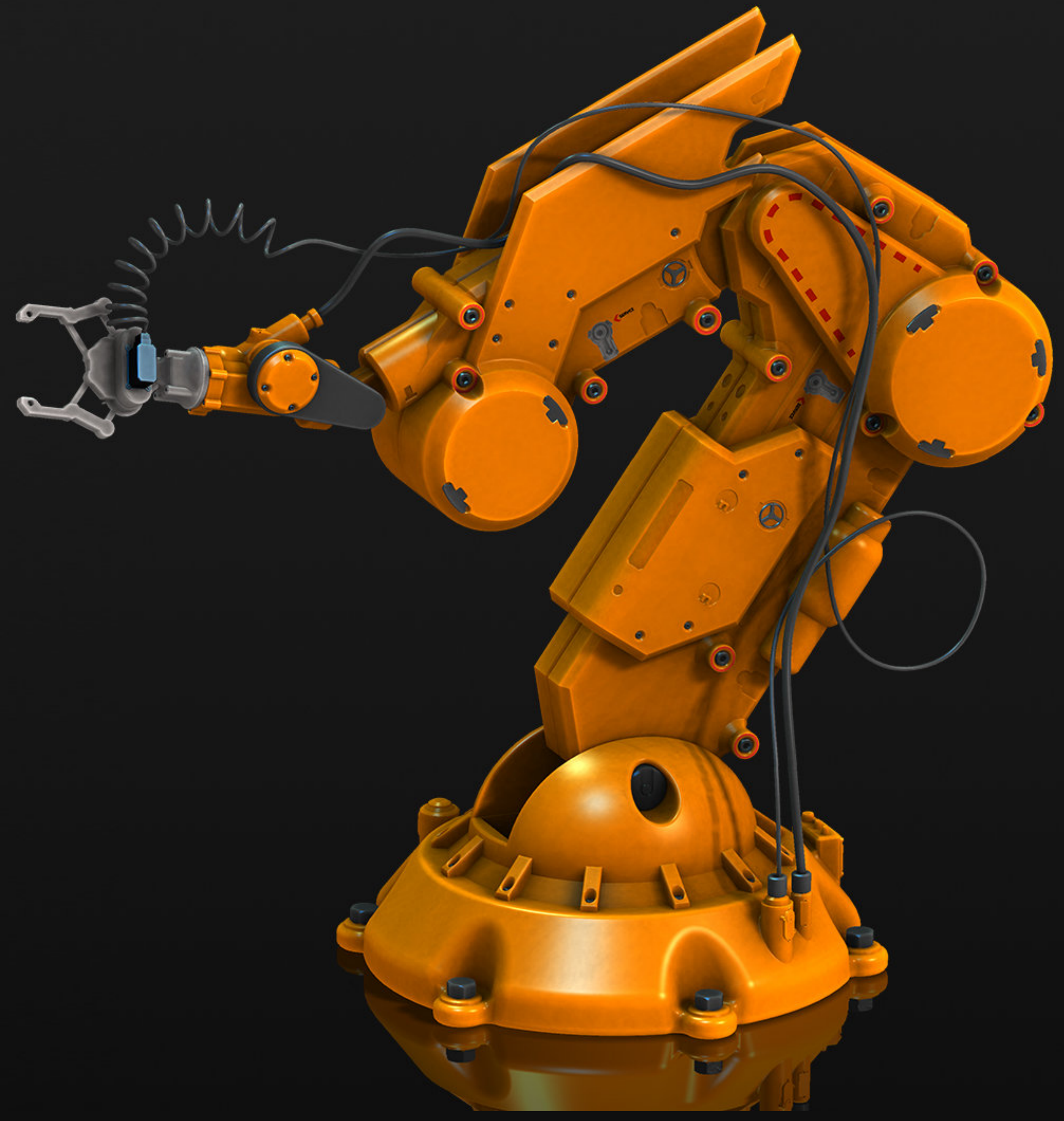
Low latency is required to maintain the **interactivity** of the application



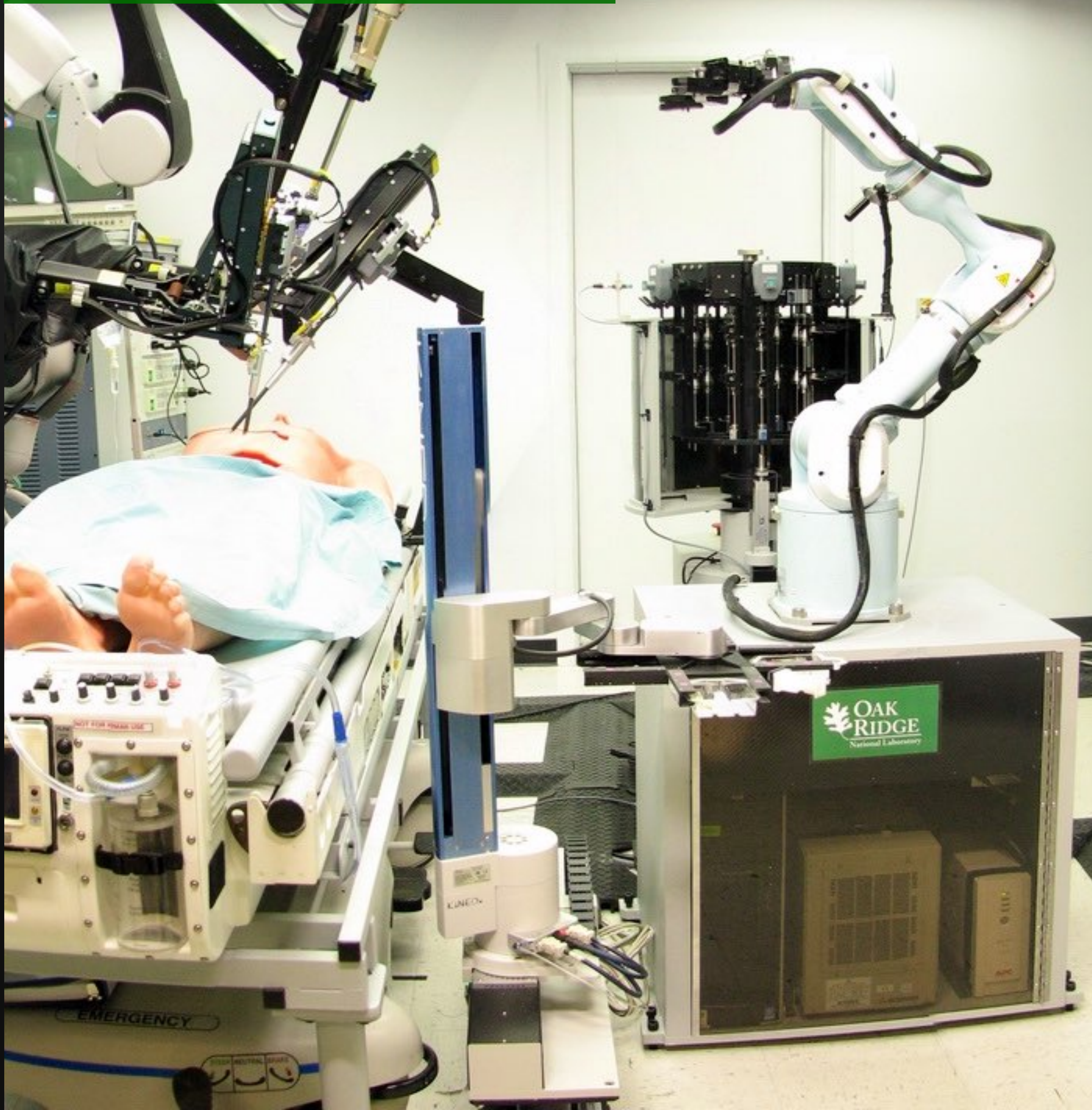
Cloud Video Gaming



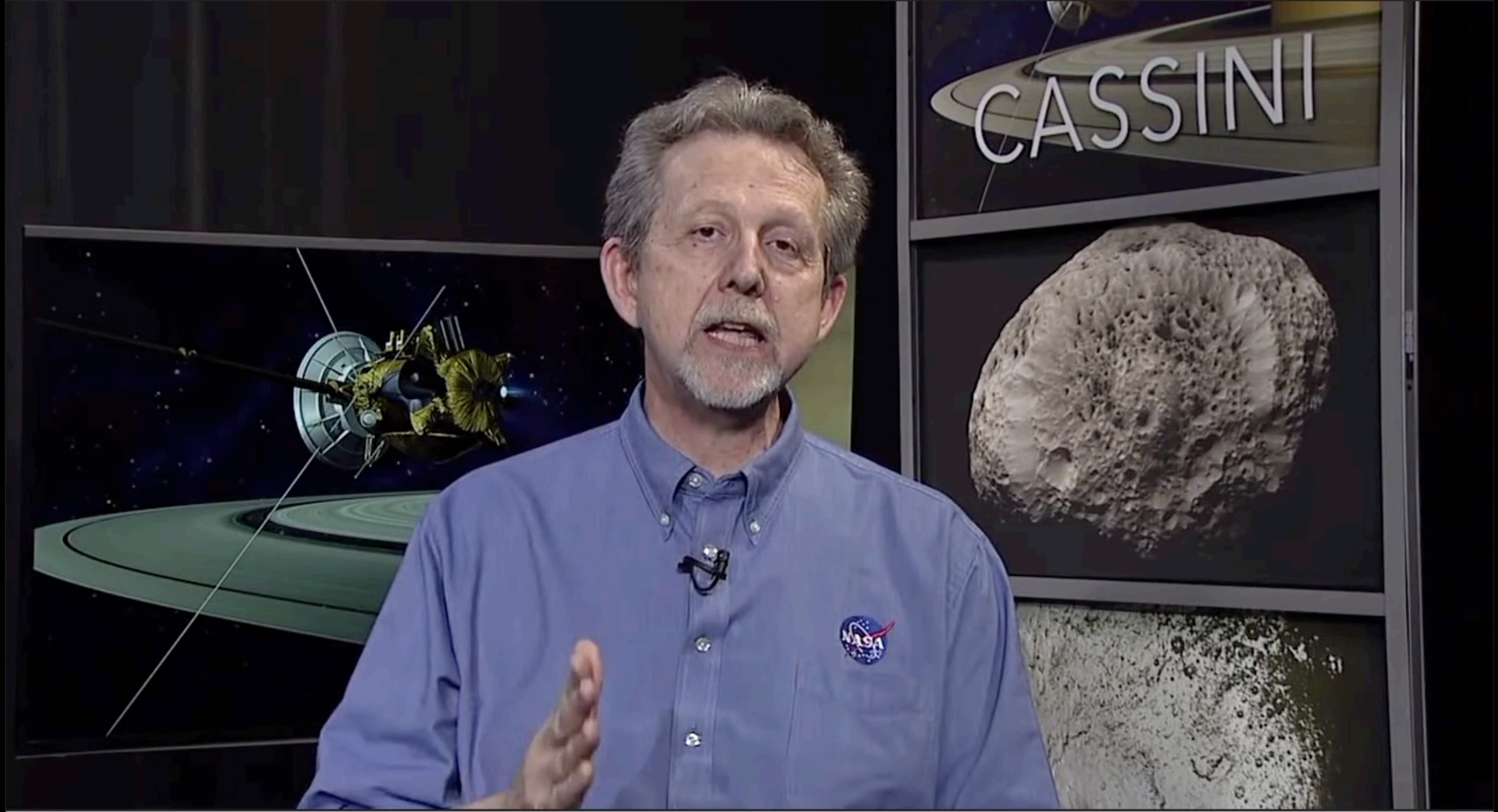
Teleoperation of Robots

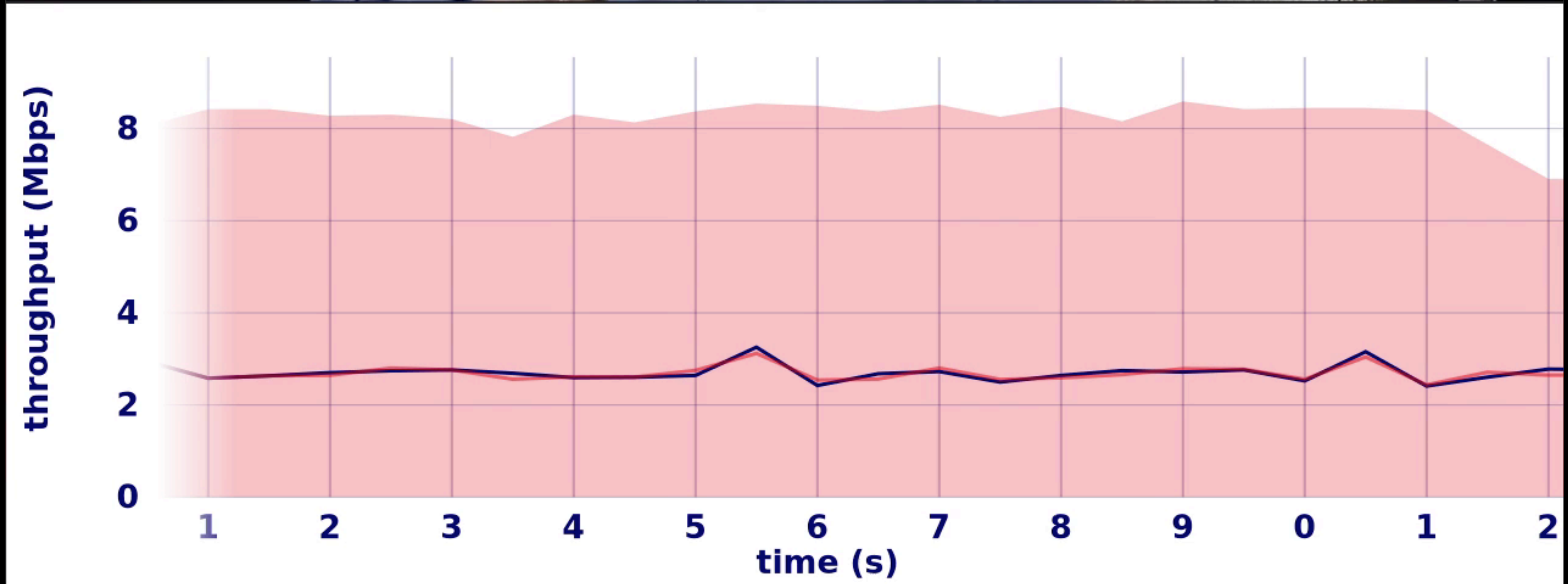


Remote Surgery



Video Conferencing

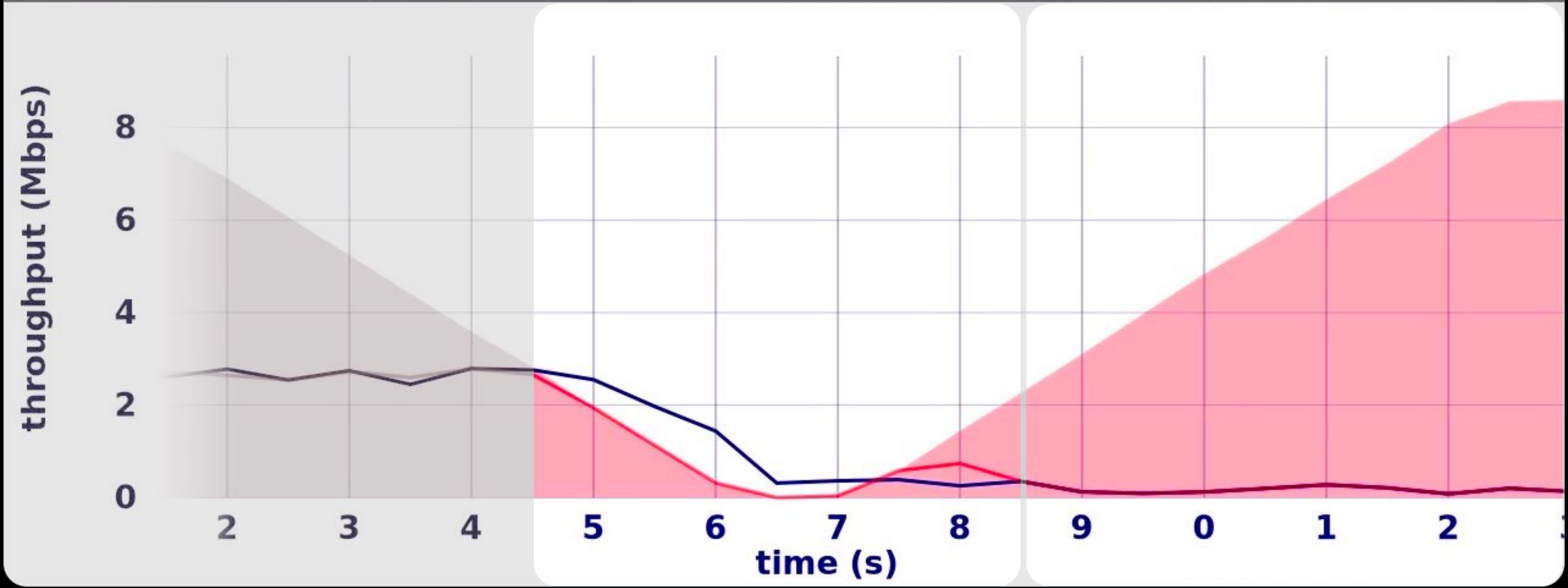




WebRTC
(Google Chrome)

slow reaction to network variations

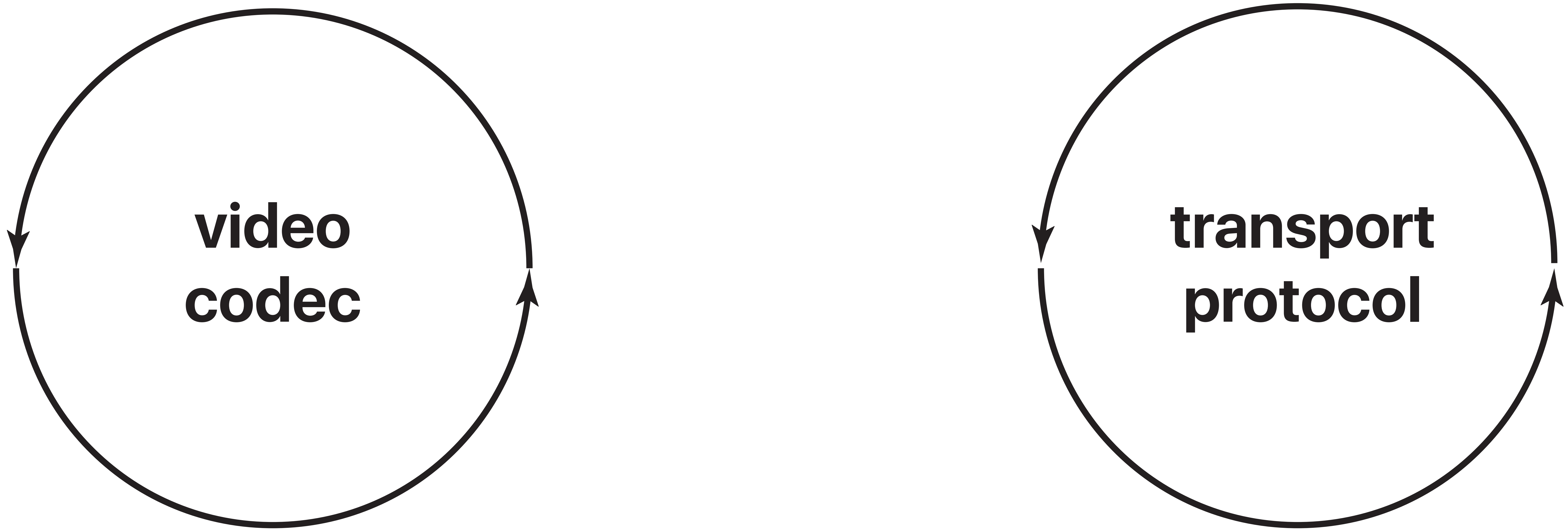
⇒ **stalls and glitches**



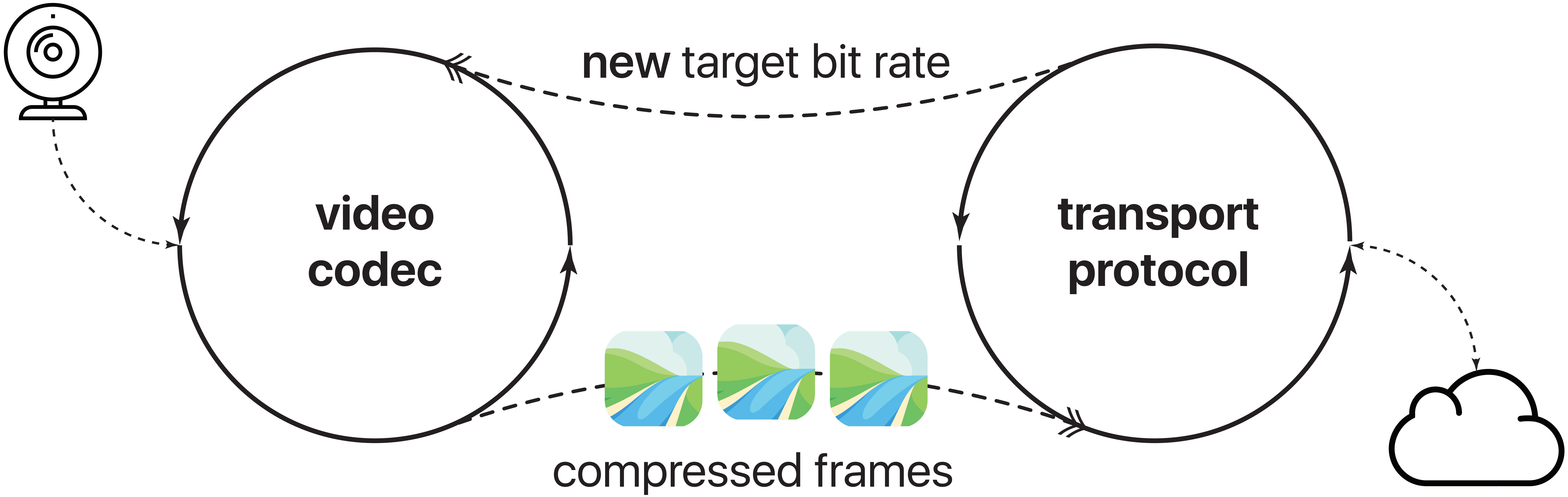
Enter *Salsify*

- Salsify is a new architecture for real-time Internet video.
  - Salsify tightly integrates a **video-aware transport protocol**, with a **functional video codec**, allowing it to **respond quickly to changing network conditions**.

Conventional design: two control loops at arm's length

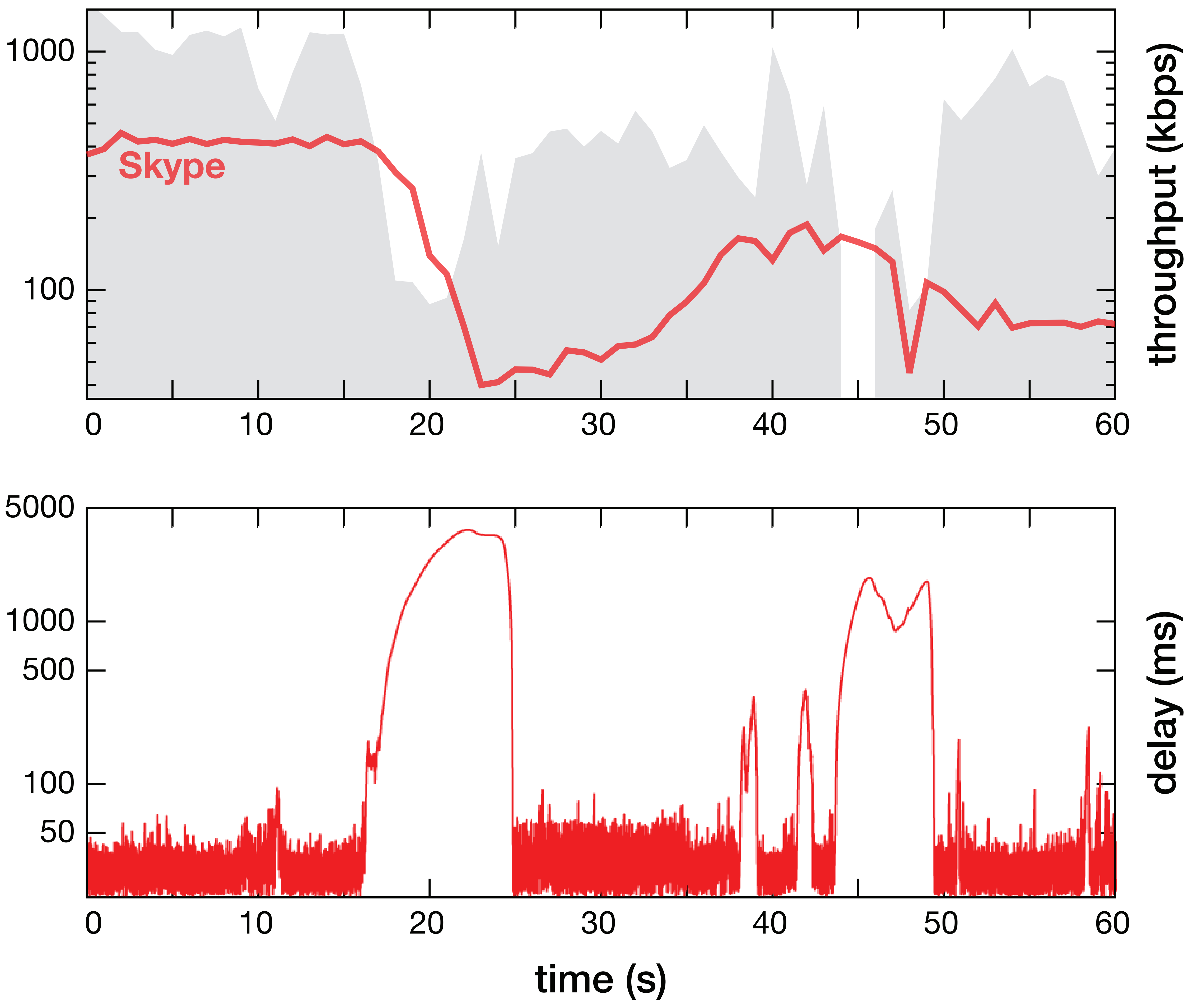


The narrow interface between codec and transport



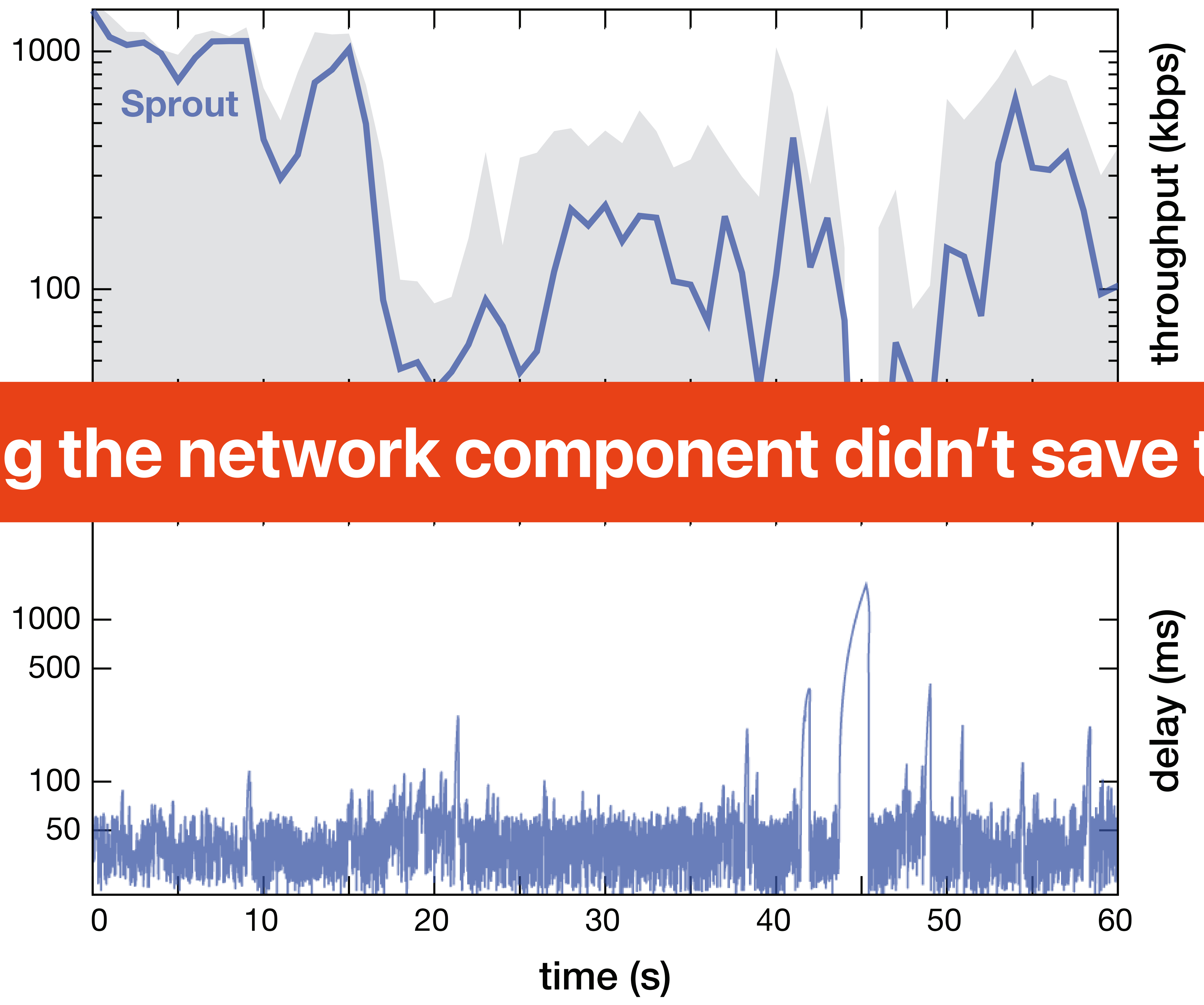
Decades of research and development on these components

“Let’s improve the transport”



K. Winstein, A. Sivaraman, H. Balakrishnan, “Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks,” NSDI’13

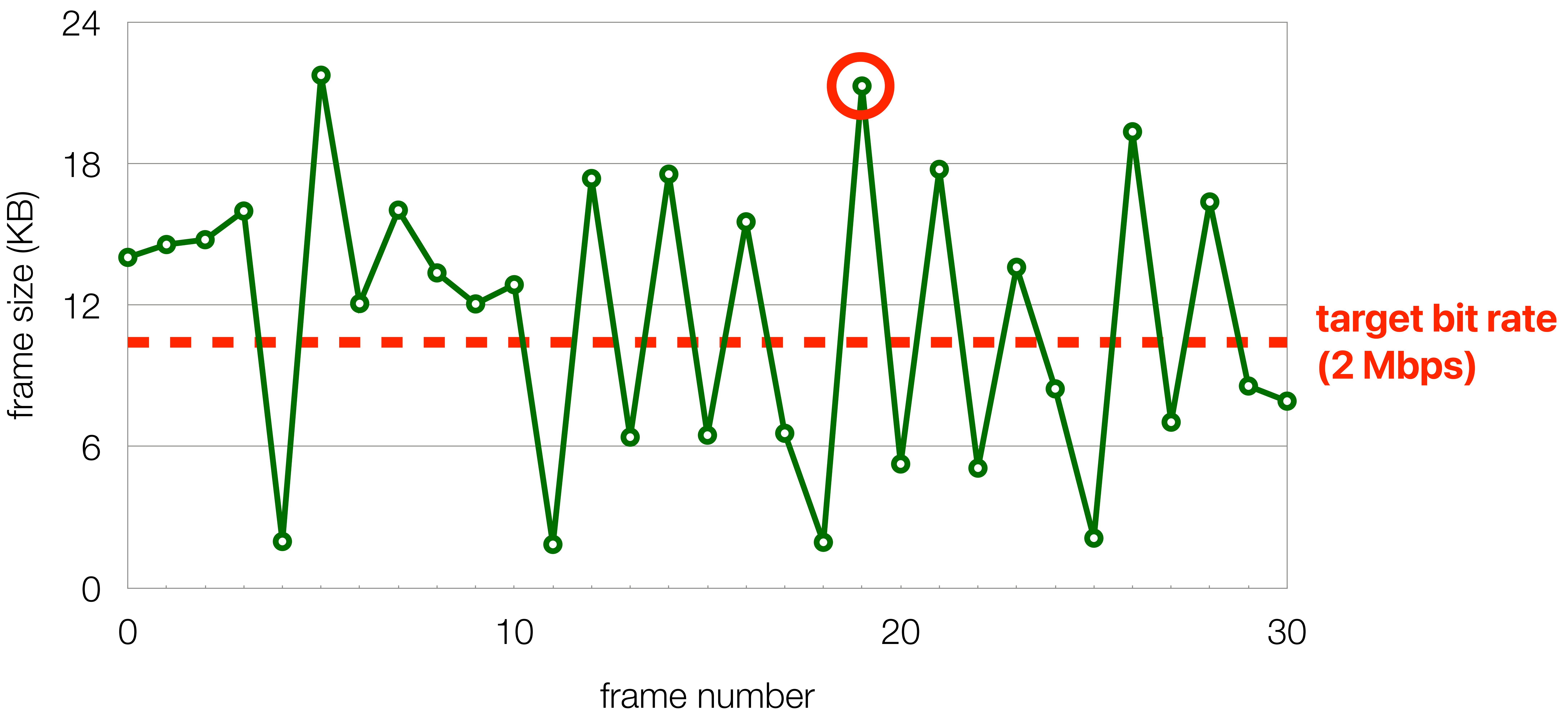
“Let’s improve the transport”



improving the network component didn’t save the day...

K. Winstein, A. Sivaraman, H. Balakrishnan, “Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks,” NSDI’13

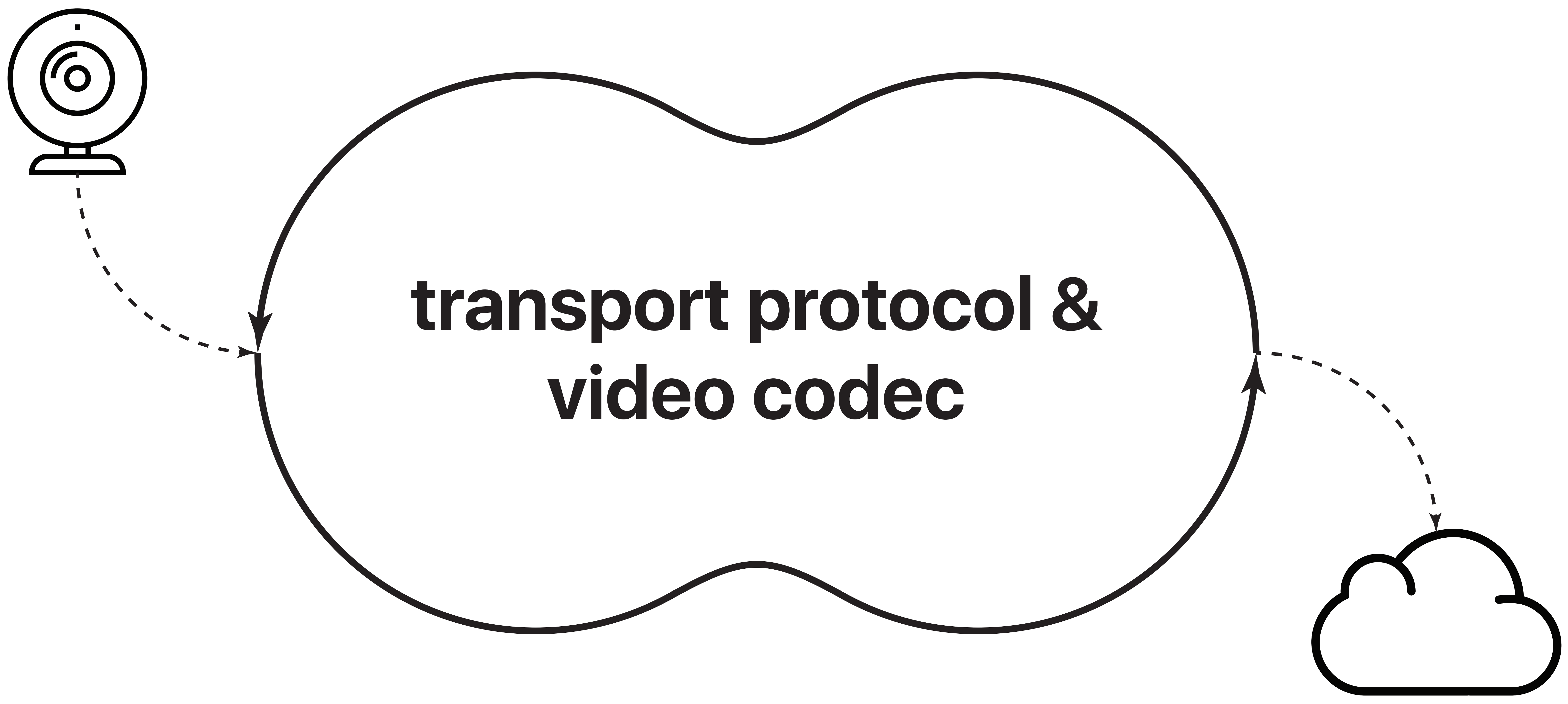
The video codec can only achieve the bit rate *on average*



The problem: codec and transport are *too* decoupled

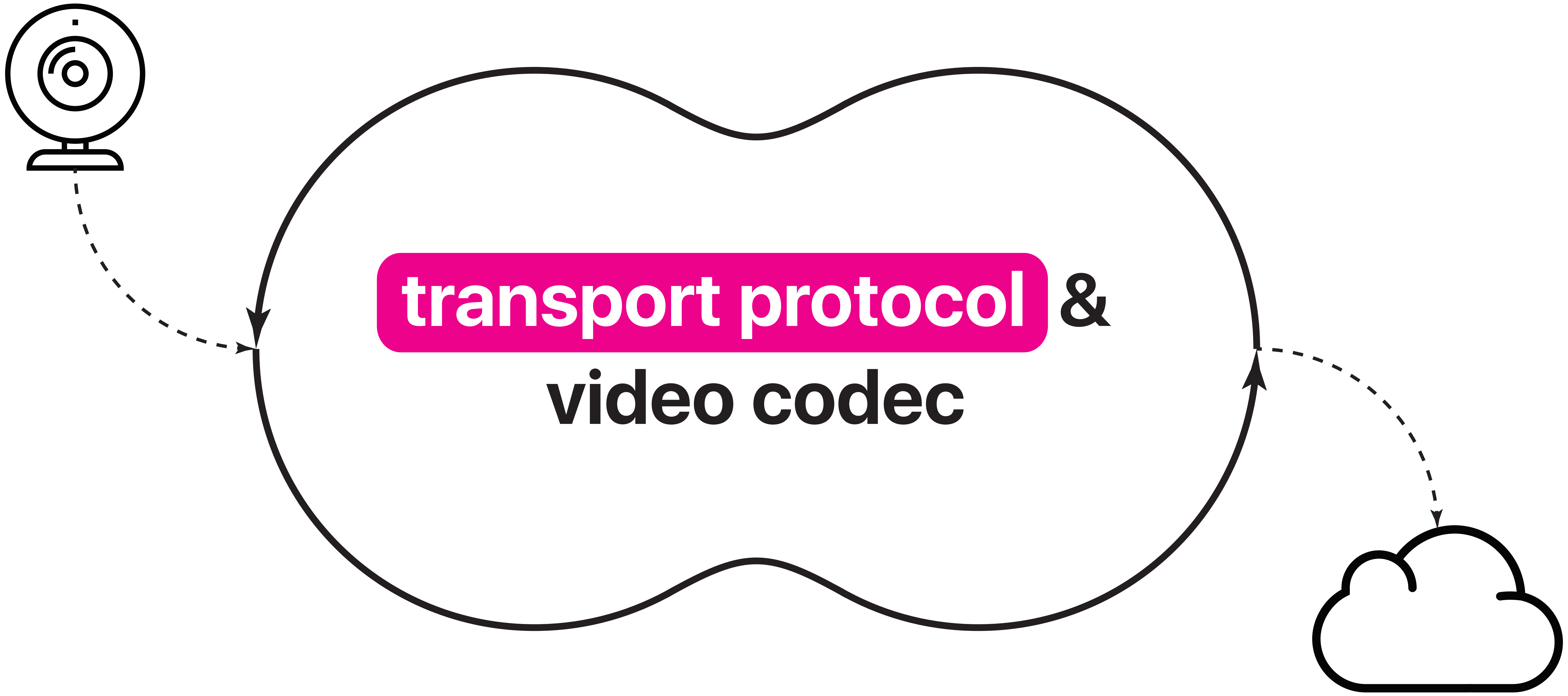
- The codec can only respond to changes in target bit rate over ***coarse time intervals***.
    - Individual frames may cause packet loss/queueing.
- The transport has little control over what codec produces.

Salsify explores a more tightly-integrated design



Salsify's architecture:

Video-aware transport protocol



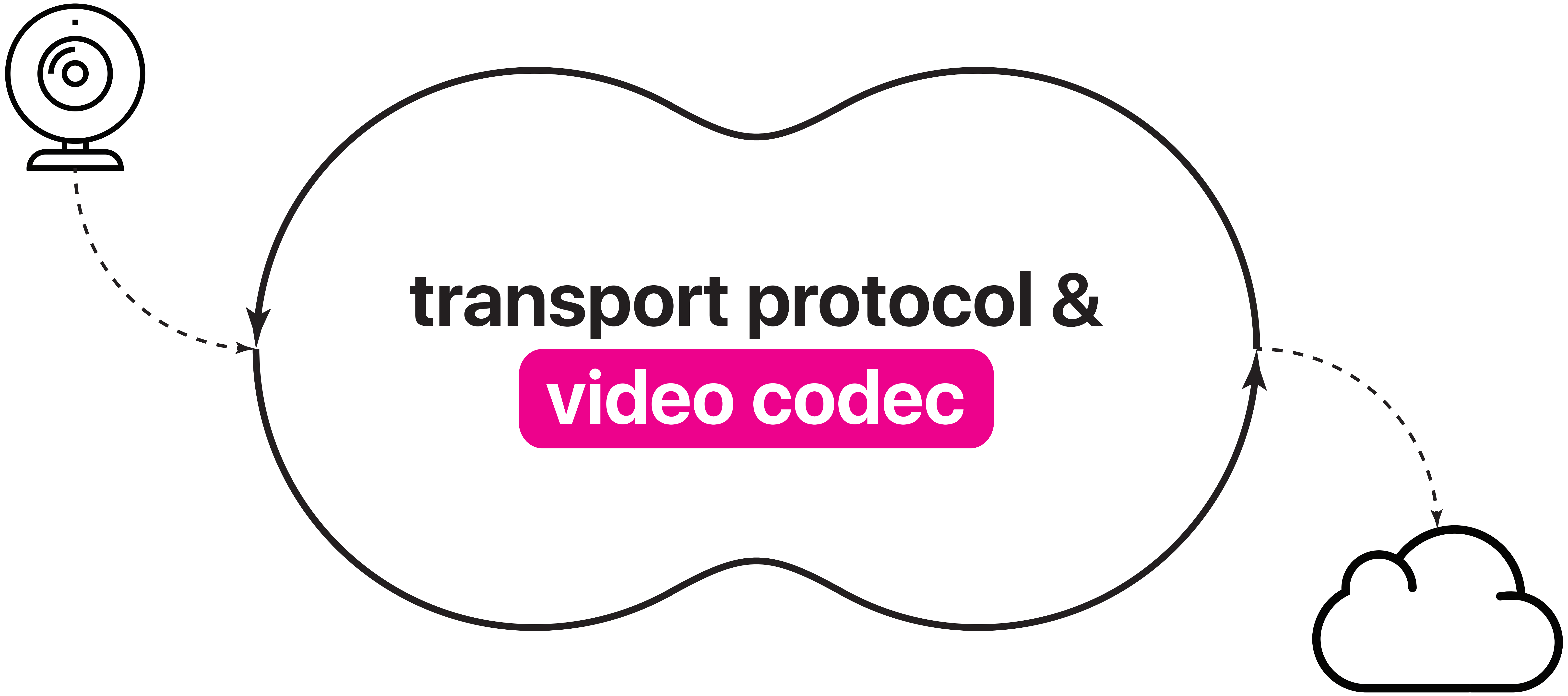
Video-aware transport protocol

"What should be the size of the next frame?"

- There's no notion of bit rate, only the next frame size!
- Inspired by *packet pair* and *Sprout-EWMA*, transport uses **packet inter-arrival time**, reported by the receiver.

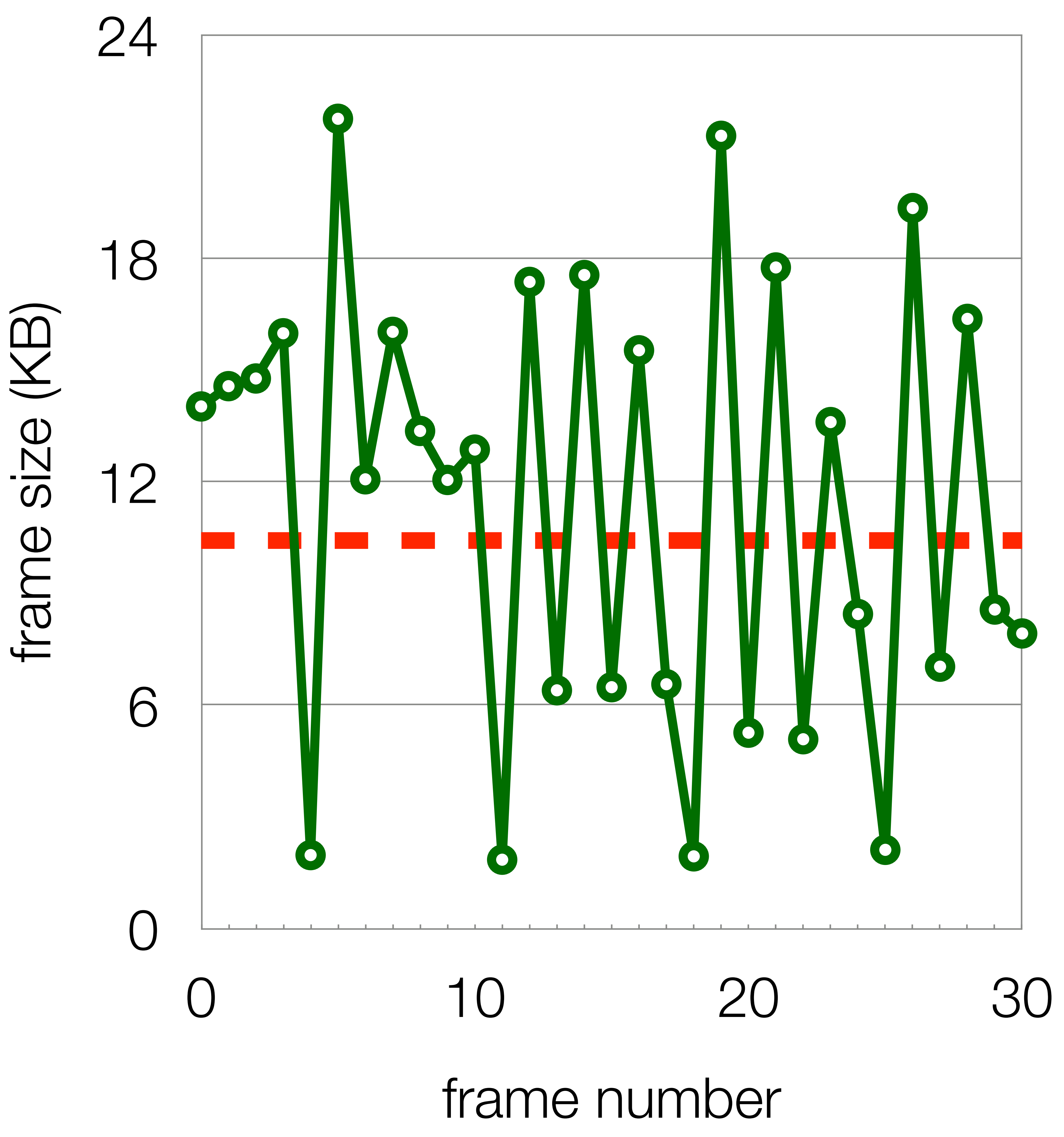
Salsify's architecture:

Functional video codec



The encoder can only know the output size *after the fact.*

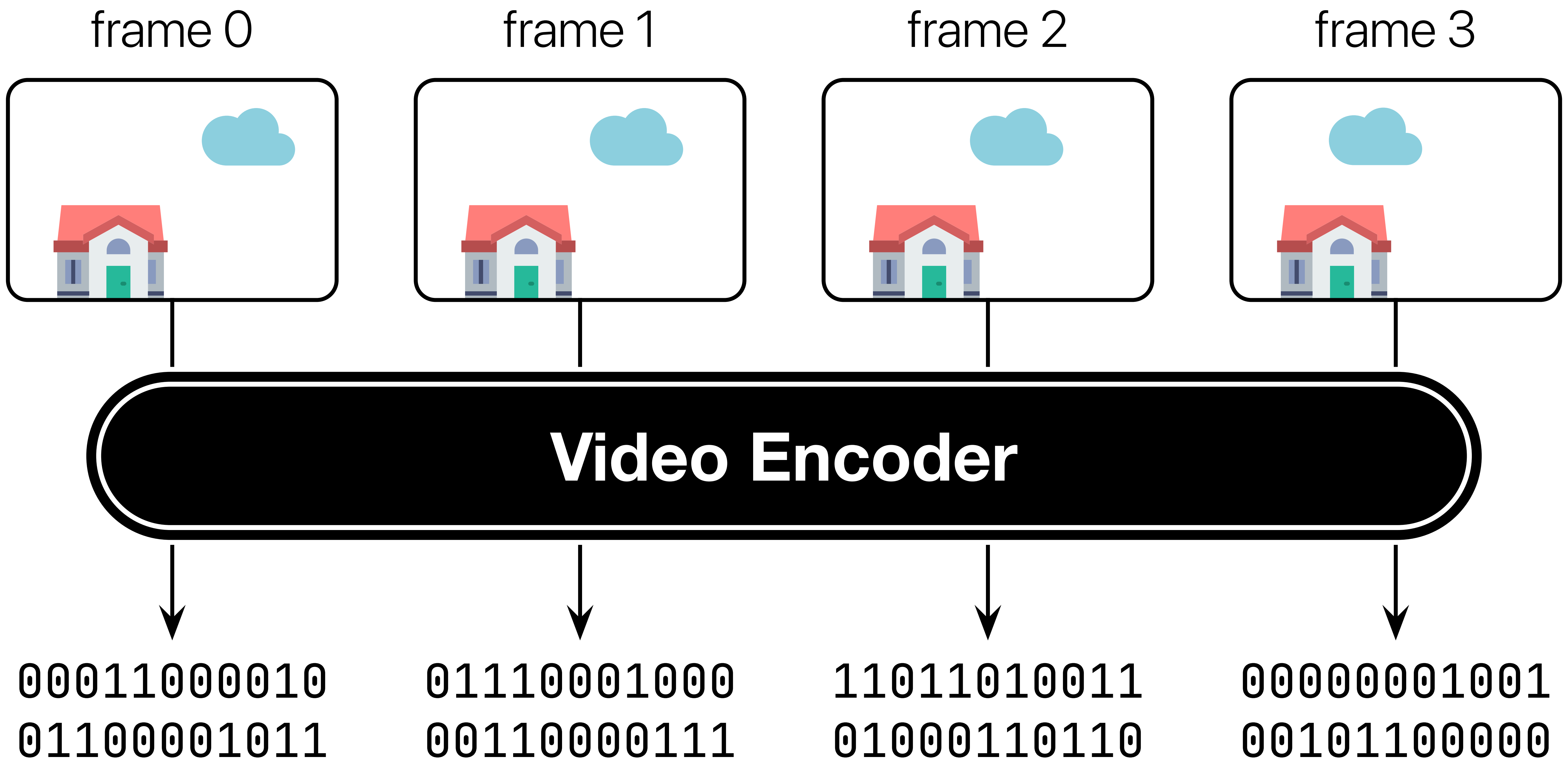
It's challenging for **any codec** to choose the appropriate quality settings upfront to meet a **target size**—they tend to over-/undershoot the target.



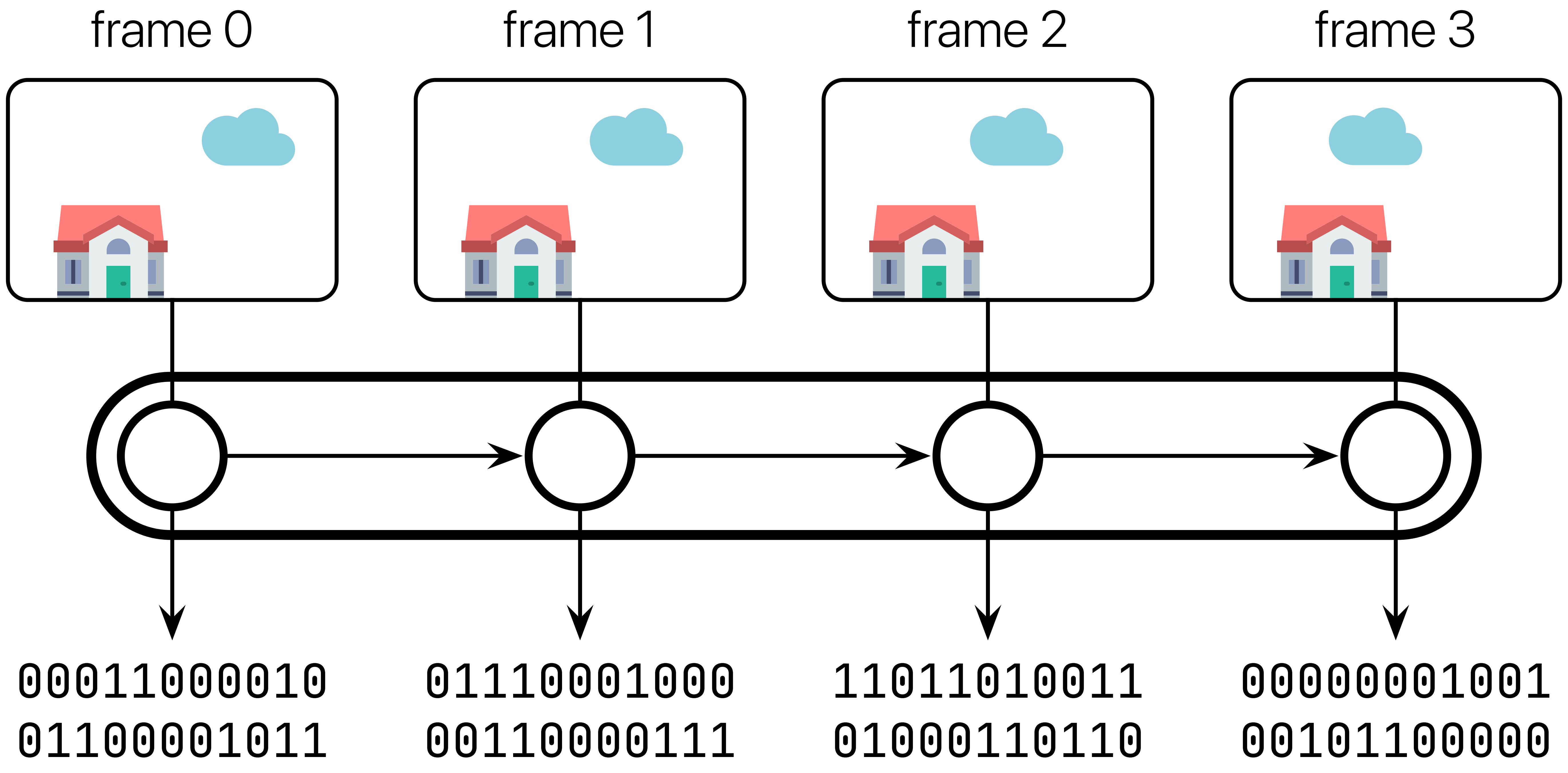
The challenge:
Getting an accurate frame out of an inaccurate codec

- **Trial and err**
  Encode with c … fits.

Video encoder turns frames into a compressed bitstream



Encoder is *stateful*



There's no way to undo an encoded frame in current codecs

`codec.encode([🌇, 🏙️, …]) → bytestream…`

The state is internal to the encoder—no way to save/restore the state.

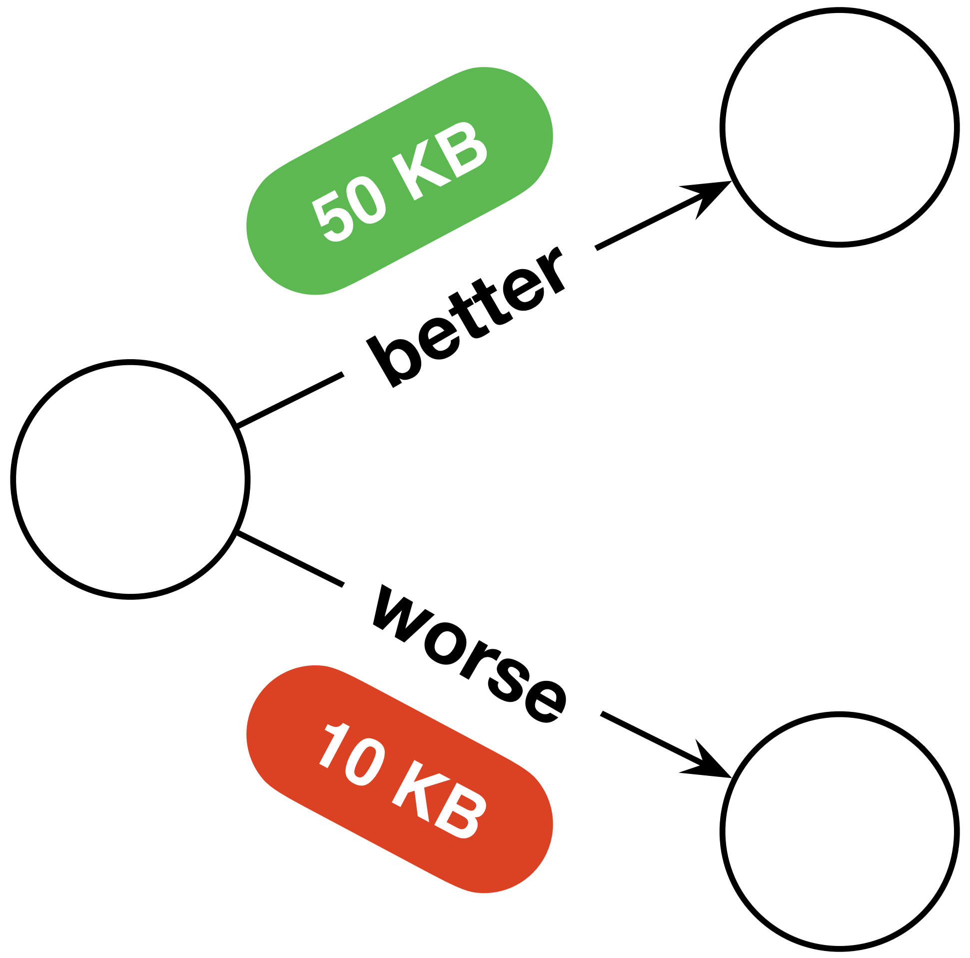
Functional video codec to the rescue

encode(*state*, 🏙) → *state′*, frame

Salsify's functional video codec exposes the state that can be saved/restored.

Order two, pick the one that fits!

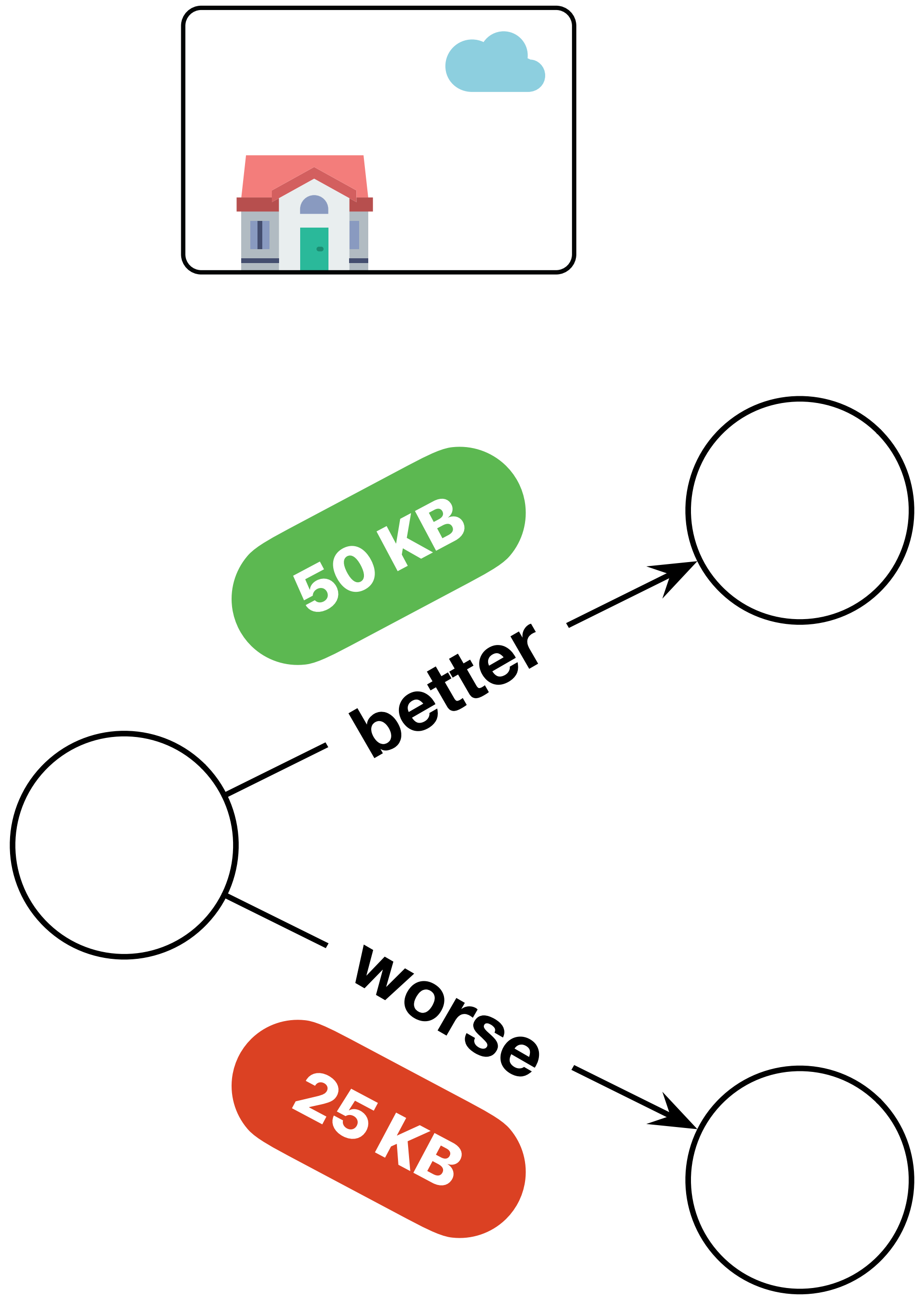
- Salsify's functional video codec can **explore different execution paths** without committing to them.
- For each frame, codec presents the transport with *three* options:
  - 🔺 A slightly-higher-quality version,
  - 🔻 A slightly-lower-quality version,
  - ❌ **Discarding the frame.**



Salsify's architecture:
Unified control loop

Codec → Transport

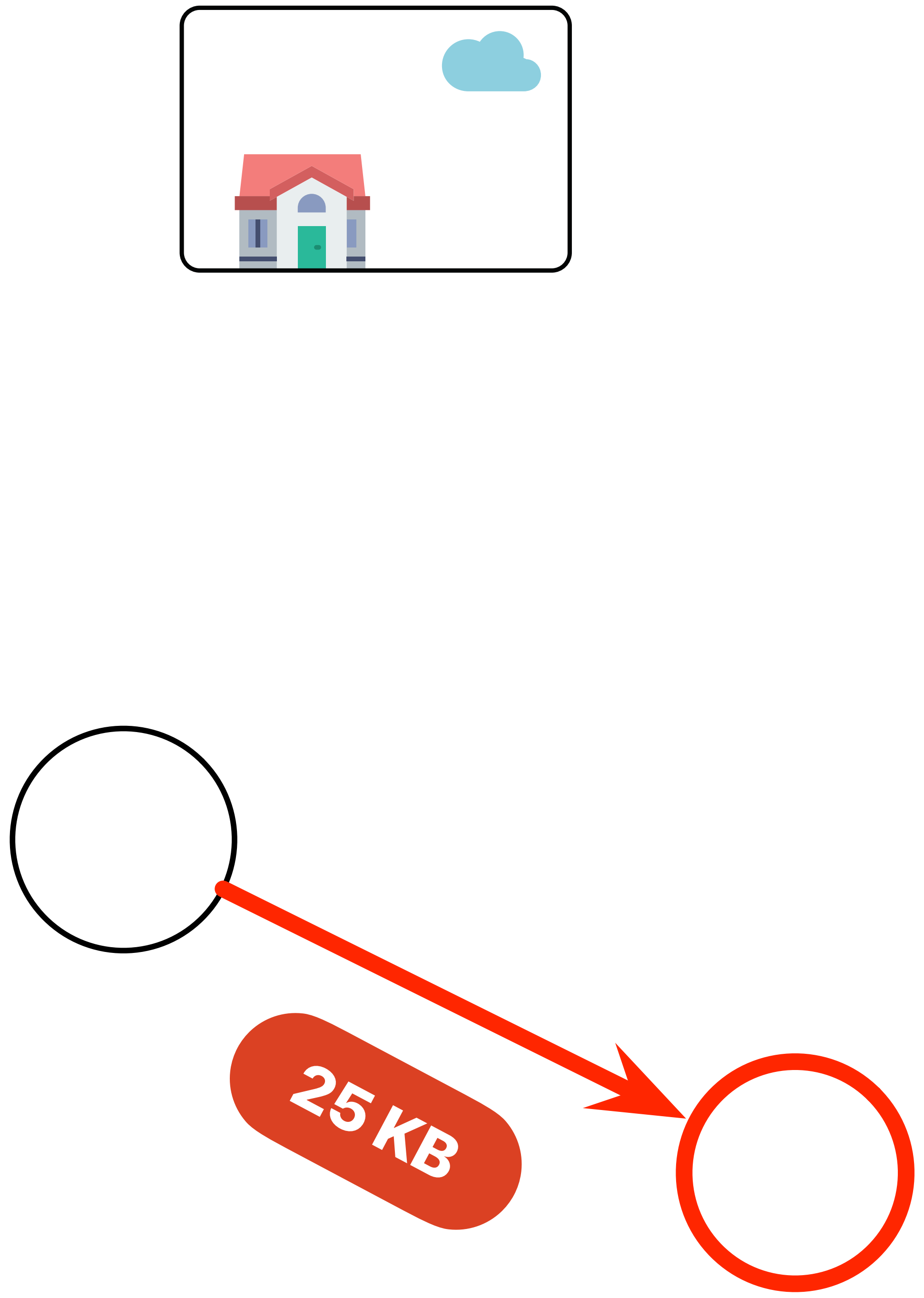
“Here’s two versions of the current frame.”



target frame size 30 KB

Transport → Codec

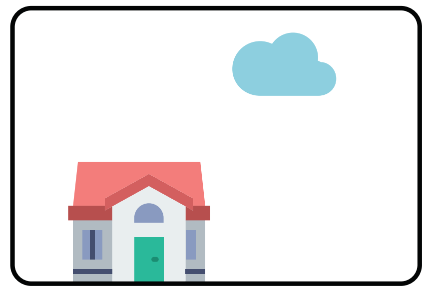
“I picked option 2. Base the next frame on its exiting state.”

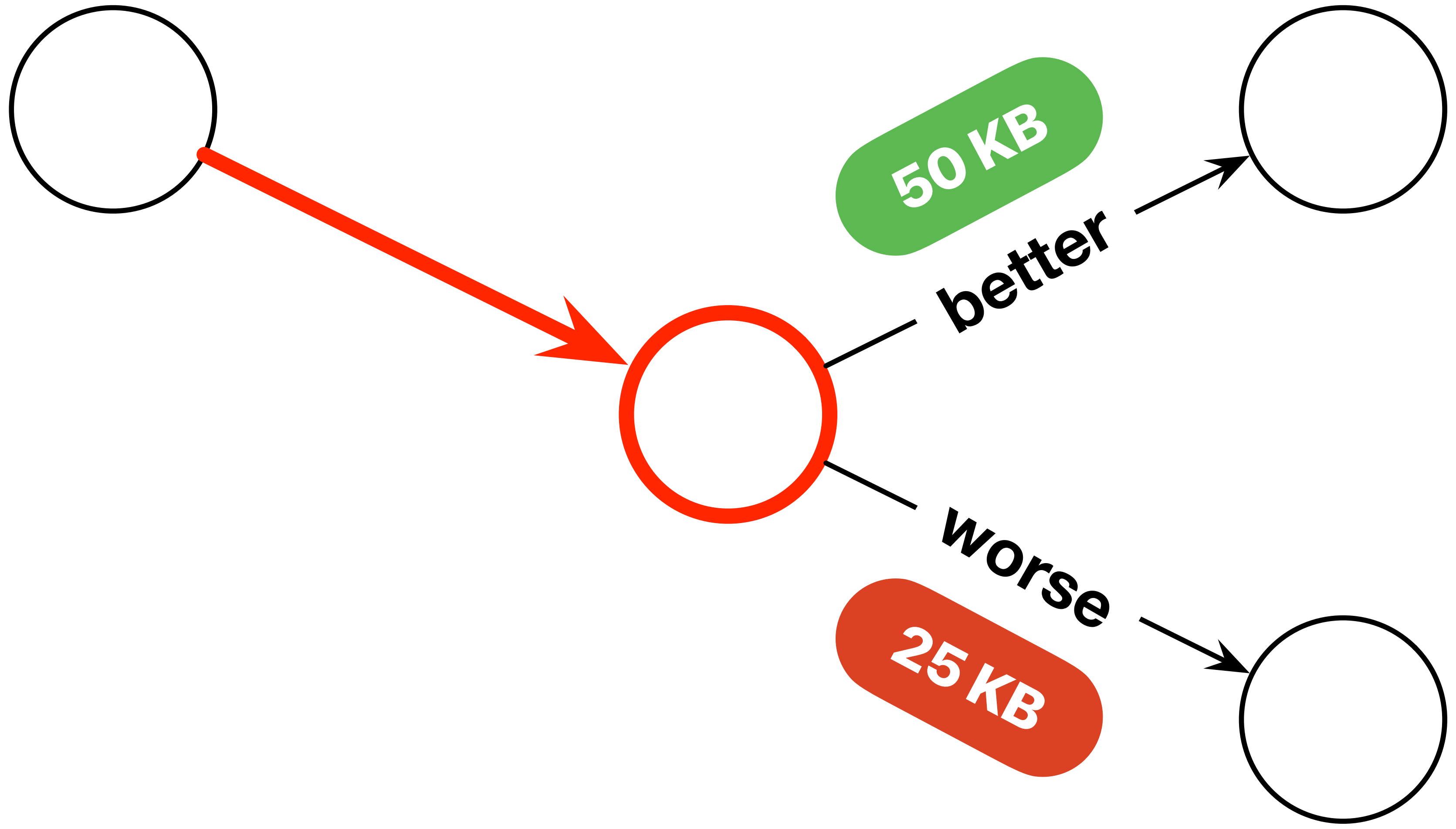


target frame size 30 KB

Codec → Transport

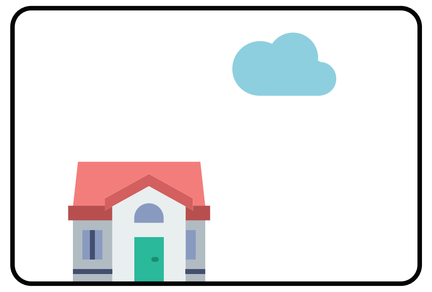
“Here’s two versions of the latest frame.”

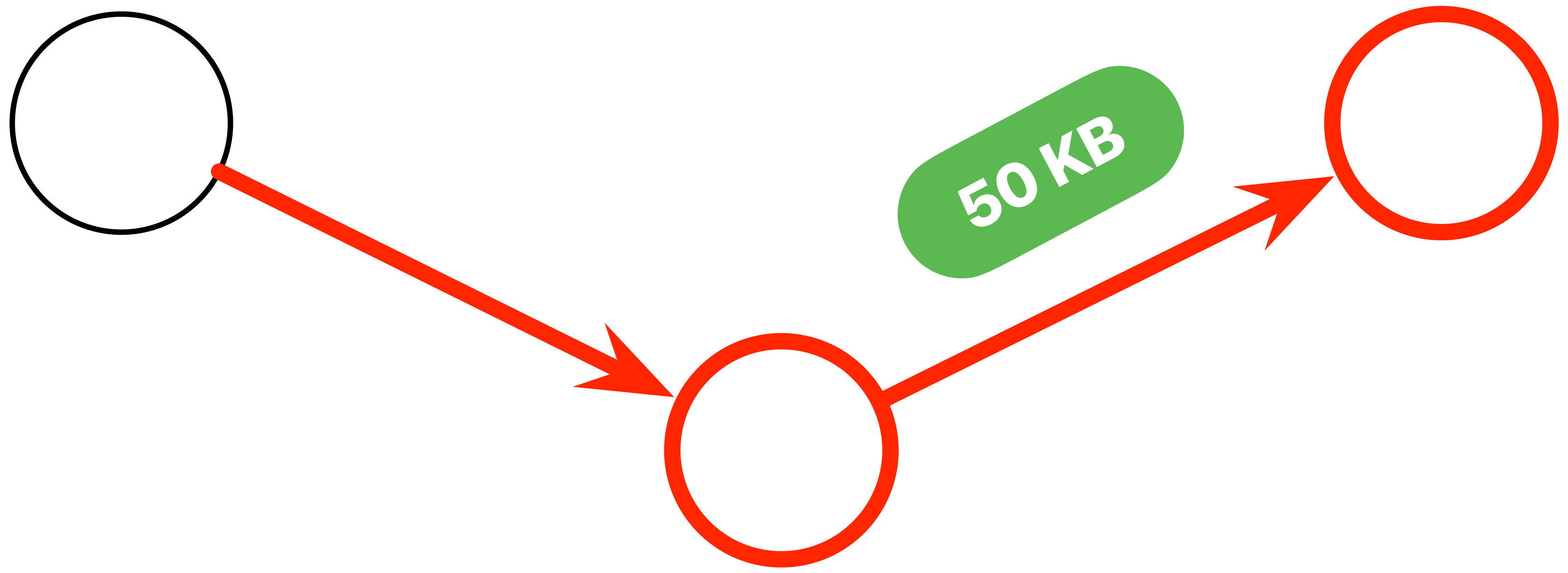




target frame size 55 KB

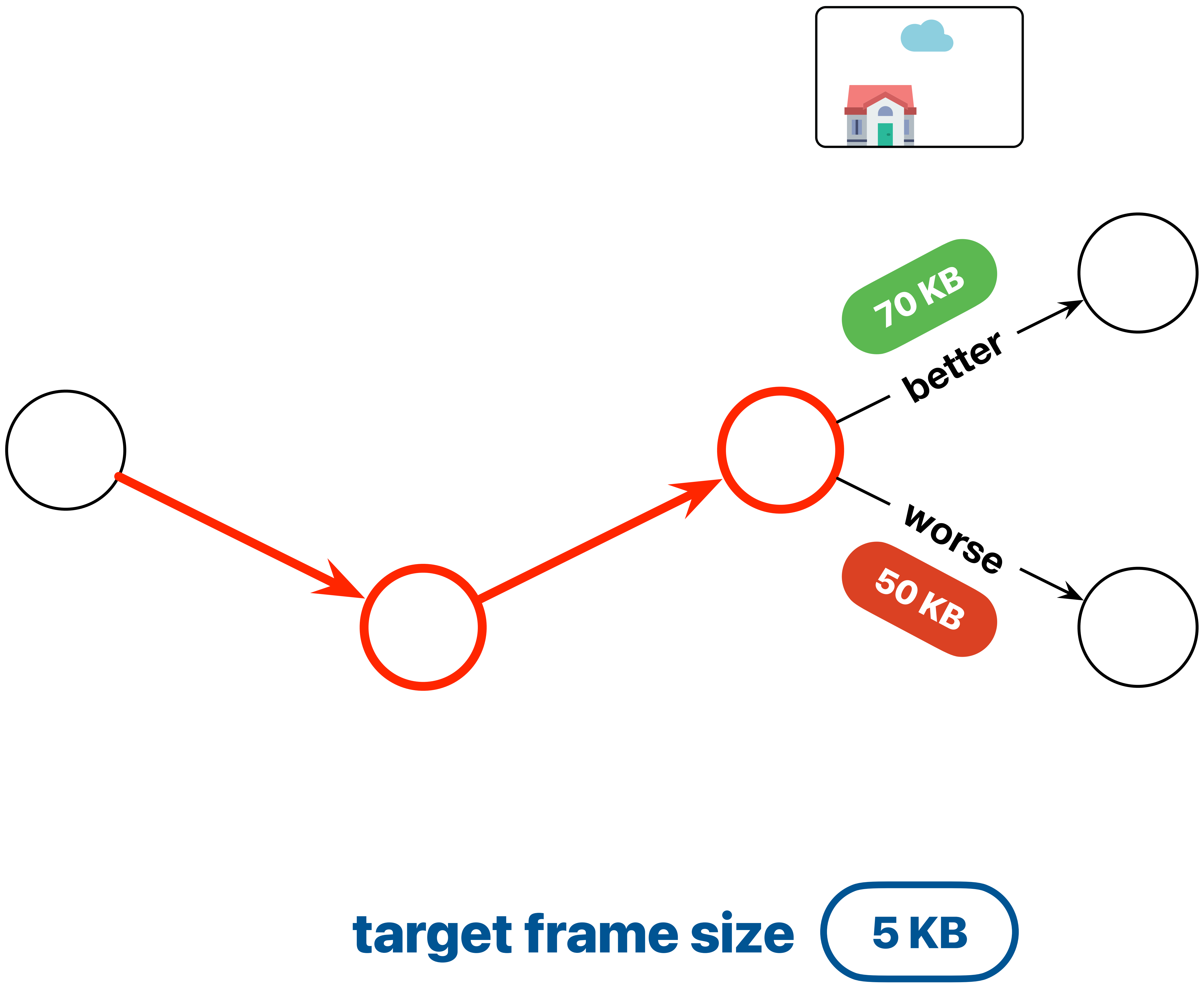
Transport → Codec
“I picked option 1. Base the next frame on its exiting state.”





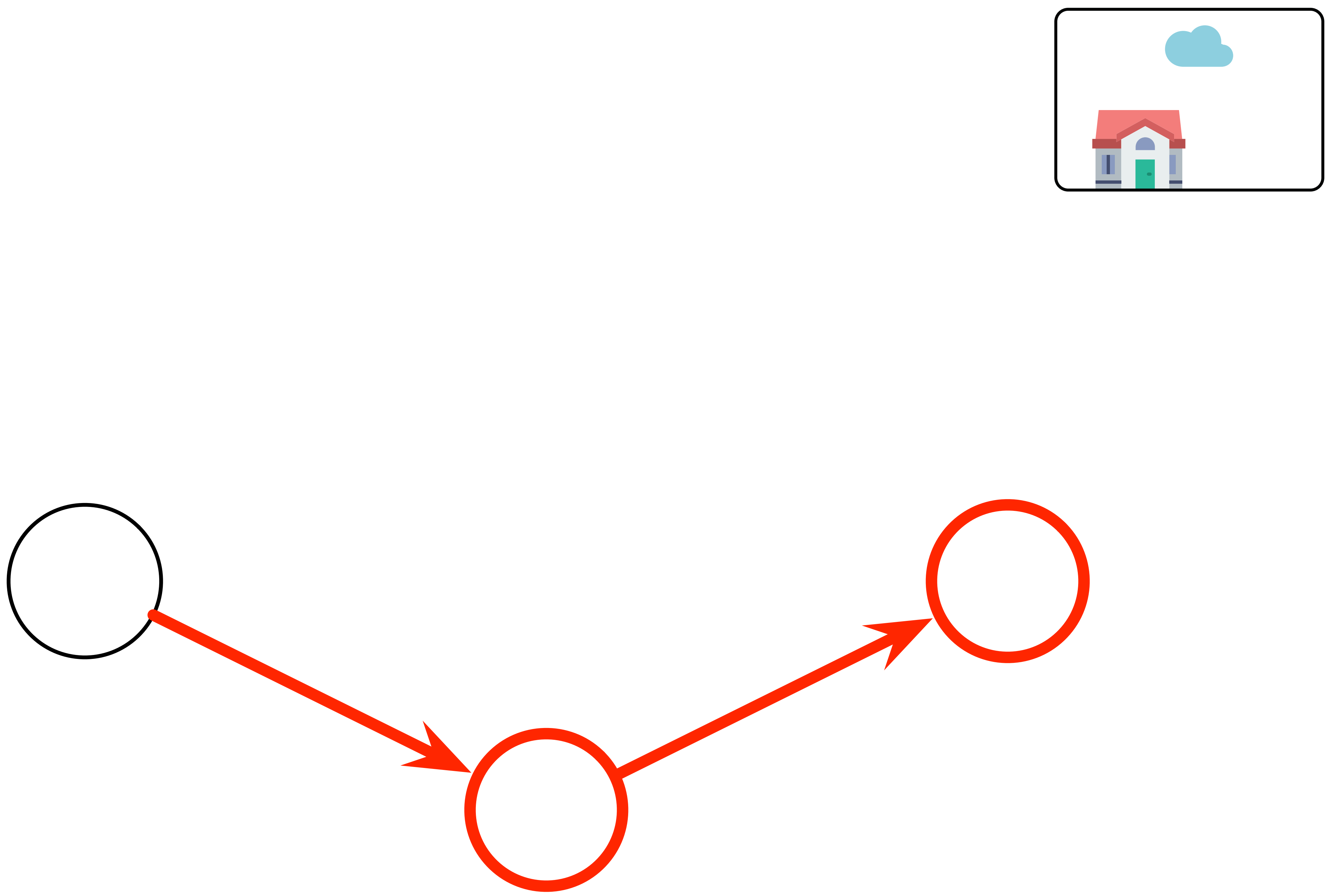
target frame size 55 KB

Codec → Transport
“Here’s two versions of the latest frame.”



Transport → Codec

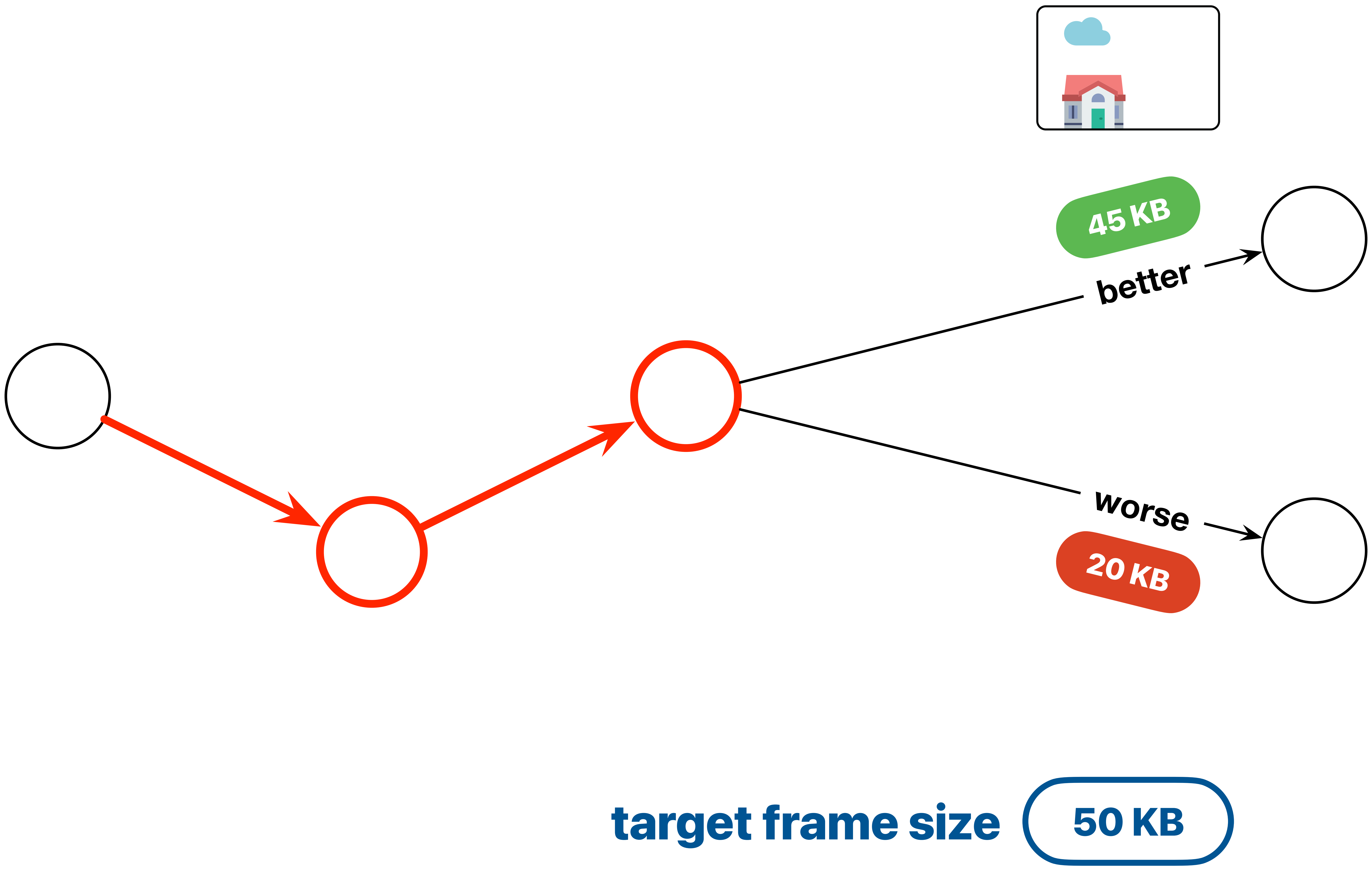
“I cannot send any frames right now. Sorry, but discard them.”



target frame size 5 KB

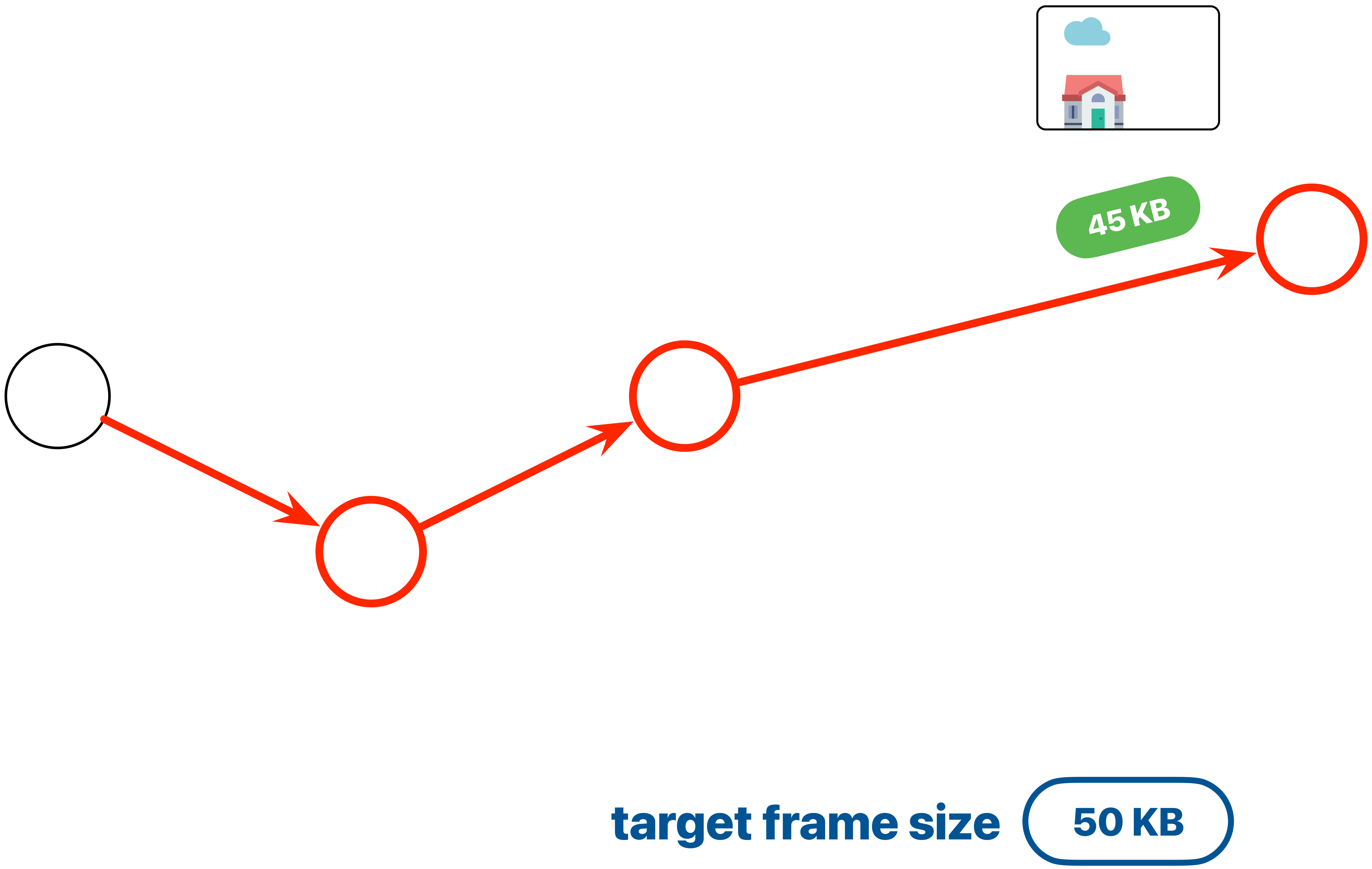
Codec → Transport

“Fine. Here’s two versions of the latest frame.”



Transport → Codec

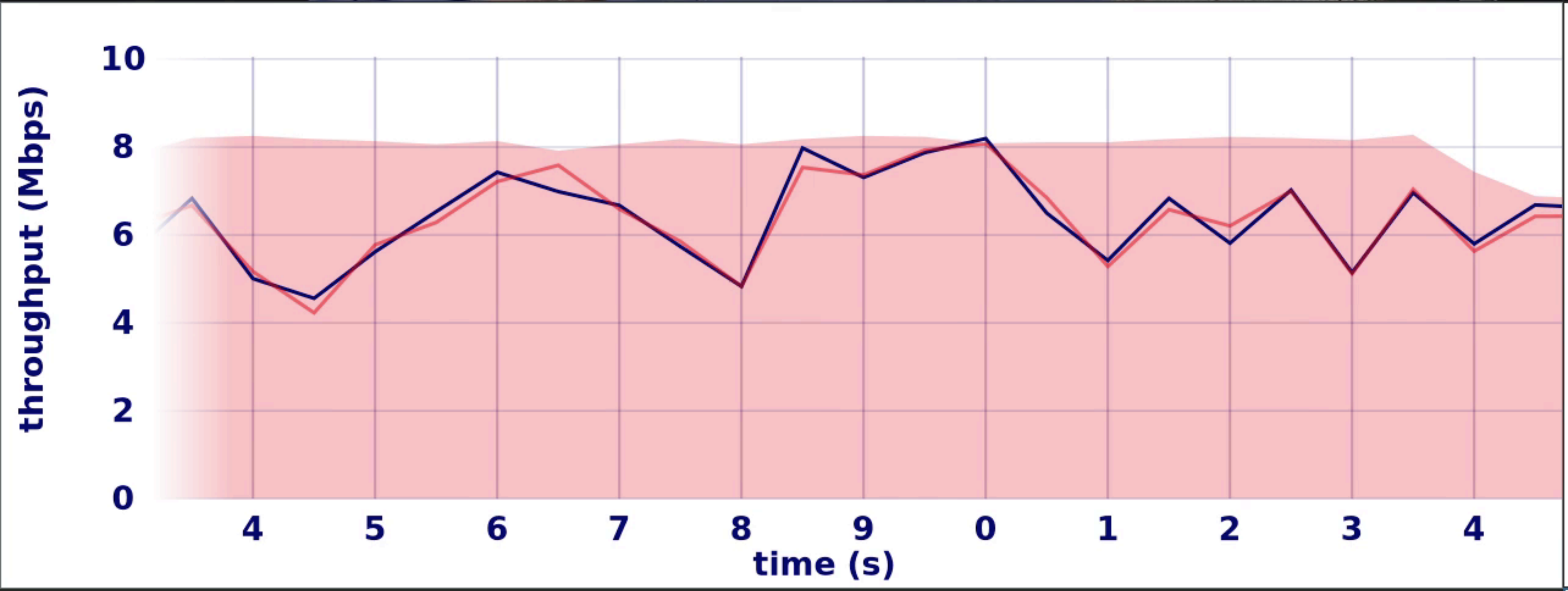
“I picked option 1. Base the next frame on its exiting state.”

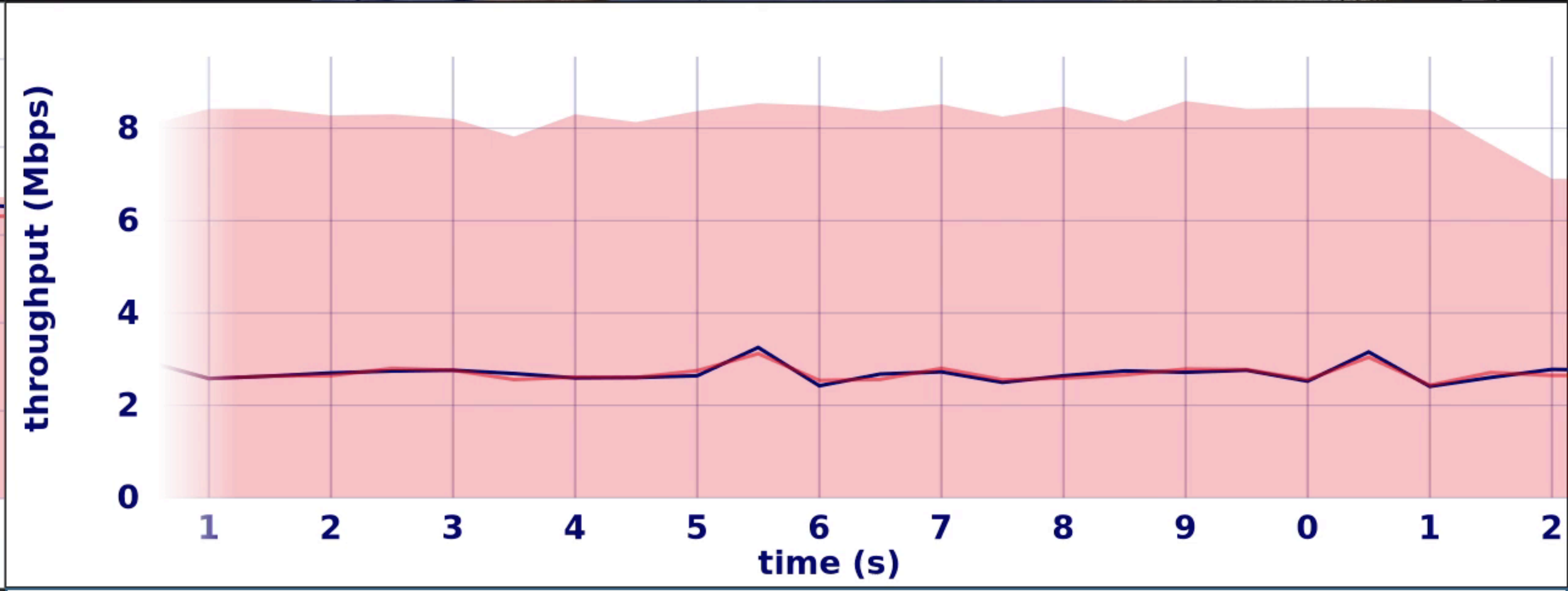


There's no notion of **frame rate** or **bit rate** in the system.
Frames are sent when the network can accommodate them.

Evaluation of Salsify

Network Variations



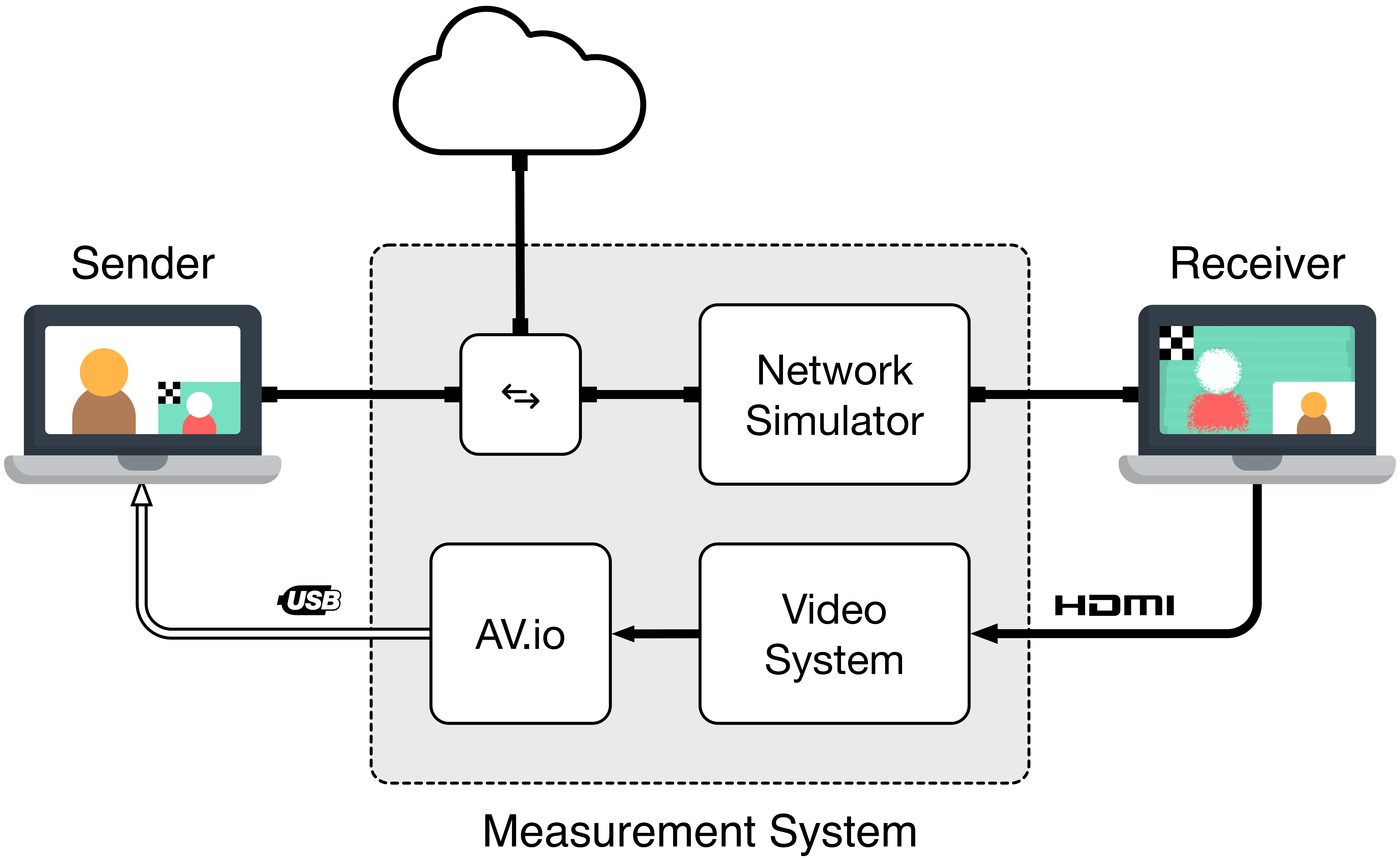


Salsify

WebRTC (Chrome 65)

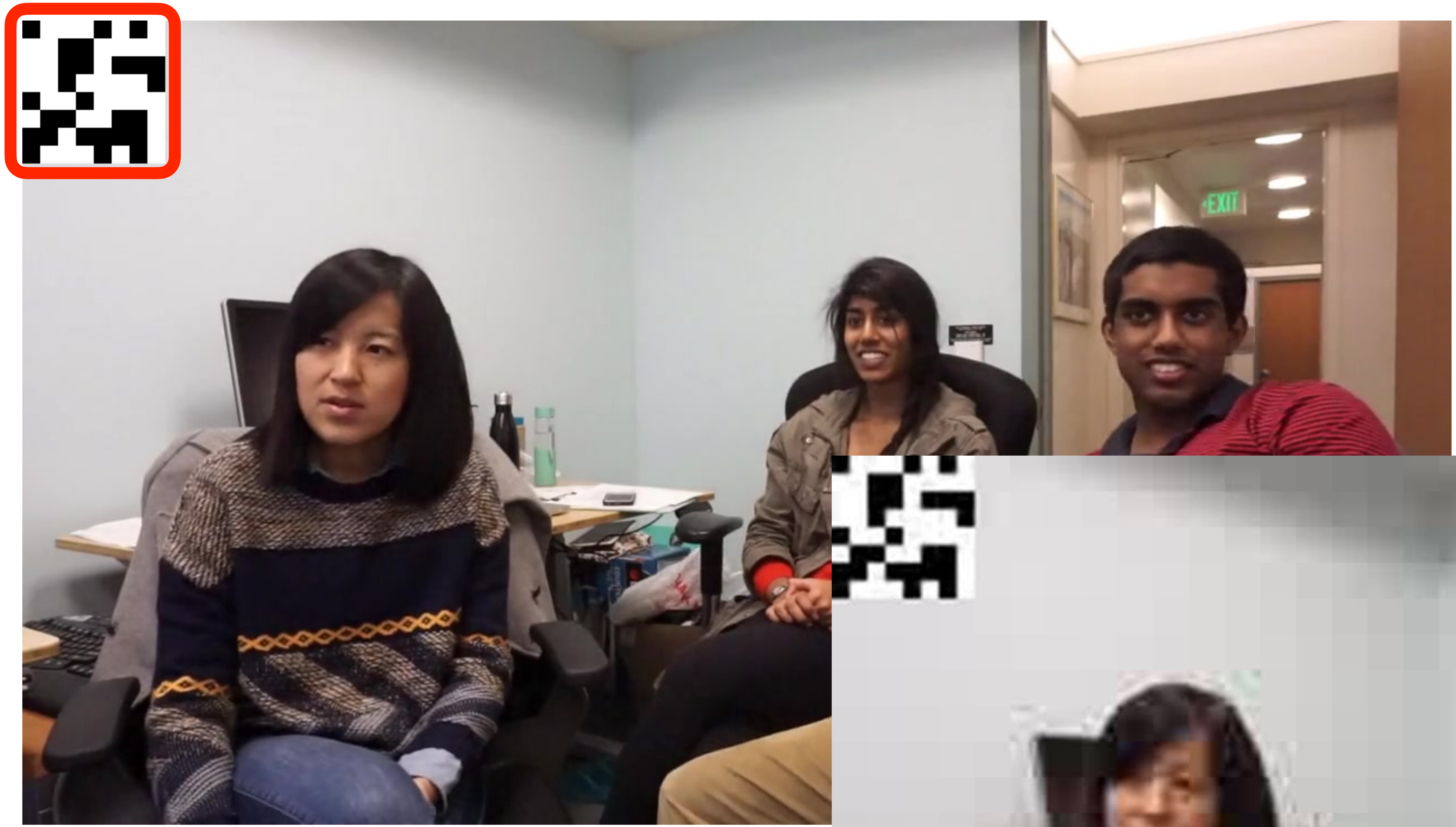
Salsify

WebRTC (Chrome 65)

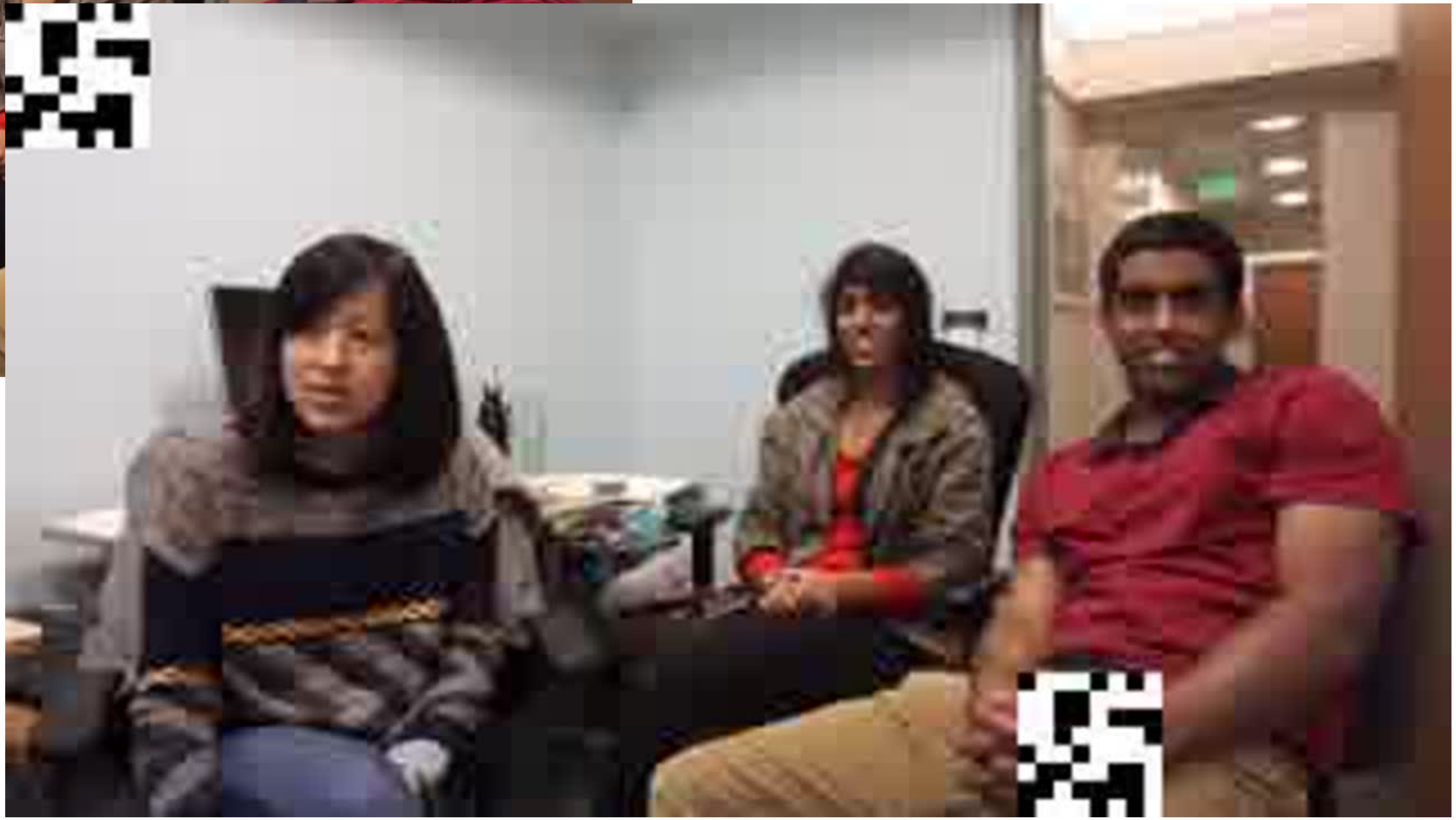


Sender
Receiver
Network Simulator
AV.io
Video System
USB
HDMI
Measurement System


emulated network
barcoded video
receiver HDMI output
video in/out (HDMI)
HDMI to USB camera
Sender
Receiver
Network Simulator
AV.io
Video System
Measurement System

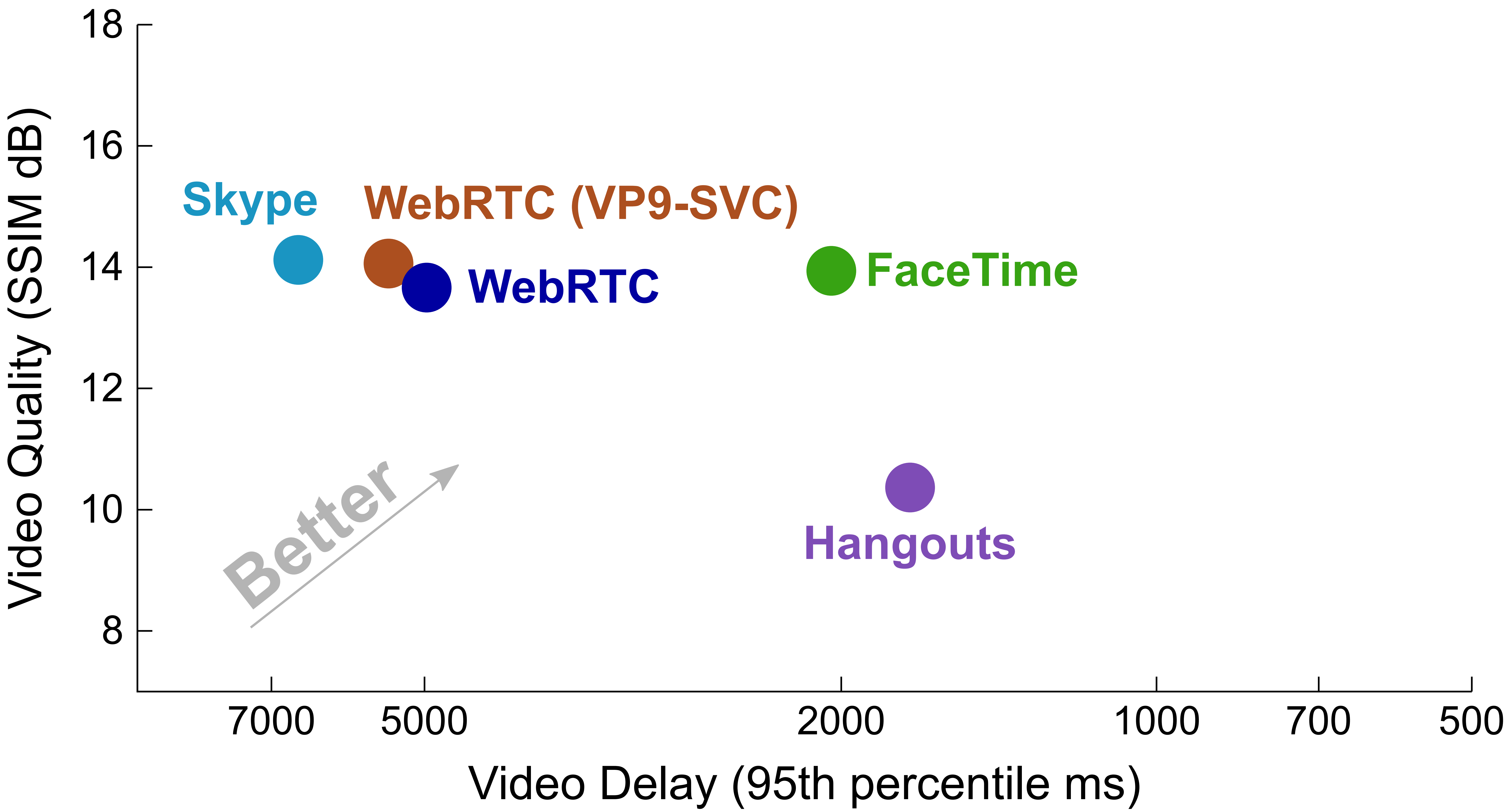


Sent Image
Timestamp: T+0.000s

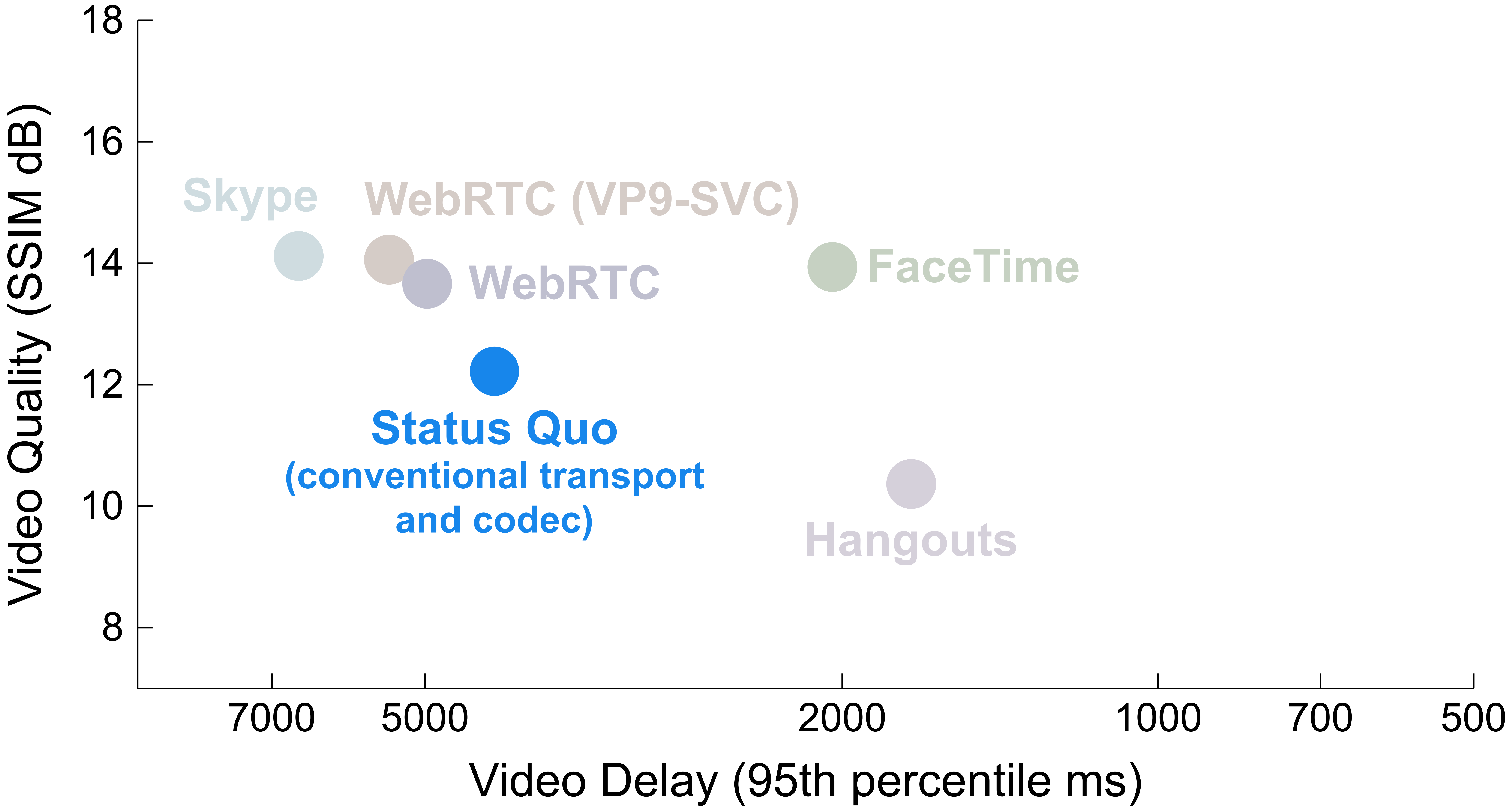


Received Image
Timestamp: T+0.765s
Quality: 9.76 dB SSIM

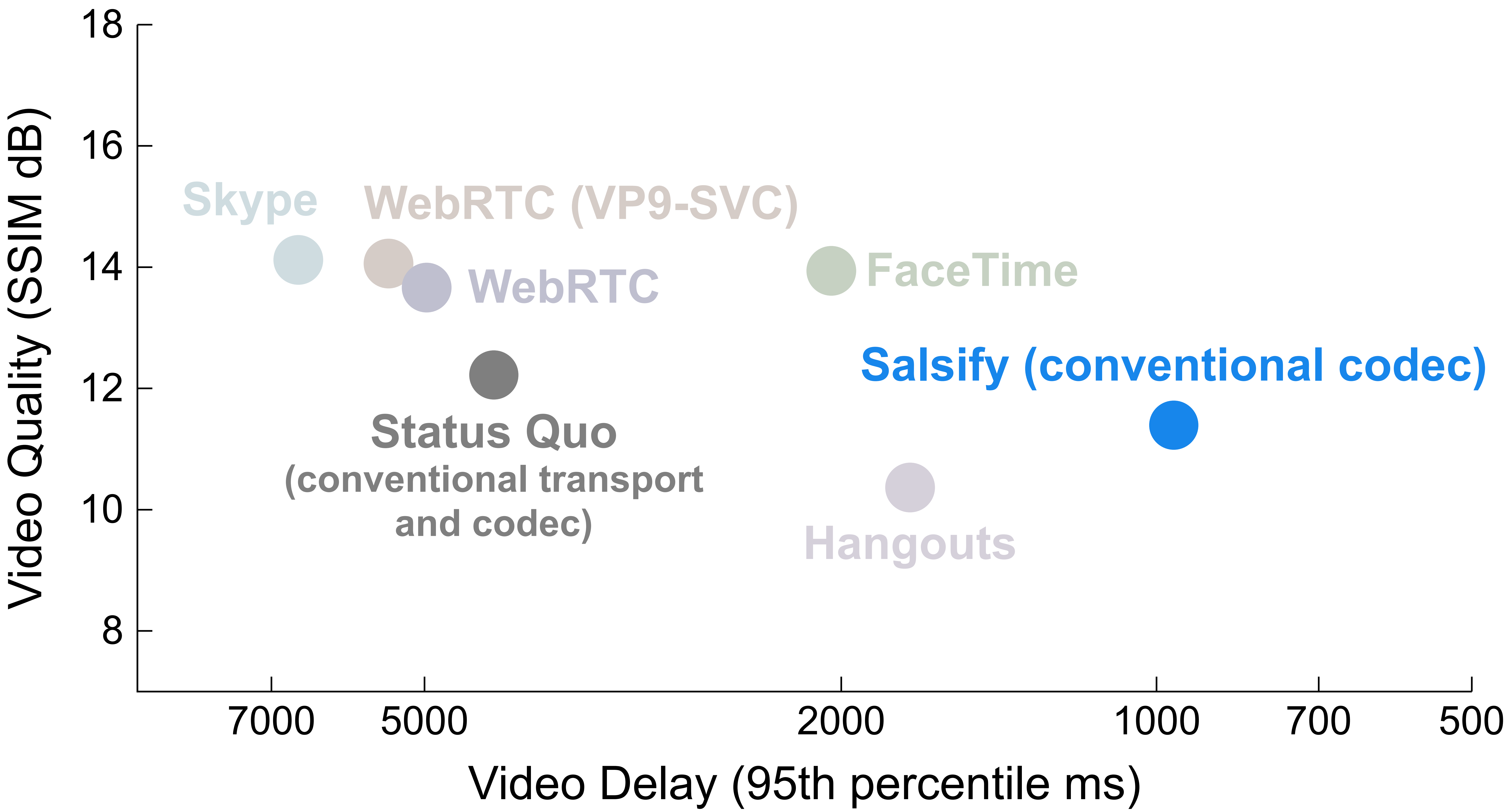
Evaluation results: Verizon LTE Trace



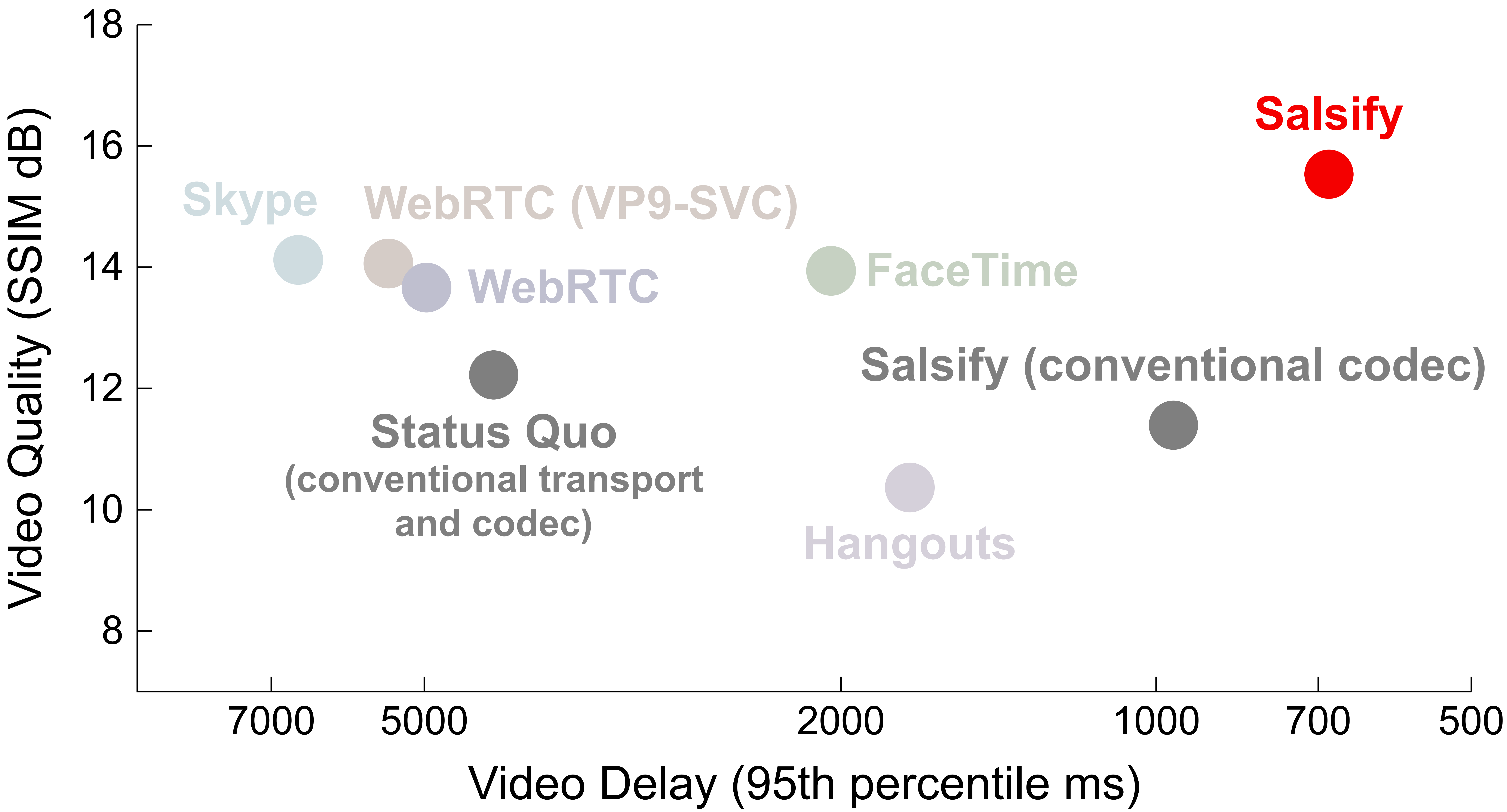
Evaluation results: Verizon LTE Trace



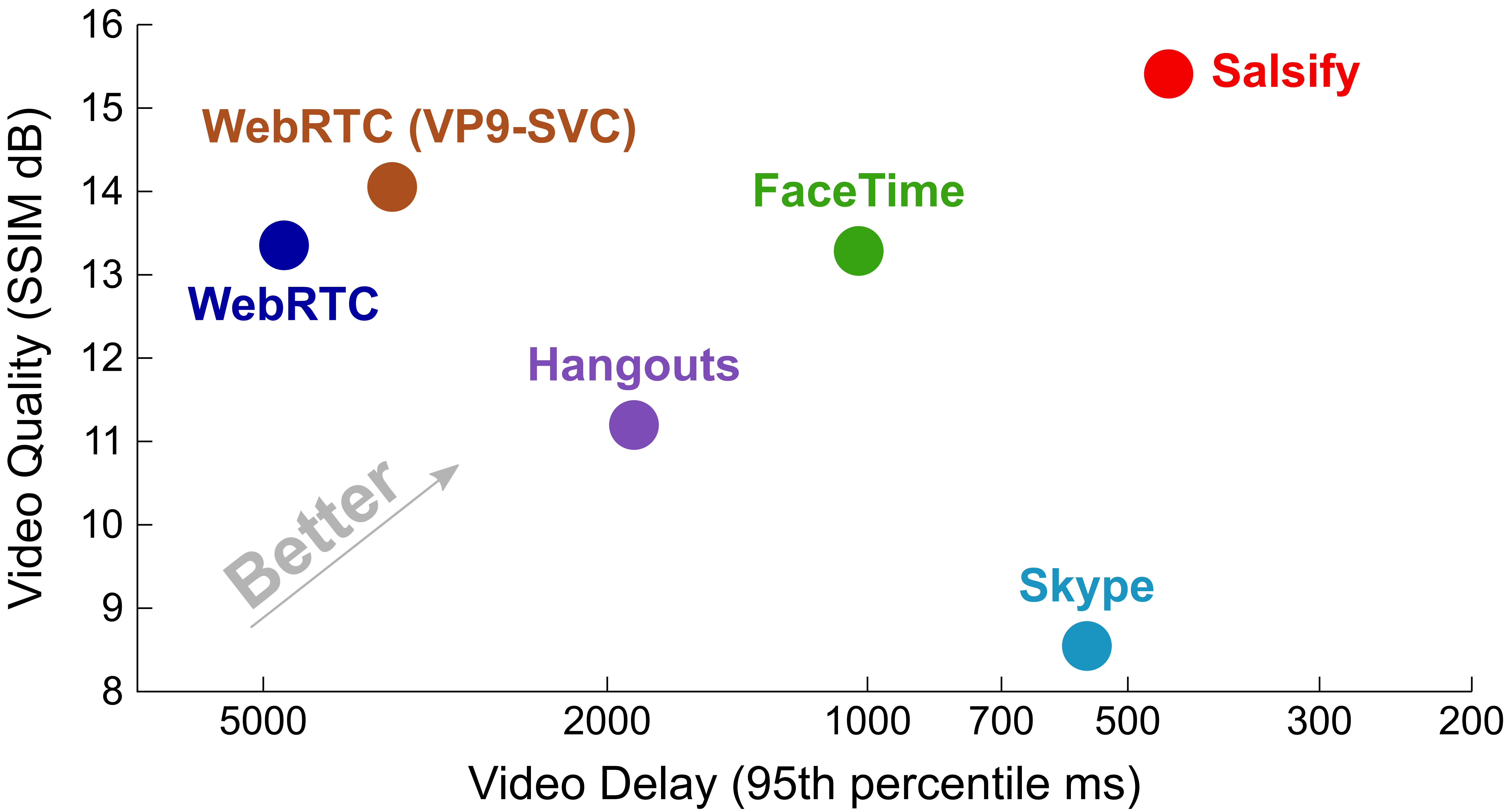
Evaluation results: Verizon LTE Trace



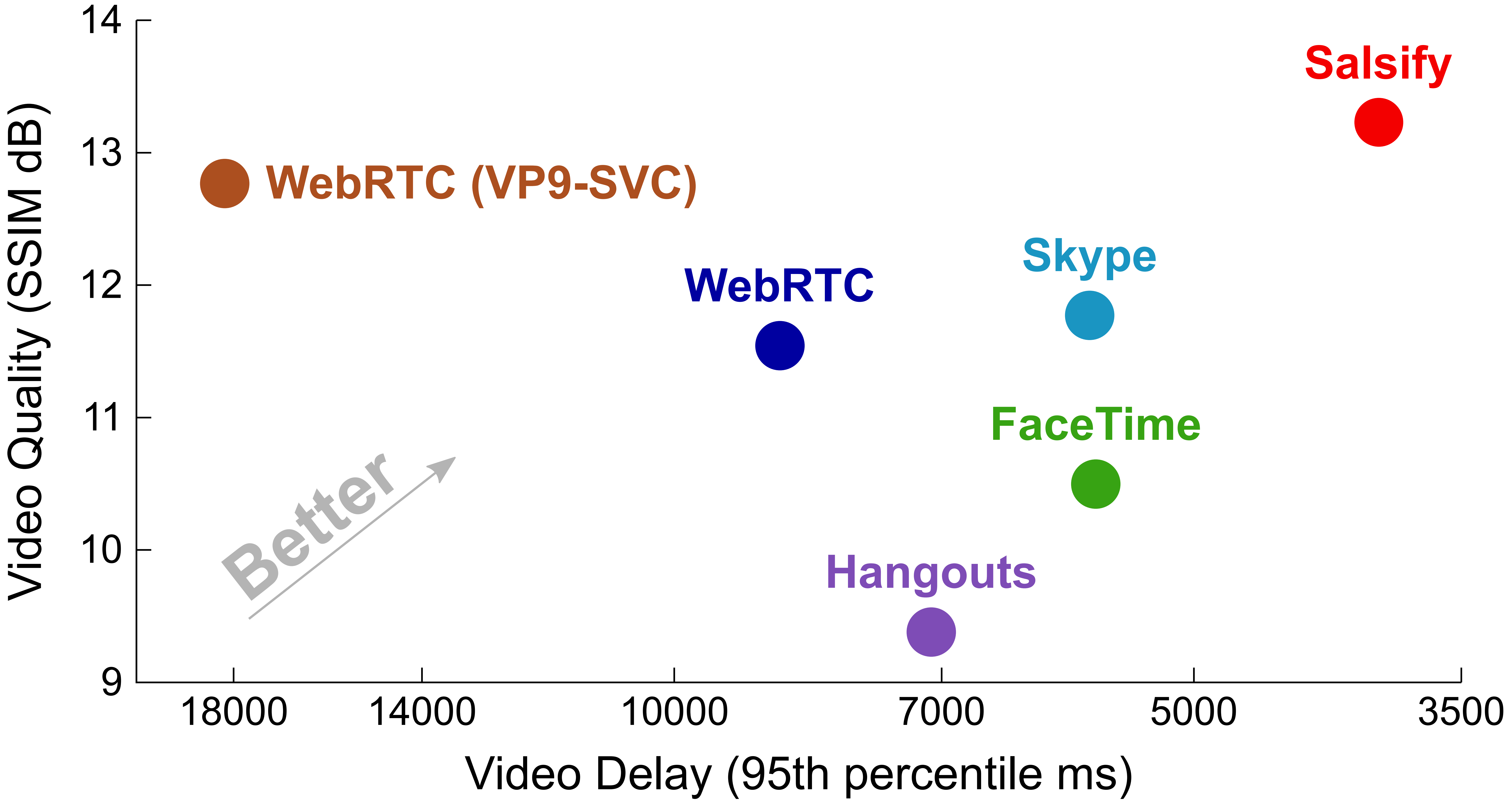
Evaluation results: Verizon LTE Trace



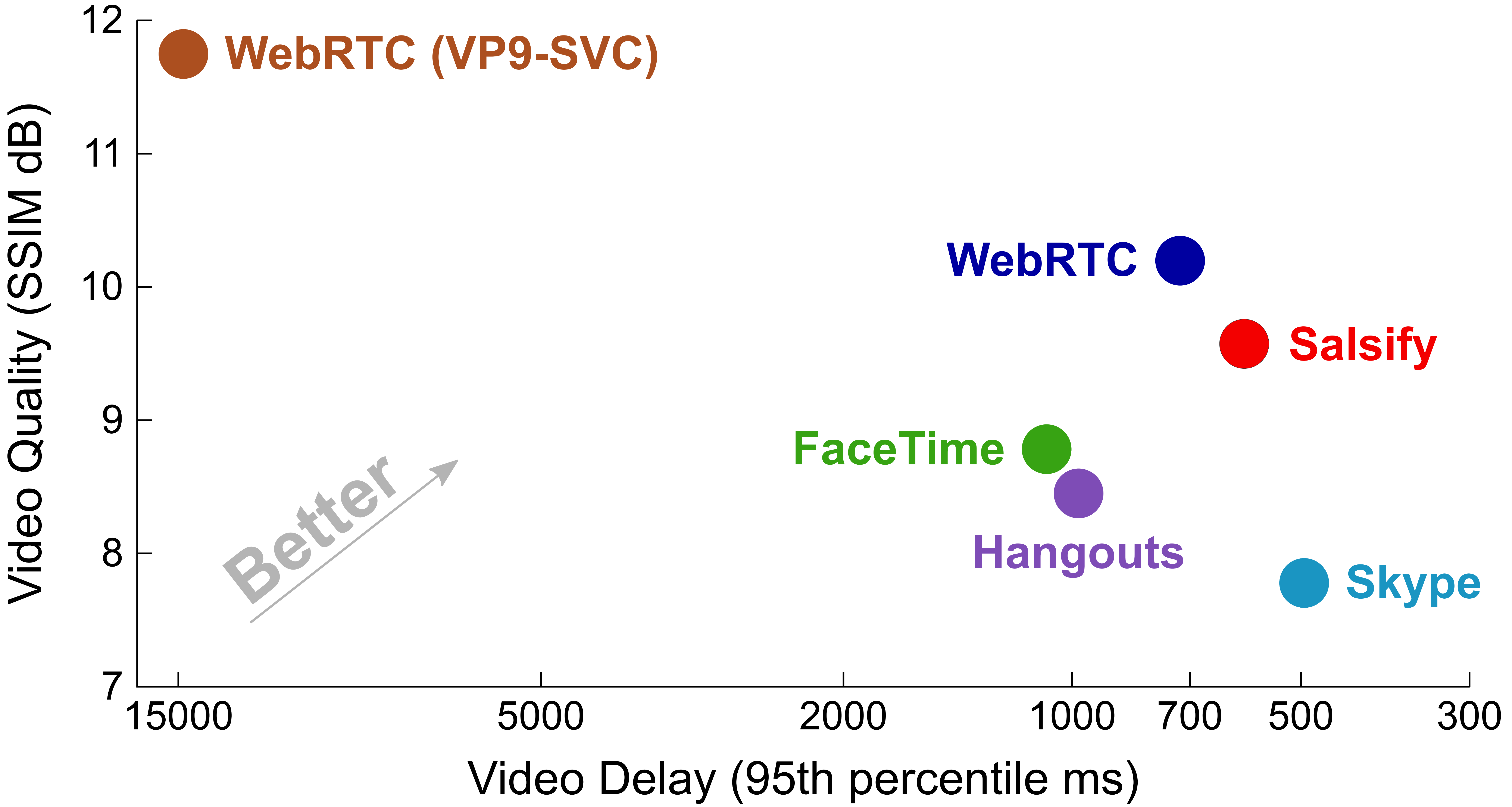
Evaluation results: AT&T LTE Trace



Evaluation results: T-Mobile UMTS Trace



Evaluation results: No variations



Individual component of Salsify are not exactly new...

- The transport protocol is a **dumbed-down** version of *packet pair* and *Sprout*.
- The video format, VP8, is **13 years old**.
- The functional codec was introduced in [Fouladi et al., NSDI '17].
  - Its compression efficiency & speed is **way lower** than commercial codecs.

It’s the architecture that’s new!

- The functional abstraction separates **the control** from **the algorithm**.
    - The system can now ***jointly*** control the codec and the transport in one control loop…
    - …and respond faster to network variability.

In this context, improvements to ***video codecs*** may have reached the point of diminishing returns, but changes to the architecture of ***video systems*** can still yield significant benefits.

Takeaways

- Salsify is a new architecture for real-time Internet video.
  - Salsify tightly integrates a **video-aware transport protocol**, with a **functional video codec**, allowing it to **respond quickly to changing network conditions**.
- Salsify achieves **4.6× lower p95-delay** and **2.1 dB SSIM higher visual quality** on average when compared with FaceTime, Hangouts, Skype, and WebRTC.
- The code is open-source, and the paper and raw data are open-access:
  `https://snr.stanford.edu/salsify`

Thank you: NSF, DARPA, Google, Dropbox, VMware, Huawei, Facebook, Stanford Platform Lab, and James.