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

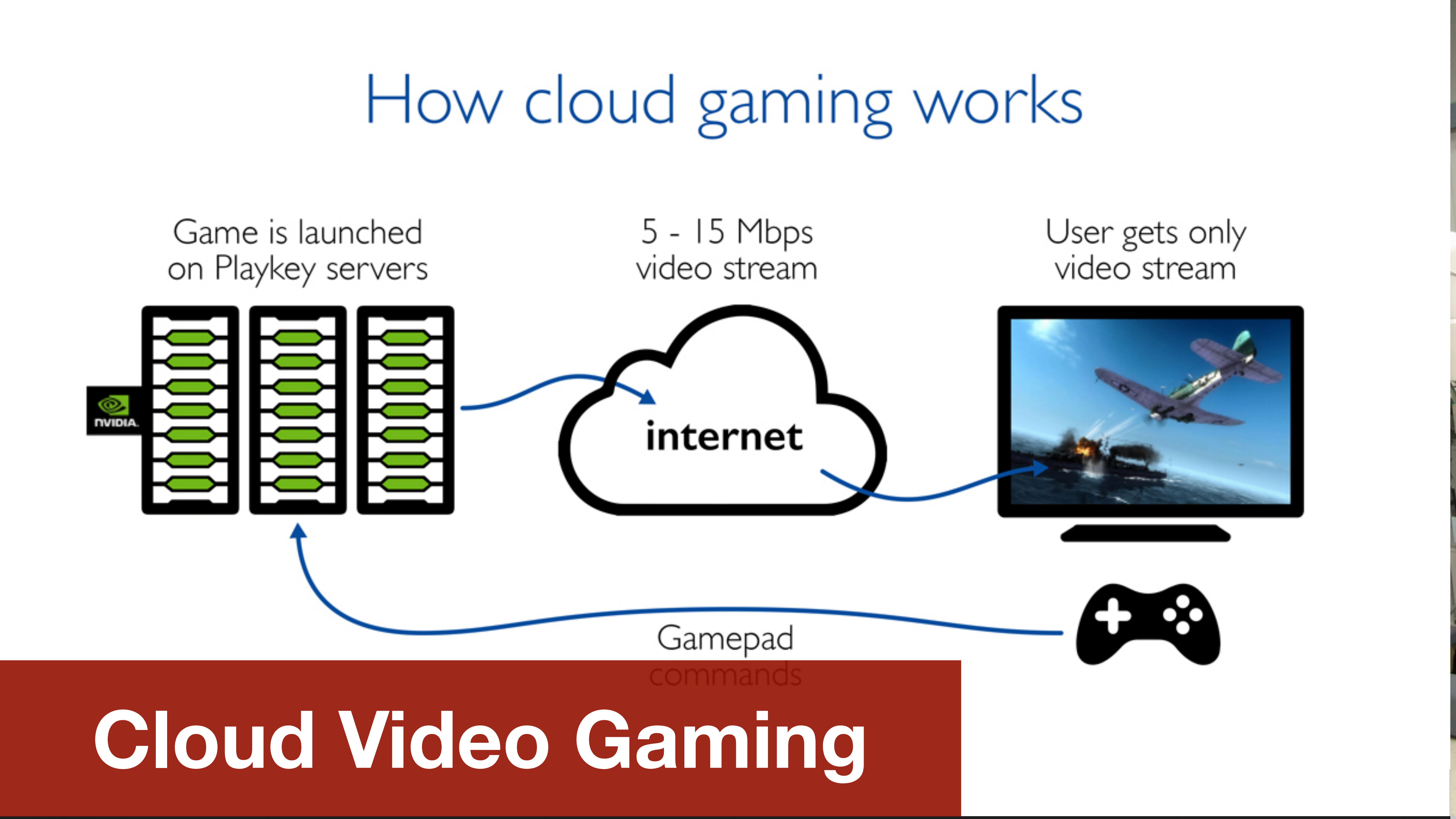
Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu[+], Riad S. Wahby, Keith Winstein
Stanford University, [+]Saratoga High School

https://snr.stanford.edu/salsify

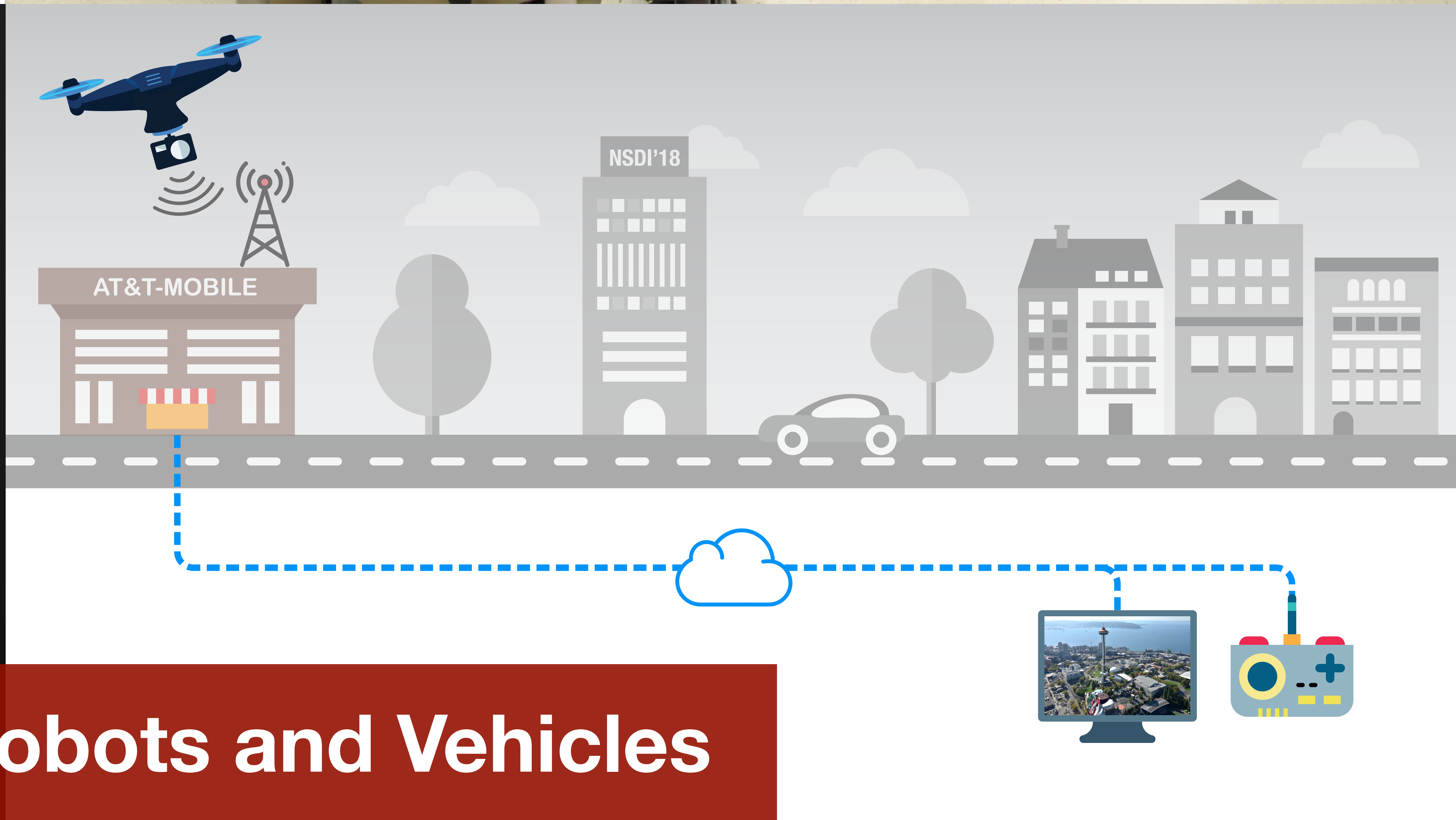
Stanford

Outline

- **Introduction**
- Salsify's New Architecture
- Measurement Testbed
- Evaluation
- Conclusions

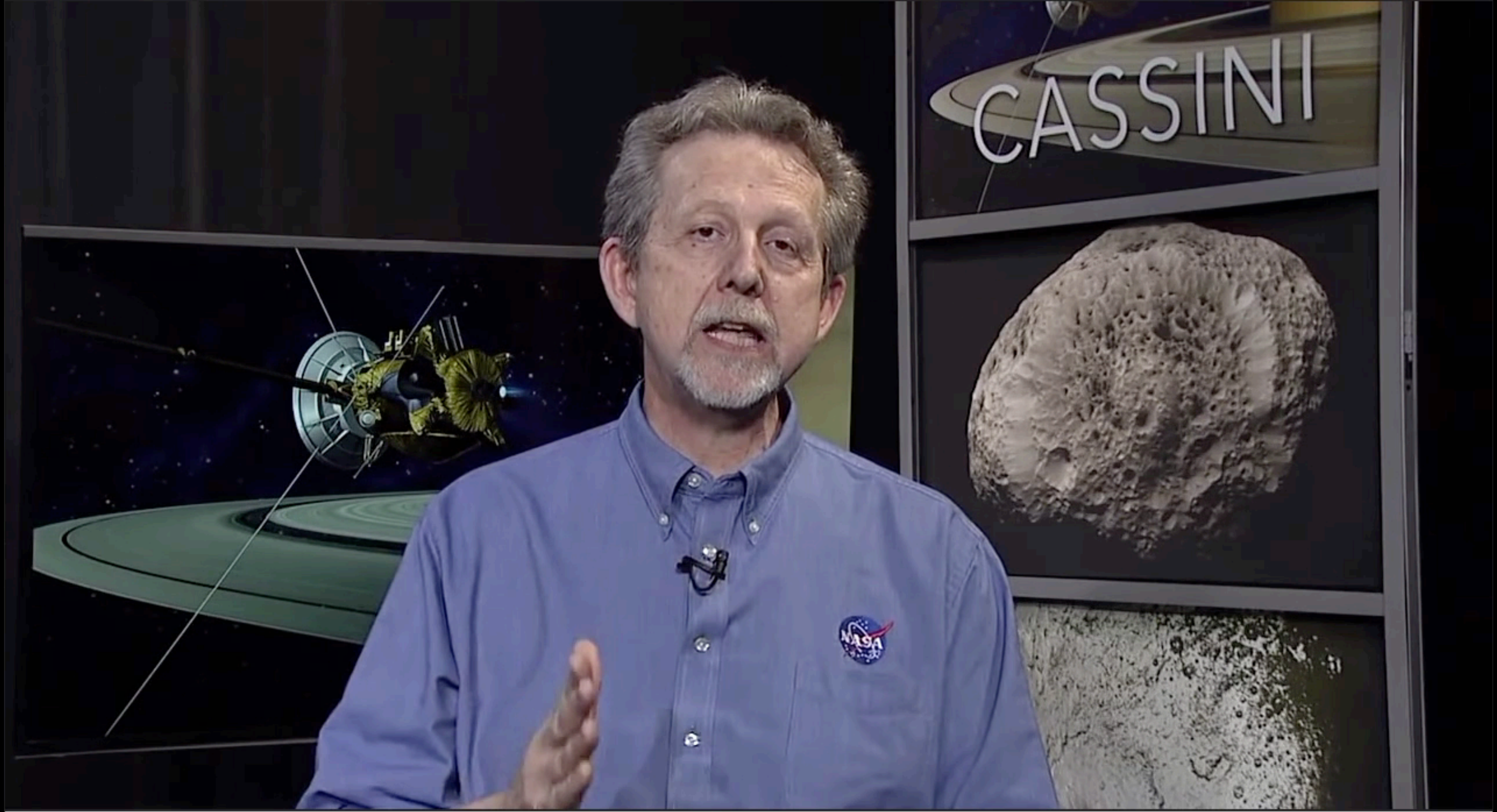

How cloud gaming works
Game is launched on Playkey servers
5 - 15 Mbps video stream
User gets only video stream
NVIDIA
internet
Gamepad commands
Cloud Video Gaming

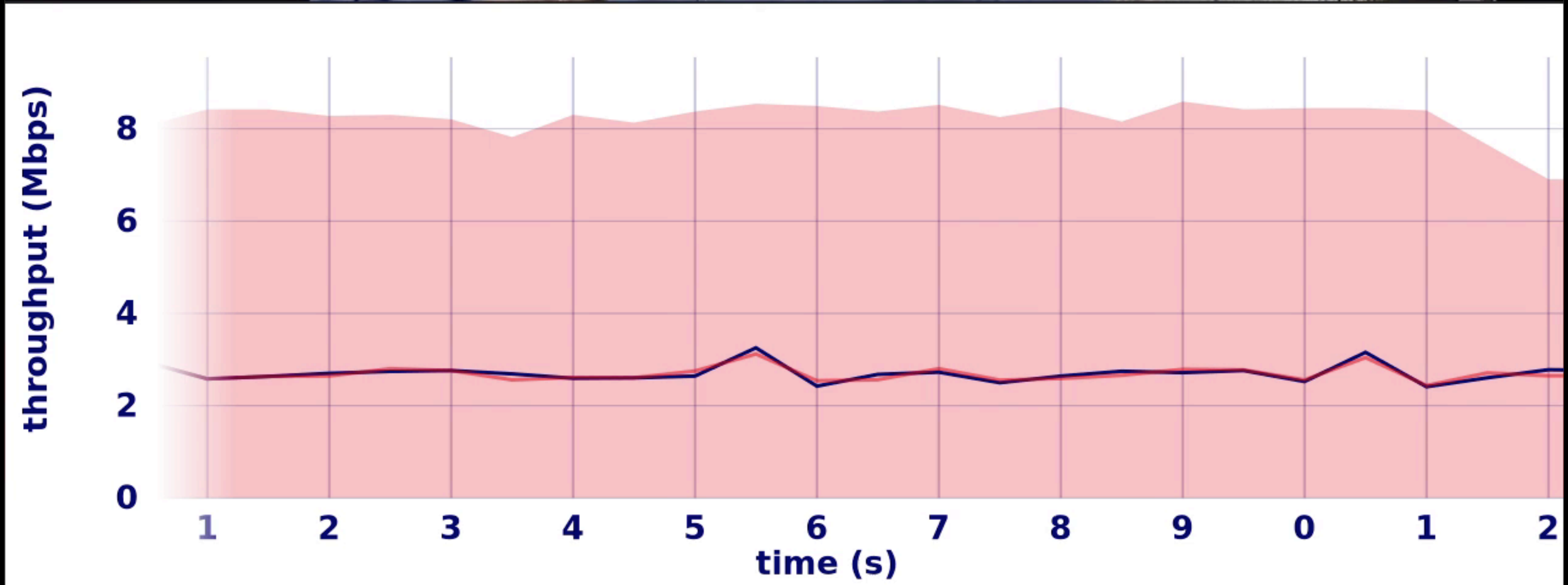

Remote Surgery
OAK RIDGE
GENERAL DYNA


NSDI'18
AT&T-MOBILE
Teleoperation of Robots and Vehicles

Video Conferencing

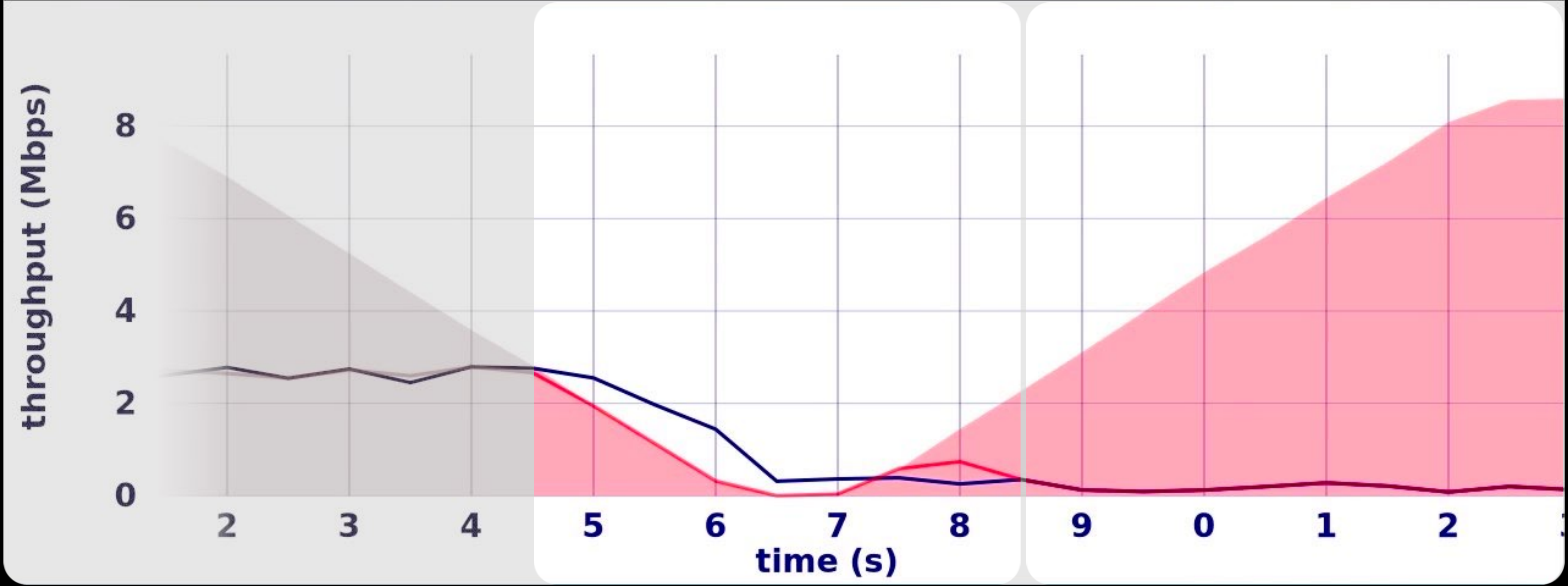

Video Conferencing *(reality)*





WebRTC
(Chrome 65)

Current systems do not react **fast enough** to **network variations**, end up congesting the network, causing **stalls and glitches**.



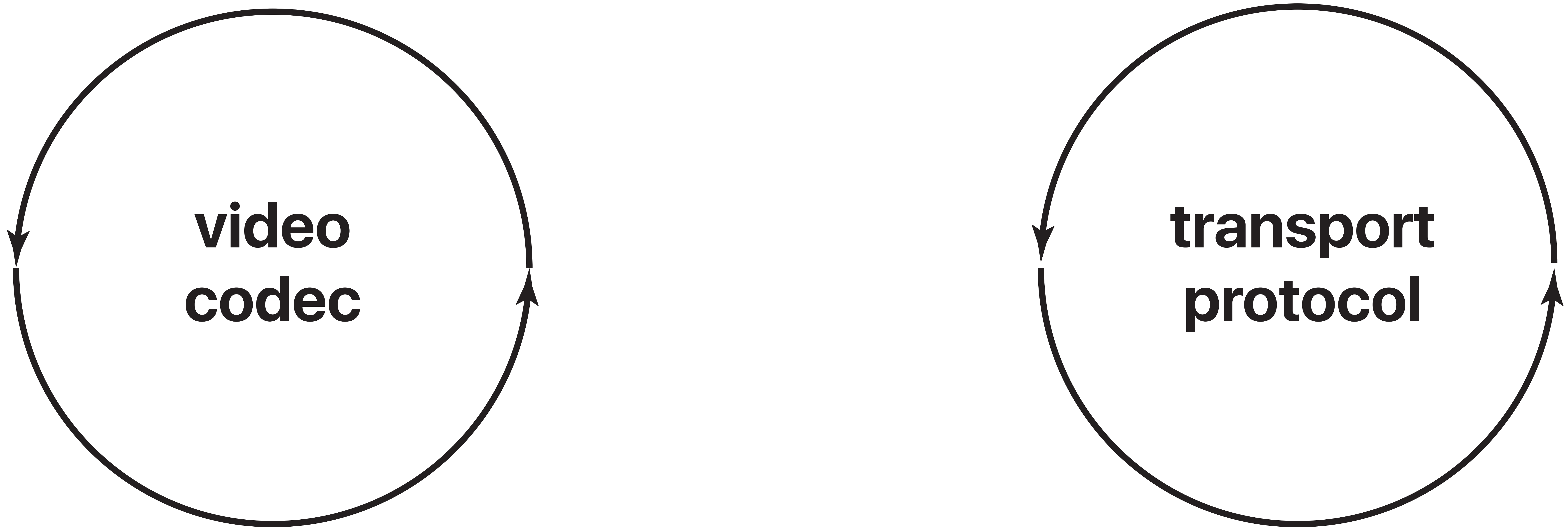
Enter *Salsify*

- 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**.
- 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.

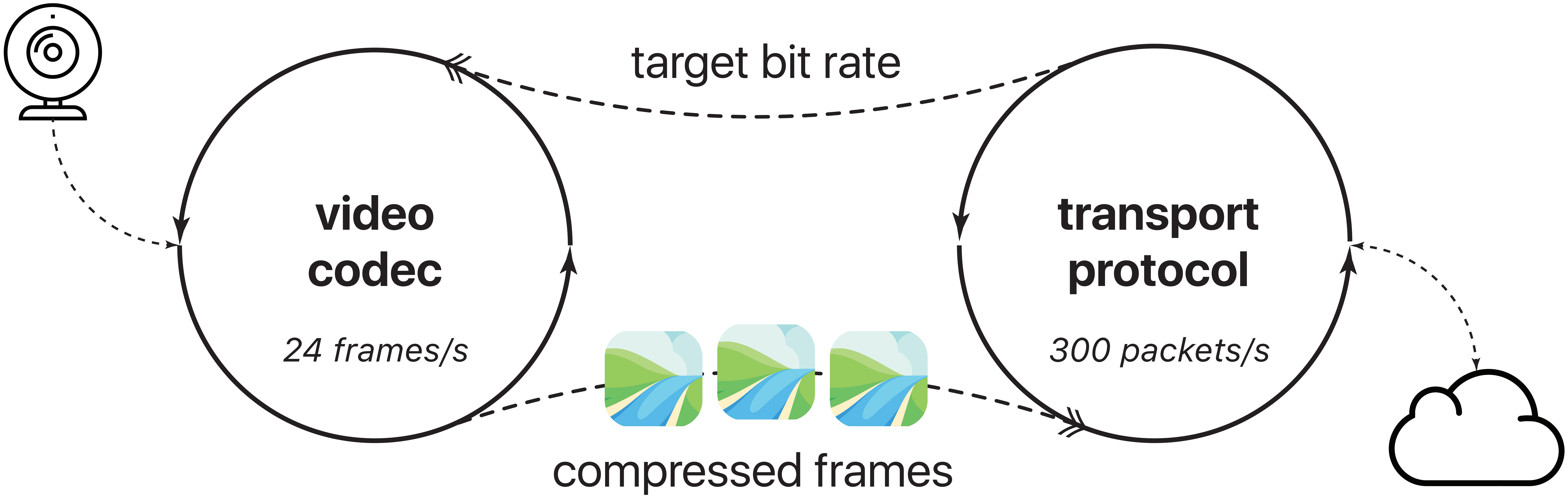
Outline

- Introduction
- **Salsify’s New Architecture**
- Measurement Testbed
- Evaluation
- Conclusions

Today's systems combine two *(loosely-coupled)* components



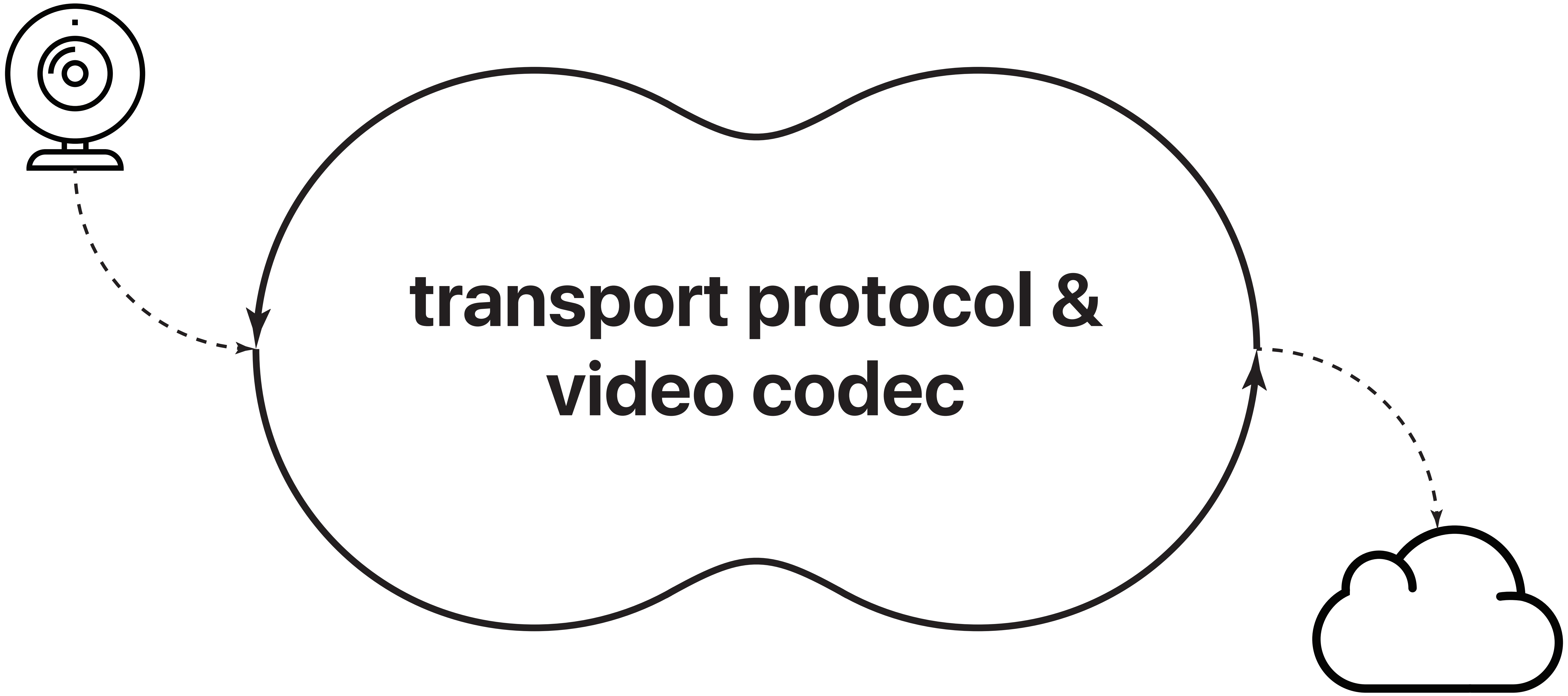
Two distinct modules, two separate control loops



Shortcomings of the conventional design

- The codec can only achieve the bit rate **on average**.
  - Individual frames can still congest the network.
- The resulting system is slow to react to network variations.

Salsify explores a more tightly-integrated design

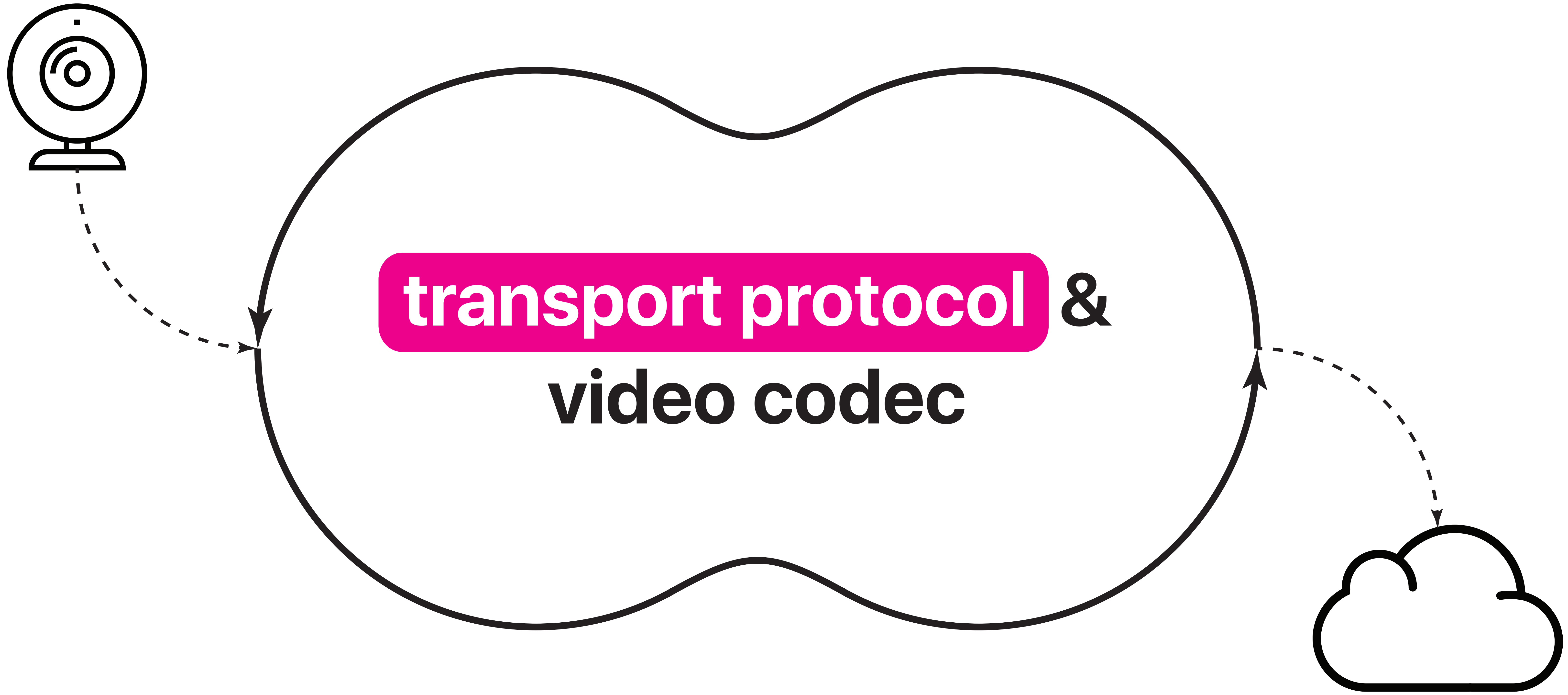


Brand-new architecture based on components we know and love!

- Individual component of Salsify are not exactly new:
  - The transport protocol is inspired by “packet pair” and “Sprout-EWMA”.
  - The video format, VP8, was finalized in 2008.
  - The functional video codec was described at NSDI’17.
- Salsify is a **new architecture** for real-time video that integrates these components in a way that responds quickly to network variations.

Salsify's architecture:

Video-aware transport protocol



Video-aware transport protocol

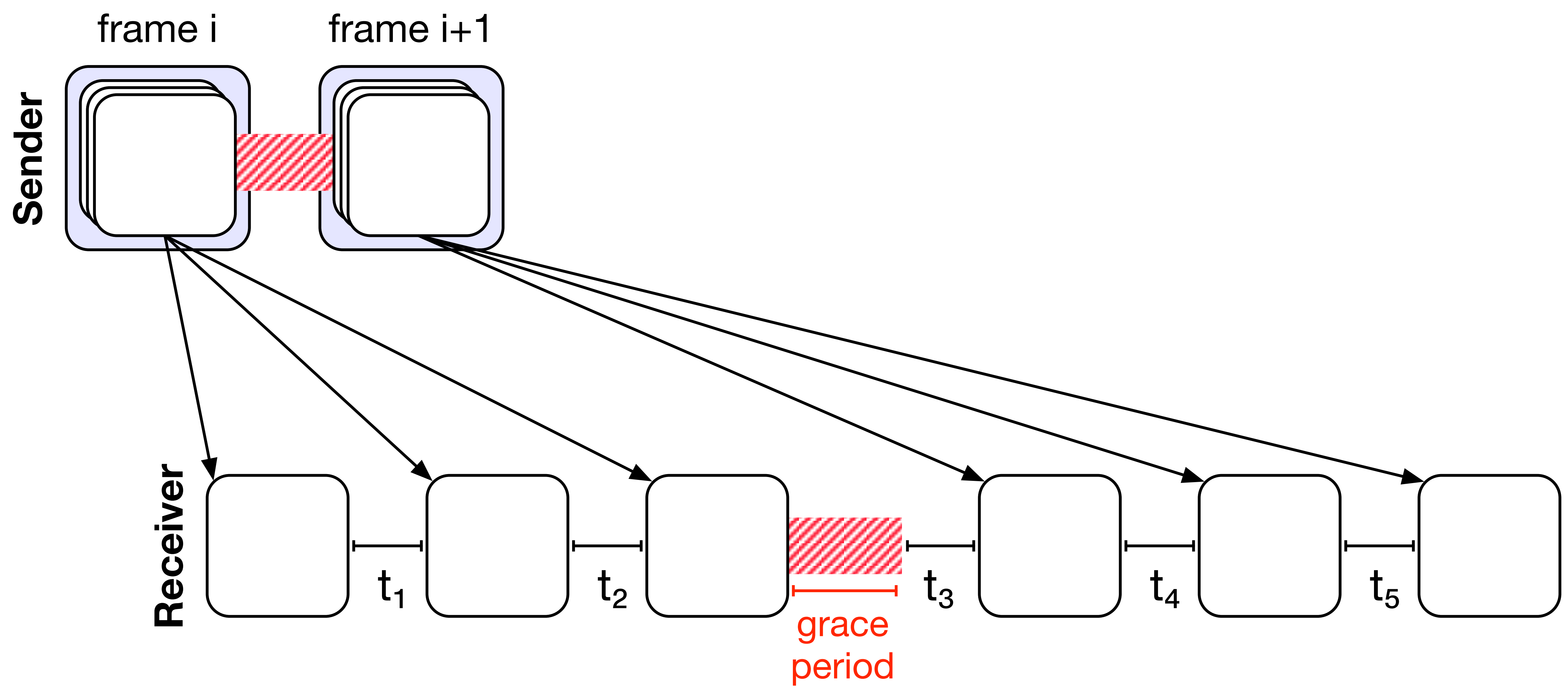
"What should be the size of the next frame?"

∗ without causing excessive delay

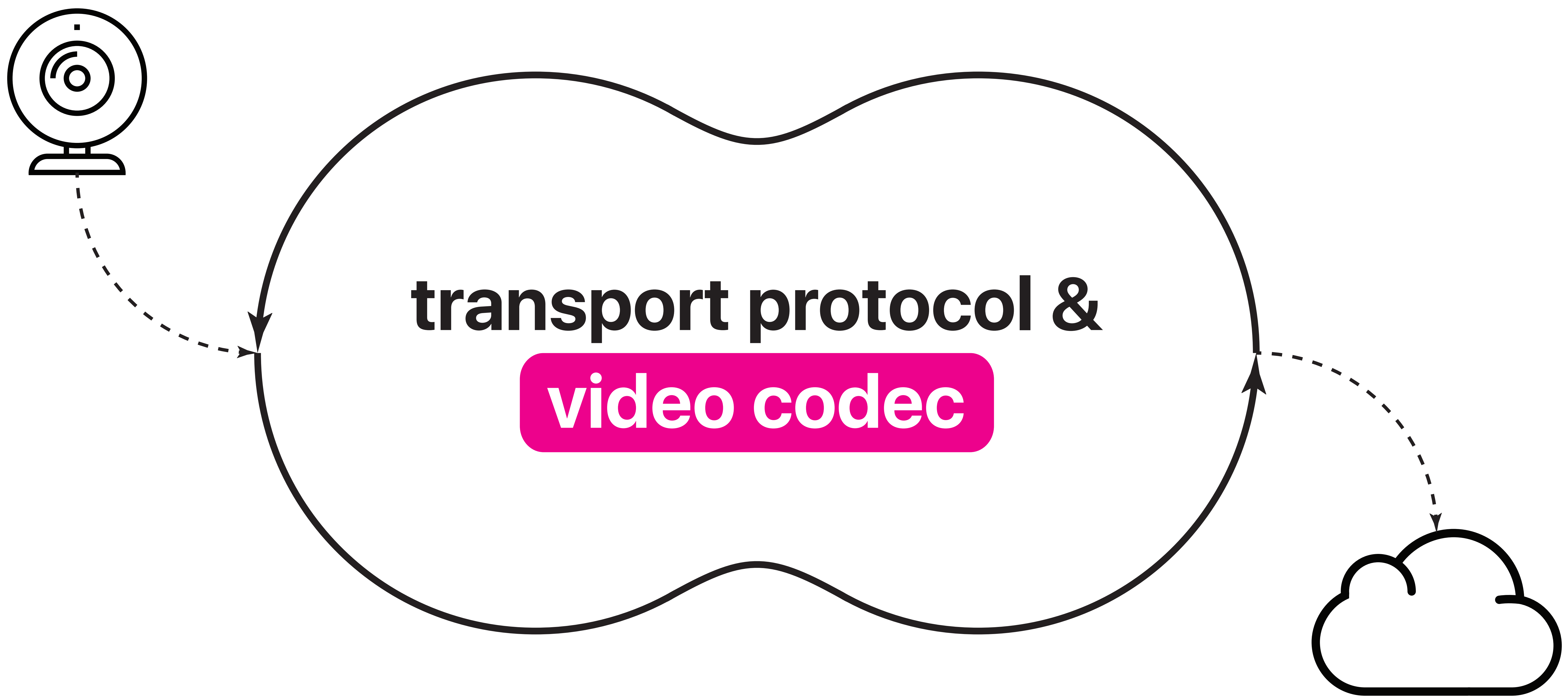
- There's no notion of bit rate, only the next frame size!
- Transport uses **packet inter-arrival time**, reported by the receiver.

The sender does not transmit continuously

- Pauses between frames give the receiver a “pessimistic” view of the network.
- Receiver treats each frame of the video as a separate packet train.



Salsify's architecture:
Functional video codec



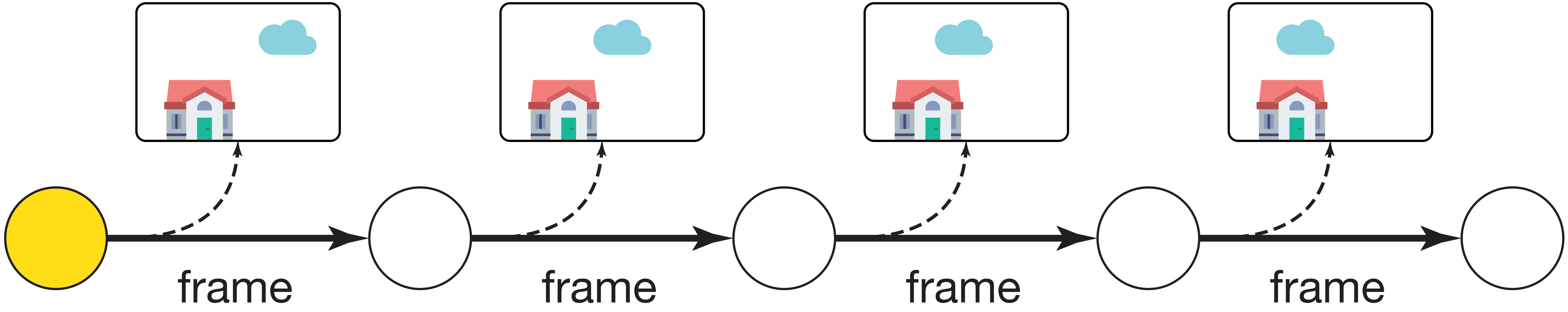
Transport tells us how big the next frame should be, but...

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.

How to get an accurate frame out of an inaccurate codec

- **Trial and error:** Encode with different quality settings, pick the one that fits.
  - *Not possible with existing codecs.*

After encoding a frame, the encoder goes through a state transition that is impossible to undo



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

```
encode(🏞,🏞,...) → frames...
```

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

Functional video codec to the rescue

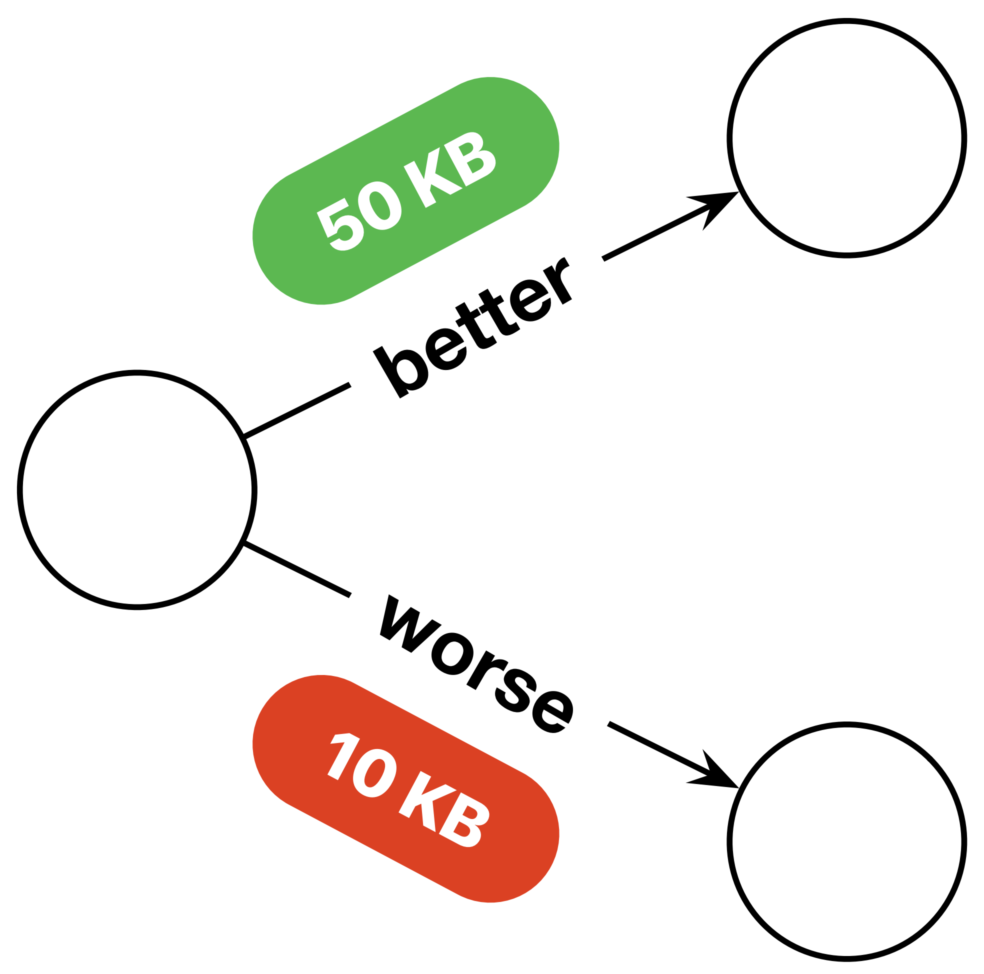
`encode`(*state*, 🏞) → *state′*, `frame`

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.

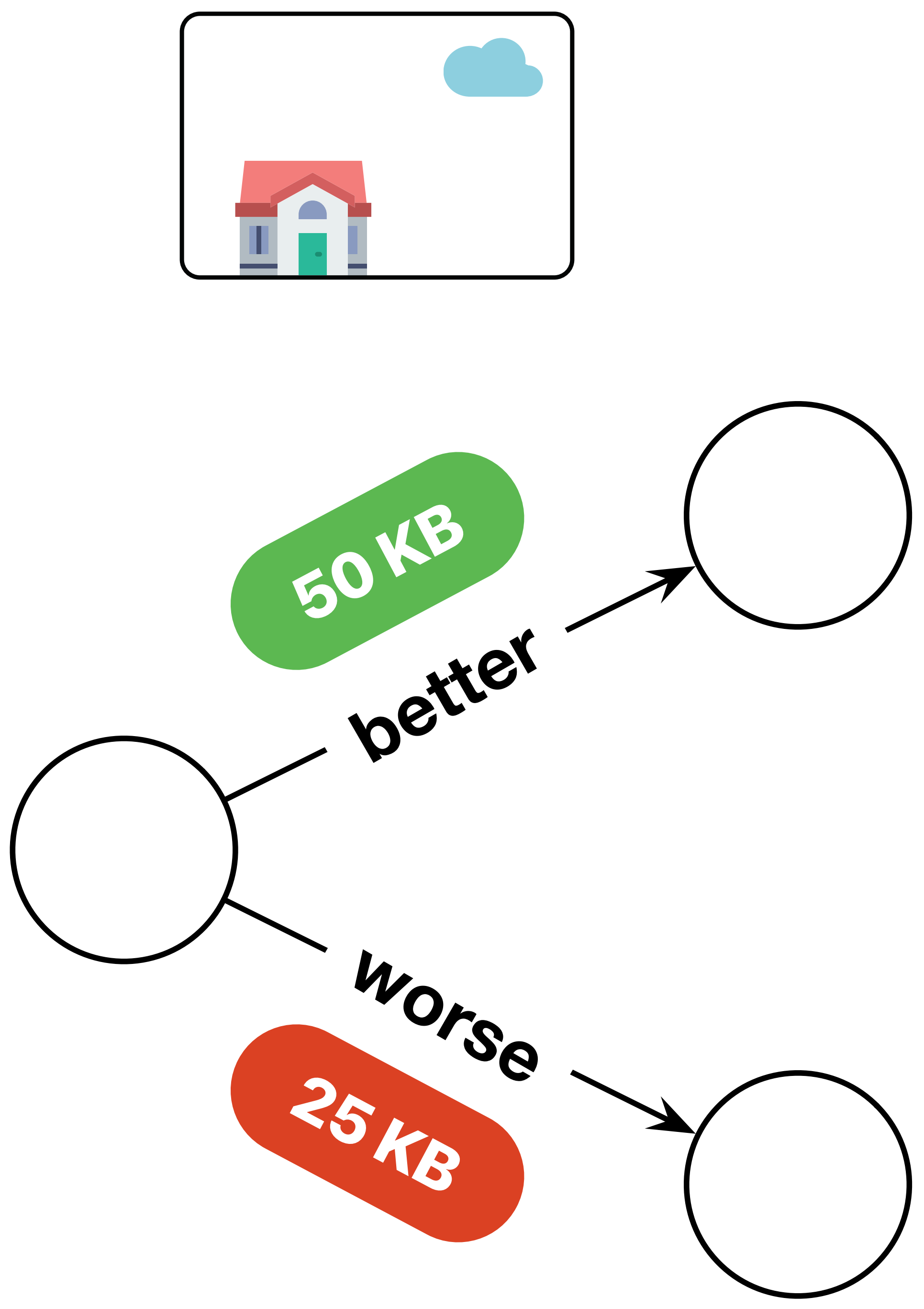
- For each frame, codec presents the transport with *three* options:
  - ▲ A slightly-higher-quality version,
  - ▼ A slightly-lower-quality version,
  - ✕ Discarding the frame.



Salsify's architecture:
Unified control loop

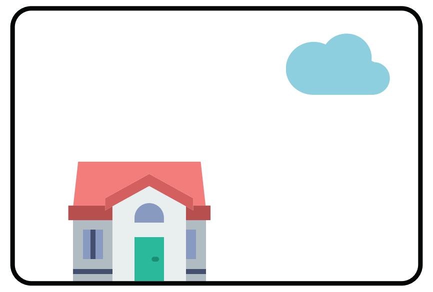
Codec → Transport

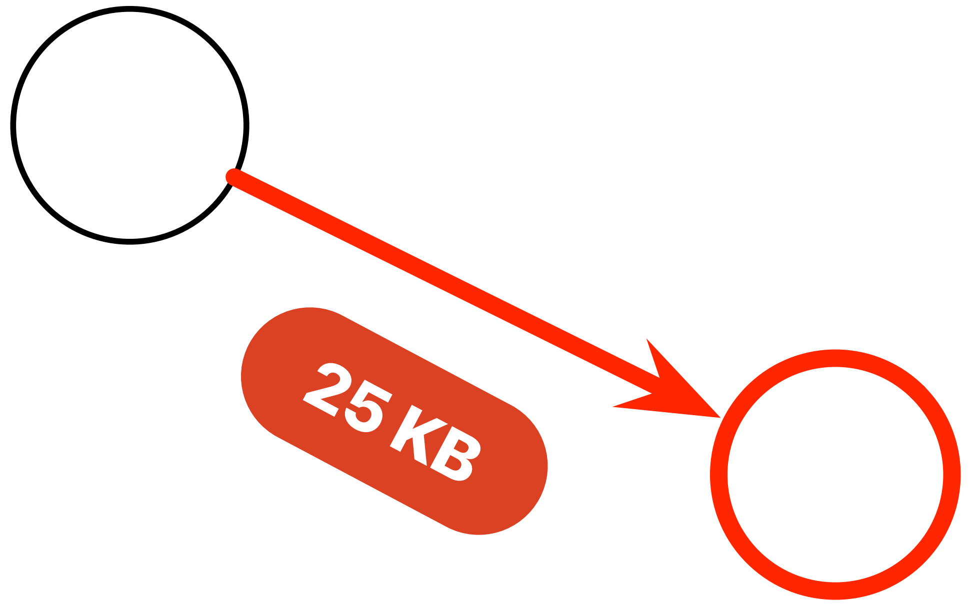
“Here’s two versions of the current frame.”



target frame size 30 KB

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

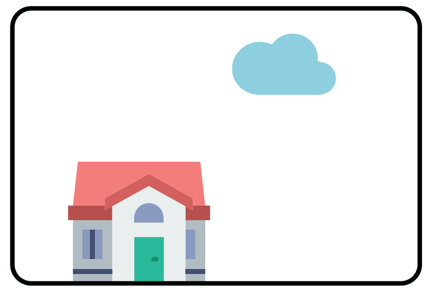


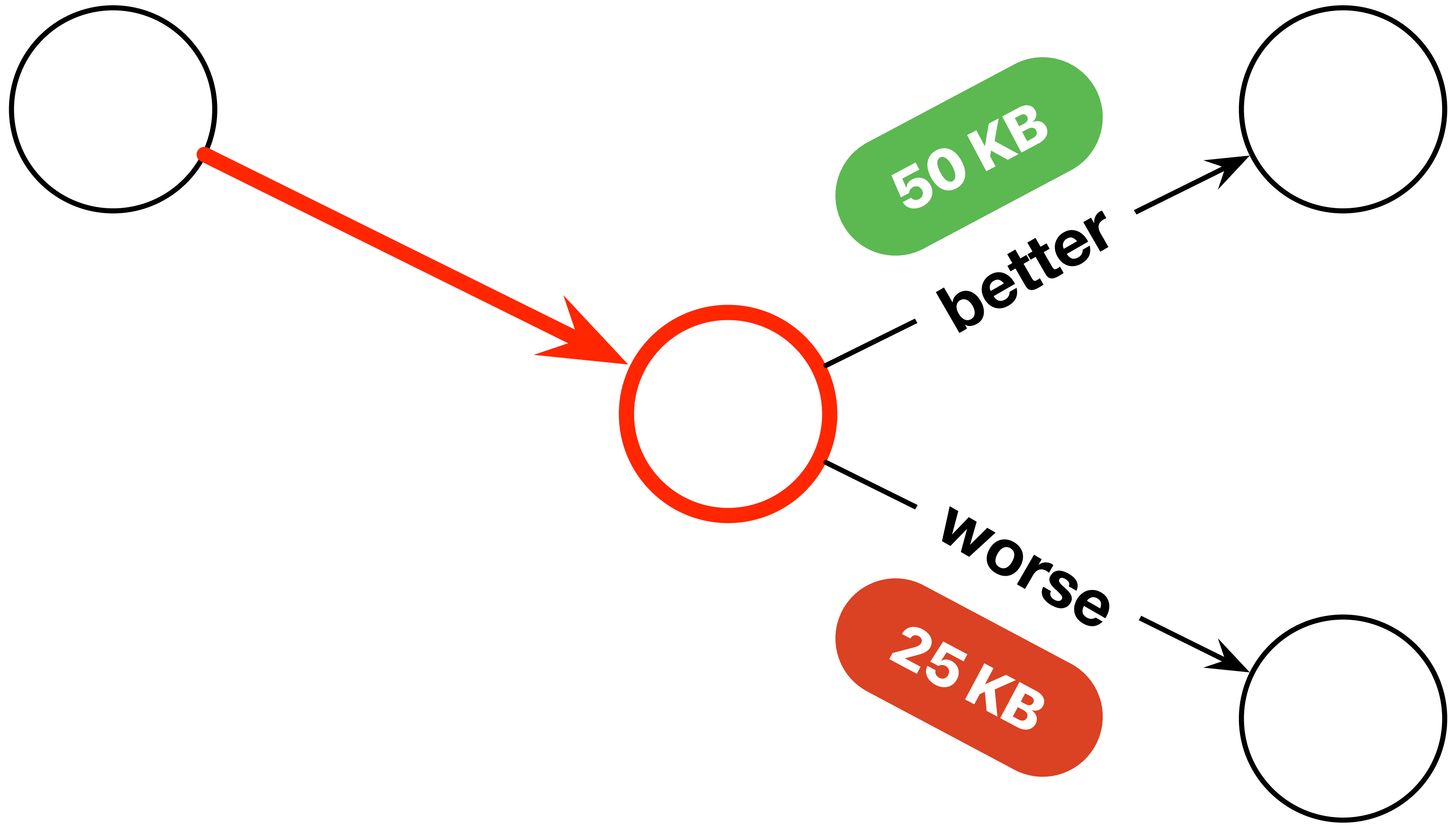


target frame size 30 KB

Codec → Transport

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

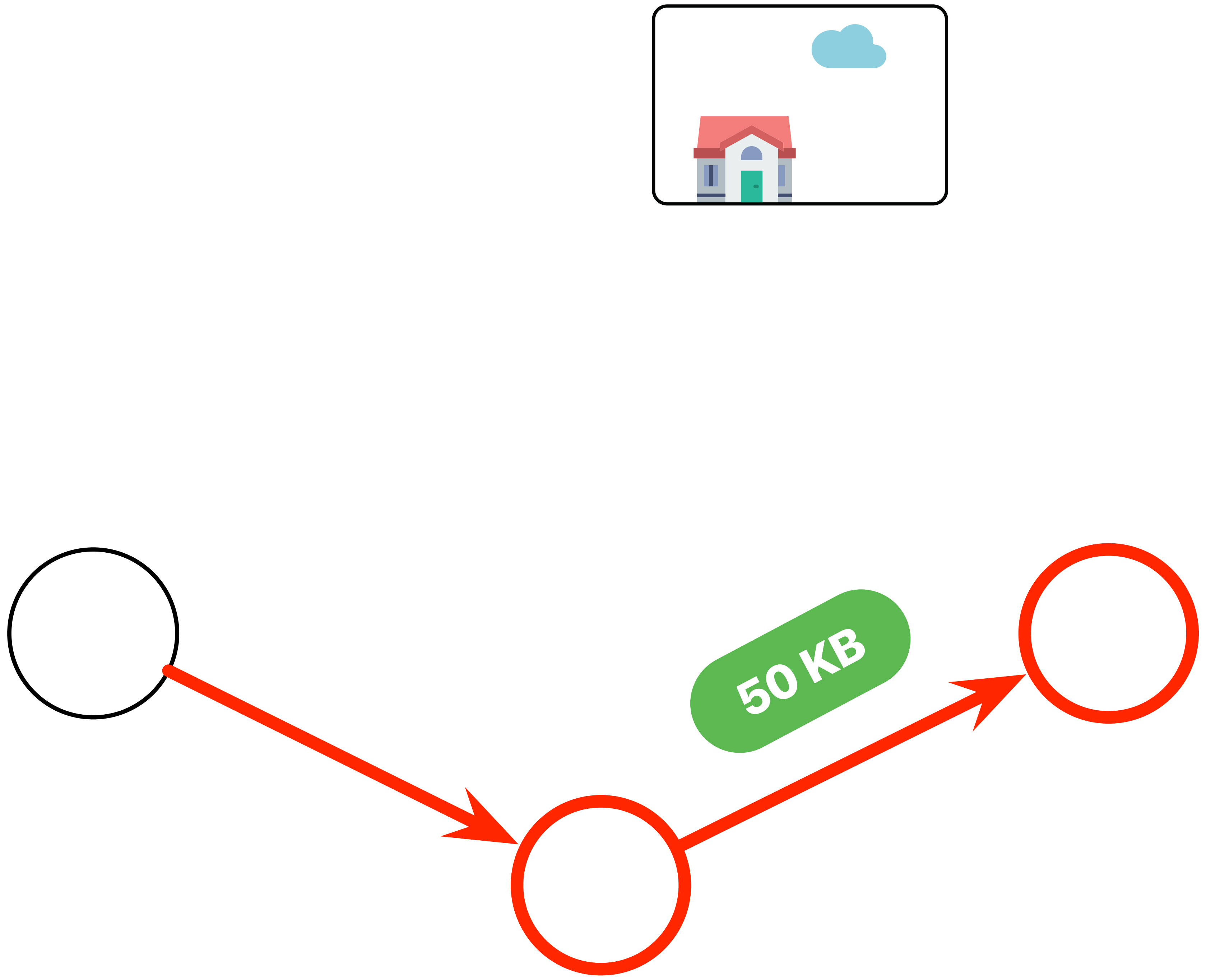




target frame size 55 KB

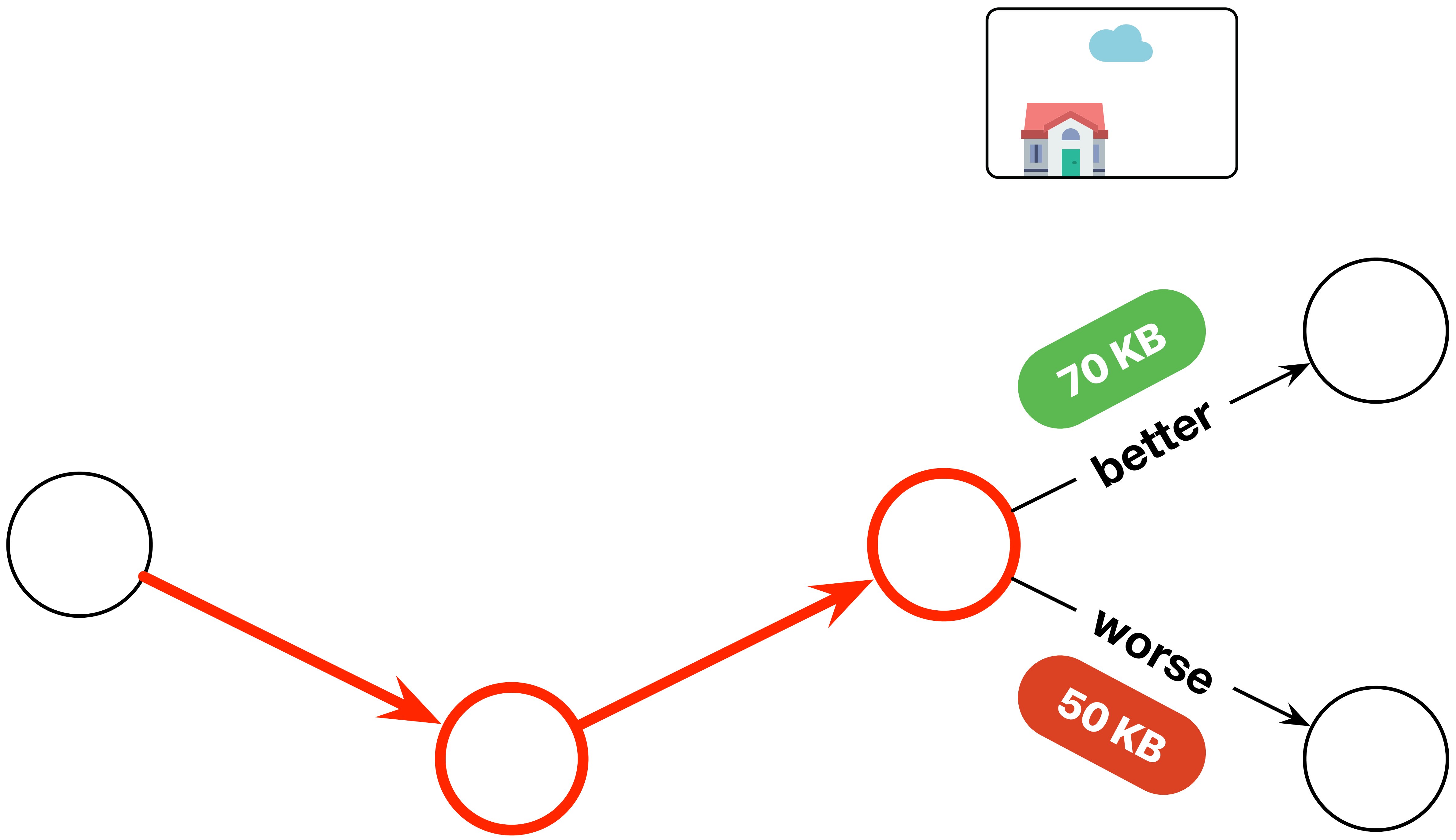
Transport → Codec

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



target frame size 55 KB

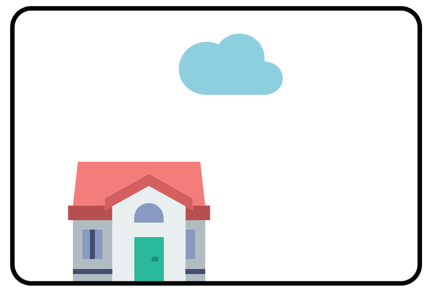
Codec → Transport
"Here's two versions of the latest frame."

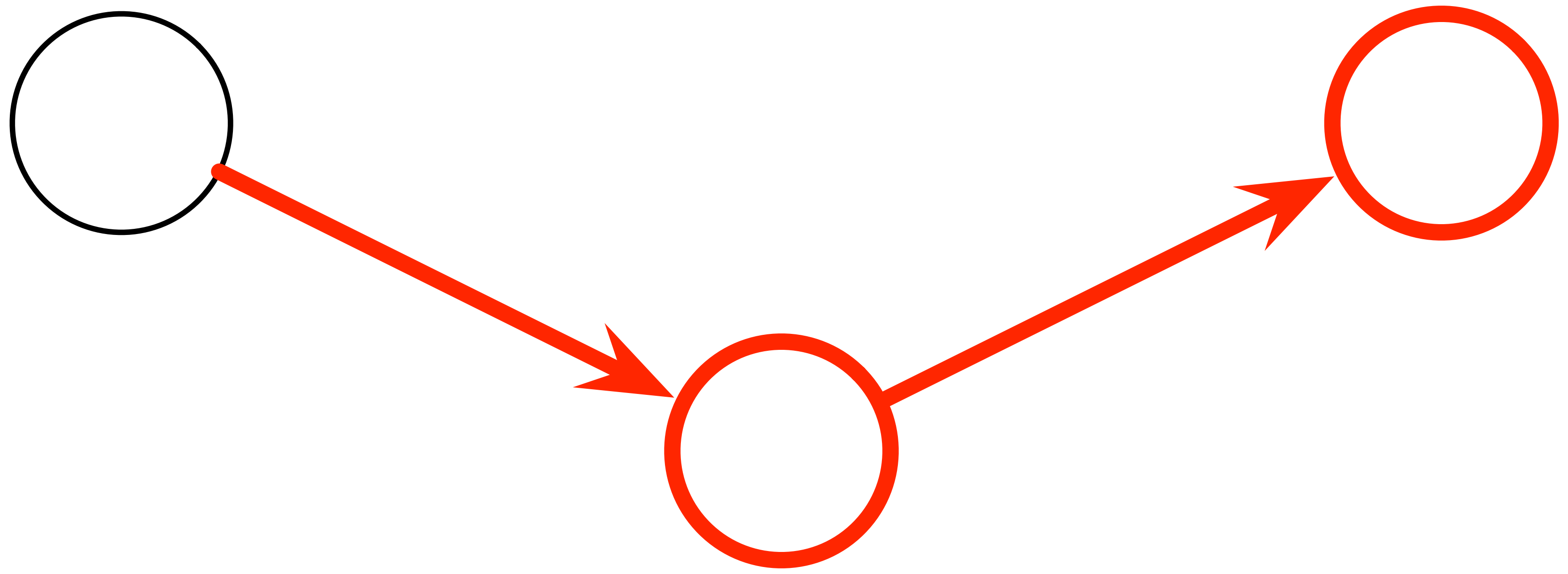


target frame size 5 KB

Transport → Codec

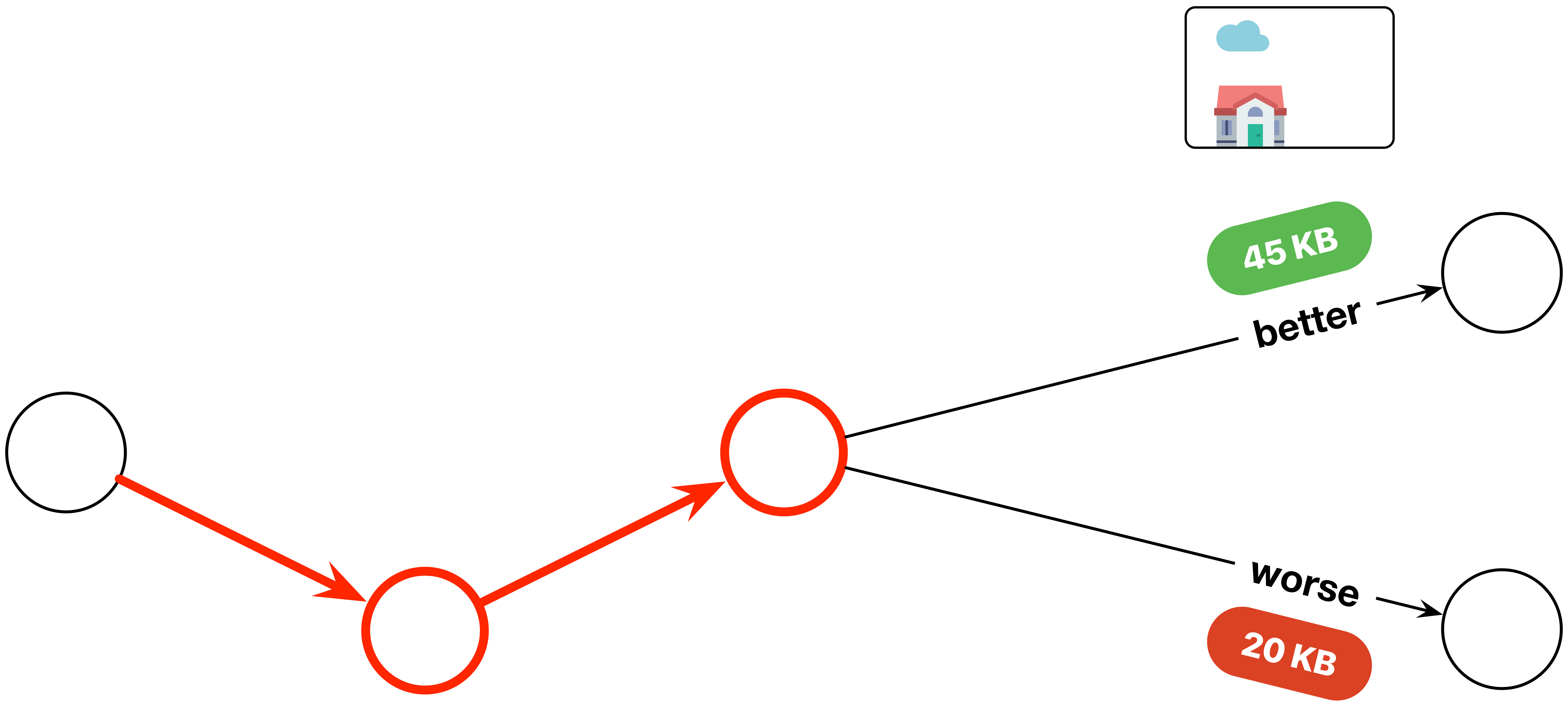
“I cannot send any frames right now. Sorry, but discard them.”





target frame size **5 KB**

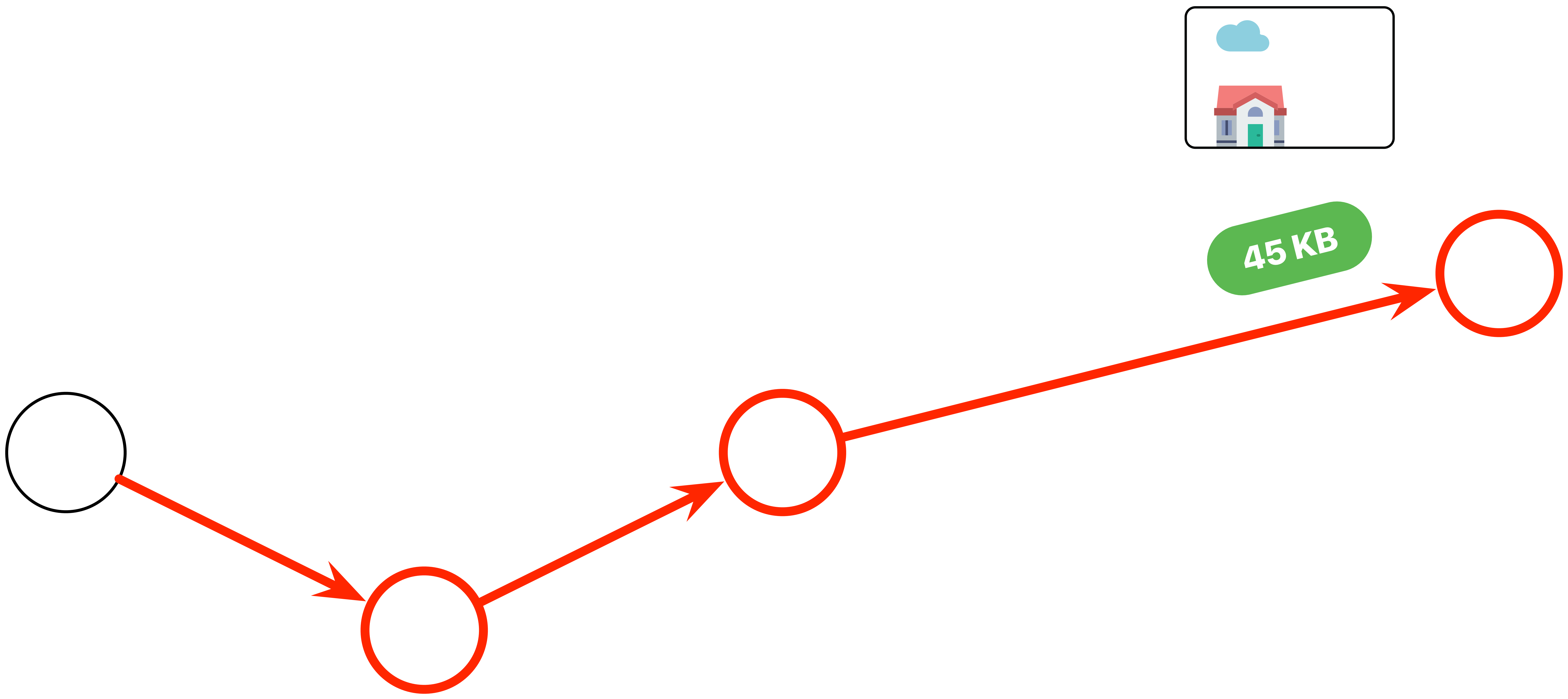
Codec → Transport
“Fine. Here’s two versions of the latest frame.”



target frame size 50 KB

Transport → Codec

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



target frame size 50 KB

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

Outline

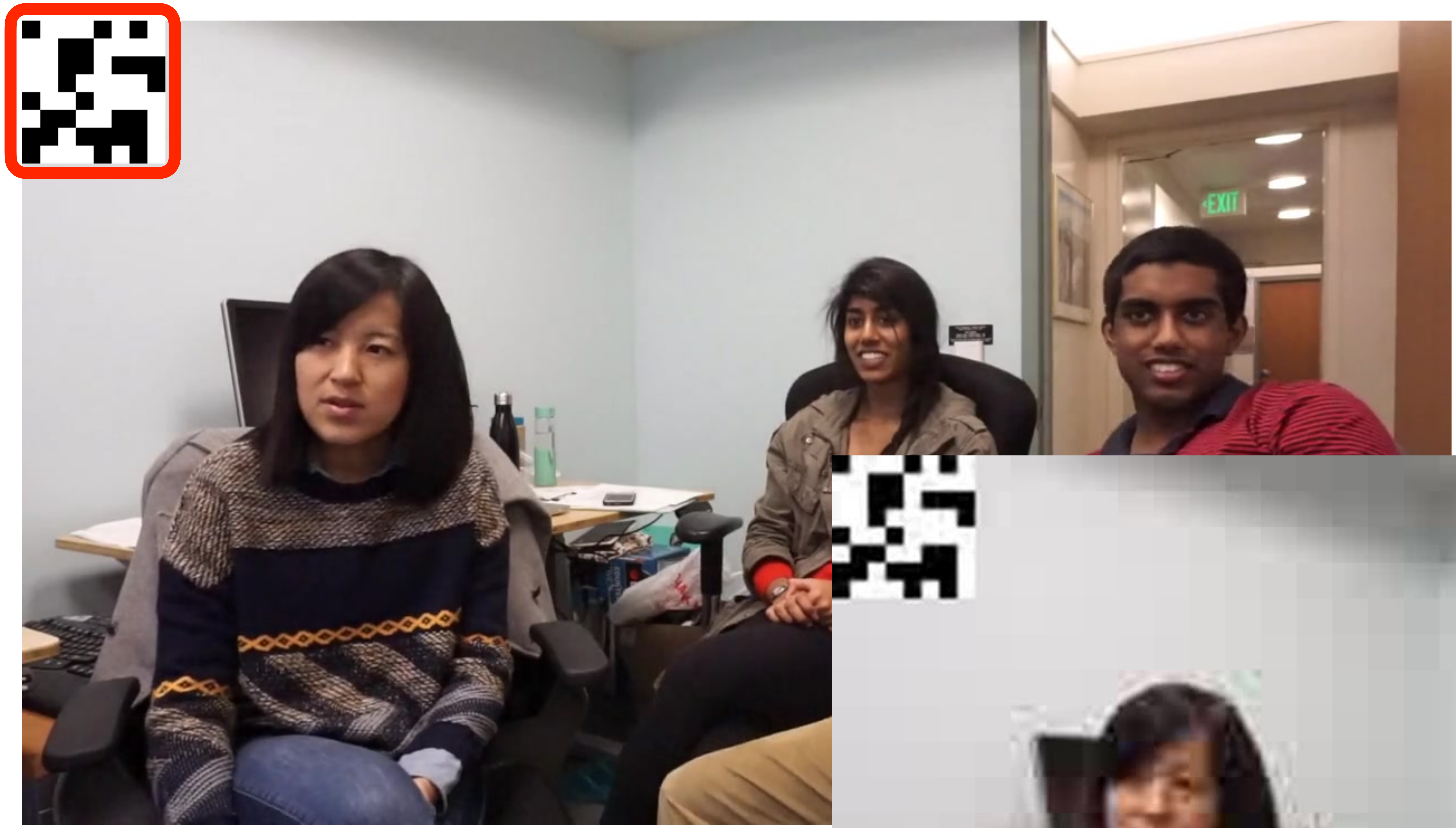
- Introduction
- Salsify's New Architecture
- **Measurement Testbed**
- Evaluation
- Conclusions

Goals for the measurement testbed

- A system with
  reproducible input video and
  reproducible network traces that runs
  unmodified version of the system-under-test.

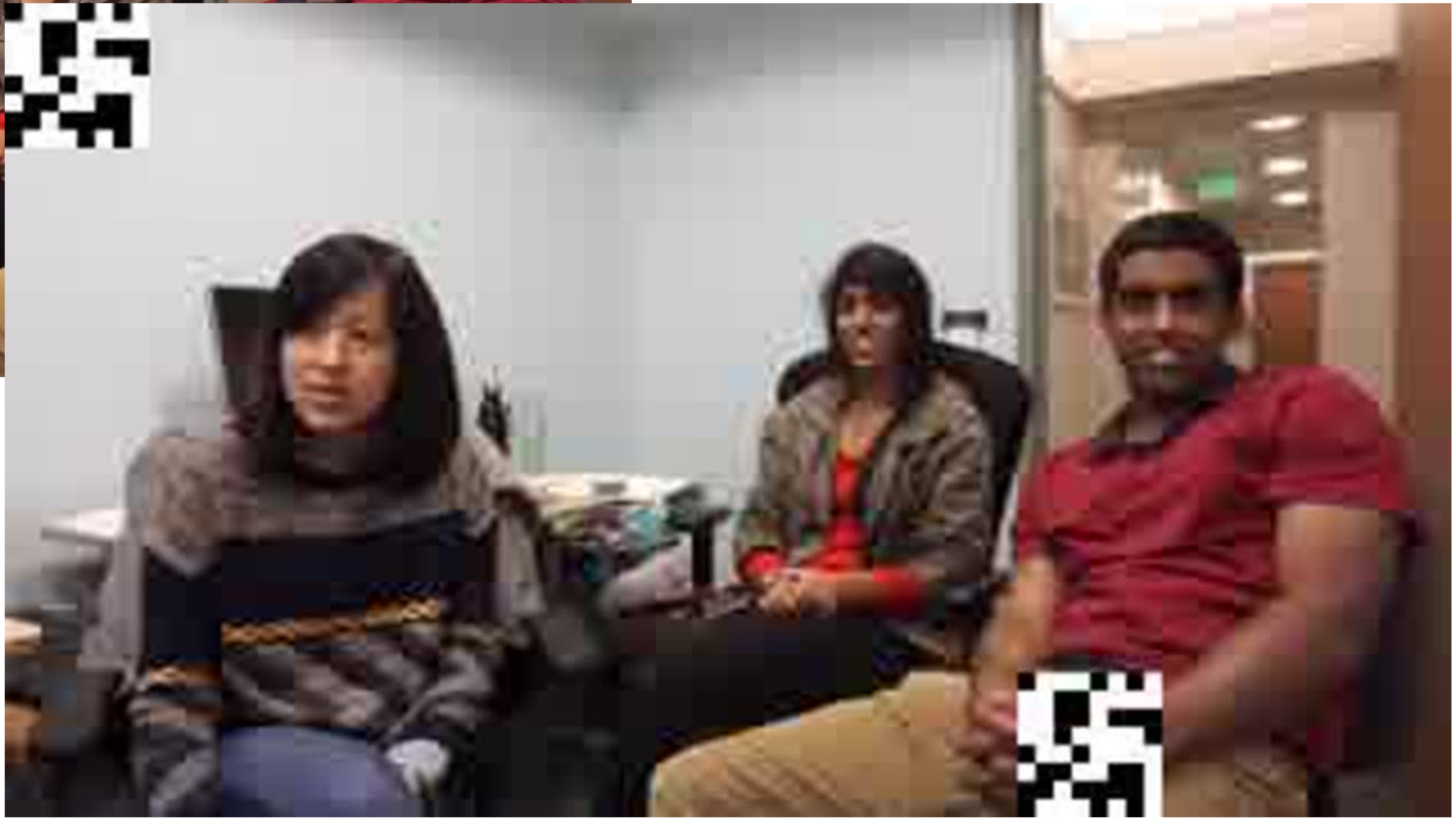
- Target QoE metrics: per-frame **quality** and **delay**.


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

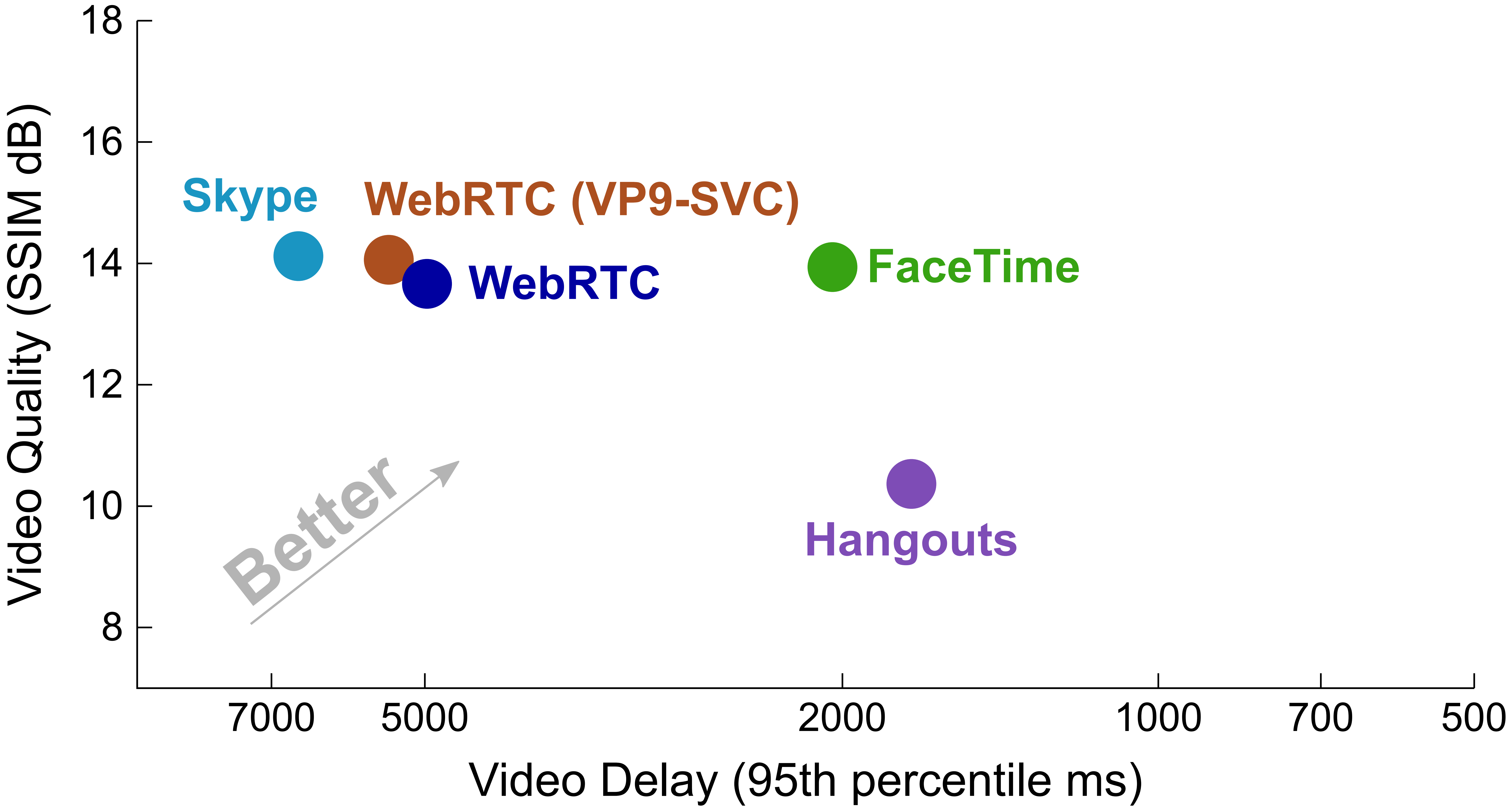
Timestamp: T+0.765s

Quality: 9.76 dB SSIM

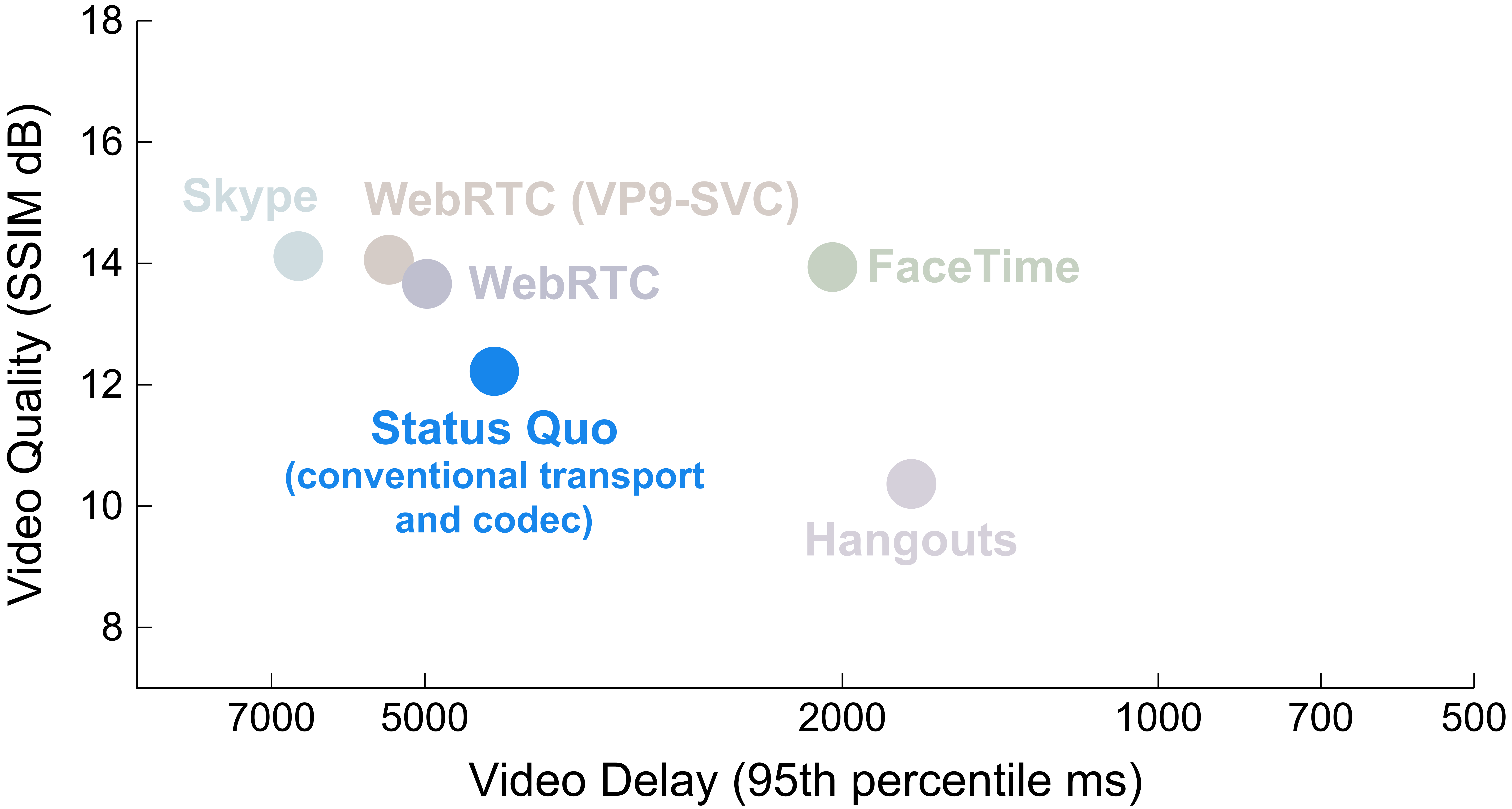
Outline

- Introduction
- Salsify's New Architecture
- Measurement Testbed
- **Evaluation**
- Conclusions

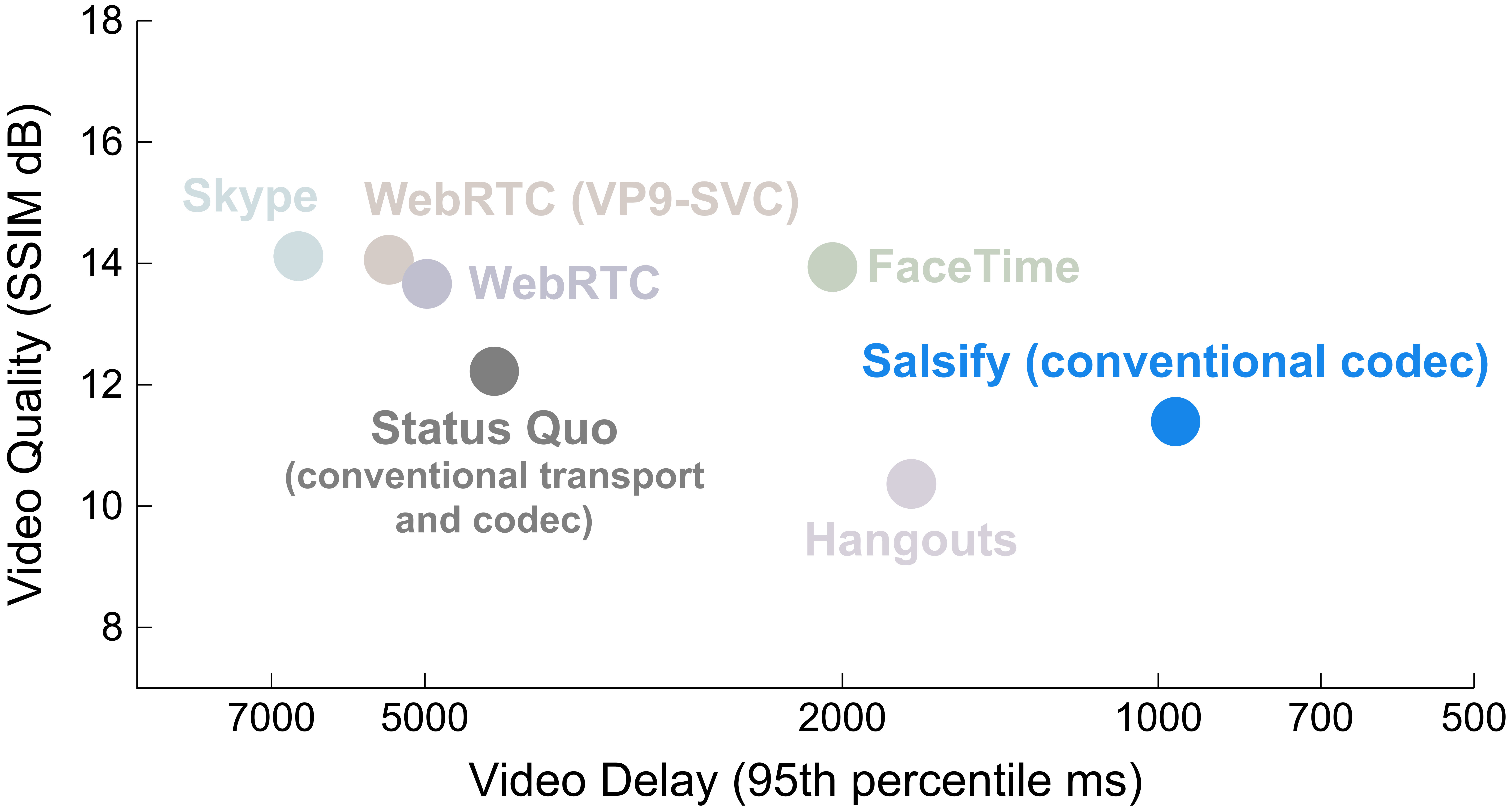
Evaluation results: **Verizon LTE Trace**



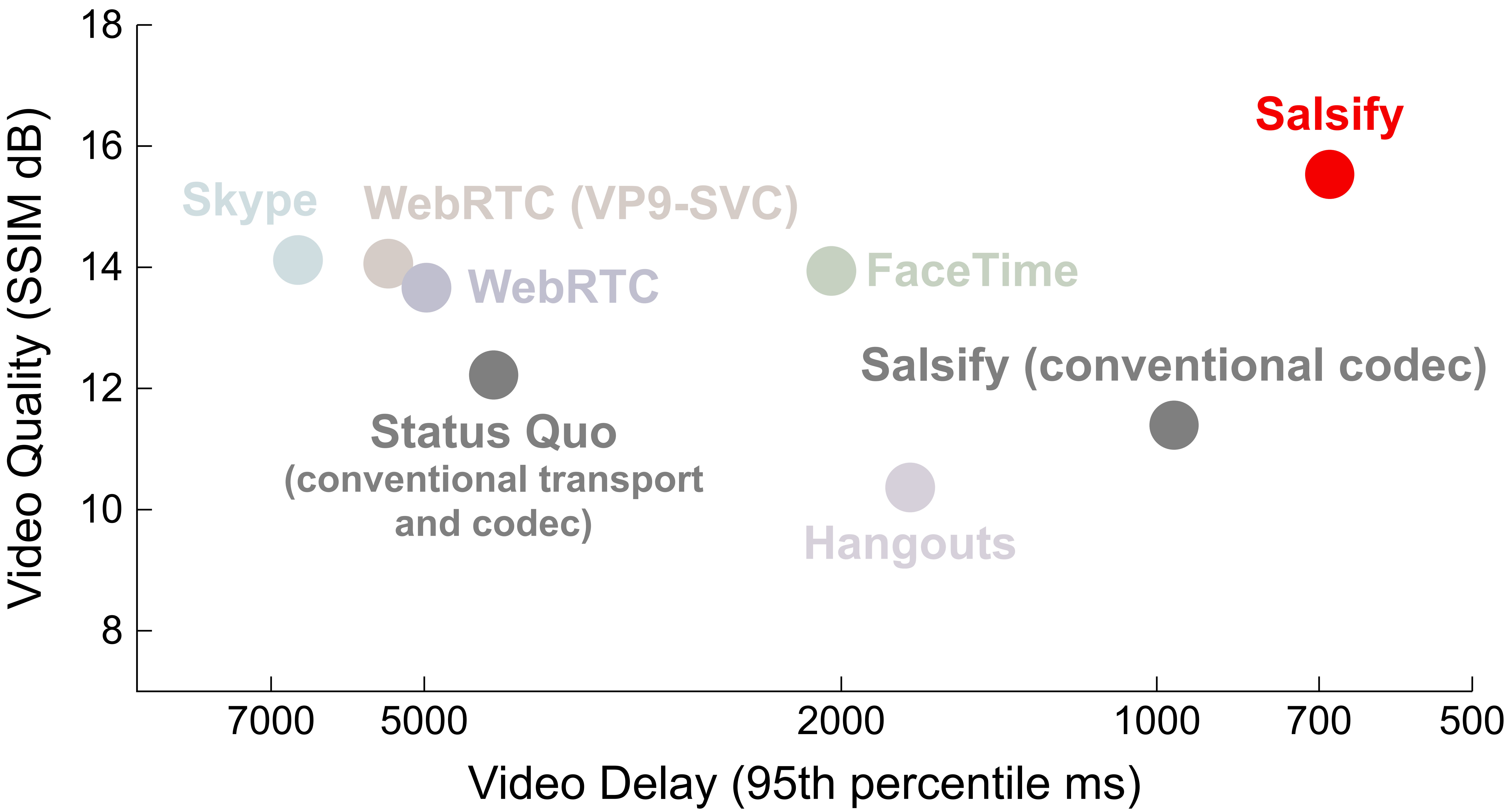
Evaluation results: **Verizon LTE Trace**



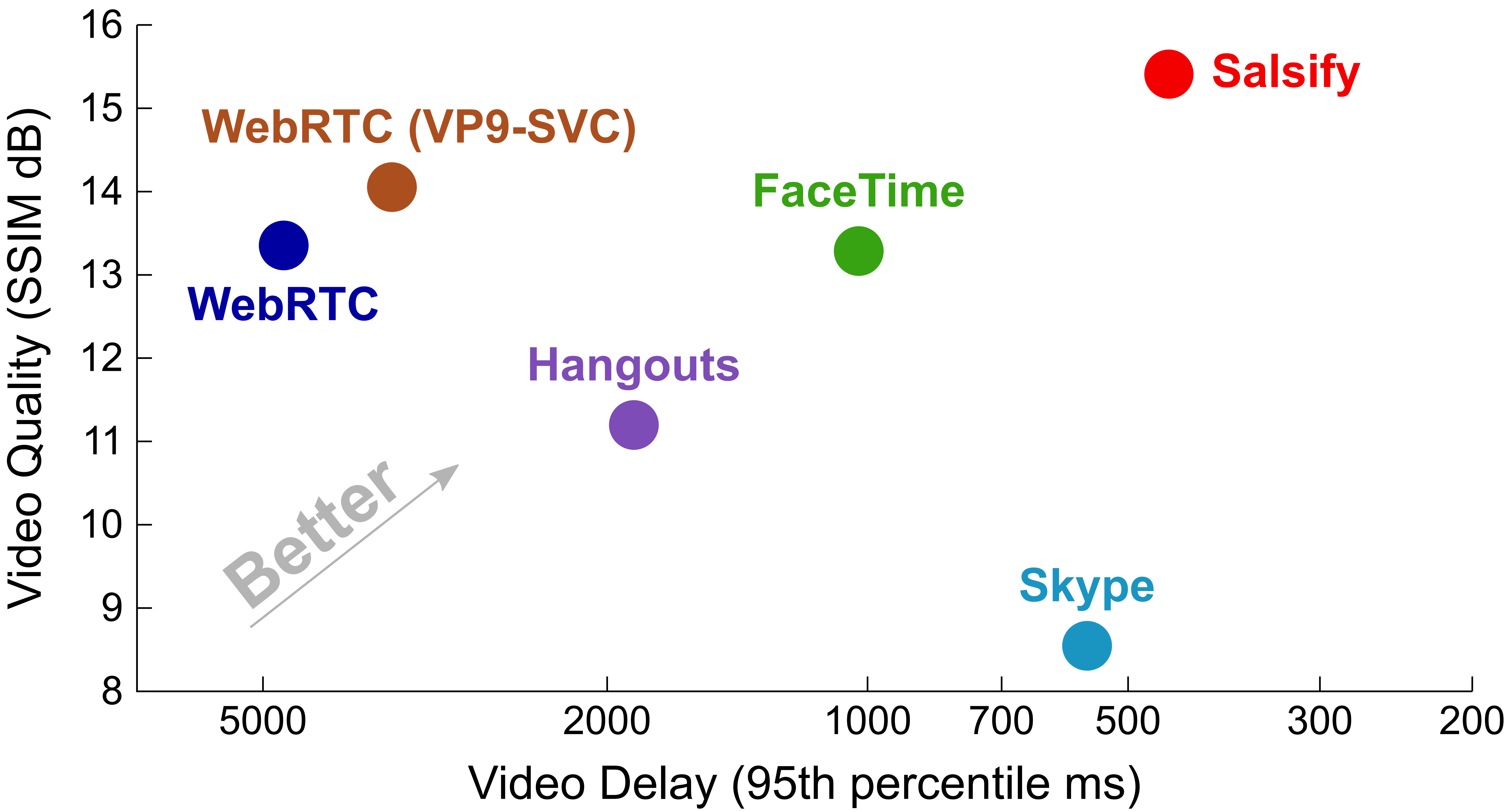
Evaluation results: **Verizon LTE Trace**



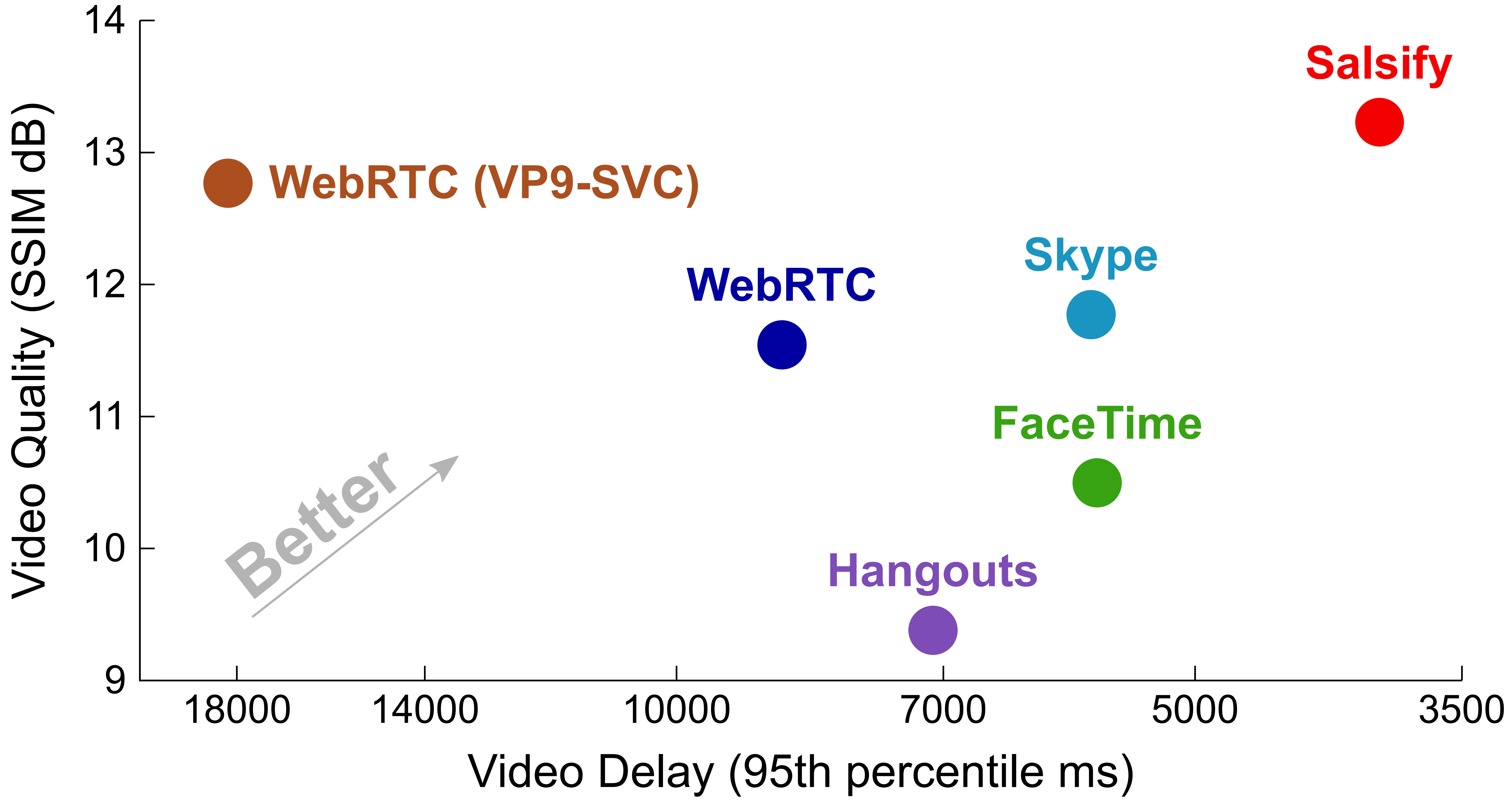
Evaluation results: **Verizon LTE Trace**



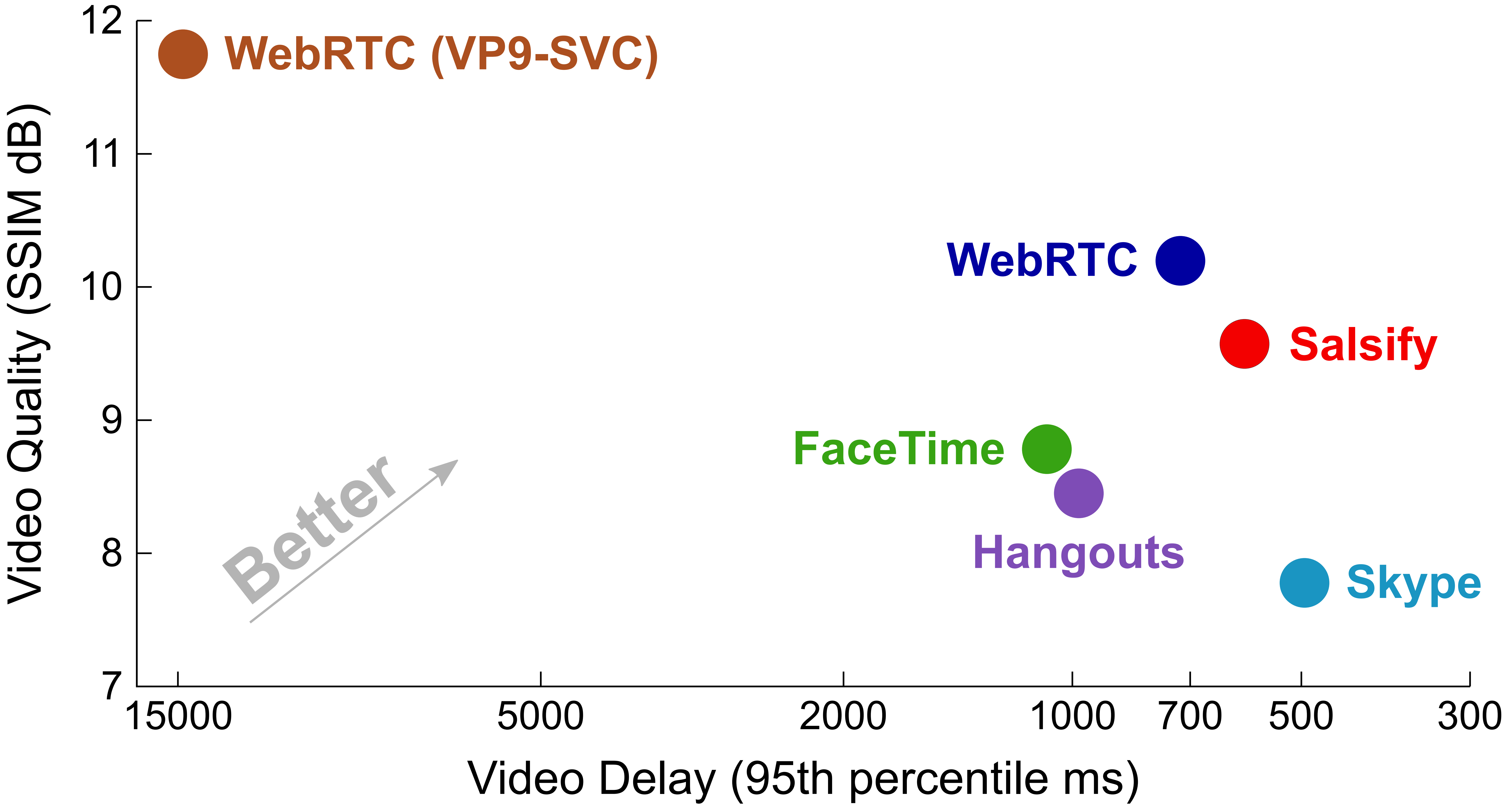
Evaluation results: **AT&T LTE Trace**



Evaluation results: **T-Mobile UMTS Trace**



Evaluation results: **Emulated Wi-Fi (no variations, only loss)**



Check out the demo videos at
https://snr.stanford.edu/salsify

Outline

- Introduction
- Salsify's New Architecture
- Measurement Testbed
- Evaluation
- **Conclusions**

Codecs have been treated as **black boxes** in video systems for a long time.

New systems have emerged from this functional interface

- NSDI'17: ExCamera
  - Using the functional codec to do massively-parallel video compression on AWS Lambda.
- NSDI'18: Salsify
  - Using the functional codec to compress frames to the right size, at the right time.
- Same interface, two different applications.

We encourage the codec designer and implementors to include save/restore state in the codecs—even if it’s large or opaque.

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.