Better Quality, Lower Delay:
Improving Realtime Video by Co-designing the Codec and the Transport

Sadjad Fouladi, Emre Orbay, John Emmons, Riad Wahby, Keith Winstein
Stanford University
Outline

• Introduction
• Design & Implementation
• End-to-end Measurement of Realtime Video
• Evaluation & Results
Realtime video: intelligent cameras
Realtime video: video conferencing
Realtime video: remote control
Realtime video in a nutshell
Many things rely on realtime video

- Realtime video applications require low delay (and good quality).
- Many of these applications may be on a variable link.
The problem: current systems do not work well on variable links

1. Transport tells encoder what “bitrate” to target

   [tens of milliseconds pass...]

2. Encoder delivers next frame

3. Transport stuck sending what encoder created

   Encoder overshot the bitrate $\rightarrow$ **blows up delay**

   Network changed $\rightarrow$ **blows up delay**
Enter Salsify

- Salsify introduces:
  - A video-aware congestion control.
  - A network-aware video codec.
  - Salsify’s traffic more closely match the network’s evolving capacity.

- Salsify consistently achieved higher video quality and lower delay than Hangouts, Skype, FaceTime, and WebRTC.
Outline

• Introduction

• Design & Implementation

• End-to-end Measurement of Realtime Video

• Evaluation & Results
Three problems with current realtime video systems

- Problem 1: No data on how big of a frame we can send without blocking.
  → Video-aware congestion control

- Problem 2: The current video codecs cannot hit the exact frame size.
  → Providing transport with multiple options

- Problem 3: It's not possible to mix-and-match different encoding streams.
  → Functional video codec
1) No data on how big of a frame we can send without blocking.

→ **Video-aware congestion control**

- Salsify’s transport protocol estimates the **frame size** that it thinks the network can handle without dropping or excessively queuing.
1) No data on how big of a frame we can send without blocking.

→ **Video-aware congestion control**

- The receiver treats each coded video frame as a packet train intended to probe the network.
- Receiver maintains moving average of packet inter-arrival times, pausing inference during grace period.
2) The current video codecs cannot hit the exact frame size.

→ Providing transport with multiple options

• Transport **late-binds** the decision of which version to send.
3) It's not possible to mix-and-match different encoding streams.

→ **Functional video codec**

• Why can't we just run multiple encoder instances with different quality levels?
Existing video codecs only expose a simple interface

\[
\text{encode}(\text{frames}[1:n]) \rightarrow \text{frames}[1:n]
\]

\[
\text{decode}(\text{frames}[1:n]) \rightarrow [\text{frames}, [...]]
\]
The decoder is an automaton
We cannot mix-and-match multiple video streams
We cannot mix-and-match multiple video streams
Video codec in explicit state-passing style

**encode**(state, image) → frame

**decode**(state, frame) → (state’, image)
3) It's not possible to mix-and-match different encoding streams.

→ **Functional video codec**

- By using the functional video codec, we can encode the options based on the current state.
1. **Transport** to **Encoder**: “Give me the next frame.”
2. **Encoder to Transport**: “Here’s a menu of $N$ different encodings.”
3. **Transport** to **Encoder**: “I sent option #3. Save that state, restore it to all N encoders, then go back to step 1.”
2. **Encoder** to **Transport**: “Here’s a menu of $N$ different encodings.”
3. **Transport** to **Encoder**: “I sent option #1. Save that state, restore it to all $N$ encoders, then go back to step 1.”
2. **Encoder** to **Transport**: “Here’s a menu of $N$ different encodings.”
3. **Transport** to **Encoder**: “I sent option #1. Save that state, restore it to all $N$ encoders, then go back to step 1.”
2. **Encoder** to **Transport**: “Here’s a menu of $N$ different encodings.”
3. **Transport** to **Encoder**: “I sent option #2. Save that state, restore it to all $N$ encoders, then go back to step 1.”
Outline

• Introduction
• Design & Implementation
• **End-to-end Measurement of Realtime Video**
• Evaluation & Results
End-to-end measurement of realtime video

- Reproducible input video
- Reproducible network traces
- Run unmodified version of the system-under-test

- Target QoE metrics:
  - Frame delay (from the camera at the sender, to the screen at the receiver)
  - Image quality
Barcoded video

Public internet and cellsim connections

Sender and receiver HDMI cables

AV.io HDMI -> USB device

Receiver HDMI output
Outline

• Introduction

• Design & Implementation

• End-to-end Measurement of Realtime Video

• Evaluation & Results
Network throughput (synthetic link)
Network throughput (synthetic link)
Network throughput (synthetic link)
Frame delay (synthetic link)
Network trace (Verizon LTE)

- Video quality (SSIM dB)
- Video delay (95th percentile ms)

- Toilet
- WebRTC
- FaceTime
- Skype
- Hangouts
- Salsify (no late binding)
- Better
Network trace (AT&T LTE)
Network trace (T-Mobile UMTS)
Takeaways

• Salsify introduces:
  
  • A video-aware congestion control.
  
  • A network-aware video codec.

• Salsify’s traffic more closely match the network’s evolving capacity.

• Salsify consistently achieved higher video quality and lower delay than Hangouts, Skype, FaceTime, and WebRTC.

excamera@cs.stanford.edu
End-to-end measurement of realtime video