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 Winston

snr.stanford.edu
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

Remote Server

Low-Latency Video Stream

Controller
WebRTC
(Google Chrome)
slow reaction to network variations
\[\Rightarrow\] stalls and glitches
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

- video codec
- transport protocol
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”

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

transport protocol & video codec
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 error
  Encode with different quality settings, pick the one that fits.

SOUNDS GOOD, DOESN’T WORK!
Video encoder turns frames into a compressed bitstream

frame 0

frame 1

frame 2

frame 3

Video Encoder

00011000010
01100001011
01110001000
00110000111
11011010011
01000110110
00000001001
00101100000
Encoder is *stateful*
There’s no way to undo an encoded frame in current codecs

codec\texttt{.encode}([\text{ображення}, \text{ображення}, \ldots]) \rightarrow \text{bytestream}...

The state is internal to the encoder—no way to save/restore the state.
Functional video codec to the rescue

\[ \text{encode}(\text{state}, \text{frame}) \rightarrow \text{state}', \text{frame} \]

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

transport protocol & video codec
Codec → Transport
“Here’s two versions of the current frame.”

![Diagram with two versions of the frame, one better at 50 KB and one worse at 25 KB, with a target frame size of 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.”
Transport → Codec

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

Throughput (Mbps)

Salsify

WebRTC (Chrome 65)
Evaluation results: Verizon LTE Trace

![Graph showing comparison of video quality and delay among Skype, WebRTC (VP9-SVC), FaceTime, and Hangouts.](image-url)
Evaluation results: Verizon LTE Trace

![Video Quality vs. Video Delay Plot]

- Skype
- WebRTC (VP9-SVC)
- WebRTC
- FaceTime
- Status Quo
  (conventional transport and codec)
- Hangouts

Video Quality (SSIM dB) vs. Video Delay (95th percentile ms)
Evaluation results: Verizon LTE Trace

[Graph showing video quality (SSIM dB) vs. video delay (95th percentile ms) with points for Skype, WebRTC (VP9-SVC), WebRTC, Status Quo (conventional transport and codec), FaceTime, Salsify (conventional codec), Hangouts]
Evaluation results: Verizon LTE Trace

- **Video Quality (SSIM dB)**
  - Skype
  - WebRTC (VP9-SVC)
  - WebRTC
  - Status Quo (conventional transport and codec)
  - FaceTime
  - Salsify (conventional codec)
  - Salsify
  - Hangouts

- **Video Delay (95th percentile ms)**
  - 7000
  - 5000
  - 2000
  - 1000
  - 700
  - 500
Evaluation results: AT&T LTE Trace
Evaluation results: T-Mobile UMTS Trace

![Graph showing video quality and delay comparison for different applications. The graph plots video quality (SSIM dB) on the y-axis and video delay (95th percentile ms) on the x-axis.](image)

- WebRTC (VP9-SVC)
- Skype
- FaceTime
- Hangouts
- Salsify

The graph indicates that WebRTC (VP9-SVC) has the highest video quality and is better in terms of delay compared to other applications.
Evaluation results: No variations

![Graph showing video quality and delay for different applications. WebRTC (VP9-SVC) has the highest video quality with lower delay compared to other applications like Salsify, FaceTime, Hangouts, and Skype.]
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.