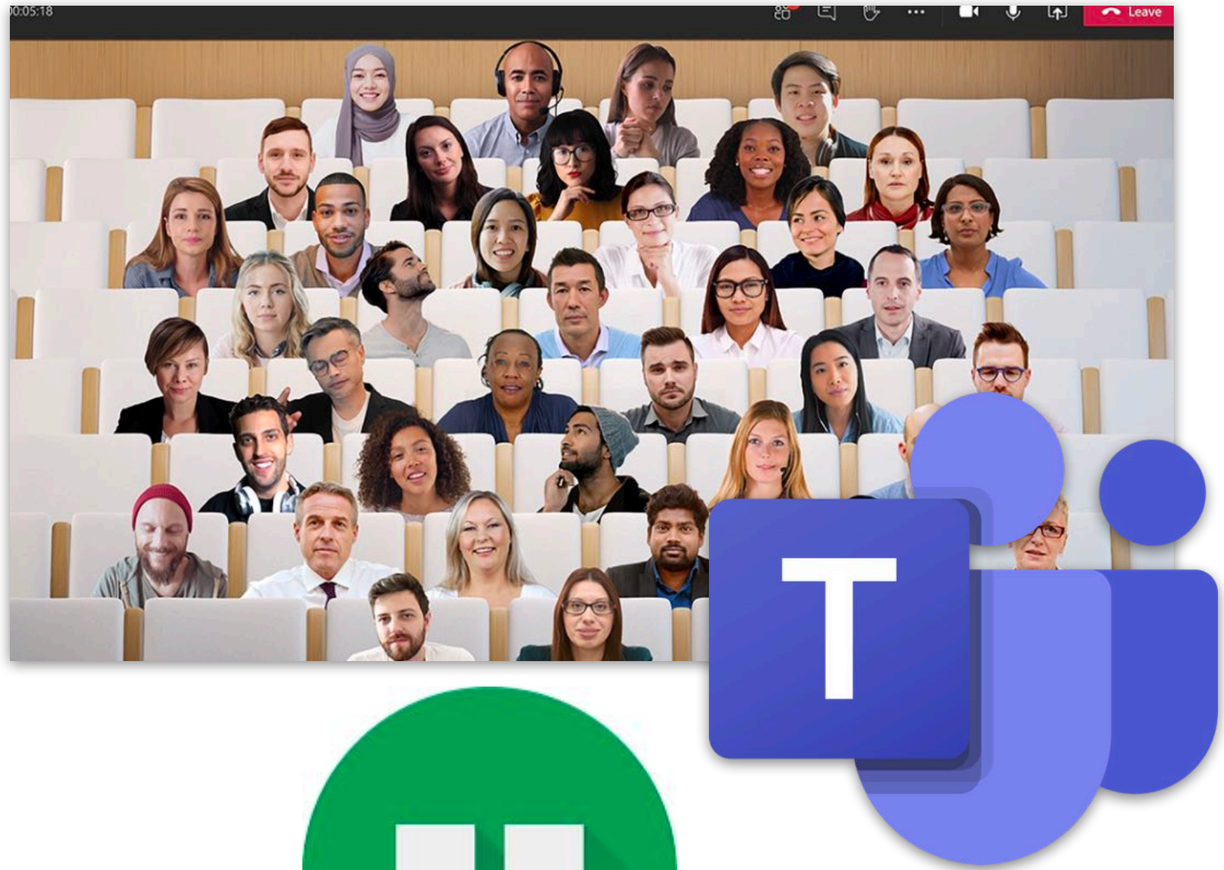
## Lecture 13:

# Video Conferencing (Discussion)

---

**Visual Computing Systems**
**Stanford CS348K, Spring 2021**

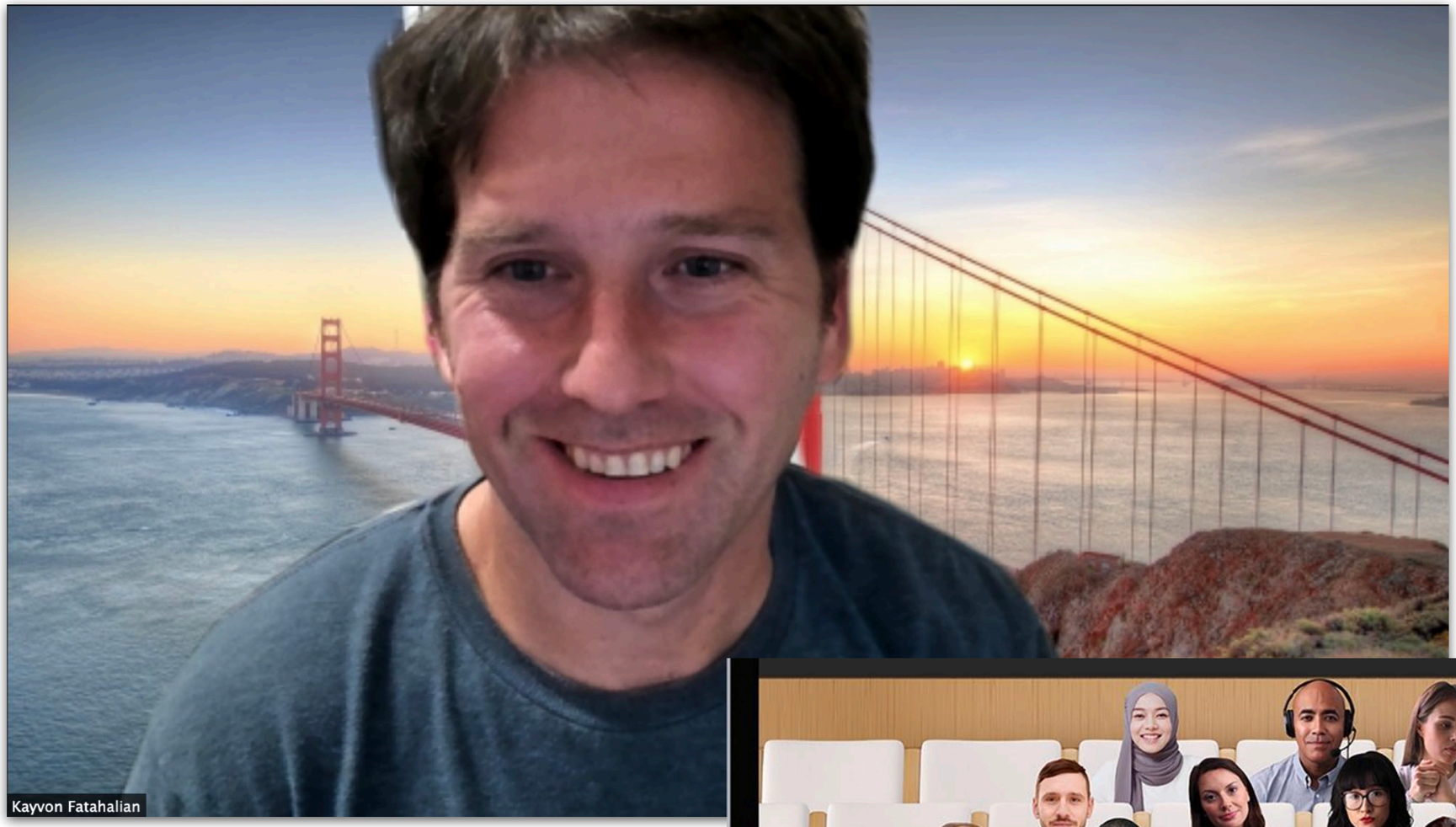
# As you can imagine, a lot of players in video conferencing in 2021 (big and small!)

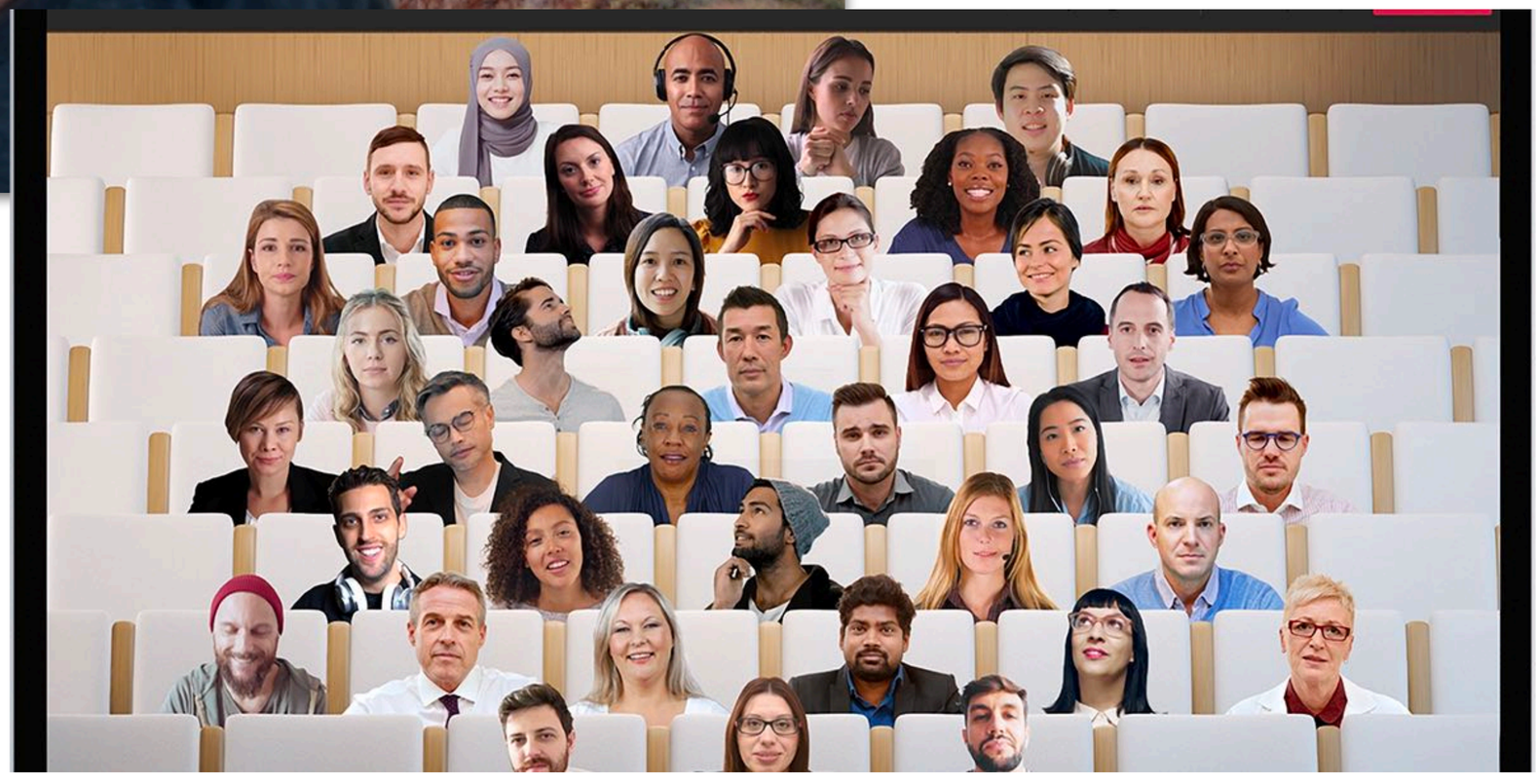



# Let's design a video conferencing system

- We want to deliver a visually rich experience similar to features of modern platforms

# Let's design a video conferencing system

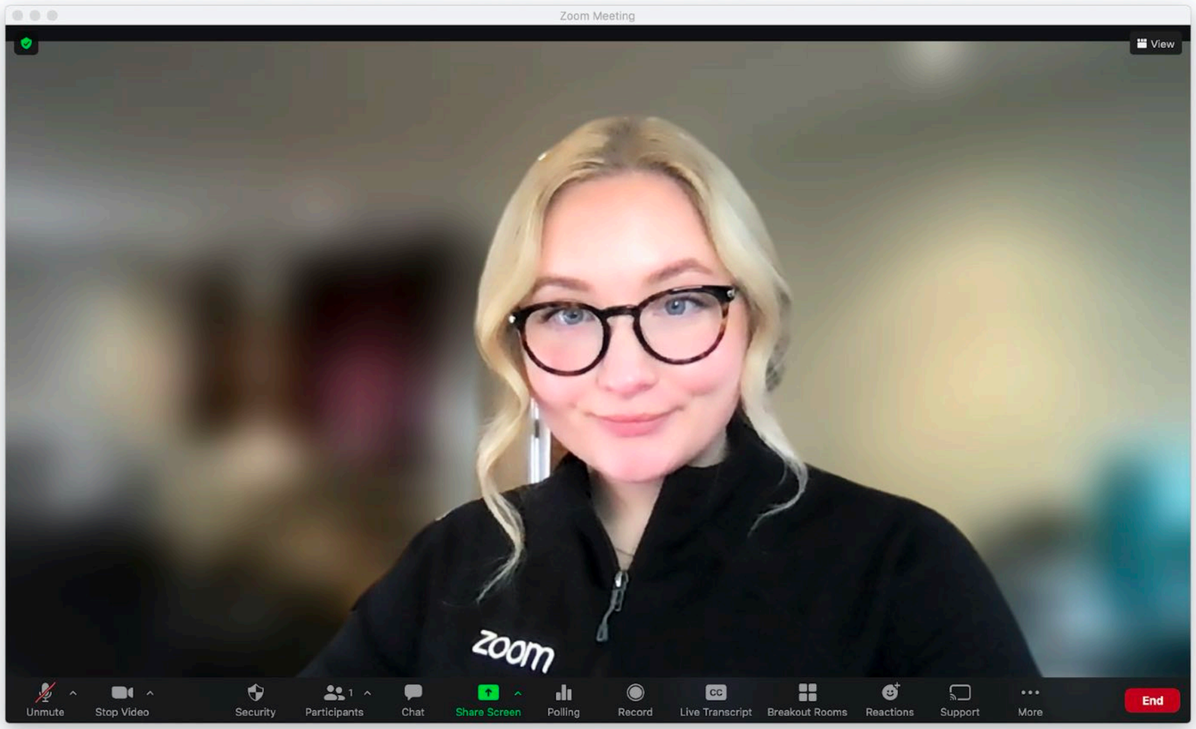


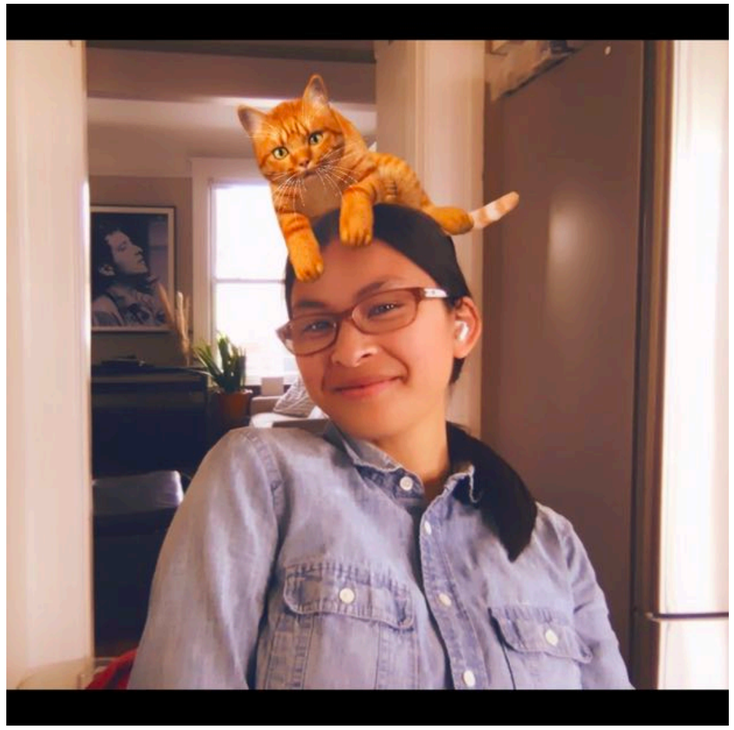
## Segment participant from background

# Let's design a video conferencing system

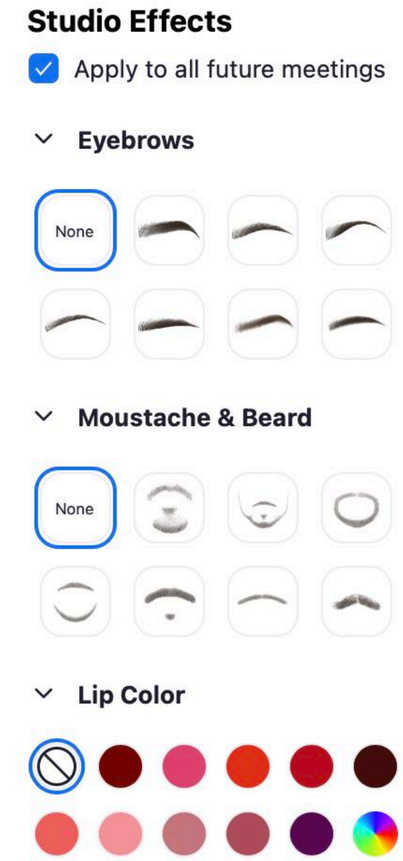
## Perform image processing to enhance look of video feed



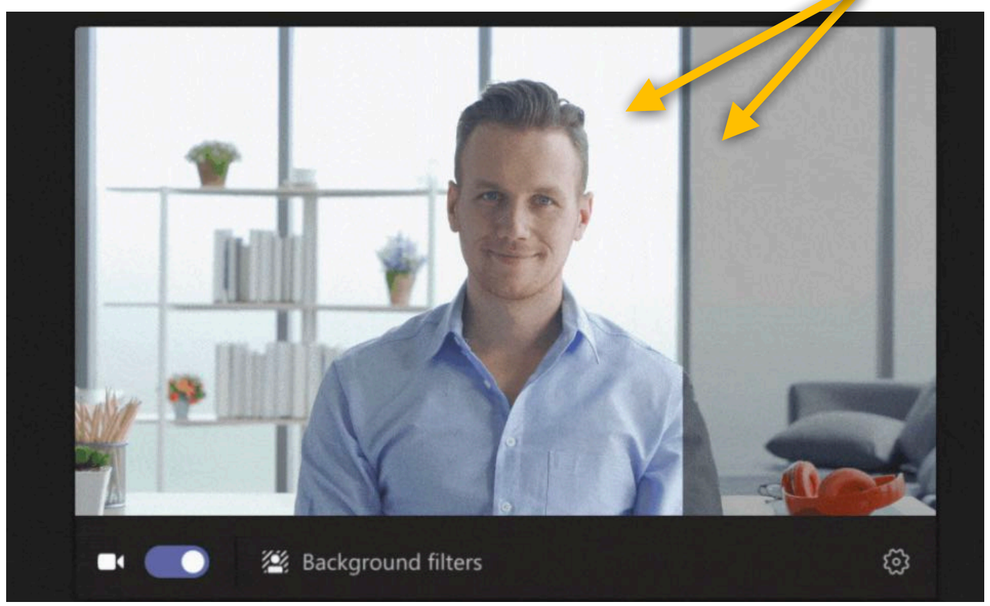

Blur background



Render additional content




Adjust lighting




# Let’s design a video conferencing system

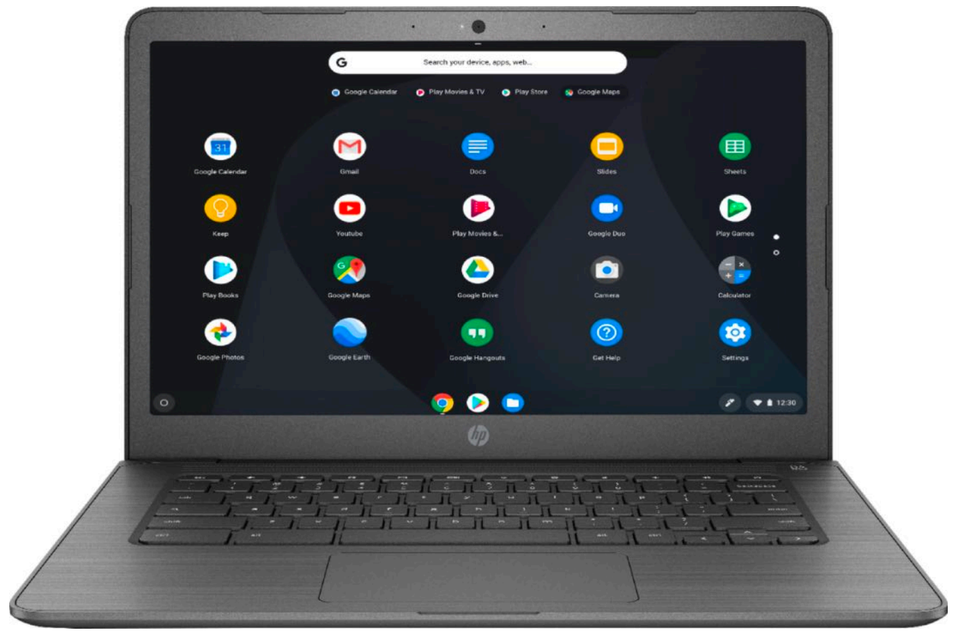
## Large gallery views: companies raced to provide 7x7 gallery in 2020


Maximum participants displayed per screen in Gallery View:

○ 25 participants ◉ 49 participants

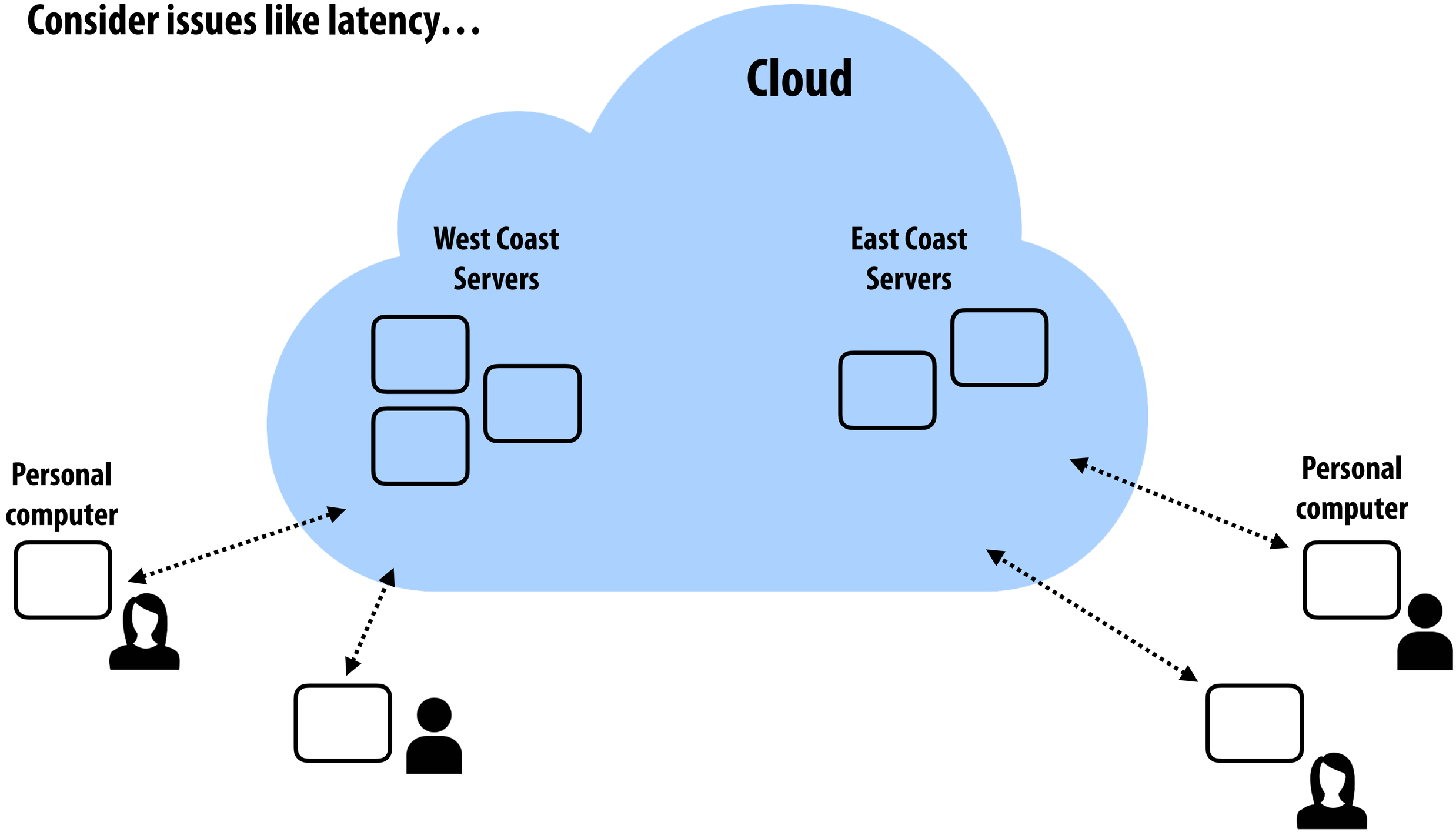
# Deliver to wide range of clients and network settings





# Setup…

## Consider issues like latency…

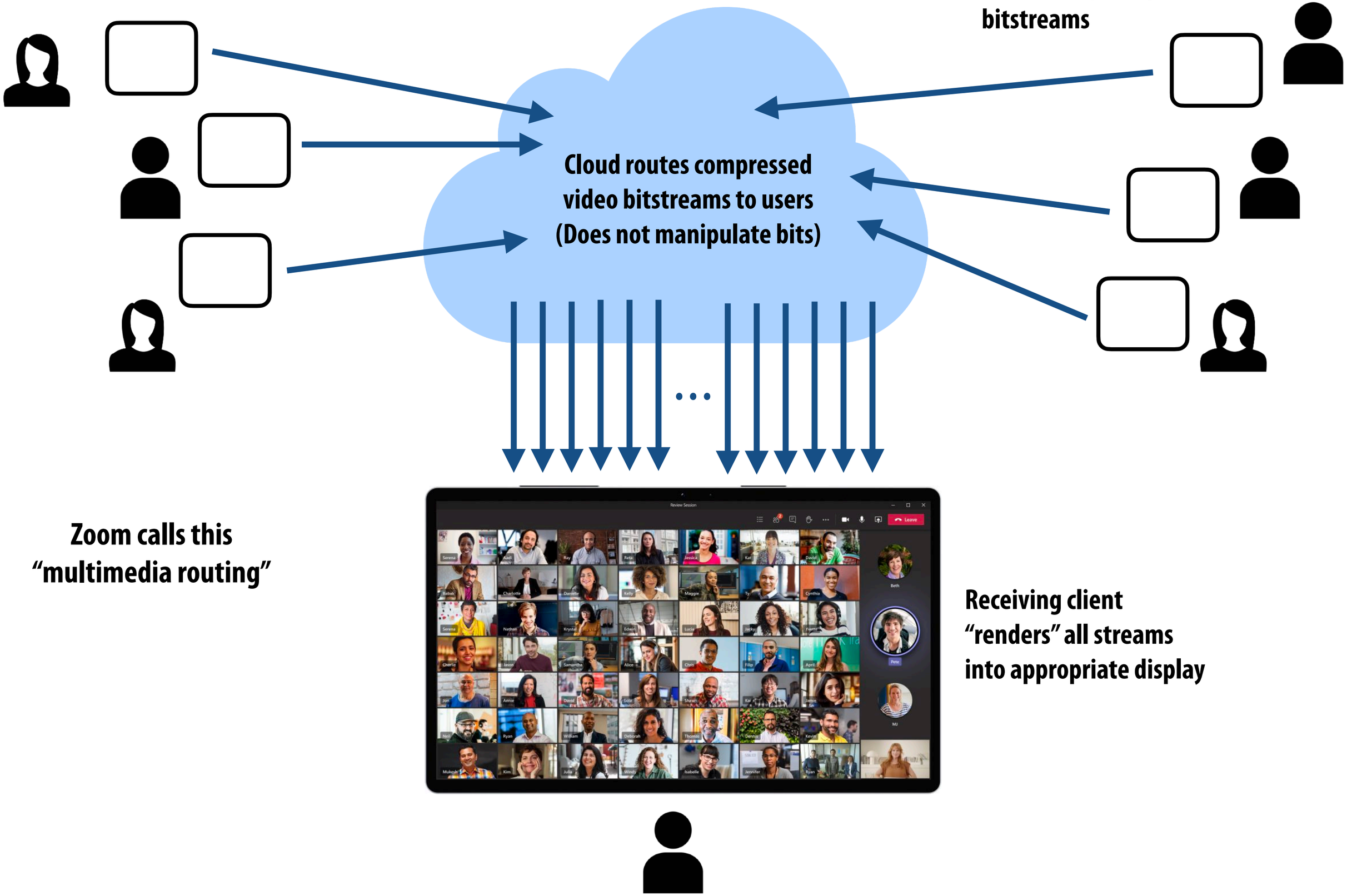



Icon credits: person by mim studio from the Noun Project, avatar by Soremba from the Noun Project

# Q. Should we transcode/process video on our cloud servers?

- What are advantages?
- What are disadvantages?

# Implementing gallery view




Zoom calls this "multimedia routing"

Receiving client "renders" all streams into appropriate display

# One drawback of this design

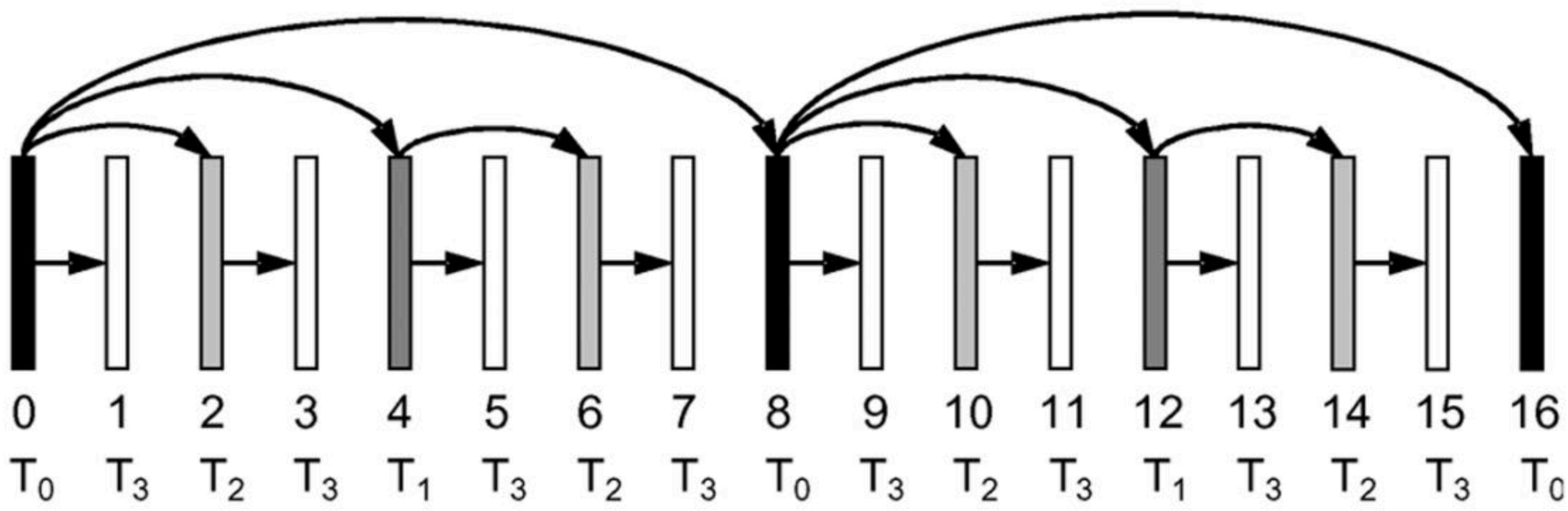
- If each client is providing a single compressed video stream, that means each person on the video call must receive the same bits right? (What if they are on different network connections)

# Scalable video codec (SVC)

SVC is an extension of H.264 standard

- "Scalable" compressed video bitstream: subsets of the bitstream encode valid video streams for a decoder
  - Implication: if packets get lost, the remaining packets form a valid H.264 bitstream, albeit at lower resolution or quality

Example: temporal scalability




Layer 0: ($T_0$) defines valid video at frame rate R

Layer 1 ($T_1$) defines bumps frame rate to 2R

…

Note how layer 0 information is used to predict higher layer information

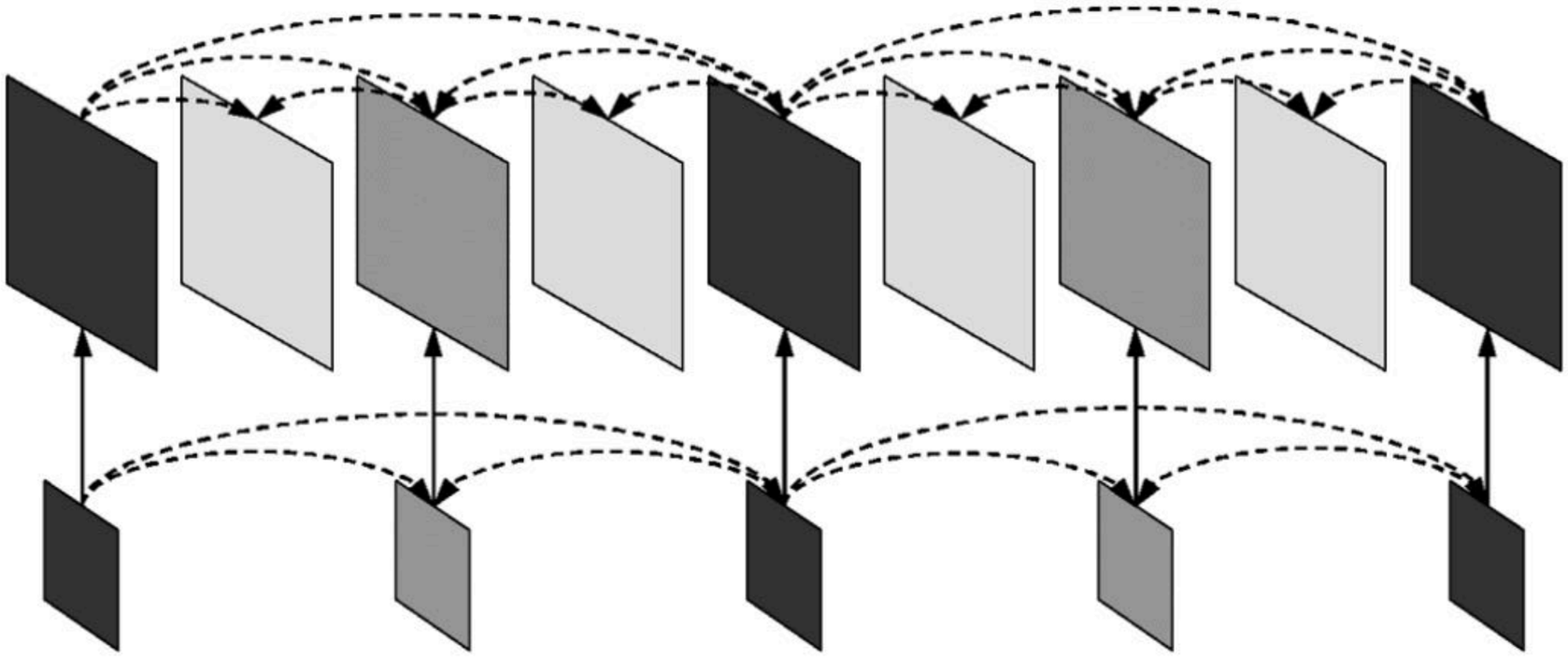
# Scalable video codec (SVC)

**SVC is an extension of H.264 standard**

**Example: spatial scalability**

**Layer 1:**
**(Higher res)**

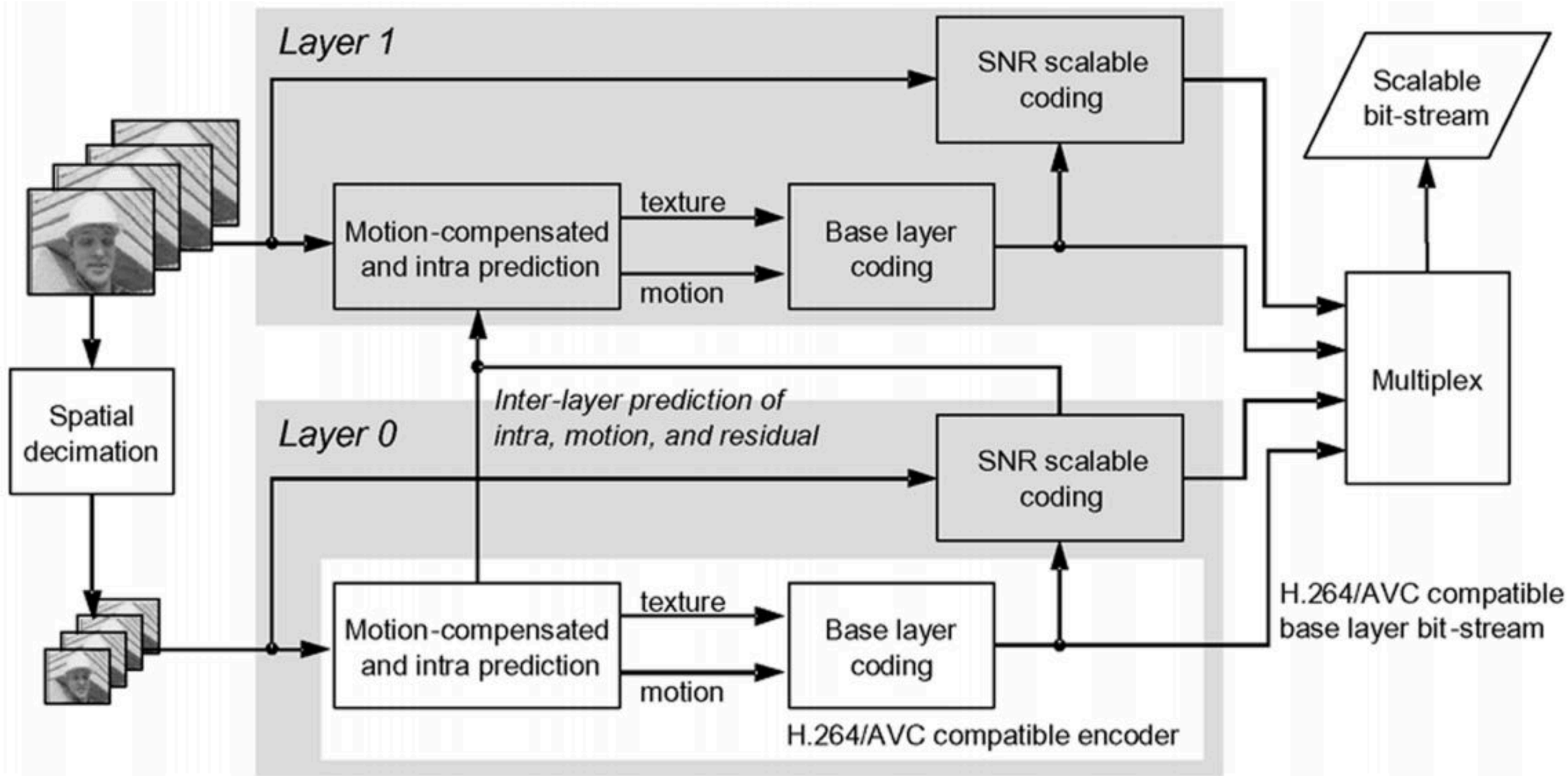
**Layer 0:**
**(Low res)**



**Again, note how layer 0 information is used to predict higher layer information**
**(Higher efficiency than independently encoding two video streams)**

**Layer 0: defines valid video at low resolution (and low frame rate)**
**Layer 1: provides additional information for higher resolution (and higher frame rate) video**

# Scalable video codec (SVC) encoder




**Costs: higher encoding/decoding costs**

**(But possible on modern clients as SVC is supported in hardware)**

# We just saw one benefit of a more capable client.

# What are other benefits of increasingly capable clients?

# Does calculus change if cloud has high performance video processing/encode/decode hardware?

## Common trend in visual computing systems: "Wheel of reincarnation"

# On the Design of Display Processors

T. H. MYER
*Bolt Beranek and Newman Inc, Cambridge, Mass.*
AND
I. E. SUTHERLAND*
*Harvard University, Cambridge, Mass.*

The flexibility and power needed in the data channel for a computer display are considered. To work efficiently, such a channel must have a sufficient number of instructions that it is best understood as a small processor rather than a powerful channel. As it was found that successive improvements to the display processor design lie on a circular path, by making improvements one can return to the original simple design plus one new general purpose computer for each trip around. The degree of physical separation between display and parent computer is a key factor in display processor design.

KEY WORDS AND PHRASES: display processor design, display system, computer graphics, graphic terminal, displays, graphics, display generator, display channel, display programming, graphical interaction, remote displays
CR CATEGORIES: 2.44, 6.22, 6.29, 6.35



## 1. Introduction

In mid-1967 we specified a research display system. This paper describes some of the problems we encountered and some conclusions we have drawn. The display will be an adjunct to an SDS-940 time-shared computer system. The chief purpose for the display and the parent computer is programming research.

When we first approached the task, we assumed we had merely to select one of the several available commercial displays. This proved possible with the analog equipment that constitutes a *display generator*; we found several display generators that combined good accuracy, resolution, and speed. However, the control part of the display, which we have come to call the *display processor*, was another story. We were not completely happy with the command repertoire of any of the commercial systems we saw; we were not sure just how to couple the display to our computer, and above all, we had serious doubts about what a display processor should *be*.

---

This work was sponsored by the Advanced Research Projects Agency under ARPA Order No. 627, Amendment No. 2, and conducted under Contract No. AF19(628)-5065, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, Bedford, Massachusetts 01730.
* And Bolt Beranek and Newman Inc, Cambridge, Mass.

## I. Introduction

In mid-1967 we specified a research display system. This paper describes some of the problems we encountered and some conclusions we have drawn. The display will be an adjunct to an SDS-940 time-shared computer system. The chief purpose for the display and the parent computer is programming research.

When we first approached the task, we assumed we had merely to select one of the several available commercial displays. This proved possible with the analog equipment that constitutes a *display generator*; we found several display generators that combined good accuracy, resolution, and speed. However, the control part of the display, which we have come to call the *display processor*, was another story. We were not completely happy with the command repertoire of any of the commercial systems we saw; we were not sure just how to couple the display to our computer, and above all, we had serious doubts about what a display processor should *be*.

Finally we decided to design the processor ourselves, because only in this way, we thought, could we obtain a truly complete display processor. We approached the task by starting with a simple scheme and adding commands and features that we felt would enhance the power of the machine. Gradually the processor became more complex. We were not disturbed by this because computer graphics, after all, *are* complex. Finally the display processor came to resemble a full-fledged computer with some special graphics features. And then a strange thing happened. We felt compelled to add to the processor a second, subsidiary processor, which, itself, began to grow in complexity. It was then that we discovered a disturbing truth. Designing a display processor can become a never-ending cyclical process. In fact, we found the process so frustrating that we have come to call it the "wheel of reincarnation." We spent a long time trapped on that wheel before we finally broke free. In the remainder of this paper we describe our experiences. We have written it in the hope that it may speed others on toward "Nirvana."

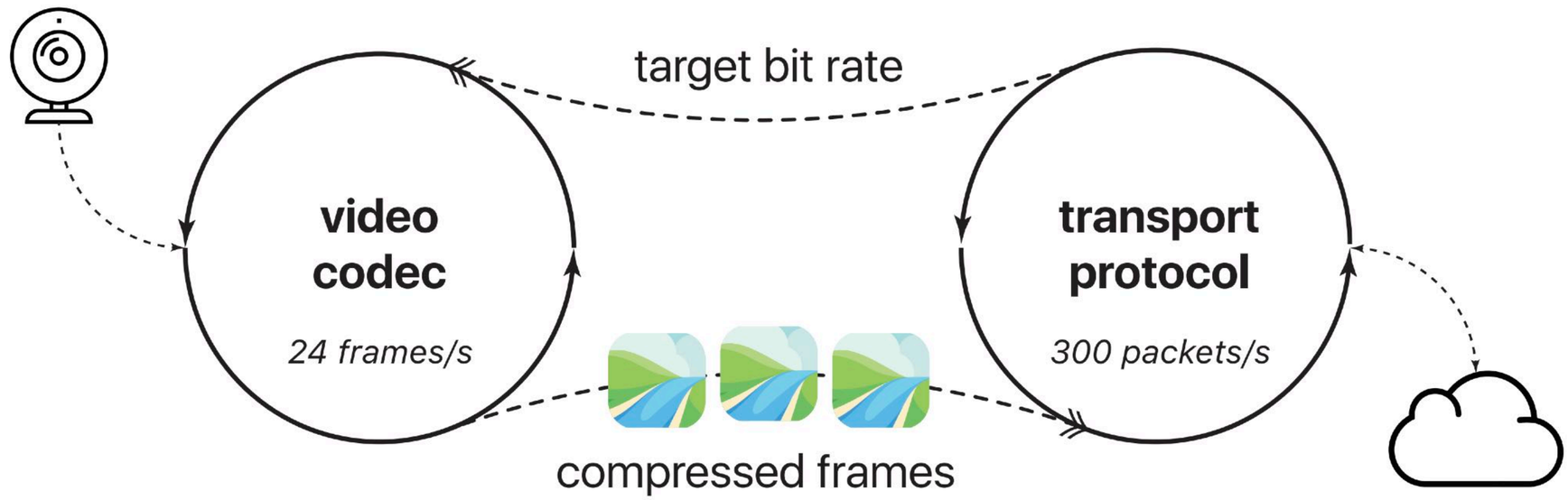
# (If time)

# Co-designing video compressor and network transport

[Credit: Fouladi et al. 2018]

# Status quo

- Video encoder proceeds to compress video frames, targeting a bit rate (on average) provided by the network protocol
- But any one frame may be too large or small (some may be hard to predict)




**Encoder: targets an average bit rate (bits/second)**

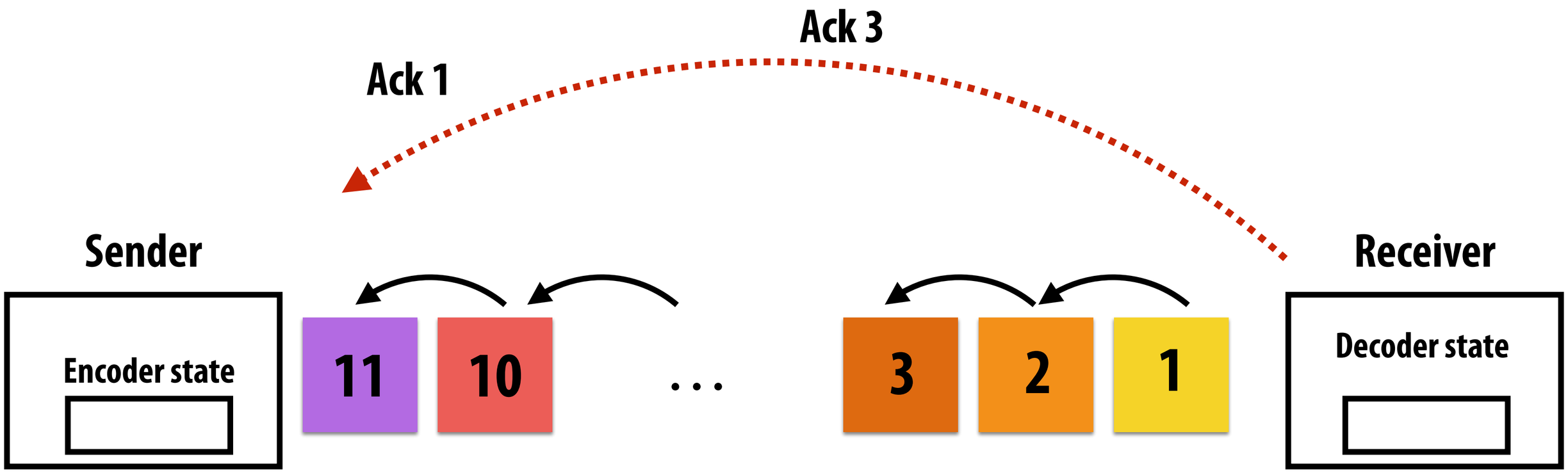
**Protocol: attempts to determine and use the available capacity of the network**

**But generates individual frames (which individually may or may not exceed network capacity)**

- If the encoder overshoots, packet loss occurs, so frames get dropped

[Credit: Fouladi et al. 2018]

# Consider challenges




Sender realizes packet carrying frame 2 has been dropped (e.g, it was too big)

But sender cannot re-encode frame at lower size because it's moved on and has different internal state

But sender cannot re-encode frame at lower size because it's moved on and has different internal state

# Stateless (functional) video encoder

```
// prob model: tables representing encoding of values in video stream
// reference_images contains three prior images
state:= (prob_model, reference_images[3]);

// just a full image
keyframe := image pixels for entire frame

// prediction_modes and motion vectors define how to predict current
// frame given decoder state
// residue is correction to this prediction
interframe := (prediction_modes, motion_vectors, residue)
```

```
// decoding a frame generates one image of pixels, and
// an updated decoder state
decode(state, compressed_frame) -> (new_state, image)

// generate an interframe approximating image given the current
// decoder state.  This operation requires expensive motion estimation.
encode-given-state(state, image, quality_param) -> interframe
```

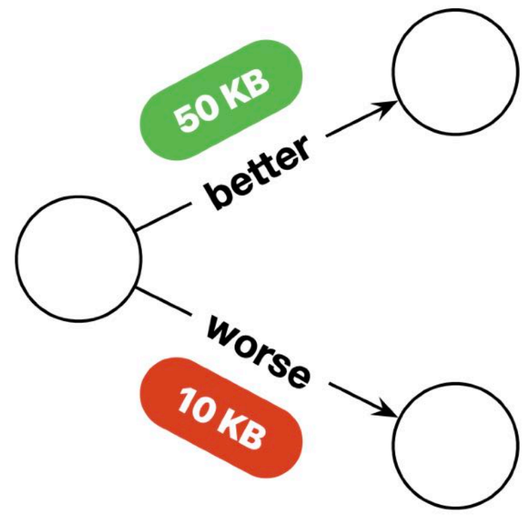
[Fouladi et al. 2017]

# Salsify: codec presents network three options

**[Fouladi et al. 2018]**

For each frame, codec presents the transport with *three* options:

- 🔺 A slightly-higher-quality version,
- 🔻 A slightly-lower-quality version,
- ✖ Discarding the frame.



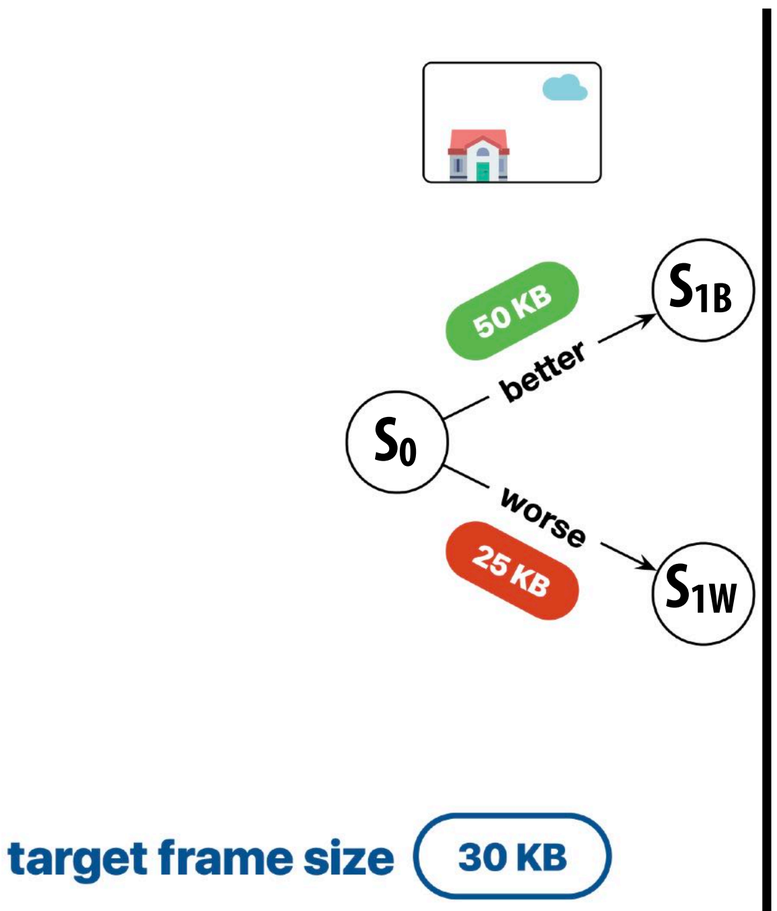

**Notice roll of functional encoder.**

**Can encode "better", reset to previous state, and then encode "worse".**

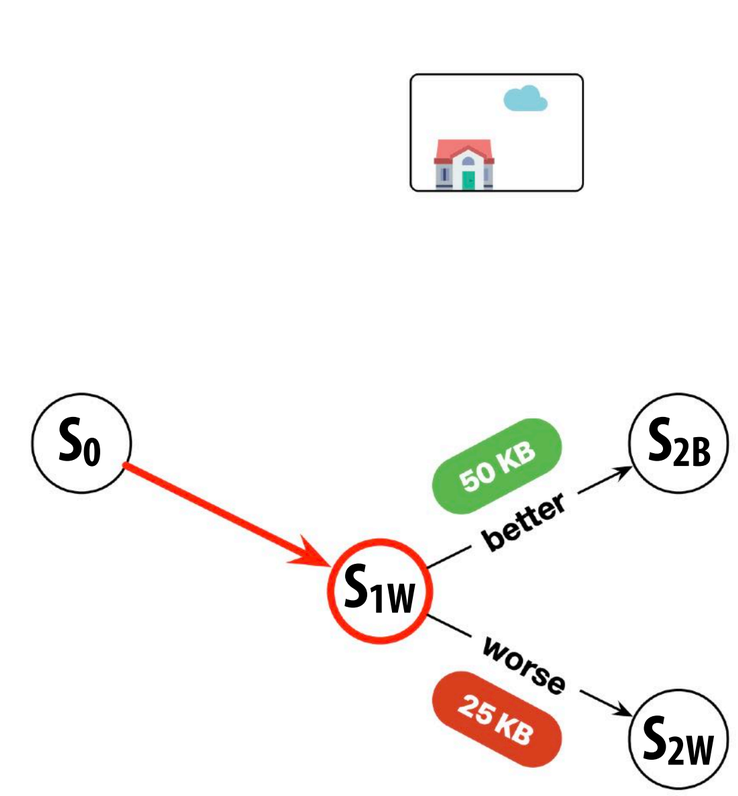
# Salsify's "video aware transport protocol: network determines what to transmit based on size of compressed frames

**Before: network tried to send whatever the compressor generated.**

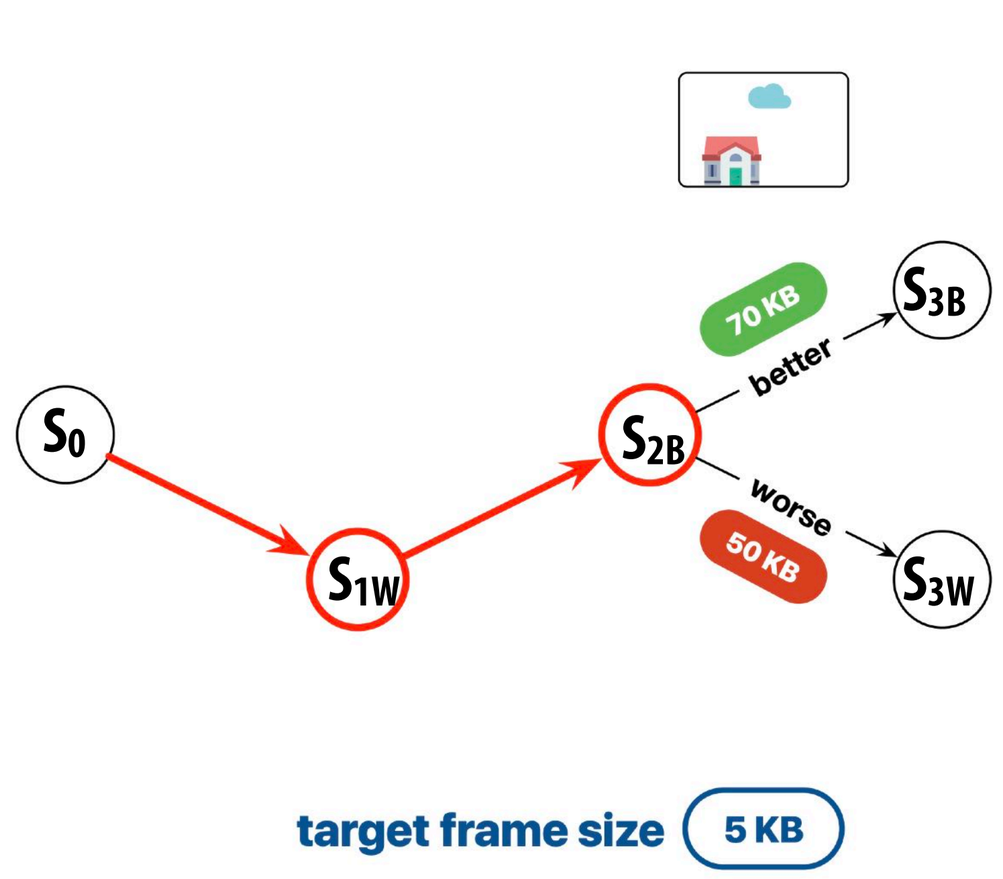
**Notice roll of functional encoder.**
**Can resume encoding from state that results from transport's choice.**




**Transport picks "worse"**



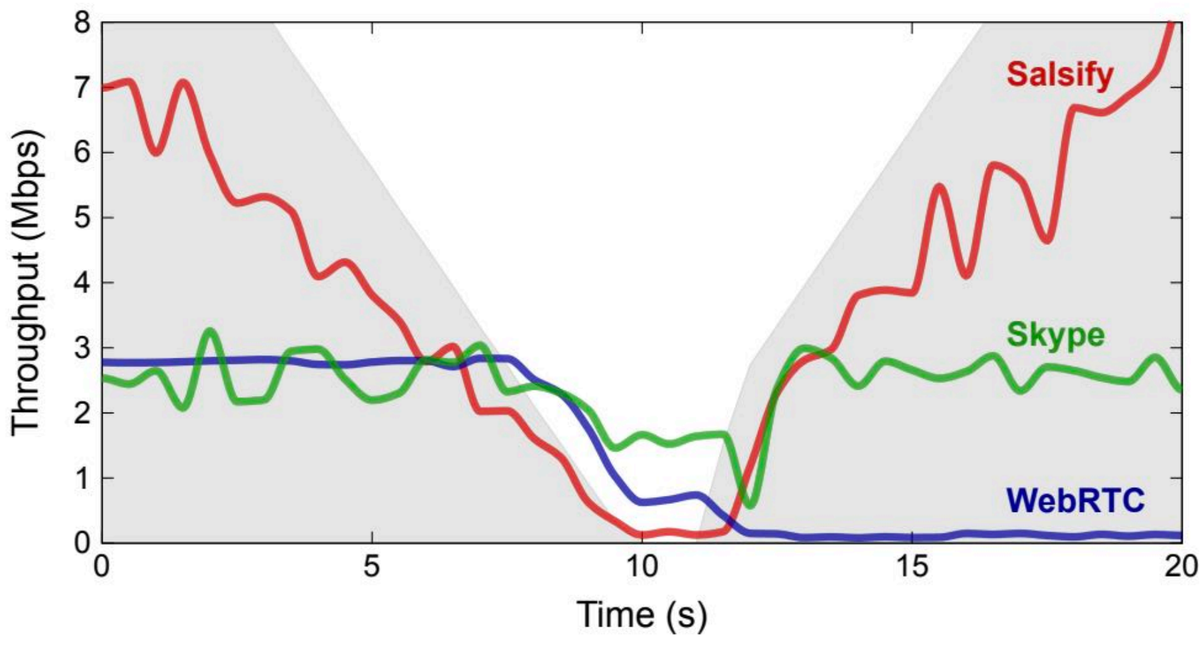

**Transport picks "better"**



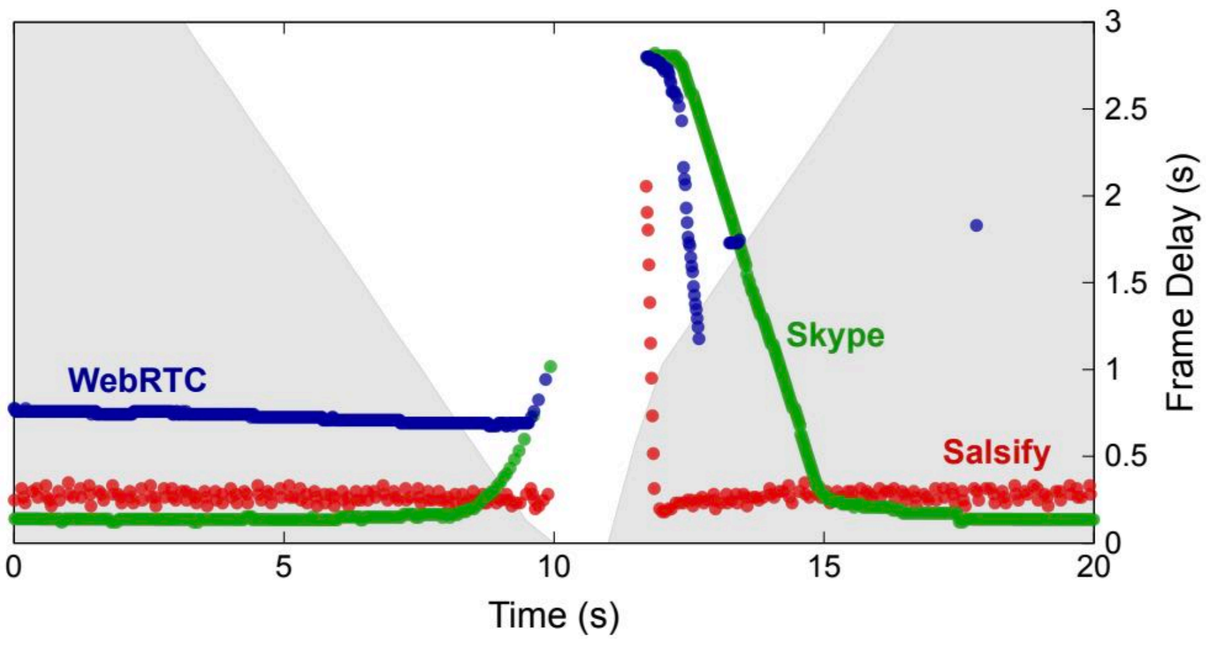

**Transport picks "none"**
**(Encoder will proceed to compress frame 4 based on state $S_{2B}$)**

[Fouladi et al. 2018]

# Much faster recovery from network changes




**(a)** Throughput




**(b)** Frame delay

**Gray region shows capacity of network:**
**(Simulating an outage at 10 seconds)**