Modeling Video Traffic Sources for RMCAT Evaluations

Our experience with the Mozilla web browser

draft-ietf-rmcat-video-traffic-model

Xiaoqing Zhu, Sergio Mena,
and Zaheduzzaman Sarker

Mozilla for transient behavior

Outline

- Why we did it
- How we did it
- Why it’s better
- Updated results
- Plots
- What’s next

Motivation: Why We Did It

- IETF95 (Buenos Aires)
  `http://www.ietf.org/proceedings/95/slides/slides-95-rmcat-3.pdf`
- Results for statistics model's transient behavior using:
  - VideoLAN's x264 as codec
  - non-standard settings (e.g., only 1 initial I-frame)
  - animation sequences → scene cuts
- Two feedback items to address:
  - Animation: not representative of video conferencing
  - x264 not widely used for live encoding
- We addressed those items:
  - Produced conferencing-like video sequence
  - Randell Jesup volunteered to help us out
    - Use codecs shipped with Mozilla (H264/VP8)
    - Thanks Randell! :-)

How We Did It

Part I. Live Video Capture

- Used Cisco Telepresence unit
- Captured video sequence:
  - Over 6 min long
  - 1080p, 30 fps
  - Conference-like content (3 participants)
  - "Light" compression: 4 Mbps
- Converted to uncompressed (yuv420p) format
  - 720p → file size: 16 GB
  - 1080p → file size: 37 GB

How We Did It

Part II. Modified Mozilla Browser

- Limited changes to Mozilla source code
- `VideoConduit.cpp`, `bitrate_controller_impl.cc`:
  - Disregard: Bandwidth data from congestion controller
  - Use instead: Fixed (hardcoded) pattern
  - Log frame sizes
- `MediaEngineDefault.cpp`:
  - Read frames (yuv420p) from a file

How We Did It

Part II. Modified Mozilla Browser

- Test file:
  - *.html* using webrtc
  - One-way conference (same host)
- Hardware: MacBook Pro (v11,4; fairly recent):
  - MBP Retina, Mid 2015
  - 16GB RAM, 2.2 GHz CPU (4 cores, 8 threads), SSD hard disk
  - OSX version: El Capitan (10.11.6)
- Workload for 1080p sequence:
  - CPU: ~35% of total usage
  - RAM: rss ~450 MB, virtual size ~5 G
  - Hard disk: able to read file @ 30 fps (logs show no lag)
  - ➢ Conclusion: **no overload**

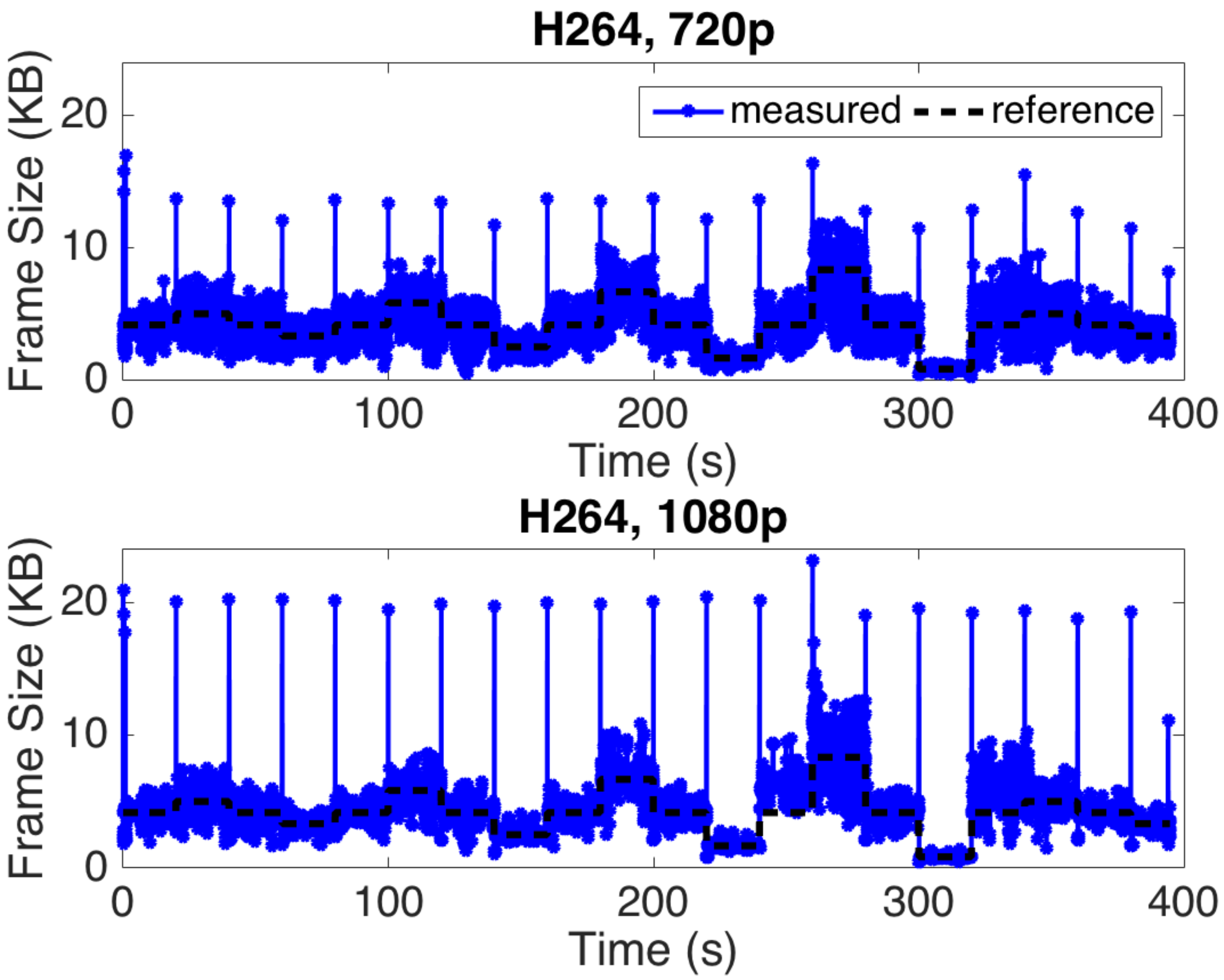
Why it's better

We addressed valid concerns from *rmcat* group:

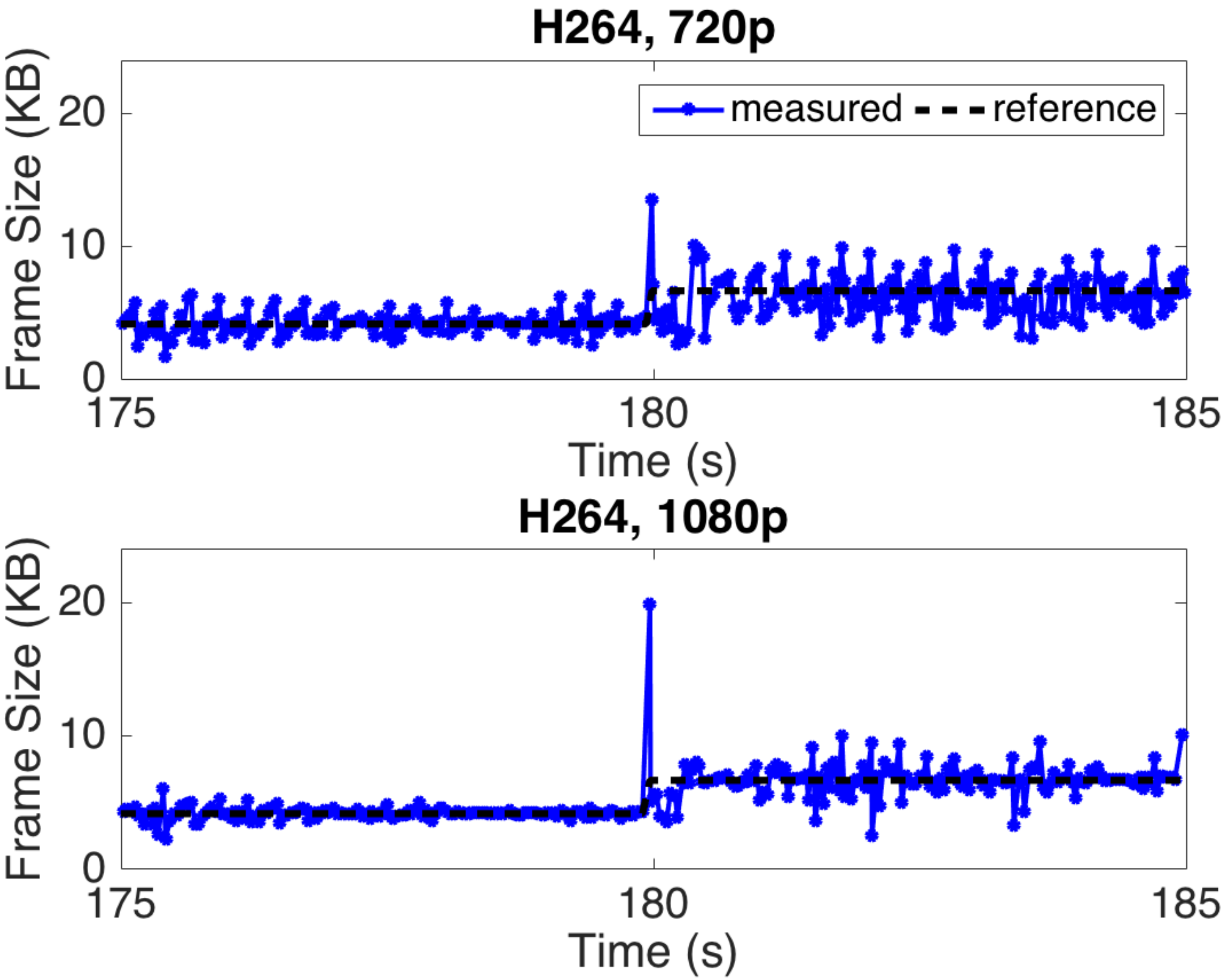
- Teleconference-like contents
- Mozilla is a widely used browser
  - We used "default" settings
  - We tried two "default" codecs (H264 and VP8)
  - **Representative use** of the browser
- Video sequence from file, rather than live camera
  - Encode right contents
  - Tests are easier to run
  - **Repeatable results** (e.g., across resolutions)

UPDATED RESULTS

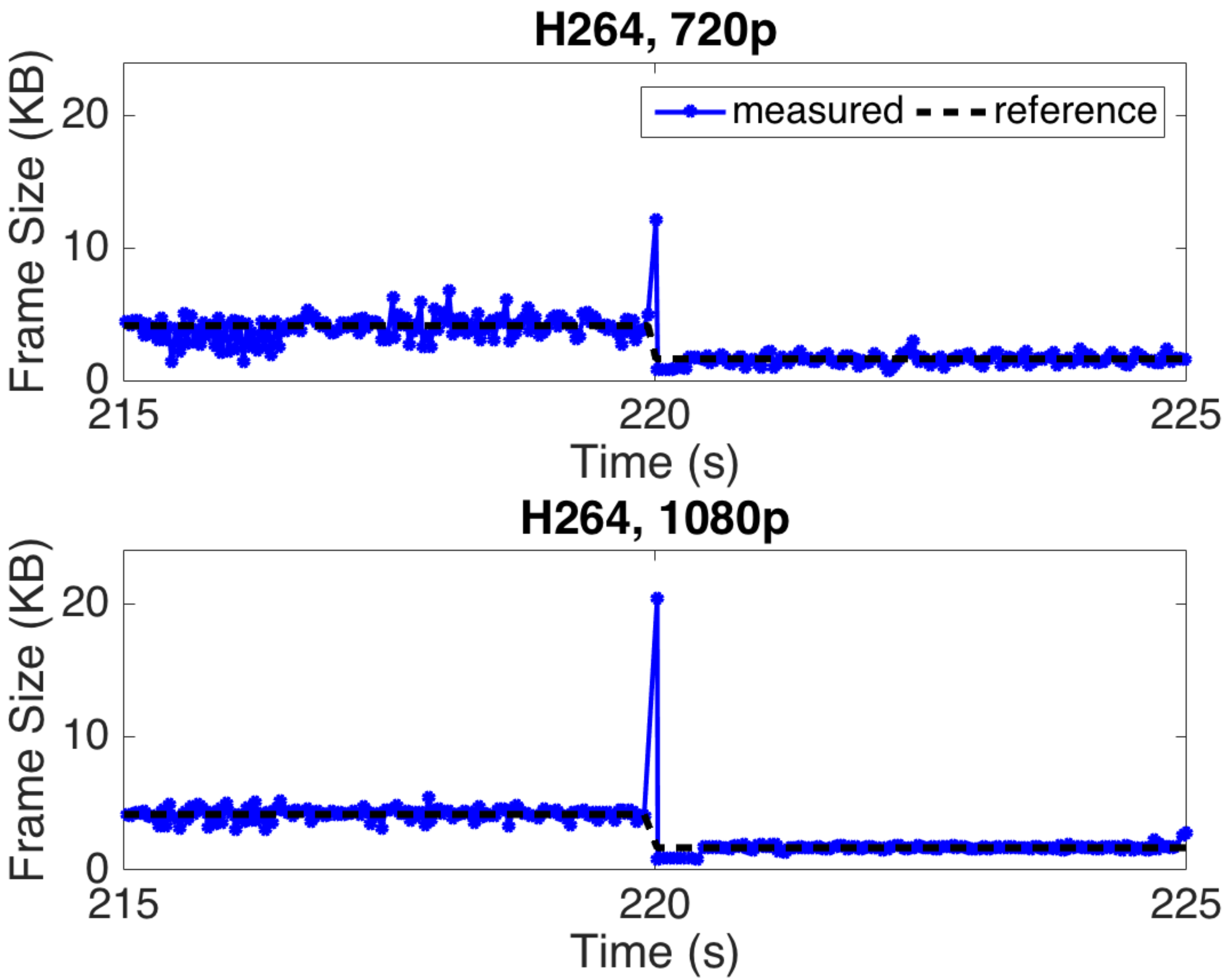
Time-Varying Target Rate with H264: Encoded Frame Sizes



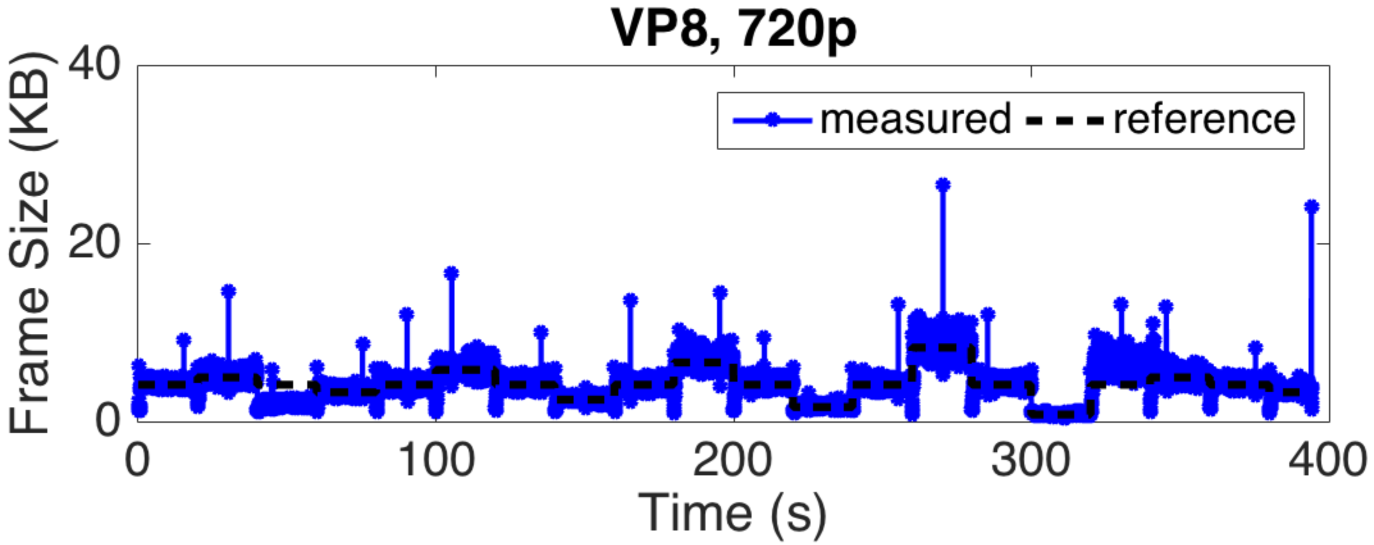
Zoom-In View of Frame Size Trace: 1Mbps -> 1.6Mbps



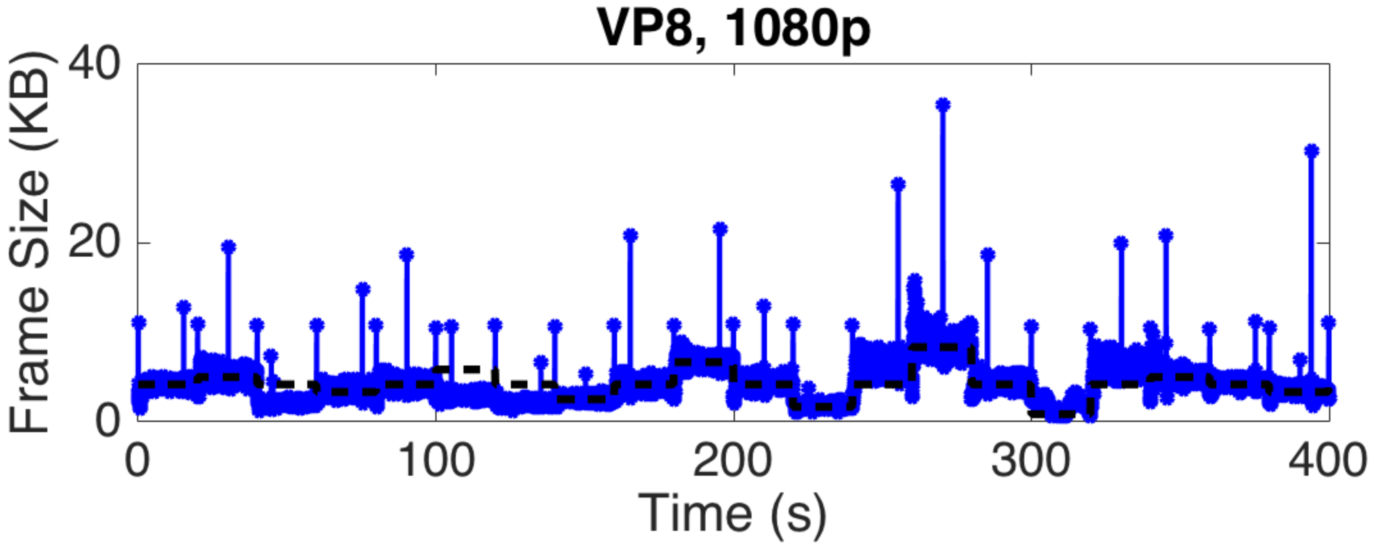
Zoom-In View of Frame Size Trace: 1Mbps -> 400Kbps



Time-Varying Target Rate with VP8: Encoded Frame Sizes

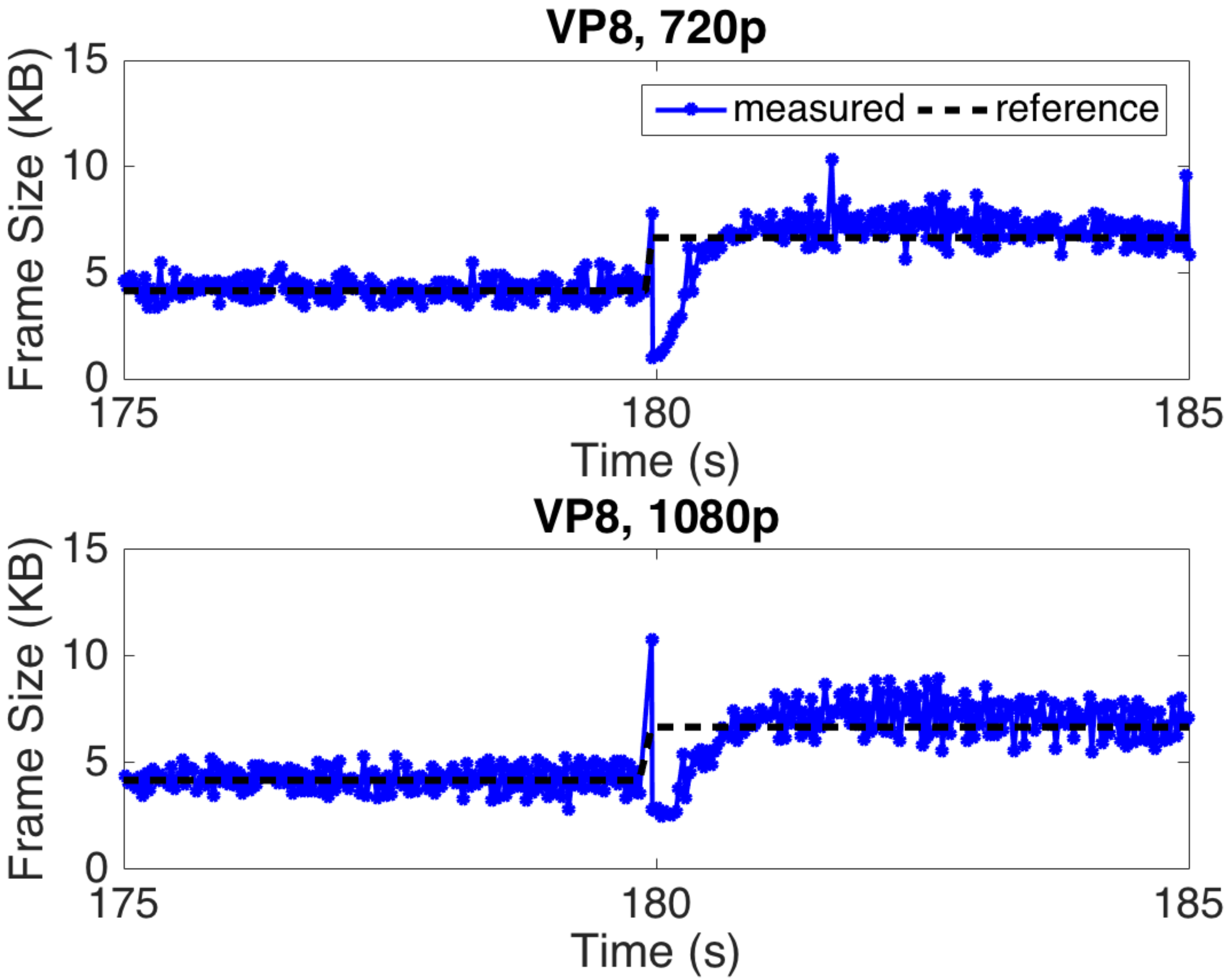


VP8, 720p

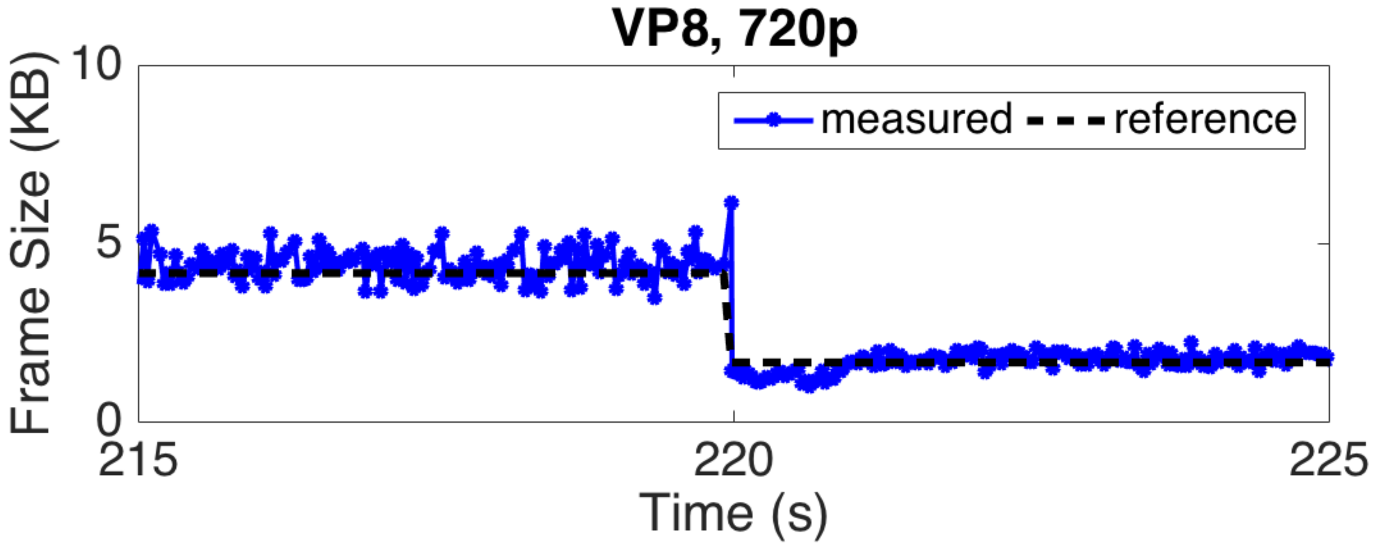


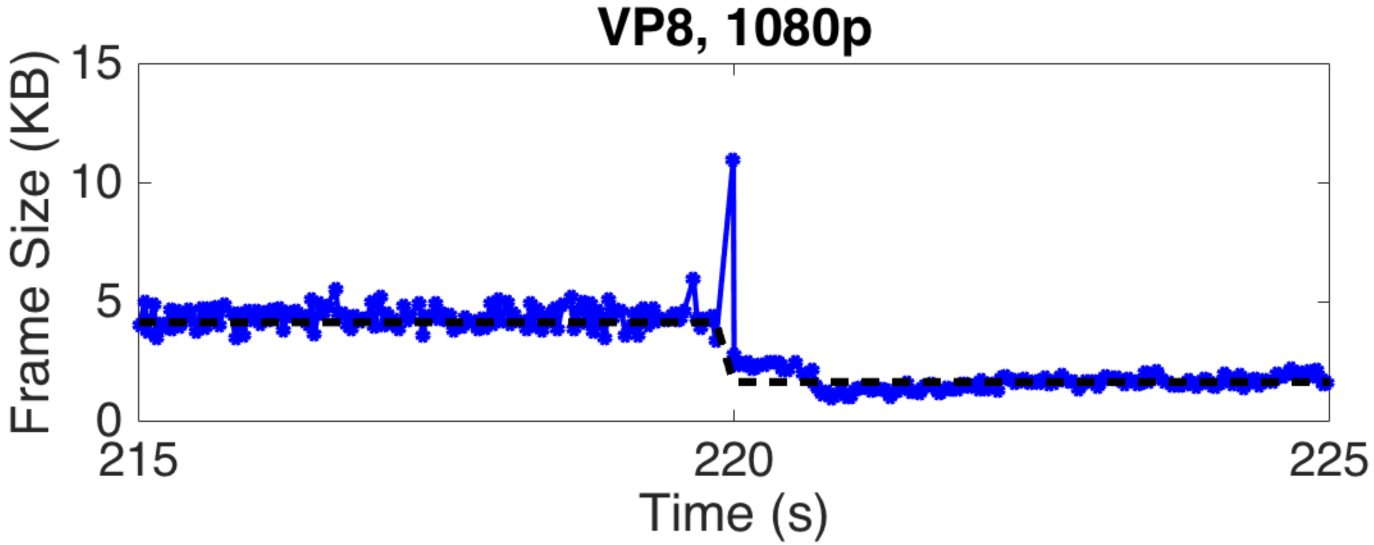
VP8, 1080p

Zoom-In View of Frame Size Trace: 1Mbps -> 1.6Mbps

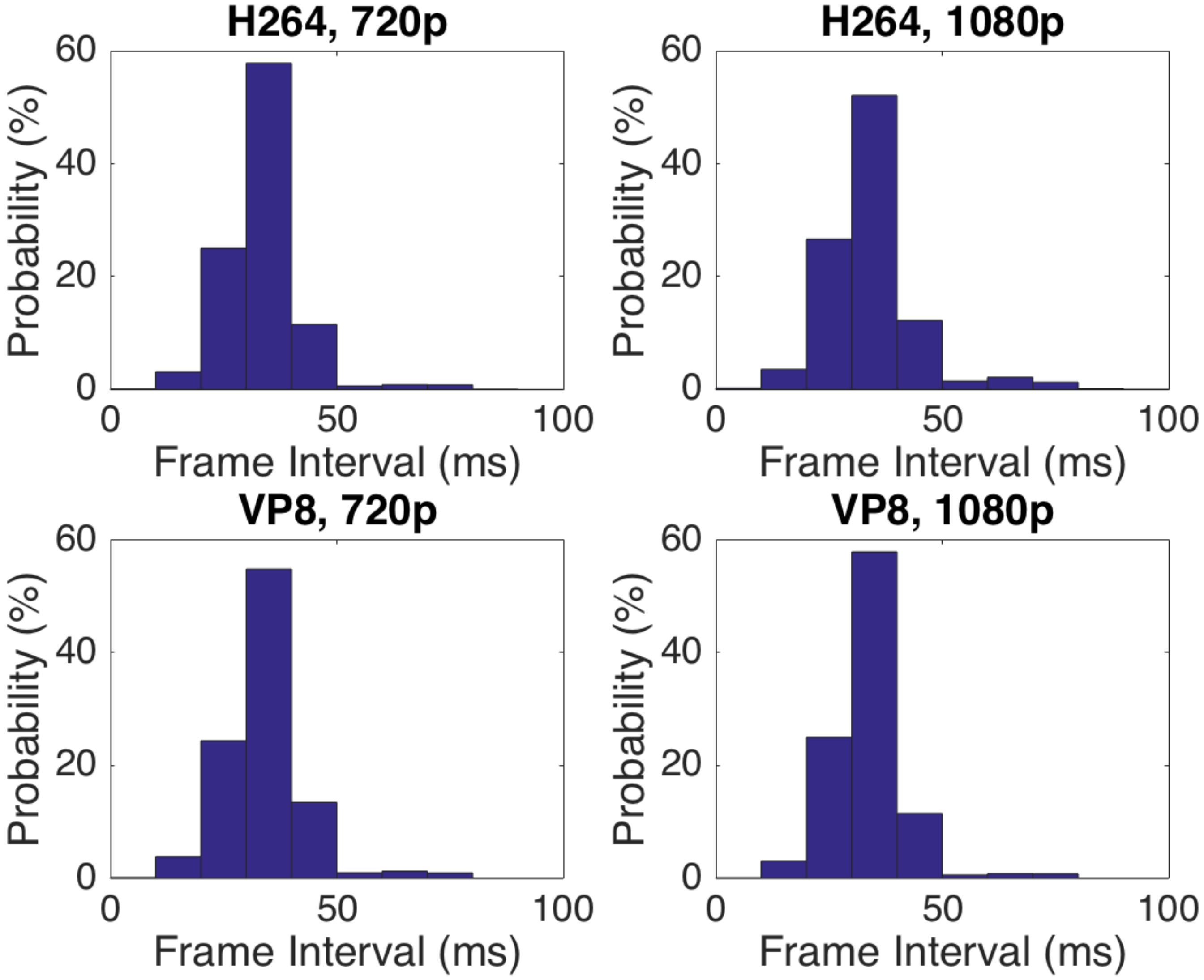


Zoom-In View of Frame Size Trace: 1Mbps -> 400Kbps





Frame Interval Distribution



Summary of Observations

- Fluctuation of frame interval around around the reference value
- The VP8 encoder occasionally cannot meet the target output rate (e.g., at t=40-60s for target rate of 1Mbps)
- Presence of big transition frames both for rate upshifting and downshifting
- A few smaller frames followed by big transition frames; transition times different for H264 and VP8(*)

(*) Following default settings for codec configurations, results not intended as codec performance comparison.

WHAT’S NEXT?

Next Steps

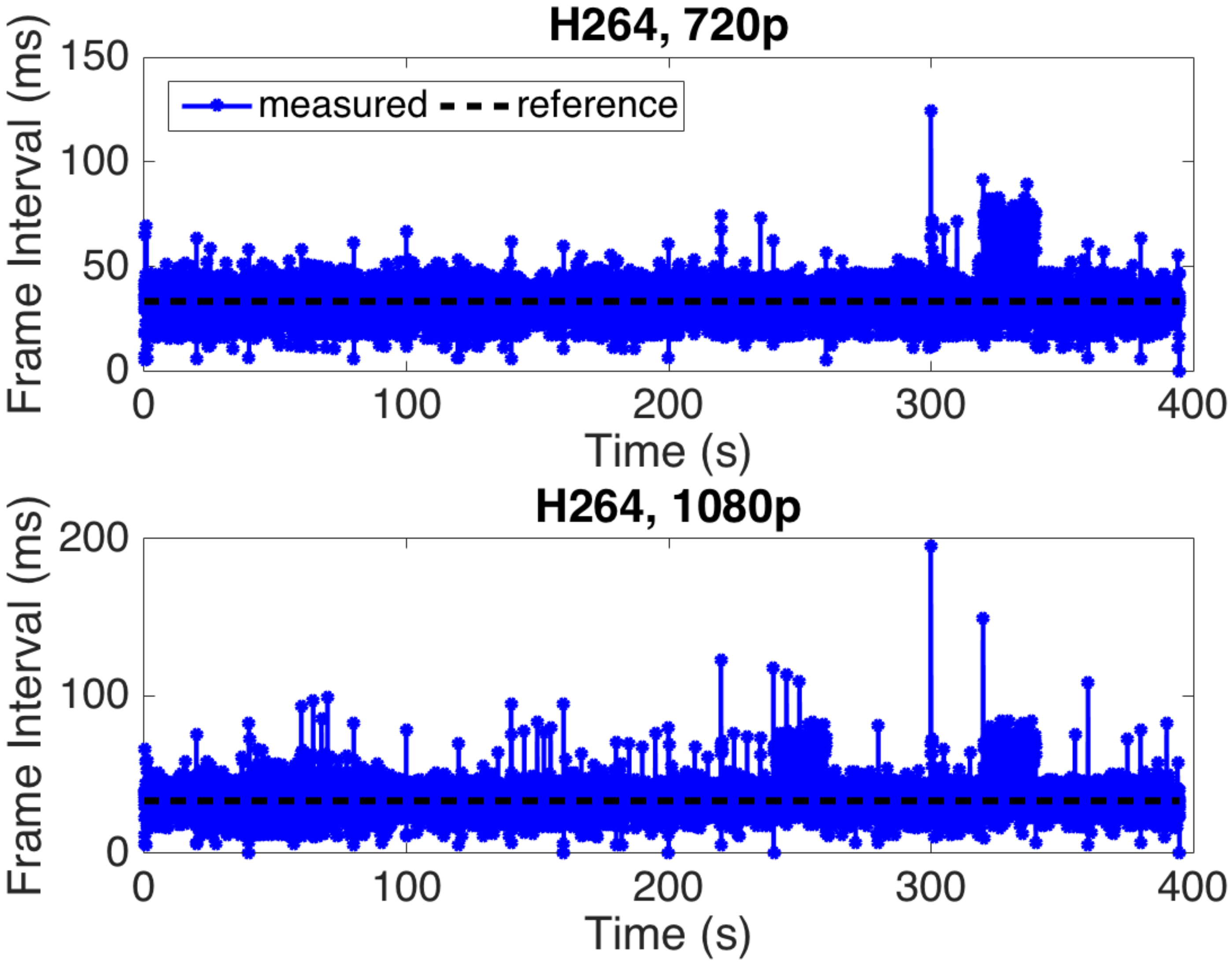
- Updates to the draft:
  - Section 5. Propose concrete values to the statistical model with our results
  - Section 7. Adjust the steady/transient threshold according to our results
- *Syncodecs*:
  - Implement hybrid model
  - Adapt statistics-based codec with our results
  - Update steady-state traces with output from Mozilla browser

Thank you

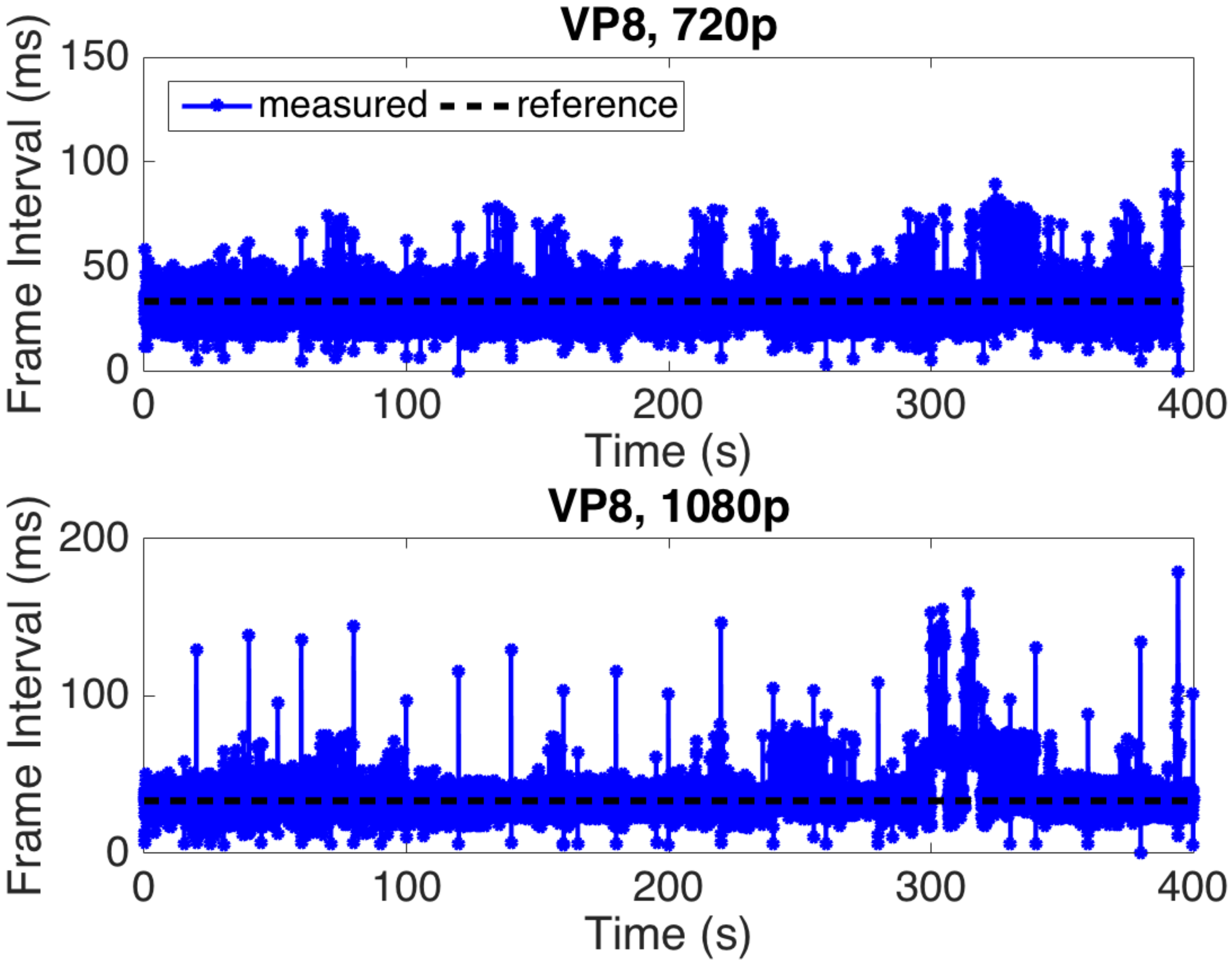
Questions?

Sample Screenshot of Video

Time-Varying Target Rate with H264: Frame Intervals



Time-Varying Target Rate with H264: Frame Intervals


VP8, 720p
Frame Interval (ms)
measured
reference
Time (s)
VP8, 1080p
Frame Interval (ms)
Time (s)