Karlsruhe Institute of Technology

Enhancing Real-time Data Monitoring Display through Video Streaming Technology

Nicholas Tan Jerome, Suren Chilingaryan, and Andreas Kopmann


Research Center

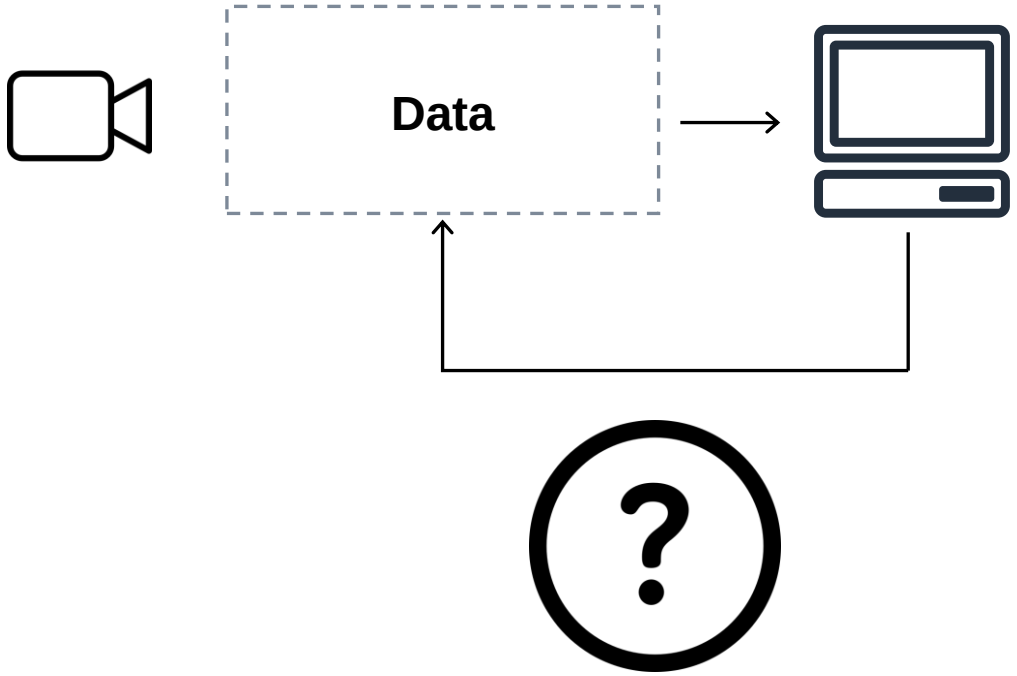

KATRIN Experiment


University

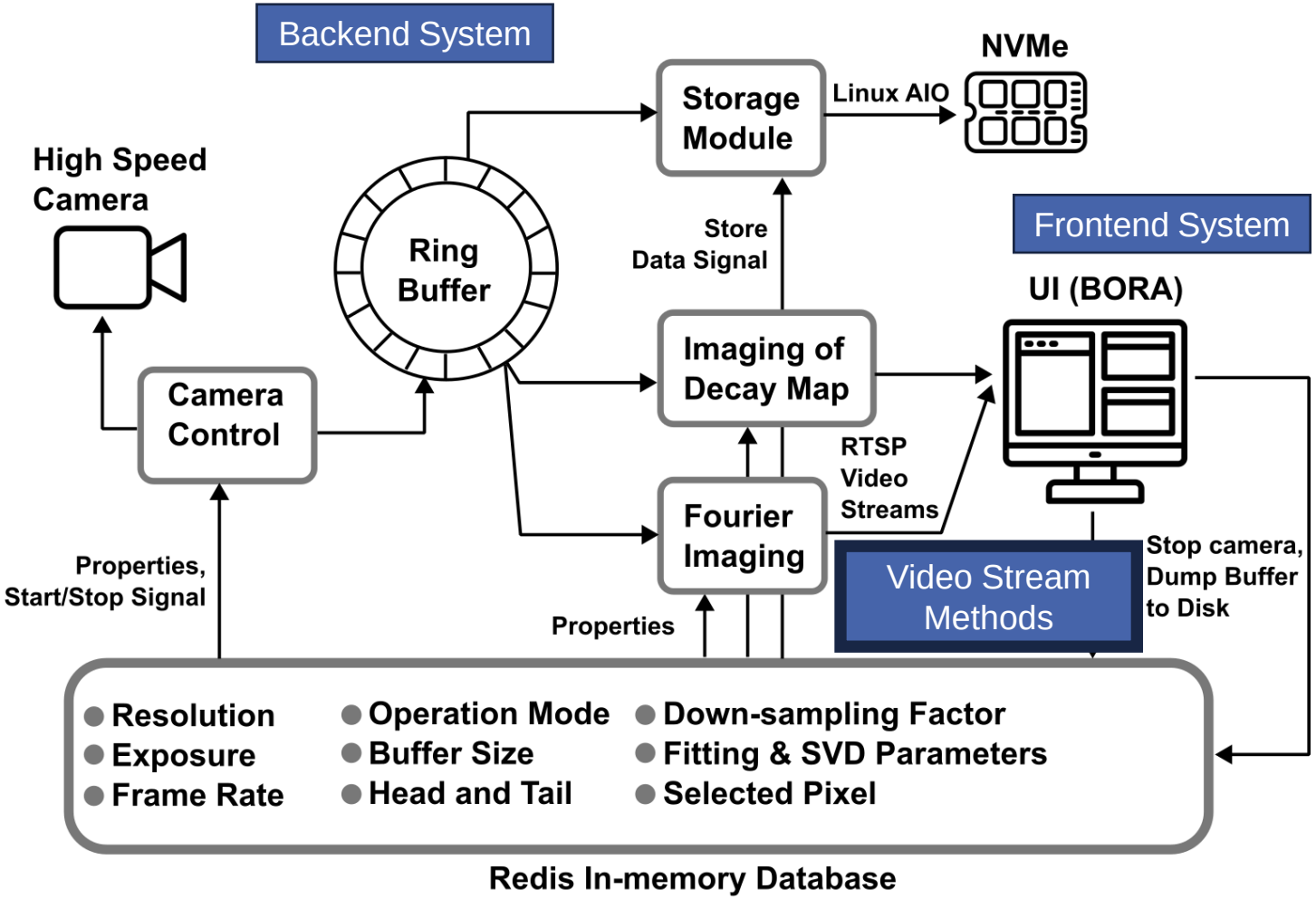
Motivation: High-speed Imaging

- Commercial high frame rate cameras are becoming affordable
- CCD and CMOS systems deliver an upper frame rate near 5000 fps, 10 GB/s
- Complicates data storage
- Lack of studies and solutions in process monitoring and quality control

Concept to deal with data at such a rate and size

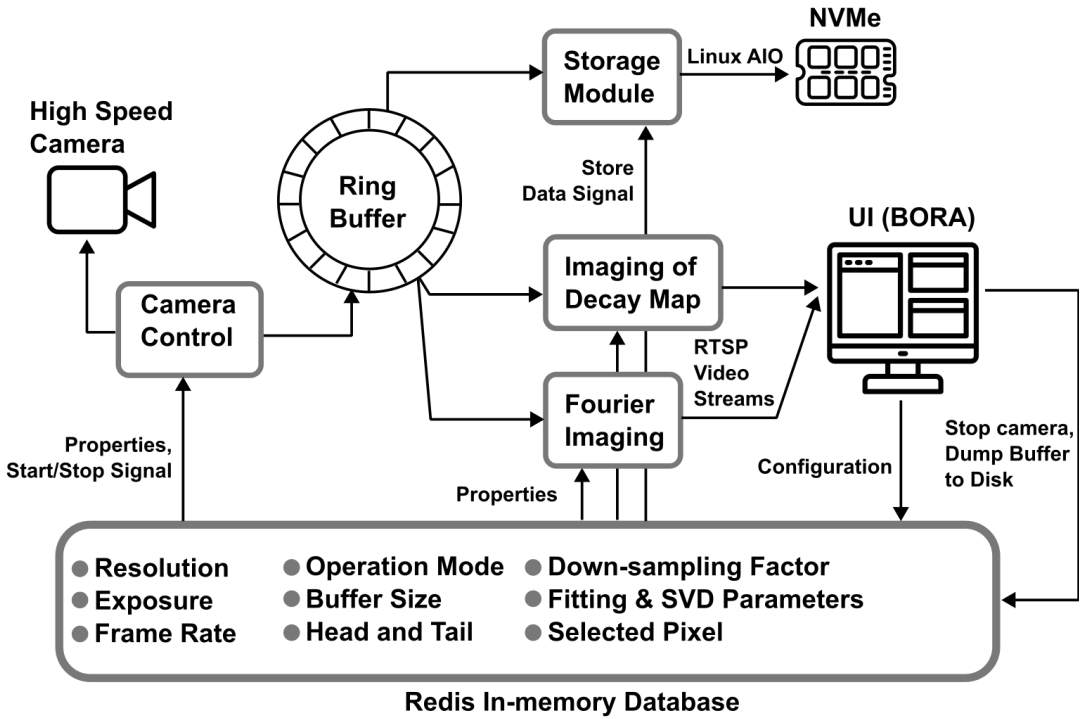


Our Idea: Encode Data in Video Streams



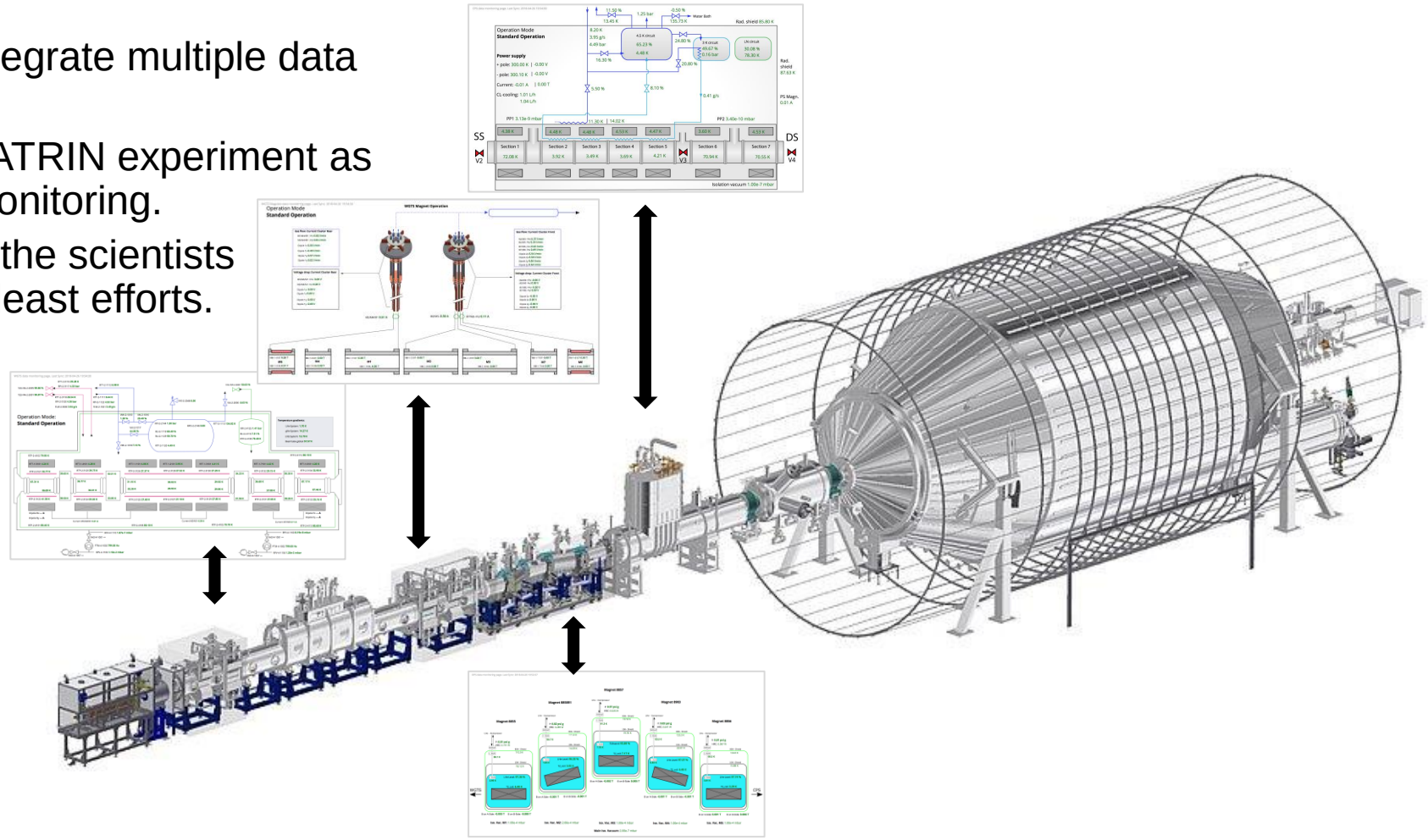
Test Setup: Backend Ring Buffer

- Simple mode: the camera captures a set number of frames and stores them in a ring buffer
- Trigger mode: the ring buffer is constantly receiving and storing the acquired data in a FIFO mode. Upon receiving a trigger signal, the data in the buffer can be isolated and processed.
- Compression mode: During compression mode, only processed data is saved to the disk
- Distribution mode: the ring buffer will be duplicated across multiple processing computers, each responsible for various data processing and storage tasks.

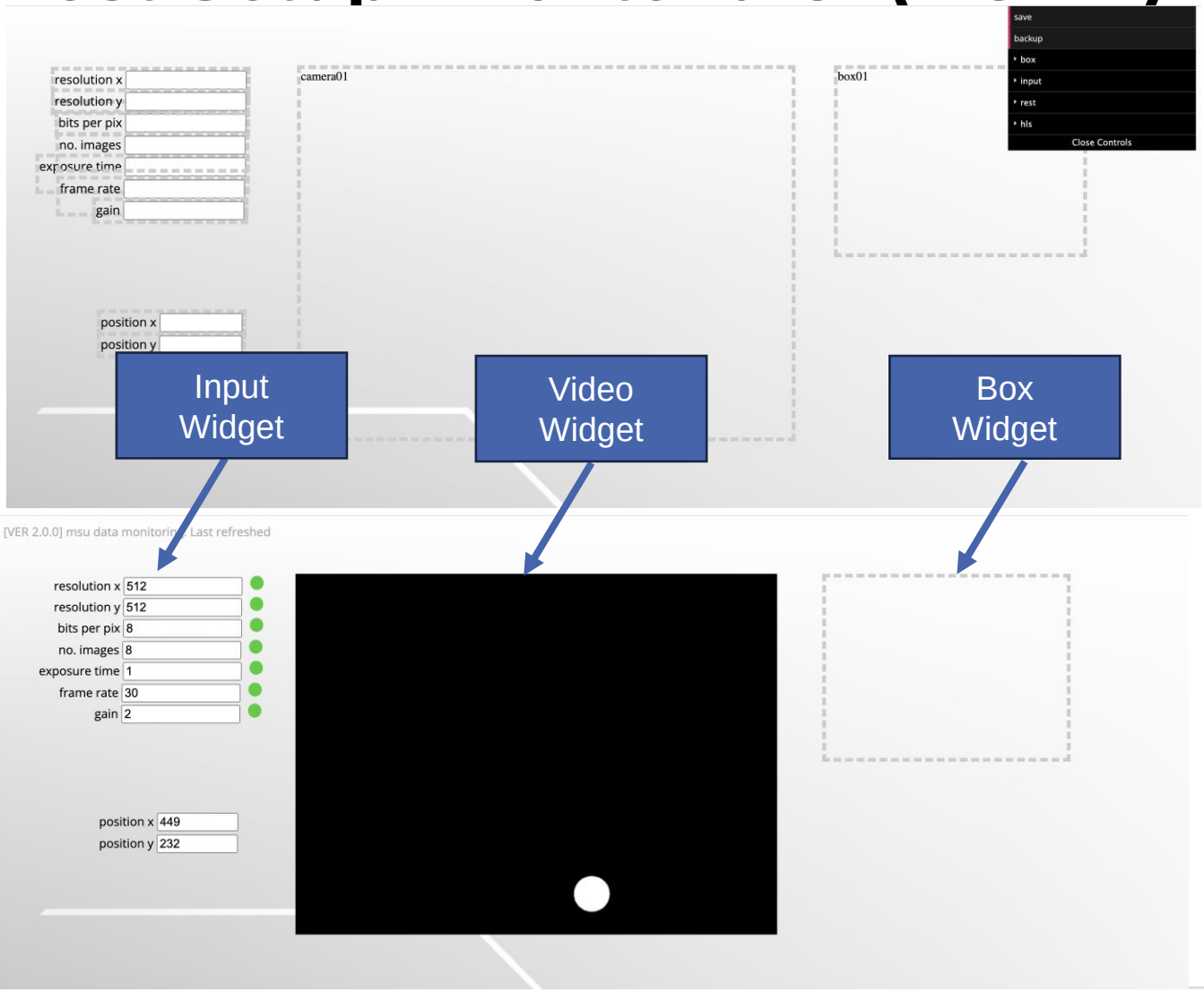


Test Setup: Frontend UI (BORA)

- Lightweight UI framework that integrate multiple data sources into a single view.
- Originally motivated within the KATRIN experiment as the visual panels for operation monitoring.
- Based on the concept that helps the scientists to build monitoring displays with least efforts.
- Notebook and Redis integration
- https://github.com/kit-ipe/bora



Test Setup: Frontend UI (BORA)



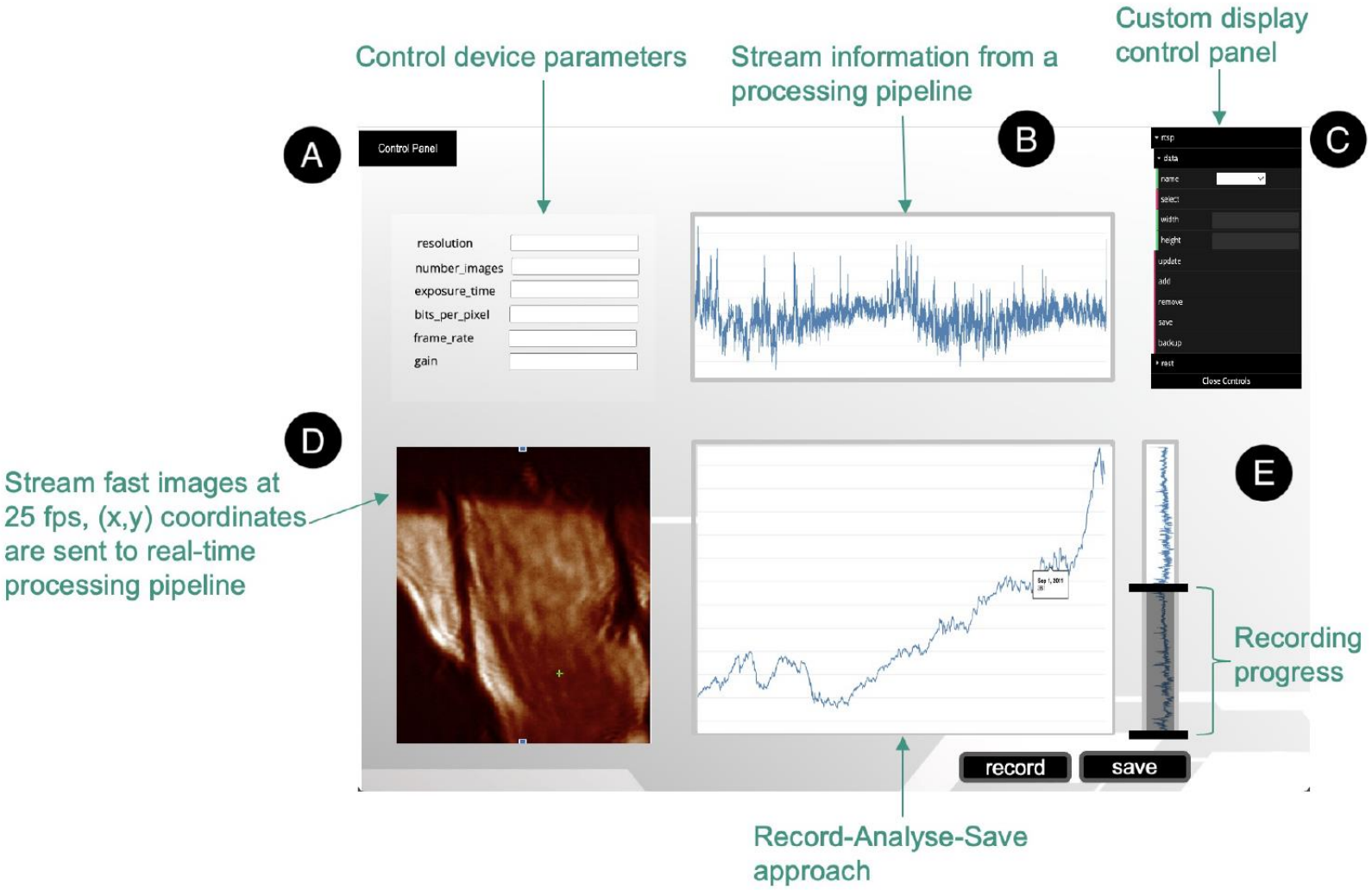
Designer

/designer

Data Display

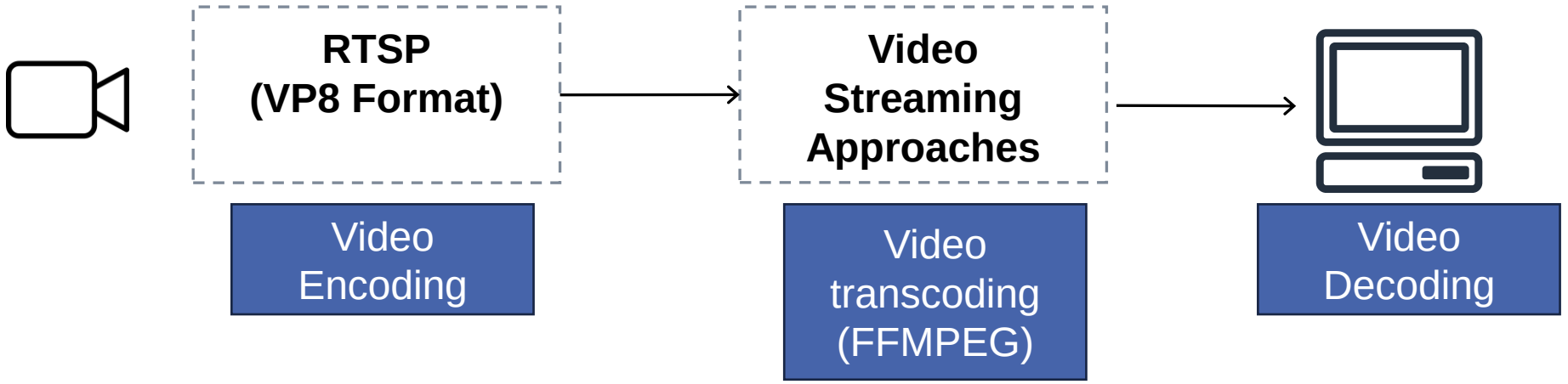
/status

Test Setup: Frontend UI (BORA)



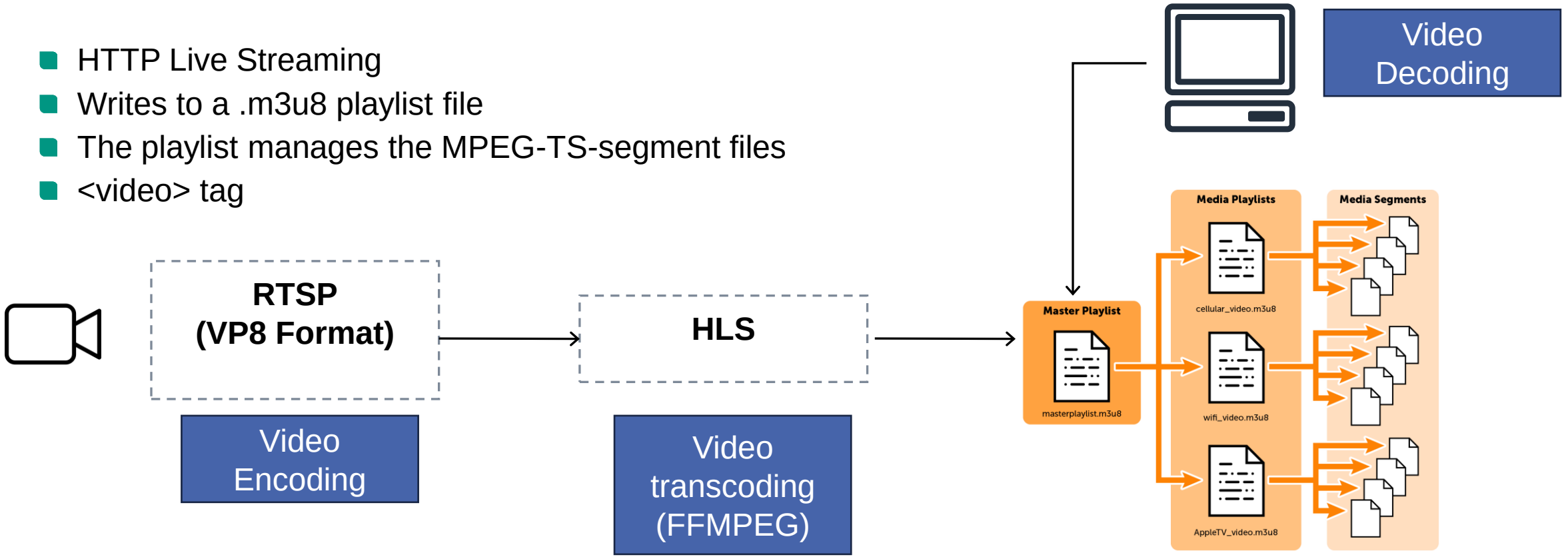
Bringing Video Streams to the Web

- Web browser does not support many video formats
- Evaluate technologies to bring video streams to browser with low-latencies:
  - HLS (HTTP Live Streaming)
  - MPEG-Websocket
  - WebRTC ((Web Real-Time Communication)



HLS

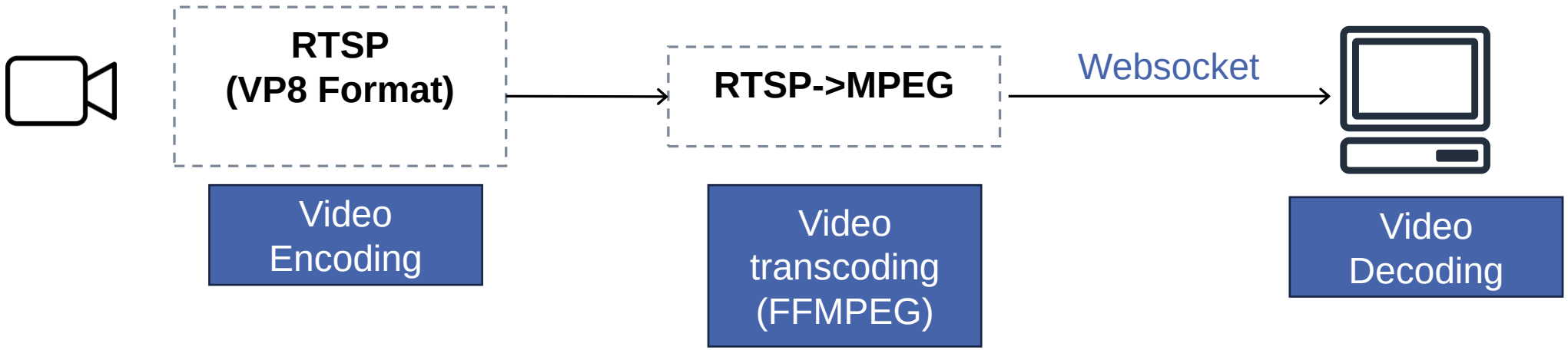
- HTTP Live Streaming
- Writes to a .m3u8 playlist file
- The playlist manages the MPEG-TS-segment files
- <video> tag




```
ffmpeg -i "rtsp://127.0.0.1:8554/test" \
       -hls_time 3 \
       -hls_wrap 10 \
       "stream/streaming.m3u8"
```

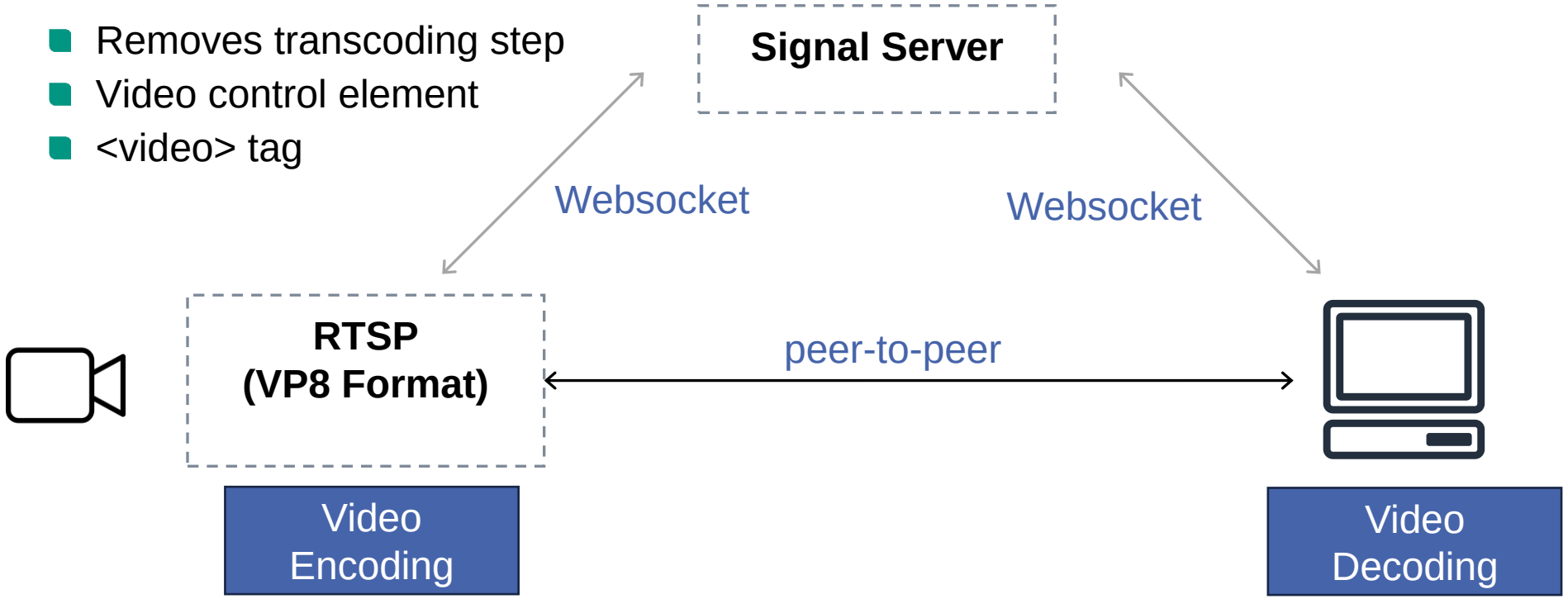
MPEG-Websocket

- FFMPEG to convert RTSP to MPEG-TS-Stream
- JS-Web-assembly MPEG1 Video Decoder
- <canvas> tag --- WebGL & Canvas 2D Renderer



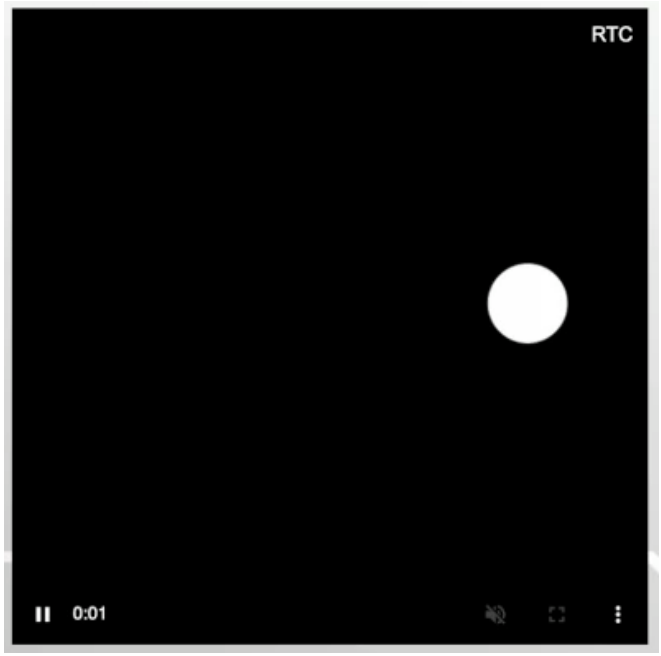
WebRTC

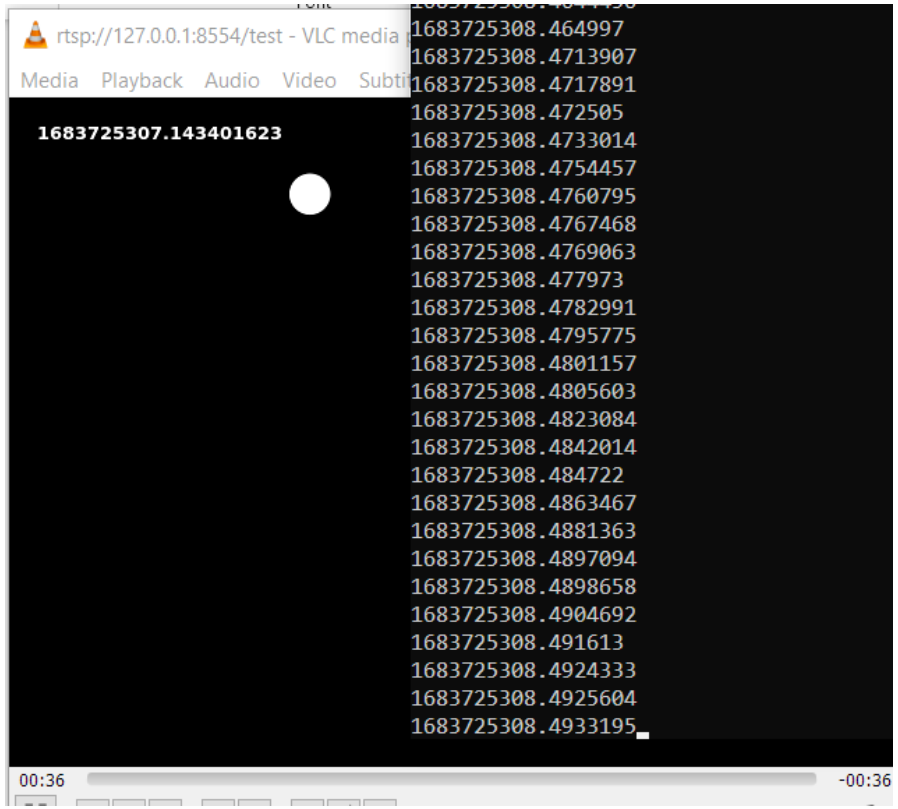
- Direct communication like P2P
- Removes transcoding step
- Video control element
- <video> tag



Evaluation

- We created a C++ prototype RTSP source using the GST template library.
- We look into transmission latency and start-up delay
- Codes are available:
  https://github.com/kit-ipe/bora/tree/master/misc/rtsp_streaming





Evaluation

- Zero start-up delay for HLS doesn’t mean good performance, rather the file is readily processed
- Higher transmission latencies for HLS and MPEG-WS show the overhead from video transcoding, and video decoding of the MPEG-TS format.
- Decoding MPEG-TS involves parsing the container and decoding the compressed video format
- Start-up delay by the WebRTC shows the overhead of the signaling procedures and the low transmission latency is the video decoding of the VP8 format.

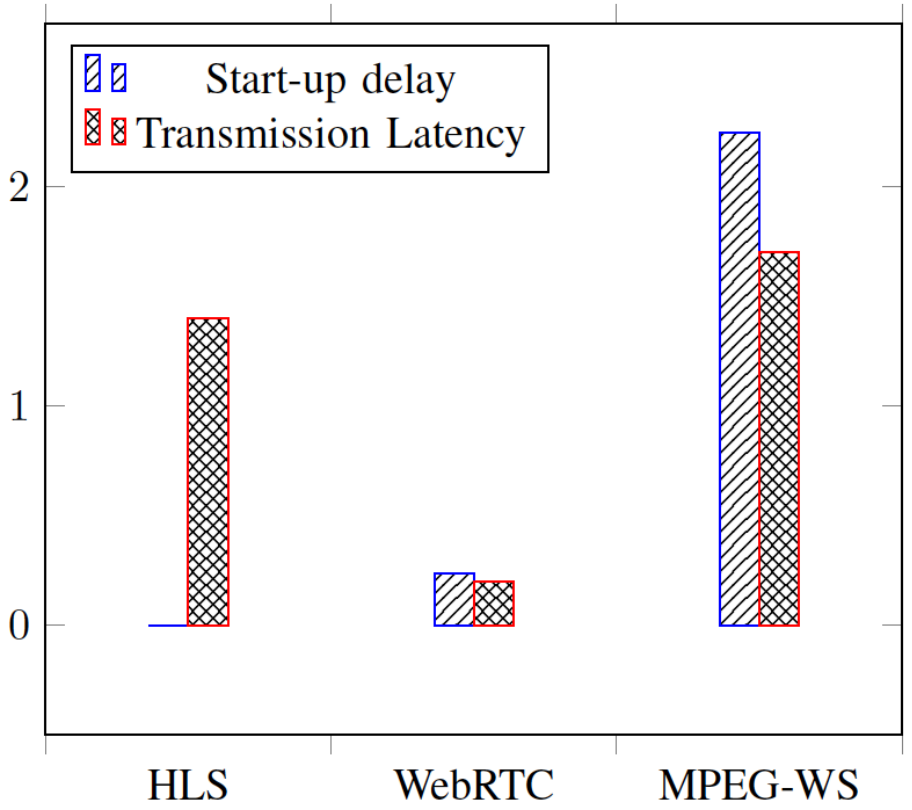
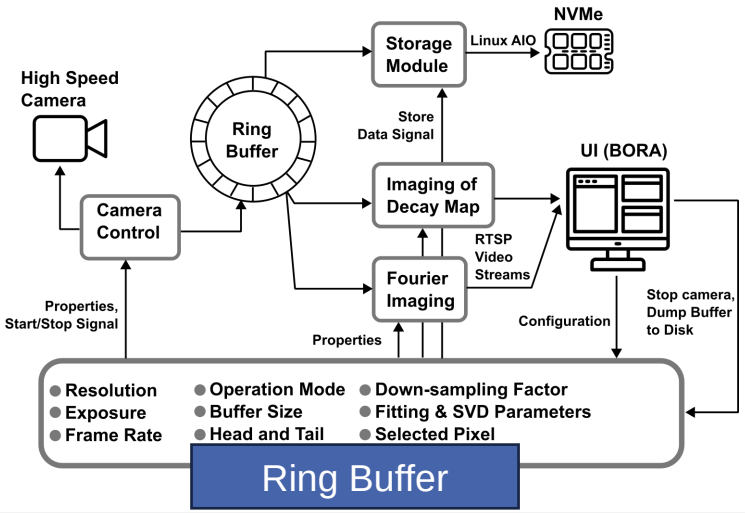
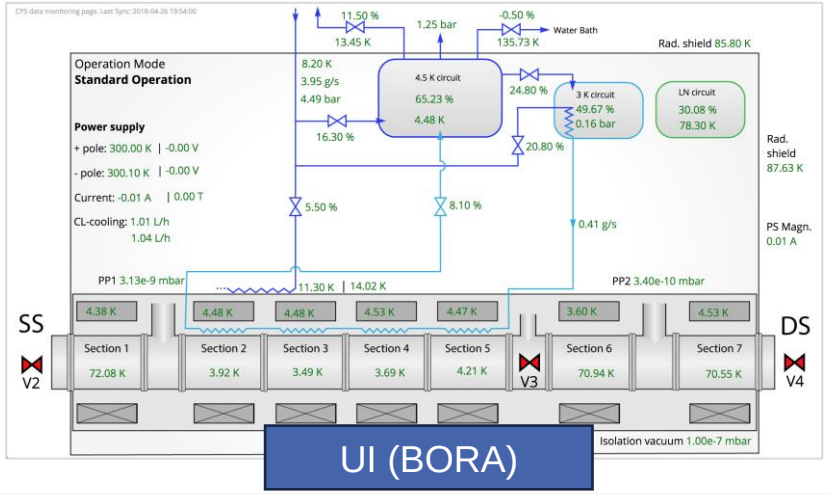


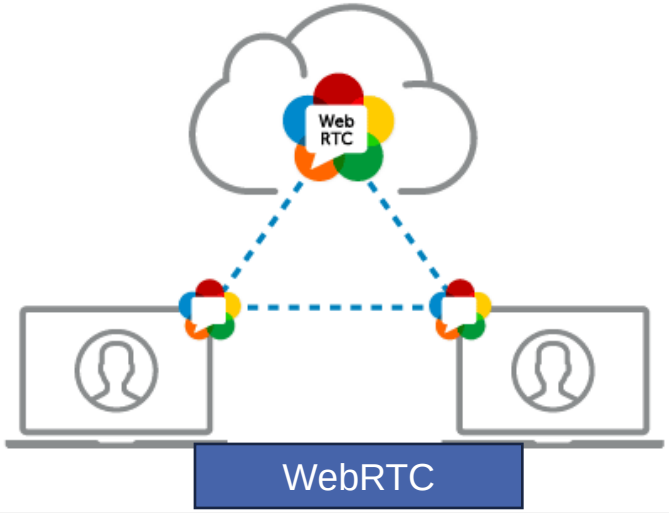
Fig. 3. Comparison of the start-up delay and transmission latency between the HLS approach, the MPEG-Websocket approach, and the WebRTC approach.

Conclusion

- Examines real-time video streaming methods for monitoring high frame rates through web browsers as the client medium
  - HLS, MPEG-Websocket, WebRTC
- The evaluation assessed the start-up delay and transmission latency of HLS, MPEG-Websocket, and WebRTC methods.
- The suggested system architecture, which integrates a shared memory ring buffer framework and the BORA frontend framework, offers a foundation for processing and displaying high-speed imaging data.







Thank You