# Synergized QoE-centric Streaming for Telerobotics

Madhurima Ganguly, Abhijan Bhattacharyya, Ashis Sau, Suraj Mahato
*TCS Research, Tata Consultancy Services*
Kolkata, India
{ganguly.madhurima, abhijan.bhattacharyya, ashis.sau, surajkumar.mahato}@tcs.com

***Abstract*—We propose an end-to-end video streaming solution for telerobotics aimed to improve the user experience despite variations in the underlying network. It uses a synergized smart spatio-temporal bitrate adaptation technique, tightly coupled with an adaptive exchange protocol semantics and smart open-loop jitter buffer adaptation. It shifts paradigm from existing GOP-based encoding and bitrate adaptation. Efficacy is proven by comparing performance against a similar telerobot system using WebRTC. Comparisons are done in terms of both objective QoE metrics, and subjective user experience study through live remote operation on the robot over the different long-haul Internet paths as well as for practical last mile channel degradation.**

***Keywords—video streaming, QoE, QoS, telerobotics.***

## I. Introduction, Motivation and Gap Analysis

The quality of experience (QoE) for a human operator, operating a remote telerobot over the public Internet, depends heavily on the live video feed received from the robot's camera. Such applications are very sensitive to poor visual quality, overshoot in end-to-end motion-to-photon or scene-to-screen delay and freezing of the received video as those phenomena drastically reduce the confidence of the operator on sanity of inference on the remote affairs and confidence on the outcome of the commands being delivered to the robot.

Video streaming application is a combination of *encoding* at acquisition-end, *packetization & exchange semantics* for application data segments in flight*, jitter-buffer management & decoding* at the rendering-end. The low-level network quality of service (QoS), which is beyond the control of the end-applications, largely affects performance of such systems. But, if the application-level QoE is modelled around judicious choice of techniques for each component of the collective process for the end-to-end stream, then that may improve the *perceived QoS* despite low-level fluctuations. This in turn affects the QoE of the entire system [1]. So, contextual application-level adaptation and loss-management techniques have been proposed [2][3][4] on reliable (HTTP on TCP) or best-effort (RTP on UDP) transports. However, all such techniques try to adapt to network degradation through adaptive bitrate (ABR) techniques which undermine the user experience [5]. Such systems were originally designed for video-on-demand applications and later adapted for teleconferencing. But,they lack the desired performance for telerobotics kind of applications under challenged network conditions [6][7]. The reasons are: *(1)* The backward error correction mechanism (BEC) or application layer FEC (AL-FEC) both undermine real-time performance while trying to regain PSNR under lossy conditions [5]. *(2)* All predominant techniques use encoders with group of pictures (GOP) structure which suffer from freezing under loss or corruption of I-frame. These are inherently slow reactive to network context. Even if network QoS is improved, inability to synch to next I-frame causes the rendering unit to freeze the video [6]. [8] proposes tuning of a hardware-accelerated H.264 encoder for latency reduction but such encoder has high dependency on the available hardware. [8] also inherits all the shortcomings of GOP based encoding. Hence it fails to stand out as a democratized approach.

Historically, video encoding and streaming protocols have been two independent lines of development causing a lack of synergy between them [9]. Though [9] tries to synergize the encoding with transport through a frame-by-frame modification deep inside VP8/VP9 structure, the modifications are intended to satisfy the applications like video on demand and not tested for telerobotics-like applications. [9] does not pose any alternative to conventional bitrate adaptation scheme that undermines QoE. Though [9] claims to perform better than WebRTC [10], [11] questions some of the performance claims. [12] has highlighted the supremacy of JPEG for latency sensitive applications. But simple JPEG is bandwidth inefficient. [13] attempted to improve BW efficiency of MJPEG through background subtraction while proving the bitrate at par with H.264 in limited cases. But it works only on grayscale images and does not take care of the adversities of transmission over Internet backhaul. Especially, it is silent on protecting the frame-to-frame delta in lossy conditions. Also, it encodes the deltas as full JPEG frame which loses the expected BW savings. Literatures like [14] propose QoE-aware ABR scheme on GOP based encoders that adaptively uses the encoding layers. That is inherently not suitable for systems like telerobotics for the reasons explained above. Such literatures are completely silent on the underlying protocol and are not tested in real Internet-backhaul. At present, WebRTC is used in practical telerobotic products and lab prototypes [15][16]. Even widely used services like Amazon Kinesis also uses WebRTC [17]. The default browser implementation uses VP8/VP9 codecs and selective retransmissions on RTP [18]. It also inherits the problems we referred. Some of the impracticalities of such retransmissions and recoveries, especially for I-frames, are highlighted in [19]. [20] provided a base-line partial approach to address the problems stated earlier. Instead of GOP based encoding and conventional protocols it used a more efficient frame-by-frame encoding using background subtraction on JPEG frames than [13]. It also proposed an adaptive packetization and protocol semantics that ensures to protect the *delta in flight* in between two frames in a tightly coupled manner. But it has no bitrate

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

adaptation or jitter buffer management. It shows an improved performance compared to WebRTC under last-mile degradation for stored frame sequences of much smaller resolution than even standard 640 X 480 resolution and are tested in local set up with negligible temporal variation. When we tried [20] in a real peer-to-peer (P2P) deployment over long-haul live streaming on the public Internet with 640X480 resolution, the system *could not maintain even 5fps framerate* and failed to cope with delay-variations in the channel.

Starting from the baseline of [20][21], this paper proposes a holistic frugal end-to-end solution, typically designed for telerobotics kind of applications. We propose a smart QoE aware *Spatio-Temporal Bitrate adaptation theorized around a QoE model tailored around the concerned class of applications.* ***Firstly***, we introduce a paradigm shift from GOP based encoding/decoding by proposing a *frame-by-frame* temporal encoding approach inspired by [20]. ***Secondly***, we introduce novel ***median-based matrix scaling*** & ***foveal Breathing*** techniques to achieve *Fixed Foveated rendering* enabled spatial bitrate adaptation. ***Thirdly,*** the protocol state machine of [20] has been modified for making it practical in the context of time-varying long-haul channels. It uses the non-blocking semantics of Constrained Application Protocol (CoAP) [22], along with the no-response option [23] and introduces an intelligent time-out and retransmission adaptation option that tightly couples the encoder states with not only the lossy-ness of the channel, but also the delay variations. ***Lastly,*** on the reception side, the protocol state machine is closely tied to a smart jitter-buffer adaptation that tracks the historical reception trend at frame-level and predictively adapts the play-out interval minimizing frame-loss, while maintaining the real-time responsiveness.

We term the system QuOVADIS (Quality Optimized Visual Anatomy-driven Dynamic & Intrinsically adaptive Streaming). It is deployed on a practical telerobot system. Efficacy is proven by comparing performance against a similar telerobot system using WebRTC [10] which is the de facto streaming technology used in telerobotics at present under real challenged conditions with both last-mile channel degradation and also in long-haul communication over the public Internet without any service-quality guarantee. Comparisons are done in terms of both objective visual QoE metrics like SSIM, VMAF, VQM, and by subjective mean opinion score (MOS) derived from user experience study through live remote operation on the robot. The overall system performance also shows improved latency and bandwidth consumption.

Section II briefly describes the theoretical model behind the QoE centric adaptation. Section III describes the entire processing chain and adaptation algorithms at different stages. Section IV describes the development efforts and actual deployments followed by the experiments, results, and analysis. Section V concludes the paper describing ongoing endeavours.

## II. Theorizing the Proposed QoE Management

We consider QoE as a composite function of the visual quality ($Q$), and perceived delay between actuation and viewing the outcome ($d$). The latter is a result of the achievable framerate ($R$) and variation in inter-frame distance ($\delta$). Again, these are related to the amount of bits being pumped into the communication pipe. We also observe that visual quality cannot be sacrificed beyond a certain limit to ensure meaningful engagement by the operator. So, following the study in [24] we first define a *foveal region* in the frame which will have the highest priority in terms of maintaining the quality (Fig. 1). The foveal region in our application is defined as a circle centered in the middle of the frame with a radius $\hat{R}$ for a given percentage $\mu$ such that: $\hat{R} = Max(ImageHeight, ImageWidth).\mu/100$.

So, the *peripheral region* (beyond the foveal boundary) first compromises quality to maintain the bitrate while still maintaining the desired inter-frame distance. Once *peripheral region* quality reaches minimum, then the *foveal region shrinking* happens to a maximum lower limit under degrading channel conditions. Similarly, when channel conditions improve, foveal region is expanded back. We call this phenomenon ***foveal breathing.*** When no further *foveal region expansion* is possible, then *peripheral region* quality enhancement is performed to take advantage of channel condition improvement.

The framerate is intrinsically adapted when the system switches between delta-frames and full frames (because of drastic variation in data to be pumped) limited by a given maximum desired rate. The desired rate may be explicitly changed given how much the system is able to cope up through the bitrate adaptation of the peripheral region and foveal breathing. So, QoE $Q = f(V_{qfov}, V_{qper}, G, R)$, where $V_{qfov}$ = quality within the foveal region, $V_{qper}$ = quality at the peripheral region, $G = \frac{A}{A'}$ ( $A$ = area under foveal region, $A'$ = peripheral area). $G$ controls the foveal breathing. Fig. 2 shows how the system states switch according to the proposed model.


Fig. 1. Illustrating foveation-based realization of intra-frame quality variation.

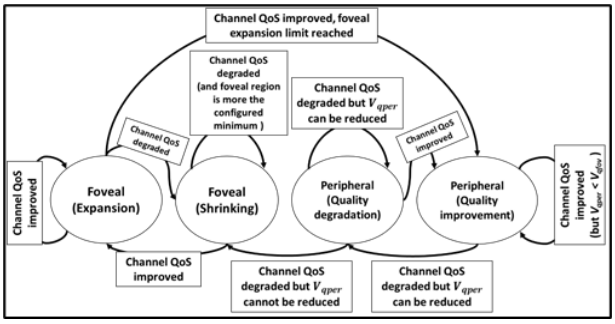


Fig. 2. Overall state-machine for spatial bitrate adaptation at frame level.

## III. QuOVADIS : Architecture and Algorithms

Fig. 3 shows the end-to-end process chain for QuOVADIS. We further explain the different key components and algorithms. Tx also maintains a *periodic timer* to receive feedback from Rx indicating reception QoS similar to that in [20]. Bitrate

adaptation in *temporal domain* is achieved by adaptively switching the encoder state between *basic* mode and *delta* mode based on the value of *Cumulative Error Rate* contained in the Periodic Timer feedback. In *basic* mode the frames transmitted are normal JPEG encoded frames. In *delta* mode it produces delta frames which uses ViBE background subtraction [25] to extract the foreground information from a frame based on scene changes from the previous frame. The proposed background subtraction based temporal encoding over MJPEG allows our system to enjoy the simplicity and low latency of MJPEG based streaming, while being bandwidth efficient. The *frame-by-frame* approach reduces delay in adaptation to channel variability and in synchronization at the receiver. The *temporal encoding* technique involving adaptive switching between *basic* & *delta* mode based on *periodic feedbacks* is inspired from [20].

The Decoder at the receiver in QuOVADIS employs the reassembly, reconstruction & loss concealment algorithms of [20] using the metadata information contained in packet headers shown in Fig. 4. The reconstructed & corrected frames are JPEG decoded and fed to the rendering unit for display.

Section II already described the basic approach for adaptation in *spatial domain*. In *spatial domain* the encoder can either be in *peripheral phase* trying to adapt the bitrate by altering the peripheral quality, or it can be in *foveal phase* where the area of the *foveal region* is adapted by tuning *G*. Encoder initially is in *peripheral phase* where neither the area nor the quality of foveal region is adapted. Therefore, the *foveal region* area is maximum and is encoded at maximum desired quality ($Q_{max}$) in this phase. But bitrate adaptation is accomplished in *peripheral region* using *adaptive median-based matrix scaling (MMS)*. In this novel technique, initially the pixels in each MCU in the *peripheral region* are quantized using normal quantization tables to form quantized matrix $\boldsymbol{M_{QNR}}$, where $\boldsymbol{M_{QNR}}$ is the original Quantized DCT-coefficient matrix for an arbitrary MCU in *peripheral region*. $\boldsymbol{M_{QNR}}$ is then made sparser or denser depending on the channel condition using a scaling factor $\boldsymbol{S}$ to form $\boldsymbol{M_{SQNR}}$ (scaled version of $\boldsymbol{M_{QNR}}$) as per Eqn. 1.

$$M_{SQNR} = ceil\left(\frac{M_{QNR}}{S}\right) X\, S \qquad (1)$$

Thus, the quantized DCT coefficients, which are zero-forced because of scaling down (division by S), are not recovered back, but the non-zero values are restored back, with subtle inexactness compared to the original value, after multiplication by S. The beauty of the algorithm is such that, the scaling is performed in such a back-to-back reciprocal operation, that the receiver need not know S unlike existing region of interest (ROI) based literatures. Because of this restoration operation, S need not be shared with the decoder and the decoder can follow the normal flow of JPEG decoding. The bitrate adaptation is accomplished by choosing the value of S adaptively in response to variations in the channel condition. S is adaptively chosen from the quantized DCT coefficients in $\boldsymbol{M_{QNR}}$. Tx computes the change in predominant error $\Delta P$ by tracking the *periodic feedback*. The encoder determines $\Delta P$ over a period of $k$ as $\Delta P = \left(\frac{P^t - P^{t-k}}{P^{t-k}}\right)$. Instantaneous value of $\Delta P$ drives the spatial bitrate adaptation depending on whether it crosses a threshold. We maintain a sorted list $\boldsymbol{L}$ containing absolute value of all non-zero and non-repeating quantized DCT coefficients for each $\boldsymbol{M_{QNR}}$. S is adaptively chosen from $\boldsymbol{L}$ depending on the value of $\Delta P$. Let, $\Psi$ be the position of median in $\boldsymbol{L}$. Let $\boldsymbol{I^S}$ be index of S in L. At each step, $\boldsymbol{I^S}$ is computed using $I^S = \Psi + \Delta\Psi$, where $\Delta\Psi$ is the *median position increment/decrement factor*. Under degrading channel conditions, i.e. when $\Delta P$ crosses a threshold, $\Delta\Psi$ is incremented using $\Delta\Psi^t = \Delta\Psi^{t-n} + \left(|\Delta\Psi^{t-n}| * ceil(\Delta P)\right)$. Whereas $\Delta\Psi$ is decremented using $\Delta\Psi^t = \Delta\Psi^{t-n} - 1$ under improving channel conditions. This dynamic determination of $\Delta\Psi^t$ is referred to as *Median Position Update* (MPU) algorithm. S is determined as L[$\boldsymbol{I^S}$]. The value of $\Delta\Psi^t$ is not allowed to cross the bound $[\Delta\Psi^{min}, \Delta\Psi^{max}]$. S cannot be increased to the maximum value in $\boldsymbol{M_{QNR}}$. As it would lead to reducing the entire $\boldsymbol{M_{SQNR}}$ to 0 after scaling thereby causing complete loss of information in peripheral MCUs. Under improving channel conditions, if S is reduced to the minimum value in $\boldsymbol{M_{QNR}}$ then no scaling is performed and *peripheral region* is encoded at same maximum quality as *foveal region*. Hence, depending on current value of $\Delta P$ indicating instantaneous channel condition, S is adaptively shifted towards right(higher values) & left(lower values) making $\boldsymbol{M_{QNR}}$ sparser or denser respectively, thereby achieving *peripheral region* bitrate adaptation. Fig. 5 demonstrates an exemplary step by step matrix operation. If S has assumed its maximum possible value, and if channel conditions further degrade depending on whether $\Delta P$ crosses a threshold, then *spatial encoder* switches to *foveal phase*. In this phase, *peripheral region* quality is not modified anymore as it is already encoded at minimum possible quality, but bitrate reduction is achieved by *foveal shrinking*. During shrinking phase the *foveal radius* is reduced by scaling down $\boldsymbol{\mu}$ by a factor $\boldsymbol{f_r}$ up to a limit $\boldsymbol{\mu_{min}}$. Whereas, under improving channel conditions, *foveal expansion* is performed. *Foveal expansion* is performed by additively increasing $\boldsymbol{\mu}$ by a factor $\boldsymbol{\mathcal{E}}$ till a limit $\boldsymbol{\mu_{max}}$. Hence, *foveal Breathing* adapts the number of pixels in *foveal region* that are encoded at maximum desired quality, thereby adapting the effective bitrate. The QoE aware Spatial Encoding algorithm therefore achieves bitrate adaptation without compromising with the quality of the Foveal region to satisfy instantaneous channel bit budget (Fig. 6).

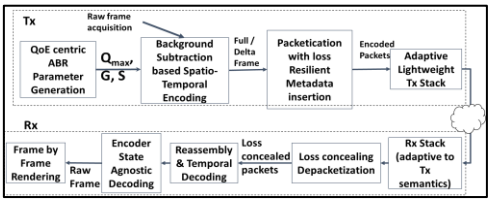


Fig. 3. The end-to-end flow for QuOVADIS.

### *A. The Protocol and Packet Semantics*

Packetization happens such that an integral number of MCUs are placed in a single packet with necessary padding bits as MCUs are not byte aligned. The MCU Payload in each packet is preceded by a payload specific header. The payload specific header comprises of metadata information required for reconstruction of frame and loss concealment on the decoder side and is inherited from [20]. Our system follows the protocol semantics & state machine of CoAP [22] with 'No Response'

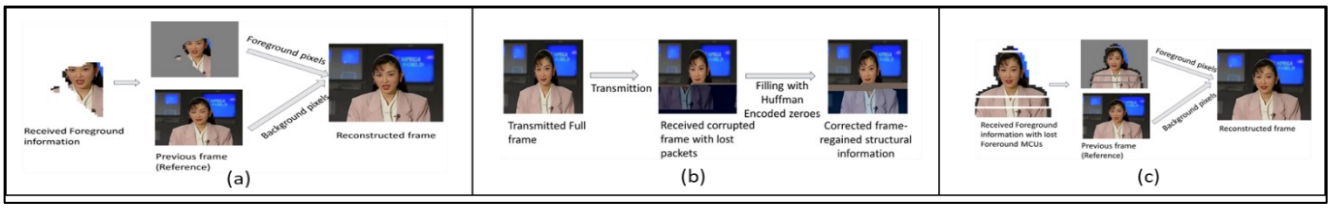


Fig. 4. (a) Reassembly of received Foreground information in delta encoded frames and filling background MCUs with 0 valued pixel information. (b) Loss compensation in full frames in basic state. (c) Loss compensation in delta frame.

[23] option. It follows the essence of [26], that tries to maintain balance between reliability and latency by adding an abstraction layer over CoAP for streaming. Critical packets like first packet of basic frame and the frame sent on Periodic Timer expiry are sent reliably via the (Confirmable)CON semantics. Whereas the rest of the packets are sent via Best Effort delivery using the (Non-confirmable) NON semantics with No Response option. Reliable transmissions are carried out in non-blocking fashion i.e., Tx after sending a packet reliably, continues with transmission of packets next in line in best effort mode while simultaneously waiting for the ACK of the sent reliable packet. The non-blocking mode helps maintain a desired system frame rate unaffected by the channel RTT. The waiting for ACK is done for an adaptively determined Timeout period. We are also proposing a new header option named *Number of Retransmissions* (**NRTx**) to the CoAP protocol options set to control the maximum number of retransmissions allowed for an individual data packet. This provides additional functionality to the MAX_RETRANSMIT field in CoAP. But, the maximum value of NRTx field is limited by the value of MAX_RETRANSMIT parameter. If ACK is not received within Timeout period, then depending on the value of associated NRTx field, Tx decides whether to retransmit the lost packet or not. If NRTx field is nonzero, then Tx performs retransmission of the lost CON packet maximum NRTx number of times. Whereas for zero value of NRTx field, Tx does not perform any retransmission. In our system, we have set the value of NRTx field for each packet sent reliably as zero. This allows us to prevent unnecessary retransmission of lost packet to cater to the real time performance of the system. If NRTx field is not present, MAX_RETRANSMIT prevails as usual. Currently we are using an arbitary option number for NRTx which will be replaced appropriately with a valid option number from IANA after standardization. The Tx, Rx protocol semantics illustrating the close synergy between encoder and transport are shown for different situations through timing diagrams in Fig. 7. Depending on the reception status of ACK, the ACK Timeout period is adapted to minimize the ACK misses. If ACK is lost, might indicate channel congestion, then TO value is increased till a maximum limit using the following formula: $TO_{new} = 1.5\,X\,TO_{prev}$. Whereas if n consecutive ACKs are received, might indicate improvement in channel conditions, then TO value is reduced till a minimum limit using the following formula: $TO_{new} = \left(\frac{TO_{curr} + TO_{prev}}{2}\right)$ where $TO_{curr}$ and $TO_{prev}$ are the Timeout values for current and previous CON packets.

### B. *Calculation Of Errors Statistics At the Receiver*

At each playout interval t, Rx determines the total number of expected packets for each frame by parsing the offset field of first packet which contains the position indicator for the last expected packet in the frame. Rx computes $E^t = \frac{N_{lost}}{N_{total}} X\,100$.

Here, $\mathbf{E^t}$ = instantaneous Error rate at time *t*; $\mathbf{N_{lost}}$ = estimated total no. of packets lost for a frame; $\mathbf{N_{total}}$ = total no. of expected packets for a frame. Using $E^t$, Rx computes the cumulative error at play out-interval ending at ***t,*** starting from time ***t-k*** when Rx last received a CON packet that drives the temporal encoding mechanism. Along with this, Rx maintains a log of $E^t$ for each ***t***. Whenever Rx receives a packet sent via Reliable mode on expiry of periodic timer at time ***j***, it computes $P^j = mode(E^i, E^{j-1}, \ldots\ldots\ldots, E^{j-k})$**.** Where $P^j$ indicates the most frequent Error Rate within the interval between ***j*** and ***j-k***. Rx piggybacks $C^t_{mk}$ and $P^j$ with Acknowledgment of the Reliable packet as periodic feedbacks to Tx. $C^t_{mk}$ and $P^j$ are indicative of end-user QoE and assist Tx in adaptively parameterizing spatio-temporal encoding functions.

### C. *Kalman Filter Based Jitter Buffer Adaptation*

The optimum settings for jitter buffer ensure smooth rendering following the delay variation in the channel. Too low jitter buffer might lead to frame losses and too high jitter buffer may cause increased rendering latency. In case of QuOVADIS, the Rx does not get any feedback from the Tx-side. So, we have to estimate the future delay by observing the delay incurred for the past received frames. The Rx determines the delay for each data packet i using the arrival $A_i(t)$ and transmission $T_i(t)$ time of the data packet using $d_i(t) = A_i(t) - T_i(t)$. Using the delay of each packet, the delay of the corresponding frame is determined based on average of the delay values for each packet as $d(t) = \sum_{i=1}^{N} \frac{d_i(t)}{N}$. One way delay gradient for a frame is calculated using the delay values of the current and previous frames as $d_g(t) = d(t) - d(t-1)$. This value is fed to the Kalman Filter model to predict the value of one-way delay gradient $d_g(t+1)$ for the incoming frame. Eqn. 2 & 3 depict the state and output equations respectively for the proposed Kalman Filter model. where, $m(t)$ is the only state variable, which is model of the one-way delay. gradient. $\omega(t)$ and $\eta(t)$ represents state and measurement noise respectively modelled as stationary Gaussian processes.

$$m(t+1) = m(t) + \omega(t) \quad (2)$$

$$d_g(t) = m(t) + \eta(t) \quad (3)$$

Iterative tuning of certain noise parameters of the KF model such as Kalman Gain ($K(t)$), Process Noise variance ($P(t)$), State Noise variance ($Q(t)$), and measurement noise variance

$(\sigma^2(t))$ with each measurement update of $d_g(t)$ is performed. The next state of the KF model is determined based on the current measurement $d_g(t)$ and current state of KF model using updated value of Kalman Gain as per Eqn. 4. The updated state of KF model serves as the *one-way delay gradient* $d_g(t+1)$ for the incoming frame. Then, we can adjust the jitter buffer for the next frame as ($d_g(t+1)$+ certain sleep time). The sleep time is determined using the maximum desired frame rate of the system. QuOVADIS Kalman Filter model follows that of *Google Congestion Control* [27]. Interested readers can refer to [27] for detailed explanation of the KF model.

$$m(t+1) = m(t) + K(t)(d_g(t) - m(t)) \quad (4)$$

## IV. EXPERIMENT, RESULTS & ANALYSES

QuOVADIS is implemented in C++ using OpenCV and Boost libraries. It captures the raw frames and entire encoding happens in our own S/W without using any special H/W accelerator or encoding in camera firmware. The system was built on Ubuntu 20.04 on a standard Intel Core i5 machine. The transmitter side of QuOVadis is ported to R-Pi3 which is housed in a telerobotic car designed for remote teleoperation. QuOVADIS was designed to live stream both stored videos and live camera feed. A parallel WebRTC implementation on JS is created with media channel for video streaming and data channel for exchanging kinematic controls and feedbacks. The WebRTC system was also designed to transmit both stored video and live camera feed. The implementation is done with 640 x 480 resolution frames. We set the maximum desired frame-rate for QuOVADIS as 15fps. By design, it relies heavily on the delta-frames to achieve this. This can be realized owing to the frugalness of our system as the encoding and decoding processes were completed within 60ms without any hardware acceleration. For full frames, the system conservatively auto-reduces the frame rate to 5fps. Based on our studies and visual experiences, we have set the minimum value of μ to 25% as reducing the foveal region beyond this will undermine the end-user QoE. Similarly, the initial and maximum values of μ were set to 50% and 75% respectively. To ensure very good visual quality with retention of colour information, we set the quality of foveal region to 90%. Initially we compare the performance of QuOVADIS with WebRTC in terms of both full referential and subjective quality metrics for stored video sequences taken from [28]. To ensure to cover different test cases comprising of static FoV, dynamic FoV, high motion, low motion, etc. we chose *Akiyo, Hall, Foreman and Tennis* sequences and rescaled all to 640 × 480 resolution. In this experiment we were interested to observe the effect of only the last-mile impairment. So, both Tx and Rx were kept in the same Network and the access point was moved *'far from - and- near to'* the test set up in a *U*-shaped trajectory. This resulted in the RSSI response shown in Fig. 8 (a). To enable full referential comparison (*SSIM, PSNR, VQM, VMAF*) we created a stream recording mechanism in the receiver and transmitter pages in the WebRTC system. For QuOVADIS we get the individual transmitted and received frames. For WebRTC the samples were all WebM encoded, and for QuOVADIS raw frames were supplied. Besides performing full referential measurements, the experiments were also conducted in the front of 50 subjects who could see both the transmission and reception and they were requested to scale the reception quality for each video for both WebRTC and QuOVADIS. The full referential quality measures and MOS from user study are shown in Fig. 8 (b) & (c) respectively. The reason was loss of GoP synchronization and failure to agile reaction to the channel variations. But QuOVADIS continued decent performance and tried to regain lost frames through its zero overhead error concealment described earlier. There was momentary freeze around deep loss of RSSI. But due to its agile frame-by-frame operation, it regained quickly as soon as RSSI started to rise just above -70 dB. The efficacy is reflected in subjective and full referential results. We then deployed the system over a long-haul P2P setting. The transmitter on the Pi-car was put in Kolkata, India. The operator console was in Bangalore, India. Both units were put in private networks behind restrictive NATs which do not allow hole punching. This ensured that WebRTC will always have to route through TURN server. We also created a UDP-based relay service, co-located with the TURN server, for QuOVADIS. The TURN and the relay servers were replicated in three different AWS instances in ***Mumbai (India), Tokyo (Japan), and Ohio (US-east)***.

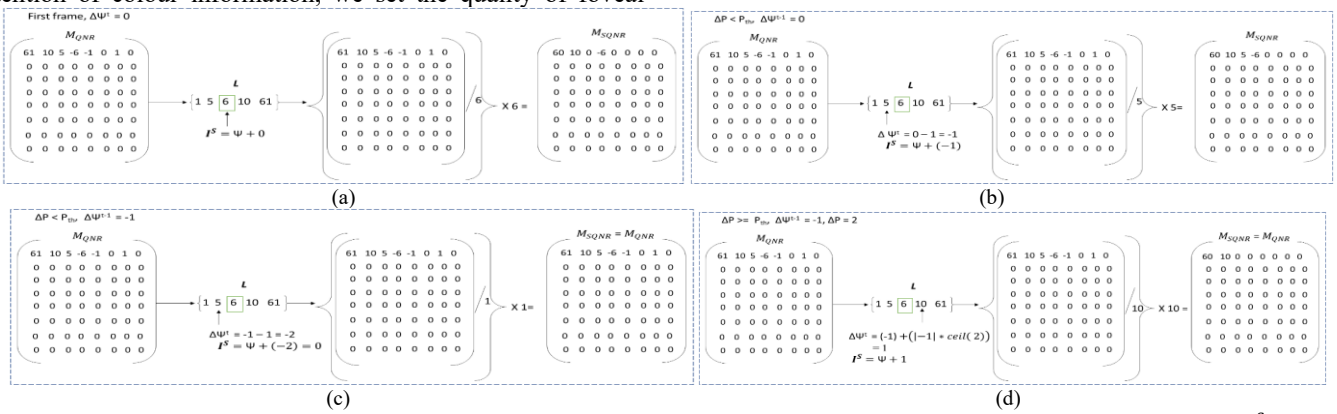


Fig. 5. Median-based Matrix Sparsification for: (a) first frame; (b) frame with $\Delta P < P_{th}$ making MQNR less sparse; (c) frame with $\Delta P < P_{th}$ and $I^S$ becomes 0 making MQNR equal to MSQNR (d) For frame with $\Delta P \geq P_{th}$ performing sparsification using MQNR.

**Algorithm 1: Foveal Breathing**

*Input*: $\Delta P$, $P_{th}$, $\mathcal{B}$, $\mu$, $\mu_{min}$, $\mu_{max}$, $f_r$, $\Delta\Psi^{t-1}$, $\Delta\Psi^{max}$, $\Delta\Psi^{min}$, $\varepsilon$

IF $\Delta P \geq P_{th}$ THEN
  IF $\mathcal{B} == True$ THEN
    IF $\mu > \mu_{min}$ THEN
      $\mu = \frac{\mu}{f_r}$
    END IF
  ELSE
    IF $\Delta\Psi^{t-1} == \Delta\Psi^{max}$ THEN
      $\mathcal{B} = TRUE$
      $\mu = \frac{\mu}{f_r}$
    ELSE
      $\Delta\Psi^{t} = MPU(\Delta\Psi^{t-1}, \Delta P)$
    END IF
  END IF
ELSE
  IF $\mathcal{B} == True$ THEN
    IF $\mu \geq \mu_{max}$ THEN
      $\mathcal{B} = FALSE$
      $\Delta\Psi^{t} = MPU(\Delta\Psi^{t-1}, \Delta P)$
    ELSE
      $\mu = \mu + \varepsilon$
    END IF
  ELSE
    IF $\Delta\Psi^{t-1} > \Delta\Psi^{min}$ THEN
      $\Delta\Psi^{t} = MPU(\Delta\Psi^{t-1}, \Delta P)$
    END IF
  END IF
END IF

Fig. 6. QoE centric Adaptive Bitrate Spatial Encoding

This way we could track performance under communication over Internet backbones running through different parts of the world. A person in Kolkata threw a ball on the floor in a given trajectory, and the person in Bangalore had to track the ball by moving the Pi-car remotely (Fig. 9). While WebRTC system was equipped with data channel for this purpose, we created a special control console for operating while observing feeds using QuOVADIS. The control commands were also relayed through the same relay server. We experimented on 50 users aged between 25 – 45 years. Each user was told to do the 'ball-tracking' exercise for 15 times in each sitting. Out of the 15 times the traffic was routed through Mumbai, Tokyo and Ohio for 5 times each. The experiment was repeated for the same subjects over a span of 5 days at different time of the day (morning, afternoon, evening). Each time the stream was recorded for full referential measures of the videos and the operators in Bangalore were told to mark the experience in a scale of 5. Fig. 10 (a) and (b) shows the full referential and MOS results respectively. The MOS results had the additional consideration for ease of operation by only looking at the video feed from Kolkata. We also measured the motion-to-photon latency for traffic routed through the said routes. For this we time synchronized two smart phones with milliseconds clock in Bangalore and Kolkata. The view of clock was streamed from Kolkata. In Bangalore the mobile clock was set by the console and reception was recorded showing the time in both the screen and on the clock. Then we paused the recorded video at different instances for each recording and measured the time difference between the clock and screen-view for each paused screen. Fig. 10(c) shows the average latency observed in the three different routes. As expected, we found larger latency variation in Ohio, followed by Tokyo and then Mumbai. Mumbai was the least as both peers were located in India. Ohio was the farthest. WebRTC experienced regular freezing with several cycles of Ball throw being missed especially for Ohio-routed traffic. At-times the quality of the reception also extremely deteriorated due to extensive compression of VP8 encoded stream in WebRTC leading to inability to do any kind of teleoperation. So was not the case for QuOVADIS where there was momentary reduction in reception rate during overshoot of end-to-end latency but it did not interfere with the teleoperation owing to its per-frame based operation. The tightly coupled QoE centric spatio-temporal encoder & protocol, along with robust loss concealment & Predictive Jitter Buffer at Rx altogether could cater to end-user QoE requirements for teleoperation.

We used Wireshark to measure the live BW consumption for each experiment. QuOVADIS is also bandwidth efficient as shown in Fig. 11. Considering the figures of Bandwidth consumption at source, the bandwidth consumption consistently reduces from Mumbai-routed traffic to Ohio -routed traffic as the transmission rate reduces with degrading channel from Mumbai to Ohio. QuOVADIS outperforms WebRTC for both Mumbai & Tokyo. But for Ohio, WebRTC pauses transmission for several seconds causing an extreme low bandwidth consumption at expense of visual quality. The BW consumption at Rx results clearly shows that QuOVADIS beats WebRTC under all the three scenarios. The BW consumed at Rx represents the feedback sent to Tx and it increases from the case we route the traffic via Mumbai to when we route it via Ohio due to degrading channel conditions.

## V. CONCLUSION

We have presented an end-to-end QoE-aware streaming solution for delay and freezing sensitive telerobotics solutions. The proposed solution can be extended to many such distributed real-time interactive systems like gaming, etc. In the presented solution the foveal region is fixed as per usual usage pattern. We are currently working on inferring the gaze of the operator from analyzing operator's self-view and adapt the foveal location accordingly, thus enhancing user experience without any dedicated head-mounted sensory-system.

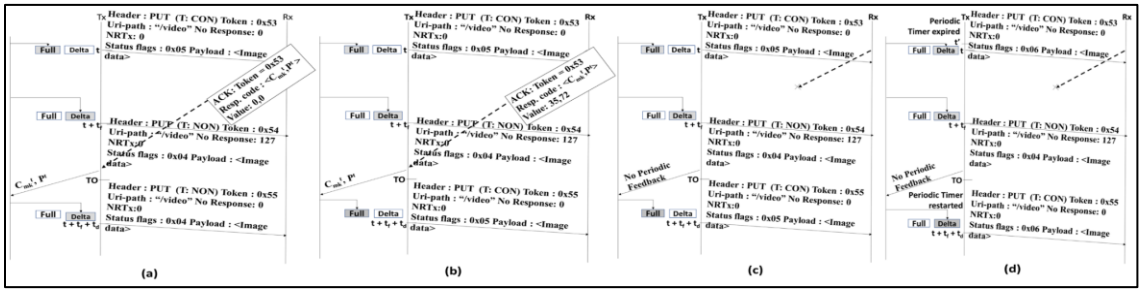


Fig. 7. Timing Diagram in different situation (a)Tx receives ACK with Periodic Feedback from Rx signifying no loss at the Rx and encoder performs bitrate enhancement.(b) ) Tx receives ACK with Periodic Feedback from Rx signifying loss at Rx above threshold and encoder performs bitrate reduction in response. (c) ACK belonging to a full frame lost making Tx send next frame as full frame.(d) ACK belonging to a frame sent at Periodic Timer Expiry lost making Tx send next frame with 'Periodic Timer status' flag set.

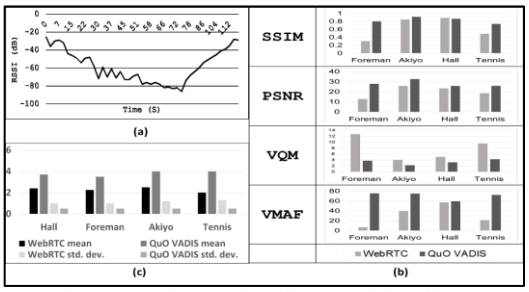


Fig. 8. (a) RSSI along test trajectory (b) Full referential quality measurements (c) MOS results under practical last mile impairments

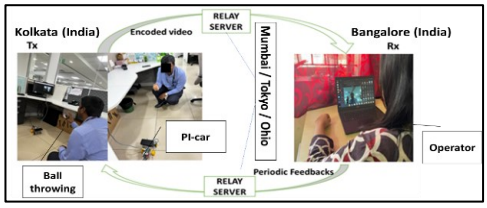


Fig. 9. Setup for long-haul telerobotic experiment

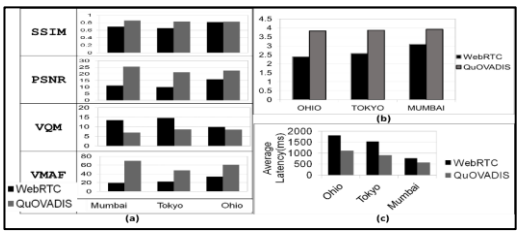


Fig. 10. Average results under long-haul : (a) full referential QoE. (b) Teleoperation MOS. (c) Avg. latency.

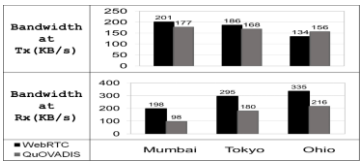


Fig. 11. Bandwidth consumption at Tx & Rx for live streaming in WebRTC & QuOVADIS via Relay servers at Mumbai,Ohio & Tokyo

## REFERENCES


[1] W. Wu, et. al., "Quality of experience in distributed interactive multimedia environments: toward a theoretical framework", 17th ACM international conference on Multimedia (MM '09), NY, USA, 2009.

[2] ISO/IEC, "Dynamic Adaptive Streaming over HTTP (DASH) Part 1: Media presentation description and segment formats," 2014.

[3] R. Pantos, W. May, RFC 8216, "HTTP Live Streaming", August, 2017.

[4] M. Ellis, D. P. Pezaros and C. Perkins, "Performance analysis of AL-FEC for RTP-based streaming video traffic to residential users," 19th International Packet Video Workshop (PV), Munich, 2012, pp. 1-6.

[5] M. Palmer, T. Krüger, B. Chandrasekaran and A Feldmann, "The QUIC Fix for Optimal Video Streaming", Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ'18), Greece, December, 2018. DOI:10.1145/3284850.3284857.

[6] S. Patra, et al, "An ITS solution providing real-time visual overtaking assistance using smartphones", 40th conference on local computer networks (LCN), IEEE, pp. 270-278, October, 2015.

[7] A. Kaknjo, et al, 2018. Real-time video latency measurement between a robot and its remote control station: causes and mitigation. Wireless Communications and Mobile Computing, 2018.

[8] Vechtomov, Vladimir. "LOW LATENCY H. 264 ENCODING FOR TELEOPERATION." (2023).

[9] S. Fouladi, et al., "Salsify: Low-Latency Network Video Through Tighter Integration Between a Video Codec and a Transport Protocol", 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18), Renton, USA, April, 2018.

[10] H. Alvestrand, RFC 8825, "Overview: Real Time Protocols for Browser-based Applications", IETF, January, 2021, doi: https://datatracker.ietf.org/doc/html/rfc8825.

[11] "What do WebRTC Monitoring Experts Think About Salsify?", https://www.callstats.io/blog/2018/07/18/what-do-webrtc-monitoring-and-analytics-experts-think-about-salsify-for-video-delay, last accessed: Jan 27, 2023.

[12] Žádník, Jakub, et al. "Image and video coding techniques for ultra-low latency." ACM Computing Surveys (CSUR) 54.11s (2022): 1-35.

[13] X. Tran, et. al., "Reducing Bitrate and Increasing the Quality of Inter Frame by Avoiding Quantization Errors in Stationary Blocks", EAI Endorsed Trans. Ind. Networks Intell. Syst, 2020, vol.7, pp. e2.

[14] T. C. Minh, et al., "QABR: A QoE-Based Approach to Adaptive Bitrate Selection in Video Streaming Services." International Journal of Advanced Trends in Computer Science and Engineering, 2019.

[15] Double Robotics - Telepresence Robot for the Hybrid Office, https://www.doublerobotics.com/, last accessed 2023/05/15.

[16] A. Sau, A. Bhattacharyya, M. Ganguly, "Teledrive: A Multi-master Hybrid Mobile Telerobotics System with Federated Avatar Control", Mobile and Ubiquitous Systems: Computing, Networking and Services. (MobiQuitous 2021), Japan, 2021, doi: 10.1007/978-3-030-94822-1_6.

[17] https://aws.amazon.com/blogs/iot/utilizing-aws-services-to-quickly-build-solutions-for-robotics-use-cases

[18] C. Perkins, M. Westerlund, J. Ott, "RFC8834: Media Transport and Use of RTP in WebRTC", IETF, 2021.

[19] N. Feamster and H. Balakrishnan, "Packet Loss Recovery for Streaming Video", International Packet Video Workshop, Pittsburgh, April 2002.

[20] M. Ganguly, A. Bhattacharyya, A. Sau and B. Purushothaman, "A-REaLiSTIQ-ViBe: Entangling Encoding and Transport to Improve Live Video Experience", International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 2022.

[21] A. Bhattacharyya, M. Ganguly and A. Sau, "Improving Perceived QoS of Delay-sensitive Video Against A Weak Last-mile: A Practical Approach" International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 2021.

[22] Z. Shelby, K. Hartke, C. Bormann, RFC 7252, "The Constrained Application Protocol (CoAP)", IETF, June, 2014.

[23] A. Bhattacharyya, et al, RFC 7967, "Constrained Application Protocol (CoAP) Option for No Server Response", August, 2016.

[24] Bastani, Behnam. "Strategies for Foveated Compression and Transmission", doi: https://research.google/pubs/pub46452/ .

[25] Barnich, Olivier, and Marc Van Droogenbroeck. "ViBe: A universal background subtraction algorithm for video sequences." IEEE Transactions on Image processing 20, no. 6 (2010): 1709-1724.

[26] A. Bhattacharyya, S. Agrawal, H. K. Rath, A. Pal. 2018. Improving Live-Streaming Experience for Delay-Sensitive IoT Applications: A RESTful Approach. In *Proceedings of 2018 IEEE Globecom Workshops (GC Wkshps)*. Abu Dhabi, UAE, 2018.

[27] Carlucci, Gaetano, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. "Analysis and design of the google congestion control for web real-time communication (WebRTC)." In *Proceedings of the 7th International Conference on Multimedia Systems*, pp. 1-12. 2016.

[28] https://media.xiph.org/video/derf