Tele-driving an electric vehicle over a private LTE network

Ashish Joglekar*† ashishj@iisc.ac.in
Alok Rawat* alokrawat@iisc.ac.in
Ashwin Gerard Colaco‡ ashwincolaco@iisc.ac.in
Aswin Prasad I S‡ aswinprasad@iisc.ac.in
Praphulla Ranjan* praphullar@iisc.ac.in

Raghava Doreswamy§ raghava@iisc.ac.in
Rajat Chopra§ rajatchopra@iisc.ac.in
Amrutur Bharadwaj*§† amrutur@iisc.ac.in
Bharath Govindraju† bharath@artpark.in
Fashid Rahman† fashid@artpark.in

Himanshu Tyagi§ htyagi@iisc.ac.in
Naveen Arulselvan*† naveena@iisc.ac.in
Preetam Patil‡ preetampatil@iisc.ac.in
S.V.R Anand† anandsvr@artpark.in
Vishal Sevani‡ vishalsevani@iisc.ac.in

*RBCCPS (Robert Bosch Centre for Cyber-Physical Systems), IISc, Bengaluru, India
†ARTPARK (AI & Robotics Technology Park), IISc, Bengaluru, India
‡CNI (Centre for Networked Intelligence, RBCCPS and EECS Division), IISc, Bengaluru, India
§Dept. of ECE, IISc, Bengaluru, India

***Abstract*—We demonstrate tele-driving operation for an electric vehicle capable of stopping itself in case of system failure over a captive LTE network deployed in a university campus. Our electronically controlled vehicle is driven remotely by an operator from a control room which receives the multi-camera real-time video feed from the vehicle over this network. Our primary contribution includes the responsive emergency braking mechanism for the vehicle, modular vehicle design based on CAN bus, low latency LTE MAC scheduler design, and modifications to popular video tool, FFMPEG to support low latency real time video streaming. Our demonstration shows complete integration of the different components, i.e., the vehicle, the LTE network and the remote driving application. Another salient feature of our system is the O-RAN compliant RAN awareness module and KPI (Key Performance Indicator) application which enables real-time network performance monitoring.**

I. Introduction

The past decade has seen remarkable progress in the research and development of autonomous driving vehicles. However, there is considerable skepticism on whether true self-driving capabilities or SAE (Society of Automotive Engineers) Level 5 autonomy can ever be achieved [1]. It seems plausible that SAE Level 4 autonomy can be achieved in the foreseeable future wherein vehicles are still driven by an AI, albeit in designated ODDs (Operational Design Domains), e.g., within 10 kms of a neighborhood, industrial complexes, etc. However, there are still considerable technological and ideological problems to be solved to reach Level 4 autonomy. On the other hand, robotic applications with remote operator control have been gathering momentum as 5G/LTE networks can offer low latencies. This opens up the possibility of offering vehicles with tele-driving services. In this work, we describe our ongoing efforts to demonstrate tele-driving features in a modified electric vehicle. The vehicle is remotely driven over a private LTE network running our custom MAC scheduler that provides low latency to applications including tele-driving. We note that though we use LTE network our O-RAN compliant near-realtime RIC xApp for monitoring RAN parameters enables seamless integration with 5G RAN (Sec. IV). The system also incorporates mechanisms in the off-the-shelf tool FFMPEG to support real-time video streaming.

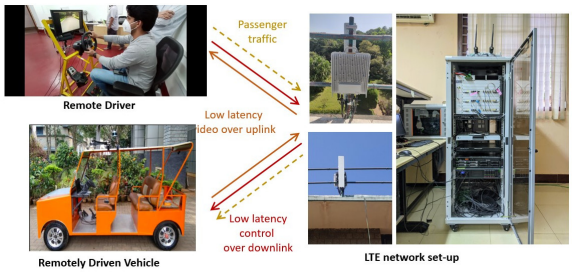

Fig. 1. Demo setup

II. Demo Setup

Our system consists of an electric vehicle which is remotely driven from a control room. The vehicle is fitted with multiple cameras to capture the outside view which suitably aids the driver sitting in a control room to drive the vehicle. The real-time video feed from multiple cameras is relayed to the remote driver over the uplink of a private LTE network. The remote driver controls the vehicle using a joystick. The commands to drive the vehicle are sent using control packets over the LTE downlink. Figure 1 shows the overall setup.

In our LTE setup, we have two RRHs mounted on top of our department building such that the entire vehicular path has complete radio coverage, and the network supports an aggregate uplink bandwidth of upto 6 Mbps for three video cameras mounted on the vehicle. The video stream is sent over UDP and the control commands are sent on the downlink over TCP. Multiple LTE radio bearers have been configured with

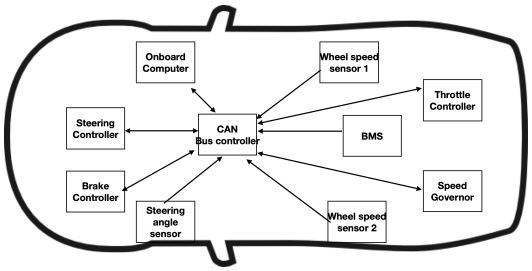

Fig. 2. Hardware architecture of the vehicle

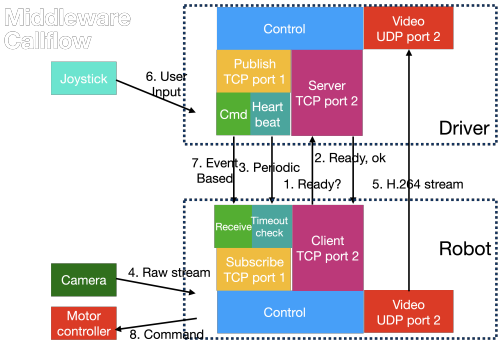

Fig. 3. Software architecture of the vehicle

different QCI values so that the custom MAC scheduler can provide appropriate QoS for different traffic flows.

Our measurements show the latency of real-time video feed from cameras was mostly within 100 ms (Sec. V), while the latency for control packets was mostly within 30 ms (Sec. VI).

III. Remote Controlled Electric Vehicle Design

Below we describe the architecture of the vehicle design, along with its important feature, emergency braking.

A. Hardware and Software Architecture of the Vehicle

For the purposes of this work, we have used a commonly-seen electric buggy model shown in Figure 1. Such vehicles have wide acceptance in closed and structured environments such as university campuses. However, an electric buggy is not inherently drive-by-wire. In other words, steering, brake and throttle are mechanically actuated rather than electronically. Preparing the electric buggy to electronically actuate the individual functions, constituted the major effort in this work. In addition, a CAN bus controller was provisioned to reliably communicate across the sub-systems.

The hardware architecture of the modified vehicle is shown in Figure 2. Brake and throttle user inputs are accepted as quantized levels. The brake input controls the position of a linear actuator that engages the vehicle's hydraulic braking system. A Digital to Analog Converter (DAC) converts the throttle input to an analog value which is fed to the EV's motor driver to control the motor's RPM. A steering angle sensor is used to monitor the performance of our hysteric on-off steering motor controller. A diagnostics dashboard displays data on the CAN bus.

The software architecture of the vehicle is shown in Figure 3. It consists of a custom middleware which has desirable properties such as discovery, synchronization, liveness (or heartbeat), automatic reconnectivity, logging. As shown in Figure 3, the middleware facilitates control of vehicle by the driver via joystick using control sublayer, while the video sublayer sends the real-time video feed from the multiple cameras mounted on the vehicle to the control center.

The vehicle is currently remote controlled by the driver in the control room. However, the planned feature evolution is for the operator to eventually supervise but not micro-manage. The vehicle will be capable of local emergency handling and recovery. A richer description of the environment such as road signs, pedestrian crossings, etc can also be relayed back.

B. Emergency Braking

We have focused on the most stringent use case of network-based robot operation, namely emergency braking. The event flow for emergency braking is as follows, 1. A vehicle camera sensor captures images of the immediate environment and transmits it over the network (δ ms). 2. Remote driver perceives the need to brake and activates the brake command (p s). 3. Brake command is transmitted over the network (δ ms). 4. Vehicle comes to a complete stop (s ms).

Clearly steps 1 and 3 are additional in remote operation. Step 2 is indicative of how quickly humans perceive and react. It is referred to as the Perception-Reaction time (p) and we take p to be 1.5s for low-volume roads as per Hopper-McGee model [2]. Using appropriate calculations (omitted here), we observe that for vehicular speeds less than 20 kmph, tele-operations support for safe stopping is possible when the network latency is less than 200 ms. For our network we observe latencies much smaller than 200 ms (Sec. VI) allowing us to support emergency braking in our application.

IV. Low Latency Scheduling Over LTE Network

We have set-up our private LTE network in the university campus for testing the remote-driving capabilities of the vehicle. User equipments are connected to the vehicle to provide network connectivity to the various sub-components of the vehicle. As mentioned in Sec. II, control signals for the vehicle that are responsible for its motion are sent over the downlink, while real time video feeds of the vehicle's cameras are streamed to the remote control center over the uplink. The control signals require very low latency for the smooth operation of the vehicle. We use L2 layer's MAC (Medium Access Control) scheduler to give priority to the data flow serving the control signals.

The MAC scheduler is one of the most important components of LTE RAN (Radio Access Network). The MAC is responsible to efficiently allocate resources to the user equipment. User data flow inside a user equipment can be virtualized by radio bearers. Each radio bearer is associated with a QCI (QoS class identifier) value used by 3GPP to provide QoS to the different traffic flows. QCI values are used by the MAC scheduler to identify the characteristics of a data flow and allocate resources to the user equipment accordingly.

Control messages for remote vehicle driving require guaranteed resources with low delay budget. We implement a

low latency scheduler which efficiently allocates resources to the different flows inside a user equipment according to the QoS characteristics associated with its QCI value. This enables vehicular control messages to have a very tight delay profile (mostly within 30 ms) even with multiple flows flowing through the network as highlighted in Sec. VI.

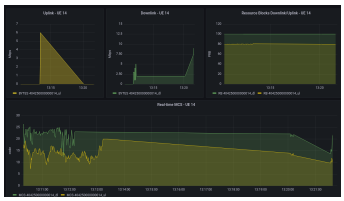

Fig. 4. Graphical display of real-time performance metrics of LTE network

Furthermore we also implement an O-RAN compliant feature known as RAN awareness as an add-on to the scheduler. The RAN awareness module is developed as an O-RAN E2-SM-KPM specification compliant Near-Real-Time RIC xApp for monitoring RAN performance parameters including UE uplink and downlink throughputs, PRB utilization, MCS, queuing delays, etc. These parameters can be made available to the application developers for making their applications network aware. Figure 4 shows the graphical display of the metrics reported by our scheduler in real-time.

V. FFMPEG Optimization

For encoding real-time video feeds from the cameras, we use the H.264 standard, as H.264 can achieve significant compression without compromising video quality. We use a popular tool FFMPEG which implements both a H.264 encoder as well as a decoder.

The cameras mounted on the vehicle output video streams at a constant frame rate of 24 fps (frames per second). FFMPEG encodes the video streams from the cameras at a constant bit-rate and sends it over the uplink LTE to control center where an FFMPEG decoder decodes the streams for display. As the maximum uplink bandwidth of LTE in our setup is about 6Mbps, we keep the sum of bit-rates of all the video streams encoded by FFMPEG from the cameras to about 5 Mbps, a value less than the maximum uplink LTE bandwidth. This enables us to prevent building of *time-lag* in the display of video streams, at the decoder, as described below.

As the vehicle is moving, we observe that the throughput experienced by the vehicle over the LTE uplink fluctuates slightly over time i.e., at times the vehicles sees a temporary drop in throughput which then subsequently recovers. As the uplink vehicle throughput reduces to lower than 5 Mbps, the video frames get queued up at the encoder and the latency of video frames at the decoder increases. This results in a build-up of *time-lag* in the video being displayed at the decoder.

When the uplink vehicle throughput resumes to a value higher than 5 Mbps, the queued up video frames at the encoder are released at a faster rate and the video queue at the FFMPEG decoder fills up. However, even though the video queue at the FFMPEG decoder fills up, the video frames do not get displayed faster. This is because FFMPEG displays the video at a constant frame rate which is the same frame rate at which the video is captured by the cameras i.e., 24 fps. Due to this the *time-lag* which is once built-up in the display of the video at the decoder doesn't recover.

We modified the FFMPEG decoder to account for this *time-lag*. We monitor the queue build up at the decoder and when we notice that at least five frames have been queued up, we skip these five frames from being displayed. We kept the number of video frames to be skipped at a time as five frames, as it did not much affect the viewing experience of the remote driver. With our optimization we observed that end-to-end latency for the display of video streams in our test runs was mostly within 100 ms. The complete details of our proposed video transmission framework including algorithms for rate control, will be presented in an upcoming paper.

VI. Demo Results

We have been testing the remote-driving capabilities of this vehicle over a private LTE network. Since we are in an early testing phase, a safety driver is present in the vehicle at the moment. Figure 5 shows the shots of the ego-view and external view while testing the vehicle.

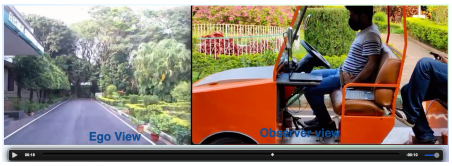

Fig. 5. Ego/external vehicle view

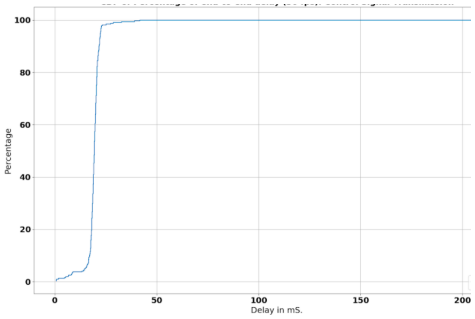

Fig. 6. Latency distribution of the control frames

Initial test results have been encouraging. The latency distribution of the control packets (Sec. II), are given in Figure 6 and as can be seen, the latencies are mostly 30 ms or less.

Acknowledgment

The project was funded by DoT (Department of Telecommunications, Govt. of India), RBCCPS, ARTPARK and CNI.

References

[1] L. Eliot, *Forbes*. [Online]. Available: https://www.forbes.com/sites/lanceeliot/2021/08/02/musing-on-that-rising-gossip-about-wanting-to-ditch-the-vaunted-topmost-level-5-of-the-self-driving-cars-autonomous-rating-scale/?sh=4e0ee8141bac

[2] K. Hooper and H. McGee, "Driver perception-reaction time: Are revisions to current specification values in order?" *Transportation Research Record, 904*, pp. 21–30, 1983.