On the Latency in Vehicular Control using Video Streaming over Wi-Fi

Pratik Sharma[1], Devam Awasare[1], Bishal Jaiswal[1], Srivats Mohan[1], Abinaya N.[2], Ishan Darwhekar[1], Anand SVR[1], Bharadwaj Amrutur[1,2], Aditya Gopalan[1], Parimal Parag[1], Himanshu Tyagi[1]
[1]*Department of Electrical Communication Engineering, Indian Institute of Science*
[2]*Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science*
{pratiksharma, devamawasare, bishalj, srivatsmohan, abinayan, ishand, anandsvr, amrutur, aditya, parimal, htyagi}@iisc.ac.in

***Abstract*—We consider the use of Wi-Fi (IEEE 802.11n/r) network for remote control of a vehicle using video transmission on the uplink and control signals for the actuator on the downlink. We have setup a network with multiple access points (AP) providing indoor and outdoor coverage, which connects an unmanned ground vehicle (UGV) to a remote command center. Additionally, our setup includes a redundant IEEE 802.11p link for sending control messages over downlink with high reliability and low latency. We study the end-to-end communication delay and complete a latency profiling for each sub-component, including the video codec and the Wi-Fi links. Furthermore, we provide guidelines for practical design choices including the optimal configuration of the scanning process during handoffs and the codec parameters for delay optimization. Overall, our proposed configuration reduces the end-to-end delay significantly in comparison with the default configuration.**

I. Introduction

We have designed an experimental setup to profile the end-to-end latency for communication between a remotely driven UGV and its control center. Autonomous or remote control driving is possible through multimodal sensing using techniques such as RADAR (RAdio Detection And Ranging), LIDAR (Light Detection and Ranging), and video streams, where video consumes the largest portion of the uplink throughput. For simplicity, we have setup a vehicle with only video streaming as sensing mechanism for remote driving and have connected it to a command center over the Wi-Fi network.

We present a systematic analysis of the end-to-end latency in the transmission of video stream over the uplink, including the video codec delays at both ends, and control command over the downlink. Based on our analysis, we have identified the key parameters that are relevant for delays and identify their best-case scenario values.

In the remainder of this section, we briefly discuss our setup, the key contributors to latency, prior art, and our specific contributions. Each of these components is elaborated later.

A. Components of our setup

We consider a scenario where a mobile node with live video stream and limited computation power is connected over Wi-Fi to a command center with a powerful server. We consider remote driving by an operator and autonomous braking by detection at the command center. Such real-time controls over network require an uplink with high throughput & low latency and a low latency downlink. In addition, edge computing capability is required at the mobile node to enable local decision making whenever needed, and Multi-access Edge Computing (MEC) capabilities in network elements to enable real-time network optimization. Keeping these requirements in mind, we have deployed a Wi-Fi network with multiple access points (APs) connected to the command center with powerful compute over fiber. Our Wi-Fi network includes both IEEE 802.11n and IEEE 802.11r, along with IEEE 802.11p. This testbed is live and will support further research and development beyond what is being reported. A block view of our end-to-end setup is depicted in Fig. 1.

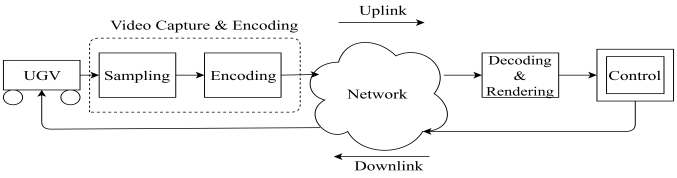


Fig. 1: Block diagram for communication between UGV and command center.

B. The main contributors to latency

Low latency is a critical requirement for real-time control over networks. In spite of significant recent interest in low latency communication, it is unclear what is the latency possible using existing network deployments. In this work, focussing on a Wi-Fi deployment, we have identified three main contributors to latency: (1) The sampling, encoding, and decoding delays of the video codec, (2) the transmission delays for the network, and perhaps most importantly, (3) the delay caused by handoffs in Wi-Fi. Note that a single Wi-Fi AP can provide coverage of $50-100$ m in outdoor deployments with foliage, whereby a mobile node will often encounter handoffs as it navigates using the network. Thus, the handoff delays dominate latency in Wi-Fi and must be carefully mitigated. In comparison, the transmission delays are negligible. In fact, the video codec delays much exceed the transmission delays and must be addressed.

C. Prior work

The paper [1] (see, also, [2]–[4]) studies a city-scale Wi-Fi deployment for vehicular communication and provides an ex-

tensive study of duration of connectivity per AP and coverage for fast-moving vehicles. At a high-level, the conclusion of this study is positive, but one must expect short-lasted connections with several handoffs. In fact, it is well-known that handoffs constitute the main bottleneck for latency in Wi-Fi during mobility ($cf.$ [5], [6]). Most of these works have the same conclusion: scanning constitutes the main component of delay. Several algorithms have been considered for optimizing the scan process ($cf.$ [6]–[8], but it remains the bottleneck. Most of this line of work considered indoor deployments, while our study is over an outdoor deployment with multiple APs.

Over the past decade, several variants of Wi-Fi 802.11 have emerged that explicitly target handover optimization. Of these, 802.11r and 802.11k seem to be the most popular ones. However, the latency reported for these standards in the literature ($cf.$ [9], [10]) seems still high, and we have included these standards in our deployment to evaluate the performance of our outdoor use case.

Another aspect considered in prior work is the effect of a poorly chosen RSSI threshold for triggering handoff (see, for instance, [7]). Prior studies have considered the adverse effect of choosing a conservative threshold resulting in repeated scans. Our concern is the opposite – a more conservative threshold results in a poor video quality during handoff.

In another development, 802.11p has emerged as a popular Wi-Fi solution for vehicular communication, offering both low latency and long-range. Several recent studies offer latency and coverage measurements for 802.11p ($cf.$ [11]). Differing from these works, we evaluate a slightly different aspect of the 802.11p link: its capability to offer a low latency redundant link for control over Wi-Fi.

In a nutshell, highly optimized and expensive hardware can give ultra-fast codecs with a few milliseconds of latency. In particular, there is no comprehensive study of latency for real deployments with live video upload from a mobile sensor node with limited computation capabilities over a wireless network.

D. Our contributions

We fill the gaps in the prior work outlined above by measuring latencies for various components in our end-to-end setup. Our deployment is live outdoor comprising multiple APs and a Wi-Fi enabled mobile node with limited computation. Furthermore, we can access and adjust the parameters of various functional elements such as video sampling, encoding, decoding, analytics, and the communication network.

One of our main contributions is to study the interdependencies between these elements and identification of important system parameters at each of these elements that can be adjusted to strike a balance for optimal latency performance in various channel conditions for our use-case. Specifically, we have identified the adjustable video codec parameters in H.264 and system parameters in FFmpeg that aided in measurement and minimization of delays. We also optimized the scanning and handoff process in 802.11r to smoothen the transition between APs during uplink transmission. Furthermore, we have optimized the threshold for RSSI for triggering handoff. In addition, we show that 802.11p is a preferred network implementation for the downlink time-critical control command transmission to ensure low latency communication.

As another contribution of this work, we demonstrate that the latency achieved in our end-to-end system can vary greatly for different values of the parameters. Since the default values are set conservatively assuming a heavily loaded system, we judiciously chose them so that the latency can be significantly reduced for our use-case of control of a single UGV over the network.

II. Experimental Setup

As shown in Fig. 2, our remote driving experimental setup consists of an outdoor Wi-Fi network infrastructure and a single remotely driven UGV. The mobile UGV sends video transmission of the surroundings to a central server connected to the Wi-Fi network, and located at the Network Operation Center (NOC). A remote driver was stationed at the NOC to observe the uplink video feed and send appropriate control messages over downlink. We used a laptop running on Linux OS, i5 processor and 8 GB RAM as the controller for remote driving.

Wireless network infrastructure consists of three 802.11n Wi-Fi access points (AP) in a triangulated set-up across the testbed at the following landmarks: (i) roof of the building at Site A, (ii) mounted on a lamp post at Site B in the center of the testbed, and (iii) roof of the building at Site C. All three APs are connected via ethernet to the central compute server. To obtain good quality seamless video transmission from UGV to the central server, during handover between two APs, a suitable Received Signal Strength Indicator (RSSI) value has to be maintained at all times. Accordingly, the three APs were positioned at an average distance of 50 m from each other. We have also placed the 802.11p based Roadside Units (RSU) at the same sites as the APs and have mounted the on-board units (OBU) on the UGV.

As a compute platform on-board the UGV, we experimented with two processors: (i) Raspberry Pi Model 3B+ with Broadcom chipset (RPi) and (ii) Nvidia Jetson TX2. In instances where the processor speed didn't have a significant impact on the latency, we have used the RPi as the default processor.

We used the kernel log to measure scanning time and enabled monitor mode on IEEE 802.11 using Aircrack-ng package with Wireshark to measure roaming time. We measured the video codec delays on the RPi by invoking a timer as soon as the frame is captured and subsequently stopping the timer as soon as the frames are encoded and sent for relay. A similar process is followed on the receiver end wherein as soon as the frame enters the decoder buffer, the timer is started. It is then stopped at the moment the frame is sent for rendering.

Finally, we measure end-to-end latency for our use-case by triggering a red light at the mobile node, detecting this light using an algorithm running on the video received at the control center, and sending the control signal for braking to the mobile node. The overall latency of the system is the time between a red light is triggered and the brake is applied.

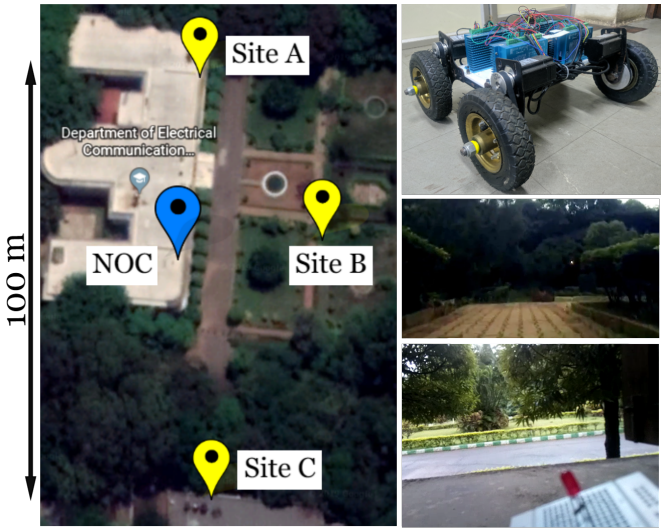


Fig. 2: Left: Representation of the Wi-Fi AP deployment at sites A, B, C and the control room at NOC in the testbed. Top Right: UGV deployed in the testbed. Middle Right: Typical Video feed as seen by driver. Bottom Right: Video Feed for emergency braking setup.

III. Communication Latency

In our experimental setup, the UGV sends uplink video transmission to 802.11n Wi-Fi APs at 2.4 GHz. The mobile UGV changes its association with the APs depending on the received signal strength. In this section, we describe this handover re-association process and present customizations that can optimize the handover process in our setting.

A. Profiling uplink latency

The handover process can be separated into two parts: scanning the signal strength of the neighboring APs and roaming from the source AP to the target AP. We first profile the handover latency, by profiling the scan and roam latencies.

1) Scanning: We found significant scan latencies at the RPi mounted on the vehicle. These experiments resulted in scanning delays upto 2.1 s for scanning the 25 channels across both 2.4 and 5 GHz frequencies. This delay was substantially higher as compared to the expected theoretical values. This is due to continuous hopping between the home and away channel to maintain a certain Quality of Service (QoS). Recall that the current channel on which the data is being transmitted is called the *home channel*, and rest all channels are referred to as *away channels*. The scanning process at RPi consists of repetition of the following two stages until all the channels have been scanned. In the away stage, the RPi sends a probe request to a subset of away channels and then waits for the probe response. During this listening time, the packets accumulate at the RPi data buffer. In the home stage, RPi returns to the home channel for clearing the data buffer. We found that majority of the 2.1 s scanning time was spent on away stages (approximately 1.502 s), and minority on home stages (approximately 0.611 s). There are three main components of away stage of the scanning [12]:

- Probe time: For the outdoor transmissions, we found that the average probe transmission time to be $2-3$ ms.
- Channel hold time: The minimum and maximum time spent on a channel listening to the probe response is denoted by $T_{\min}$ and $T_{\max}$, respectively. In our setup, $T_{\min}$ is lower bounded by 10 ms as specified by Broadcom chipset of the RPi. The default $T_{\max}$ value is significantly high, for the high AP density case.
- Channel hopping time: We found average channel hopping time to be $2-3$ ms.

Optimization of the transmit probe time, minimum channel time, and channel hopping time were not possible due to either hardware constraints or non-discovery of APs. Thus, only parameter that we can optimize in our setting is $T_{\max}$, keeping in mind that $T_{\max} \geq T_{\min}$. For the default $T_{\max}$ value in the RPi, we computed the average scan time for the three Wi-Fi APs operating on three non-overlapping channels 1, 6, and 11, and four out-of-network APs not controlled by us. We found the average scan time as 144.44 ms with a standard deviation of 9.76 ms.

2) Roaming: We found that the average roaming latency in 802.11n was around 42 ms.

TABLE I: Optimized scanning delays for three channels

$T_{\max}$ (ms)	11	12	13	14	15	16	17	18	19
APs found	4	5	5	7	7	7	7	7	7
Total time (ms)	43	46	48	52	54	58	61	64	66

B. Customizing uplink latency

We introduce customizations that result in significant reductions in the scan and roam latencies in the 802.11n network for our use case.

1) Scanning: In our experiment, we designed the wireless network such that the UGV is always covered by an AP with a good communication link, and hence we expect minimal retransmissions. We experimentally found the optimal $T_{\max}$ by scanning the APs at different values of $T_{\max}$ and selecting the one that leads to minimum scan delay while discovering all seven APs. Table I shows the relation between $T_{\max}$, the scan latency, and the number of APs discovered. We observed that the scan latencies increase monotonically with $T_{\max}$, however $T_{\max} \leq 14$ ms leads to the discovery of less than 7 APs. Therefore, we chose $T_{\max} = 14$ ms, and the corresponding scan delay was around 52 ms.

2) Roaming: For successful completion of the handover process, there are mainly three courses of action that are required to be completed during the roam stage: (1) association request and response, (2) re-association request and response, and (3) EAPOL key exchange. EAPOL key exchange requires several retransmissions in 802.11n for a secure key exchange, and this process is further optimized in 802.11r. This optimization is achieved by pre-sharing of the PMK-R0 key among all

APs in the same mobility domain resulting in the generation of the R1 key [13], and elimination of the authentication step. Hence, we enabled 802.11r at the APs by OpenWrt, and at the RPi by wpasupplicant2.6 to reduce the roam delay.

Our next customization was changing the default RSSI threshold for the handover initiation. The default RSSI threshold of -89 dBm is unsuitable for real-time video transmission, and the video had multiple lags at this RSSI value and the Quality of Experience (QoE) for the remote driver was too poor to make any meaningful control decisions based on the video feed. We experimentally found that an RSSI threshold of -68 dBm for the handover initiation leads to a good video quality experience.

With these customizations, we found that the roaming delay ranges in $20-32$ ms, and we reduced the average roam delay from 42 ms to 26 ms.

C. Downlink customization

In the remote driving application, the controller sends control messages over the downlink that require low latency. For the 802.11r network, the average downlink communication latency is found to be 9 ms when associated with an AP, and 35 ms during the handover. As an alternative, we experimented with 802.11p network for the downlink communication, which eliminates the need for association [14].

To enable this, we mounted an OBU on the UGV and RSUs on the periphery of the road to communicate with the OBU. All the RSUs are connected by a wired backbone to the command center. The OBU broadcasts beacon messages consistently at predefined intervals to announce its presence to the RSUs. The command center sends the same control message redundantly through multiple RSUs to ascertain packet delivery to the OBU. Subsequently, the OBU acknowledges all the control commands it receives. For 802.11p, the average round trip time was found to be 2.8309 ms with a standard deviation of 0.086 ms. In addition, we observed no packet loss for the UGV mobility over a straight-line path in the experimental testbed with three RSUs.

IV. Video Codec Latency Profiling and Optimization on Uplink

Low-latency video transmission over the uplink is crucial for remote driving. This section details profiling studies of the sensitivity of the uplink video pipeline to various parameters, along with customizations of the parameters for best achievable latency performance.

Implementation note: We use the FFmpeg [17] video transmission platform for our remote driving use-case. For video coding, we use the H.264/AVC [15] or MPEG-4 standard for video coding – the current industry standard for video compression[1] and an ideal video codec choice for the computation-limited processor on the remotely driven vehicle. In our experiments, we set parameters like video resolution, constant rate factor (CRF), preset value, and frames per second (FPS) for H.264 using FFmpeg.

[1]A newer H.265/HEVC coding standard has been released, but its adoption has been slow primarily because it consumes about $10X$ processing power for encoding and decoding from H.264, without a considerable reduction in bitrate below 1080p resolution [16]

A. Profiling

For a remotely driven vehicle, the total uplink latency is the delay between the occurrence of an event in the environment of the vehicle and it being displayed to the remote driver. We can divide this latency into 4 parts – Sampling, Encoding, Network, and Decoding & Rendering delays – and study how each part varies with several parameters that can be changed.

1) Sampling: The sampling delay is simply determined by the frame rate of the camera. For instance, a camera recording video at 30 FPS induces a maximum sampling delay of $1/30$ s or 33.33 ms. We show the variation of the encoding time with frame capture rate (FPS) in Fig. 3a. Note that for the 720p resolution curve, it stops at 24 FPS because the RPi processor is not capable of processing higher frame rates.

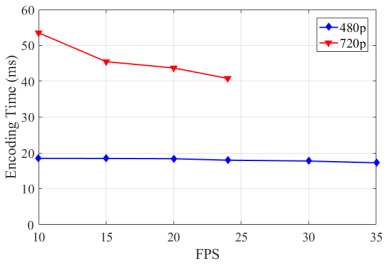
2) Encoding: The encoding delay of H.264 is determined by several factors, but is perhaps dominated by the motion correction component. We modify encoder behavior by changing parameters such as FPS, constant rate factor (CRF) value, resolution, and the presets of FFmpeg.

The CRF is a quality and bitrate control setting for the H.264 encoder. The value ranges from 0 to 51 with lower values giving a better quality video at the expense of a higher bitrate, i.e., CRF 0 signifies lossless video, and CRF 51 signifies maximum compression. The specified CRF factor determines the (roughly) constant output bitrate and perceived video quality. The effect of CRF value on encoding delay is presented in Fig. 3c. Note that lower CRF values result in higher video quality and result in higher encoding delays. Further, the overall trend remains the same irrespective of the processor used.

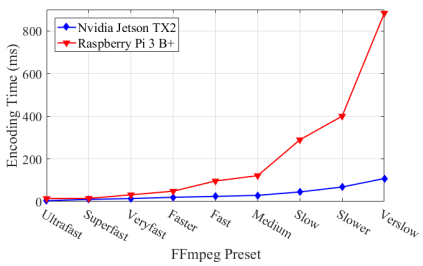
FFmpeg provides collections of settings for video encoding in the form of presets. Each preset adopts different strategies for encoding and presents a trade-off between compression ratio and bitrate. For instance, a slower preset will provide better compression resulting in lower bitrate than higher presets for the same video quality. The effect on encoding time with preset is presented in Fig. 3b for FPS 30, CRF 24 and 480p resolution.

3) Network: Since different parameters result in different bitrates, the encountered network latency changes. Note that this component of delay is stochastic and varies with the channel conditions, channel occupancy, size of each packet, and the bitrate of each transmission. Fig. 4 illustrates how network latency varies with CRF for 480p and 720p video resolution. The indoor values were obtained in the lab environment with line of sight. A lower CRF value results in better video quality but results in higher network latency and packet losses. In Fig. 4, the outdoor setting readings for CRF values below 24 have not been considered since significant packet drop was observed resulting in extremely poor video quality.

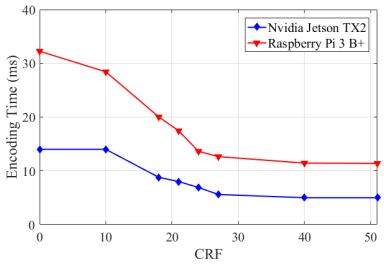
4) Decoding and Rendering: Video frame decoding and rendering is the final step in the video transmission process



(a) Encoding Time vs FPS at 480p/720p , CRF 24 and 'Ultrafast' Preset on RPi



(b) Encoding Time vs FFmpeg Preset of two processors at 480p, CRF 24 and 30 FPS.



(c) Encoding Time vs CRF of two processors at 480p, 30 FPS and 'Ultrafast' Preset.

Fig. 3: Encoding time variation when changing different parameters.

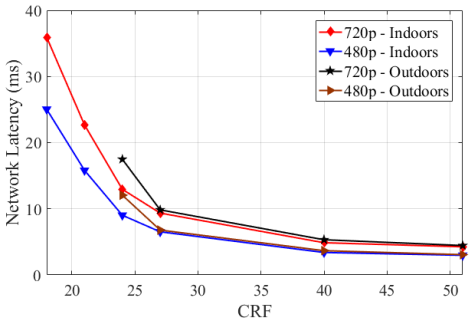


Fig. 4: Network latency vs CRF at 480p/720p, 30 FPS and 'Ultrafast' preset on RPi.

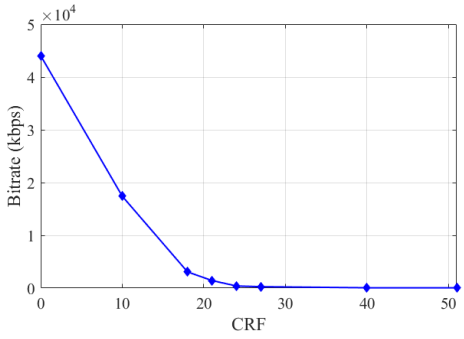


Fig. 5: Output Bitrate vs CRF at 480p, 30 FPS and 'Ultrafast' preset on RPi.

before the video is displayed to the remote terminal. Before being decoded and rendered, frames are buffered in a queue which can build up and add delay. During the course of our experiment, the unoptimized decoding time is found to be below 10 ms initially and is observed to steadily increase and stagnate around 50 ms after a while. The rendering time is approximately 100 ms.

5) Impact of processor type: We conduct our experiment on both RPi and TX2 CPU platforms, where the latter includes a more powerful processor and a graphics card. We observe that irrespective of the processor, the trends of encoding time and decoding time are similar. The comparison between the performance of these processors has been illustrated in Fig. 3c and Fig. 3b.

B. Customization of parameters

The constituent parts described in Fig. 1 have inter-dependencies, and a trade-off is essential to reduce the overall latency. We describe below our customized choices for the tunable parameters of the previous section for achieving low latency.

1) FPS (Frames per second): FPS has an upper bound since the computational capability of the encoding processor is limited. While the reduction in encoding time with an increase in FPS for 480p resolution is minimal, there is a significant drop in encoding time for 720p resolution (Fig. 3a). The RPi does not handle an FPS above 24 for 720p resolution since the encoding time exceeds the sampling time. Hence, 480p resolution is considered and the best results are obtained for 30 FPS.

2) CRF: The CRF value is primarily concerned with the encoding time. However, it also has an impact on network latency because of its inverse variation with output bitrate (Fig. 5). The encoded video above CRF 35 is not suitable for control through remote driving. Since there is minimal variation in the encoding time from CRF 28 to 35, CRF 28 is considered as the optimum value to achieve a balance between encoding time and video quality. As seen in the graph, there is an exponential increase in bitrate as the CRF value decreases beyond 18. Keeping the scalability issue in mind, a single UGV cannot be permitted to operate across a bandwidth this large.

3) FFmpeg Preset: The use of a faster preset decreases the encoding time for both the processors, albeit leading to a lower compression ratio. It is found that the 'Veryfast' preset is the best compromise between output bitrate and encoding time.

4) Decoding and Rendering Algorithm: To reduce the decoding delay, we reduce the size of the decoder frame buffer (in the FFmpeg source) to 1 frame from the default of 3 frames. This is observed to not significantly affect the rendered video output.

V. End-to-End Latency Optimizations and Conclusions

Guided by our latency analysis of video codec outlined in the previous section, we find that the best performance for

	Latency value (ms)
Maximum Sampling	33.33
Encoding	13.8 ± 2.79
Network	12.4 ± 3.825
Decoding & Rendering	12.16 ± 3.03
TOTAL	71.68 ± 5.31

TABLE II: Optimized latency values for video transmission over uplink

	Default (ms)	Optimized (ms)
Scanning	143.88 ± 9.76	54.5 ± 4.47
Roaming	41.75 ± 8.01	26 ± 8.33
TOTAL	186.63 ± 12.32	80.8 ± 8.53

TABLE III: Delays during handoffs

FFmpeg running over an RPi is attained for 30 FPS, 28 CRF, 'Veryfast' preset and 480p video resolution. FFmpeg with these settings, along with decoding buffer-size changes, attains 67.14% reduction in latency in comparison to the default values; see Table II.

Furthermore, we configure the Wi-Fi network to introduce selective scanning, reduce the $T_{\max}$ value appropriately, and deploy IEEE 802.11r to reduce the communication delays to as shown in Table III. The total latencies during regular operation and handoff before and after optimization are given in Table IV. Also, when we use IEEE 802.11p for communicating control messages over the downlink, the latency reduces further to 2.8309 ms; a complete integration of 802.11p link in our setup is ongoing.

Finally, we measure the end-to-end delay between triggering a red light at the UGV and receiving the stop command in the experiment described in Section II. The total time taken to complete this process is observed to be around 91 ms with a downlink latency component of approximately 9 ms. We note that this low downlink latency is attained after switching to UDP communication and setting the highest preference for UDP packets. Further, in normal operation, the codec and network delays are of roughly the same order.

VI. Acknowledgements

This work was supported in part by the 5G Testbed Project of the Department of Telecommunications (DoT), Government of India, in part by the Robert Bosch Centre for Cyber-Physical Systems (RBCCPS) at the Indian Institute of Science, and in part by the Centre for Networked Intelligence (a Cisco CSR initiative) of the Indian Institute of Science. We thank Alok Rawat and Chetan Kumar for help with WiFi network deployment, Venkatesh Bharadwaj and Prateek Jha for the setup of the mobile UGV, and B. Srikrishna Acharya and Raghava G.D. for useful discussions.

	Default		Optimized	
	Regular operation (ms)	Handoff (ms)	Regular operation (ms)	Handoff (ms)
Uplink	210 ± 16.83	396 ± 12.14	71 ± 5.31	149 ± 5.85
Downlink	12 ± 3.45	198 ± 7.55	9 ± 2.93	89 ± 5.67
Processing	13 ± 1.44	13 ± 1.44	13 ± 1.44	13 ± 1.44
TOTAL	235 ± 16.91	606 ± 15.25	93 ± 5.89	251 ± 7.52

TABLE IV: The end-to-end delay

References

[1] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular internet access using in situ wi-fi networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '06, 2006, pp. 50–61.

[2] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal, "Vehicular opportunistic communication under the microscope," ser. MobiSys '07, 2007, pp. 206–219.

[3] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan, "Interactive wifi connectivity for moving vehicles," ser. SIGCOMM '08, 2008, pp. 427–438.

[4] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular content delivery using wifi," ser. MobiCom '08, 2008, pp. 199–210.

[5] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 93–102, Apr. 2003.

[6] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, "Reducing mac layer handoff latency in ieee 802.11 wireless lans," in *ACM Int. Works. Mobil. Mgmt. Wireless Acc. Prot.*, 2004, pp. 19–26.

[7] Y. Liao and L. Gao, "Practical schemes for smooth mac layer handoff in 802.11 wireless networks," in *Int. Symp. World Wireless, Mobile Multim. Net.*, 2006, pp. 181–190.

[8] G. Athanasiou, T. Korakis, and L. Tassiulas, "Cooperative handoff in wireless networks," in *IEEE Int. Symp. Personal, Indoor Mobile Radio Commun. (PIMRC)*, 2008, pp. 1–6.

[9] H. Ahmed and H. Hassanein, "A performance study of roaming in wireless local area networks based on ieee 802.11r," in *IEEE Biennial Symp. Commun.*, 2008, pp. 253–257.

[10] S. Feirer and T. Sauter, "Seamless handover in industrial wlan using ieee 802.11k," in *IEEE Int. Symp. Indust. Electr. (ISIE)*, 2017, pp. 1234–1239.

[11] A. Paier, R. Tresch, A. Alonso, D. Smely, P. Meckel, Y. Zhou, and N. Czink, "Average downstream performance of measured IEEE 802.11p infrastructure-tovehicle links," in *IEEE Int. Conf. Commun. (ICC)*, May 2010.

[12] D. Murray, M. Dixon, and T. Koziniec, "Scanning delays in 802.11 networks," in *IEEE Int. Conf. Next Gen. Mobile App. Serv. Tech. (NGMAST)*, 2007, pp. 255–260.

[13] K.-H. Chi, C.-C. Tseng, and Y.-H. Tsai, "Fast handoff among ieee 802.11r mobility domains," *J. Info. Sci. Engg.*, vol. 26, no. 4, pp. 1345–1362, 2010.

[14] A. Jafari, S. Al-Khayatt, and A. Dogman, "Performance evaluation of ieee 802.11 p for vehicular communication networks," in *IEEE Int. Symp. Commun. Sys. Net. Digital Signal Process. (CSNDSP)*, 2012, pp. 1–5.

[15] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[16] M. B. Dissanayake and D. L. Abeyrathna, "Performance comparison of hevc and h. 264/avc standards in broadcasting environments." *J. Info. Process. Sys.*, vol. 11, no. 3, 2015.

[17] FFmpeg Developers, "ffmpeg tool (Version be1d324) [Software]," 2016, Available from http://ffmpeg.org/.

[18] L. Merritt and R. Vanam, "Improved rate control and motion estimation for h. 264 encoder," in *IEEE Int. Conf. Image Process. (ICIP)*, vol. 5, 2007, pp. V–309.