Real-Time Liquid Wireless Transport for Video Streaming in Rural and Agricultural Applications

E. K. A. Permatasari[†], E. Gossling[†], M. Nadim[†], S. Babu[†], D. Qiao[†], H. Zhang[†], M. Luby[‡§], J. W. Byers[¶], L. Minder[§], and P. Aggrawal[§]
{erma,evang,nadim,sarath4,daji,hongwei}@iastate.edu,{luby,lorenz,pooja}@bitripple.com,byers@bu.edu
[†]Iowa State University, [‡]ICSI, [¶]Boston University, [§]Bitripple Inc., USA

ABSTRACT

Wireless networks are essential building blocks of rural broadband. However, rural wireless is subject to both environmental factors and complex, fast-varying network dynamics and uncertainties. Supporting advanced applications with stringent throughput and latency guarantees (e.g., 360° 4K live streaming for farming observations) in such challenging networks demands whole-stack innovations including those at the transport layer. To this end, we propose the Liquid Wireless Transport Layer (LiqTL) that enables low-latency, data-intensive video streaming services over heterogeneous wireless networks, and, in this work, we focus on the *Liquid Processing Module (LiqPM)* that leverages the state-of-the-art RaptorQ fountain code to enable real-time, reliable data transport over lossy wireless networks. We implement a baseline prototype of LiqTL in the ARA wireless living lab and deploy a mobile 360° 4K video streaming application to evaluate its performance. The robustness of LiqTL is evaluated in terms of quality of experience (QoE) through the metric of displayed frames per second (FPS). Our results show that LiqTL over a single path improves the FPS as compared with alternative solutions, and LiqTL over multiple paths can provide a stable performance around 28–30 FPS in challenging environments.

CCS CONCEPTS

• **Computer systems organization → Real-time system architecture**; • **Networks → Network reliability**.

KEYWORDS

Real-Time Data-Intensive Video Streaming, Liquid Wireless Transport Layer (LiqTL), Liquid Processing Module (LiqPM), ARA Wireless Living Lab

ACM Reference Format:
E. K. A. Permatasari[†], E. Gossling[†], M. Nadim[†], S. Babu[†], D. Qiao[†], H. Zhang[†],, M. Luby[‡§], J. W. Byers[¶], L. Minder[§], and P. Aggrawal[§]. 2024. Real-Time Liquid Wireless Transport for Video Streaming in Rural and Agricultural Applications . In *Mile-High Video Conference (MHV '24), February 11–14, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3638036.3640806

ACM ISBN 979-8-4007-0493-2/24/02...$15.00
https://doi.org/10.1145/3638036.3640806

1 INTRODUCTION

Supporting reliable, real-time, and high-quality video streaming services over wireless networks for agricultural use-cases in rural areas poses unique challenges. The diverse environmental factors such as weather, terrain, foliage, and crop types and densities heavily constrain the performance of wireless networks. In addition, interactive rural and agricultural use cases such as teleoperation necessitate very low latencies, in the range of milliseconds. Therefore, off-the-shelf application protocols specifically designed for video streaming over TCP, such as RTMP and HTTP Live Streaming (HLS) protocols [10], which introduce delays up to seconds, are unsuitable. To address these challenges, we propose the Liquid Wireless Transport Layer (LiqTL) framework to enable reliable, real-time, and high-throughput data transport across complex wireless networks in the presence of fast-varying network dynamics and uncertainties.

Our LiqTL framework uses the state-of-the-art RaptorQ fountain code [7, 11] to provide high reliability and maintain low latency while delivering streaming services over wireless links in challenging network environments. At a high-level, the LiqTL framework consists of three key building blocks: a liquid processing module (LiqPM); a rate adaptation module; and liquid data traffic steering, switching, and splitting (LiqTSSS) functionality. As a first step towards realizing LiqTL, this study focuses on the design and implementation of the LiqPM building block.

LiqPM utilizes the RaptorQ encoder to transform application-generated data into liquid data [1]. Packets containing liquid data are then transmitted through heterogeneous wireless interfaces via end-to-end tunnels to the LiqPM receiver. Our LiqPM enables low-latency and efficient data delivery by taking advantage of liquid data's key properties [11]. Namely, (1) RaptorQ encoder can efficiently generate as much liquid data as needed on the fly; and (2) RaptorQ decoder can recover source data blocks efficiently from any subset of received liquid data of sufficient cardinality, which occurs when the amount of received liquid data is equal to or slightly more than the size of the application-generated data. These two properties enable reliable, real-time data transport with low overhead. Notably, and unlike retransmission-based mechanisms to ensure reliable transport, LiqPM does not incur the delay introduced by detecting data loss and then retransmitting lost data.

Using the ARA wireless living lab [16], we implement LiqTL and evaluate its performance in real-world agriculture farm settings. Our previous study of single path LiqTL [7] has demonstrated a robust performance for supporting real-time, data-intensive applications. This shows that the use of RaptorQ codes and liquid data in LiqTL helps combat the dynamics and uncertainties in wireless networks. LiqTL also natively supports the use of heterogeneous wireless networks to further improve the reliability, timeliness, and throughput of data transport, by leveraging the diversity provided by different wireless networks. Our real-world measurement study in this paper shows that multipath LiqTL significantly improves the user-perceived quality of experience (QoE) for real-time video streaming applications.

Compared with other multipath transport layer approaches [2, 3, 5, 9, 15], LiqTL is unique in the following respects. First, LiqTL leverages RaptorQ and liquid data to proactively ensure reliable, real-time data transport with low overhead, thus reducing the delay introduced in retransmission-based protocols. Secondly, the nature of liquid data enables LiqTL to support seamless handover under high mobility and wireless channel uncertainties. Thirdly, we evaluate the performance of LiqTL in supporting real-time video streaming applications in real-world farm settings, and we demonstrate the strength of LiqTL in real-time, high-throughput data transport in the presence of mobility and uncertainties in rural wireless networks.

In summary, our contributions in this paper are as follows. We first propose the LiqTL framework that leverages the RaptorQ code and liquid data for reliable, real-time, and high-throughput data transport in wireless networks; Next we implement a field deployable prototype of LiqPM, a key LiqTL building block for liquid data processing, and the prototype can work with both a single network interface and multiple network interfaces to support reliable, real-time transport of liquid data; And finally, we provide an experimental evaluation framework for LiqTL over 5G rural wireless networks, and evaluate the performance of LiqTL in supporting real-time video streaming services for precision agriculture.

The remainder of this paper is organized as follows. We present the LiqTL framework in Section 2 and detail the design and implementation of LiqPM in Section 3. We describe the real-world LiqTL implementation over 5G networks and the experimental setup in Section 4, and we evaluate the performance of LiqTL in Section 5. Section 6 concludes the paper.

2 REAL-TIME LIQUID WIRELESS TRANSPORT

In what follows, we first present the intuition behind leveraging the RaptorQ codes in the design of the Liquid Wireless Transport Layer (LiqTL), then present the architecture of LiqTL and detail its key modules for enabling real-time, reliable, and data-intensive data transport over wireless networks.

2.1 RaptorQ Code

In our current design of LiqTL, we use the state-of-the-art RaptorQ codes [8, 11] to enable real-time, reliable data transport. RaptorQ is based on LT codes [6], and it is one of the most efficient fountain codes available today.

At a sender, a source block of data is partitioned into source symbols, from which the RaptorQ encoder generates repair symbols. Source and repair symbols are sent to a receiver in the payloads of UDP packets. We use the term *liquid data* to refer to the combination of source symbols and repair symbols, because these symbols have properties analogous to those of a liquid [7].

At the receiver, the RaptorQ decoder is used to reconstruct the source block from any sufficient number of received source and repair symbols, i.e., from any sufficient amount of received liquid data. RaptorQ has the following key properties:

(1) *Expandability*: A RaptorQ encoder can generate as ma-ny repair symbols as needed on the fly from a source block, i.e., as much liquid data as needed can be generated.
(2) *Interchangeability and low overhead*: A RaptorQ decoder can recover a source block from any portion of liquid data that is the size of the source block; in rare cases slightly more liquid data is required.
(3) *CPU efficiency*: We use a RaptorQ implementation that provides linear time encoding and decoding. This allows low CPU utilization to encode or decode any size source blocks.

2.2 Liquid Wireless Transport Layer (LiqTL)

Towards reliable, high throughput, and low-latency wireless data transport, we propose the Liquid Wireless Transport Layer (LiqTL) as a layer between the applications and UDP in the network stack. We implement LiqTL in user-space to enable seamless deployment to existing applications. LiqTL takes advantage of the existing UDP transport protocol, allowing liquid data to traverse middleboxes in the Internet. Three key functions of LiqTL include the following: (1) For each given application data block, generate liquid data as needed; (2) Adapt liquid data generation rate based on in-situ network conditions; and (3) Determine the network interface used to transmit liquid data for the overall optimization of liquid data transport in the network with a multitude of applications.

Fig. 1 depicts the design of LiqTL. The key building blocks of LiqTL are as follows:

(1) *Liquid processing module (LiqPM).* At the sender side, LiqPM is responsible for transforming application data into liquid data ready to be transmitted over the network. At the receiver side, LiqPM recovers/decodes the application data from the liquid data received. RaptorQ encoder and decoder are used at the sender and receiver side, respectively, to perform the involved key functions.
(2) *Rate adaptation.* To optimize overall network utility in delivering real-time data from all the applications and users, LiqTL adjusts the amount of liquid data sent to the network depending on in-situ network conditions.
(3) *Traffic steering, splitting, and switching (LiqTSSS).* In a network where heterogeneous network interfaces are available, it becomes crucial to determine the interface to which each piece of liquid data should be transmitted over. To ensure reliable, high-throughput, and low-latency data transport and to optimize the overall network performance, LiqTSSS carefully considers factors such as in-situ wireless link performance and end-to-end path congestion in making decisions on the amount of liquid data to be delivered through the individual interfaces over time. The nature of liquid data [7] enables LiqTSSS to natively support seamless handover under mobility and wireless channel uncertainties.

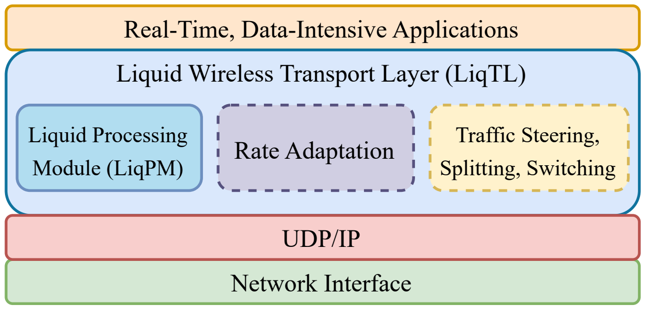
Real-Time, Data-Intensive Applications
Liquid Wireless Transport Layer (LiqTL)
Liquid Processing Module (LiqPM)
Rate Adaptation
Traffic Steering, Splitting, Switching
UDP/IP
Network Interface

Figure 1: Design of Liquid Wireless Transport Layer (LiqTL).

As a first step towards realizing LiqTL, this study focuses on implementing and evaluating the base building block Liq-PM, with rate adaptation and LiqTSSS as part of our future work.

3 LIQUID PROCESSING MODULE

As far as our liquid networking framework is concerned, the sender LiqPM (S-LiqPM) handles the real-time application data intended for the receiver. S-LiqPM generates liquid data from the application data using the RaptorQ encoder and determines the rate at which liquid data shall be sent to the receiver. Similarly, the receiver LiqPM (R-LiqPM) recreates the application data from the received liquid data utilizing the RaptorQ decoder. Moreover, R-LiqPM continuously sends feedback to the sender with the state of the received liquid data to assist S-LiqPM in making decisions. Such a feedback is useful to adjust the amount of fountain encoded data to be generated depending on real-time network conditions, but again note that these control packets are **not** used to facilitate (latency-inducing) retransmissions. To enable the seamless integration of LiqPM into the existing network stacks and applications, we make use of the tunnel interface for liquid data exchange between S-LiqPM and R-LiqPM. Fig. 2 illustrates the high-level view of the end-to-end LiqPM workflow.

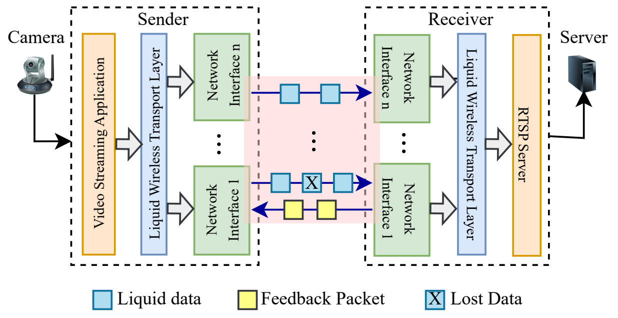

Figure 2: The proposed end-to-end LiqTL architecture. Any video streaming framework such as GStreamer [12] can be integrated to our solution to provide video streaming services.

3.1 LiqPM Implementation at Sender Side

Fig. 3 illustrates our current implementation of LiqPM at the sender side (S-LiqPM) and its key functions: (1) application-data interception and source block generation, (2) RaptorQ encoding, and (3) liquid data packetization.

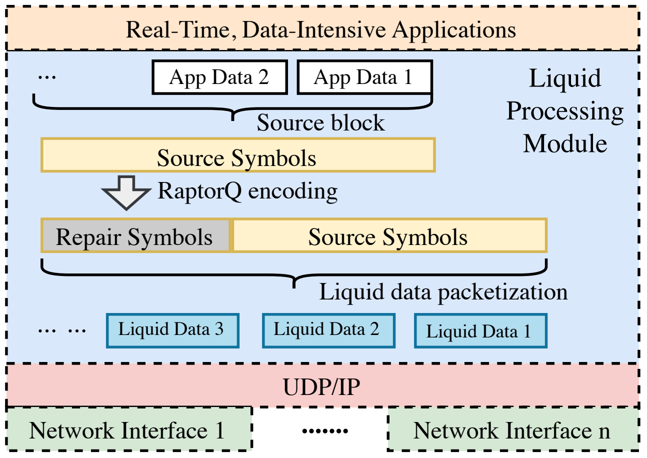

Figure 3: Current implementation of the Liquid Processing Module (LiqPM) at the sender side.

First, S-LiqPM intercepts the application data that has been processed in its interface. The data packets intercepted for a specific duration of time (defined as the *LiqPM block timer* and is set to 5 ms in our experiments) are taken as the input to the source block generator. The intercepted data packets are concatenated to generate the *source block*. Then, the source block is taken as the input to the RaptorQ encoder to generate *repair symbols*, which together with the source symbols form the *liquid data*. After that, the liquid data undergoes packetization, i.e., each source and repair symbol is copied into the payload of a UDP packet (with an appropriate header added) and sent to the receiver.

The timeout value of the LiqPM block timer can be tuned depending on the network conditions as well as the requirements of the user application. Currently, we use a static value for the block timer that is configurable by the application at execution time. The setting of this value highlights an important tradeoff. The smaller the value of the block timer, the shorter each data block tends to be. This means that with bursty loss patterns (typical), the per-block packet loss will also be more bursty. Thus more protection (more RaptorQ encoding) must be sent, which leads to more bandwidth usage. Larger values of the block timer, on the other hand, mitigate bursty losses better, but introduce proportional latency to enable block recovery.

3.2 LiqPM Implementation at Receiver Side

At the receiver side, R-LiqPM reverses the sequence of functions discussed in S-LiqPM. On receiving liquid data, R-LiqPM performs liquid data depacketization, i.e., uses the UDP header to identify the liquid data carried in each packet. As soon as the R-LiqPM receives the required amount of liquid data, it runs the RaptorQ decoder to recover the source block. Recall that R-LiqPM continuously sends feedback packets to S-LiqPM that include: (1) the highest sequence number received so far, (2) the total number of UDP packets carrying liquid data that have been received, and (3) the amount of liquid data required for recovering the current source block. Once the decoder is able to recover the source block, the original set of data packets are re-generated, which are then handed over to the application.

3.3 LiqPM Multipath Extension

Characteristics of rural environments coupled with user mobility may result in frequent wireless channel fluctuations, a problem for immersive applications. Consequently, it becomes vital to enable network services that can leverage multiple network connections to improve resilience, throughput, and latency. By leveraging inherent features of the RaptorQ fountain code such as expandability and interchangeability, the LiqPM instance can effortlessly extend its functionality beyond a single network interface [7].

Our current design of LiqPM to accommodate multiple network interfaces is shown in Fig. 3. Similar to the single interface model, the multipath version of LiqPM also employs three main functions. The sender and receiver exchange liquid data over multiple paths through a single tunnel interface. The only addition to be made at S-LiqPM in the multipath extension is to specify the distribution of liquid data sent over each interface.

Finally, in order to allow S-LiqPM to collect accurate network statistics and tune the distribution of liquid data among multiple interfaces in real-time, reliability on the feedback path is also a consideration. The feedback mechanism presented in this paper is an initial implementation that attempts to send feedback along the most reliable path; a more robust multi-path feedback scheme is in our plan for future work. In our current prototype implementation, we distribute the liquid data in proportion to the maximum capacity offered by each path. For instance, if Path-1 (through Interface-1) offers 40 Mbps and Path-2 (through Interface-2) offers 60 Mbps, the liquid data is distributed in the ratio of 2:3 between interfaces 1 and 2. Dynamic tuning of liquid data distribution among multiple interfaces is part of future work.

4 EXPERIMENTAL SETUP

To demonstrate the effectiveness of the proposed LiqTL solution, we deployed and evaluated it in the ARA Wireless Living Lab [16], an at-scale experimental testbed deployed in Central Iowa. ARA is part of the NSF Platforms for Advanced Wireless Research (PAWR) program, and the goal is to enable research and development of rural-focused wireless technologies. A key feature of the ARA platform is the availability of nodes enabled with multiple network interfaces operating at multiple frequency bands such as TV Whi-te Space (TVWS) (460–776 MHz), mid-band (3.4–3.6 GHz), and mmWave (27.50–28.35 GHz).

4.1 Hardware

For the experiment, we set up a mobile farm observation system using an Insta360 Pro 2-360° VR camera equipped with six lenses, each streaming 4K video to the data center at 30 frames per second (FPS) with a maximum bit rate of 60 Mbps. The camera was mounted on the rooftop of a truck as shown in Figure 4. During the experiment, we drove the truck across the field under varying network conditions. The truck is equipped with an ARA User Equipment (UE) consisting of multiple wireless radios operating under different frequency bands, thus providing network heterogeneity. For streaming the video from the camera, we make use of two wireless radios: (1) a Skylark massive MIMO (mMIMO) [4] CPE module operating in TVWS band offering up to 25 Mbps uplink (UL) and 100 Mbps downlink (DL) over a distance of 10 km, and (2) a COTS Quectel module operating in the mid-band Ericsson network offering up to 100 Mbps UL and 600 Mbps DL over a distance of 4 km. A Supermicro X12 compute node, equipped with Intel Xeon D-1736NT processor and running Ubuntu 20.04 LTS Server operating system and is installed inside the UE box, hosts both the wireless radios and the camera. Fig. 4 shows the UE box, the wireless radios, antennas, and the camera. On the other end of the wireless links from the UE, we have two base stations (BSes): (1) a Skylark BS at Wilson Hall and (2) an Ericsson BS at Curtiss Farm. The Skylark mMIMO component of UE connects to the BS at Wilson Hall while the Quectel COTS UE module connects to the Curtiss Farm BS. A logical diagram of the end-to-end experiment framework is shown in Figure 5.

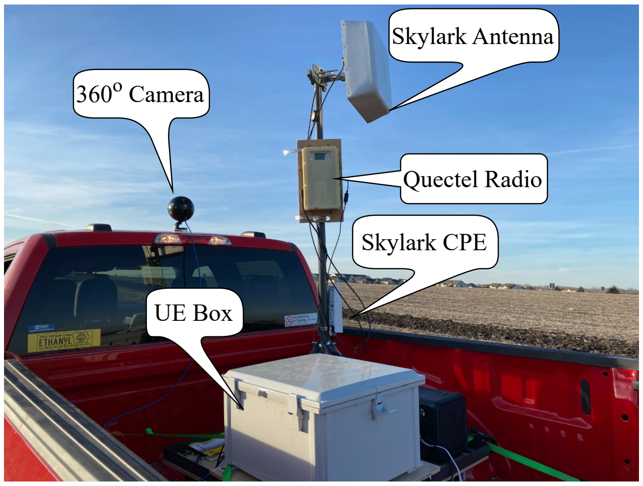

Figure 4: Mobile UE used in the experiment.

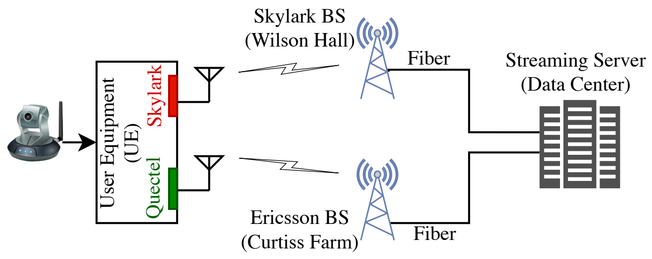

Figure 5: End-to-end view of farm observation application.

4.2 Software

For the farm observation, we livestream the video from the UE in the field to the streaming server at the data center via Real Time Streaming Protocol (RTSP) using the GStreamer framework [12]. The GStreamer RTP plugin takes the stream from the camera and sends it to the data center over multiple wirelress links. Once reaching the BS, the traffic is routed to the data center over the fiber backhaul. Users can watch the video from the streaming server using any of the RTSP-enabled applications such as Open Broadcaster Software (OBS) studio [13]. In fact, the XR applications can make use of our framework to view the video from XR headsets with the help of RTSP/Real-Time Messaging Protocol (RTMP)/HTTP Live Streaming (HLS) protocol. The liquid processing module establishes a liquid tunnel between the UE and the streaming server, and implements both RaptorQ encoding and multipath support without requiring any changes to the existing applications. The tunnel enables ultra-reliable low-latency communication over multiple network network interfaces, enabling multipath communication over two different 5G networks, as explained in Section 3.

4.3 Experiment Description

The experiment involves driving the truck carrying the UE through the Curtiss Farm field as shown in Fig. 6, while the UE continuously streams the video to the data center over single/multiple wireless links. The truck starts at Point A and stops at Point E (which are 250 m apart), as denoted in the zoomed-in region of Fig. 6, moving at a constant speed of about 8 km/h to simulate the slow speed of agriculture vehicles in the farm field. The overall duration of an experiment trial is around 120 seconds. For Skylark, except between Points C and D, the UE is in line-of-sight (LoS) with Wilson Hall BS. On the other hand, the COTS UE (Quectel) module is in LoS with Curtiss Farm BS only from Point A to Point B, while for the other parts of the route, Quectel is in non-LoS (NLoS) due to foliage. It is important to note that between Points C and D, both Skylark and Quectel are in NLoS. During the experiment, we collect the performance metrics using *GstPerf* [14].

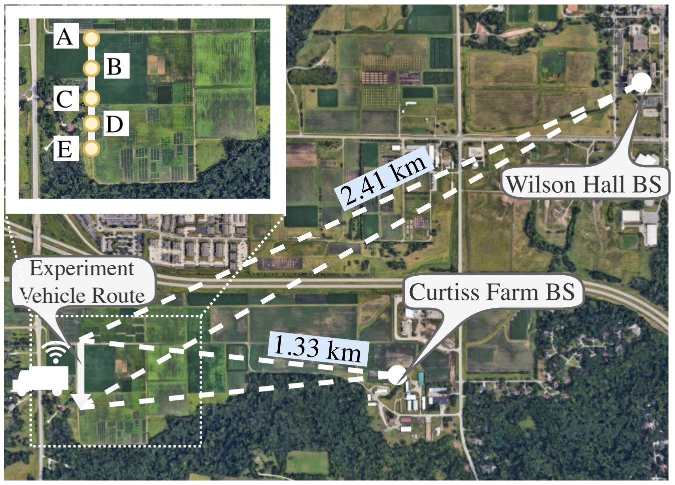

Figure 6: A field view of experimental settings.

5 PERFORMANCE EVALUATION

To evaluate the performance of LiqTL, we use Frames per Second (FPS)—the number frames successfully received and displayed at the receiver per second—as the performance metric of Quality of Experience (QoE). We choose the experiment location in such a way to demonstrate the benefits of using liquid encoding under single-path and multipath settings. As discussed in Section 4.3, the foliage makes Quectel operate with the Ericsson mid-band network under link disruptions, thereby forcing the video streaming application to depend more on the Skylark link. Since Skylark does not provide sufficient bandwidth in streaming compared to the Ericsson network, the application makes use of liquid encoding to ensure better QoE. We set the video streaming configuration to 4K 2D resolution at 30 FPS with a bit-rate of 15 Mbps for our experiment. To demonstrate the benefits of liquid networking, we include five modes of operation in our experiment: (a) Liquid over multipath, (b) Liquid over Quectel, (c) Quectel without liquid, (d) Liquid over Skylark, and (e) Skylark without liquid.

5.1 Evaluation Results

Fig. 7 plots the CDF of FPS from our video streaming application under different modes of operation during an experiment trial. Among the five modes, Liquid over multipath yields the best QoE with FPS almost always higher than 27. It is also important to note that the modes operating with liquid support offer better quality than their non-liquid counterparts. As far as the UE wireless modules are concerned, Quectel outperforms Skylark in both liquid and non-liquid modes due to its higher UL/DL bit rates. It is also interesting to observe that the non-multipath modes are insufficient to offer consistent FPS over the course of the experiment, highlighting the significance of liquid-enabled multipath mode in QoE-sensitive real-time streaming applications.

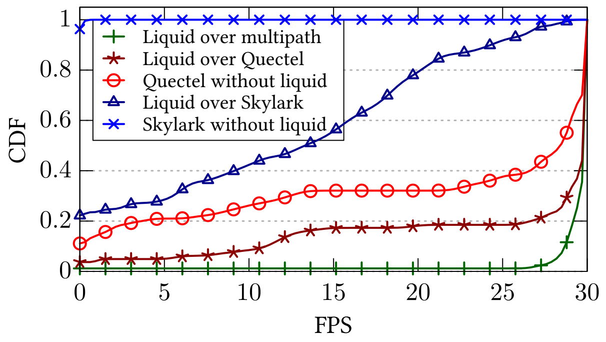

Figure 7: CDF of measured FPS over an experiment trial.

Fig. 8 plots the measured FPS for each operation mode at different UE locations along the experiment route, which starts at Point A and ends at Point E, as shown in Fig. 6. Recall that Skylark enters NLoS between Points C and D due to the water tower, while Quectel remains in NLoS beyond Point B. As a result, we can observe from Fig. 8(c) that, even though the Quectel link may support

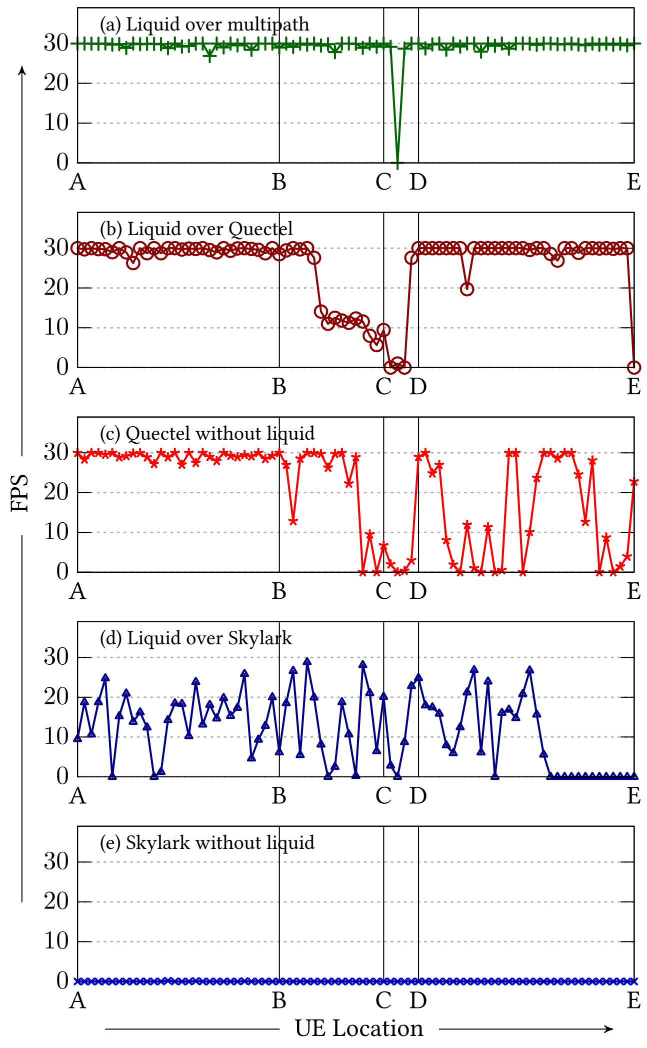

Figure 8: Measured FPS vs. UE location during the same trial as in Fig. 7.

up to 100 Mbps UL under ideal channel conditions, it exhibits unstable FPS performance beyond Point B in our experiment. In comparison, when LiqPM is activated, Quectel was able to compensate some of the packet losses during part of the route, hence improving the FPS, as shown in Fig. 8(b). This effect was particularly noticeable beyond Point D when LoS is recovered. Similar observations can be made for the Skylark link by comparing Figs. 8(d) with (e). In fact, due to its limited UL (≤ 25 Mbps), Skylark alone is unable to sustain any meaningful video streaming, while its performance improves significantly with the liquid support.

In contrast, it is clear that liquid over multipath offers a consistent high FPS except for a brief period of time, between Points C and D, when both Skylark and Quectel are in NLoS (justified by near-zero FPS in both Skylark and Quectel plots between C and D) and both links experience extremely bad channel conditions. In other parts of the route, Skylark and Quectel cooperate with each other in multipath mode to ensure a high FPS.

6 CONCLUSION

In this paper, we propose Liquid Wireless Transport Layer (LiqTL) as an innovative solution to enable real-time, data-intensive applications across wireless networks in challenging environments. We describe the details of the Liquid Processing Module (LiqPM)—a key building block of LiqTL—that leverages the state-of-the-art RaptorQ fountain code to enable real-time, reliable data transport over lossy wireless networks. We evaluate a baseline prototype of LiqTL in the ARA wireless living lab and demonstrate the superior performance of single-path and multi-path LiqTL with a video streaming measurement study in the farm field.

In future work, we will investigate the other two building blocks of LiqTL, namely rate adaptation and LiqTSSS. For both multipath rate adaptation and LiqTSSS, we intend to build on analogous mechanisms in Multipath TCP [3], while exploiting opportunities that are specific to transport of liquid data. We will further investigate the benefits of LiqTL by conducting more experiments under various wireless network conditions and deepen our evaluation with additional performance metrics such as latency, stall ratio, and structural similarity.

ACKNOWLEDGMENT

This work is supported in part by the NIFA award 2021-67021-33775 and NSF awards 2212573, 2112606, and 2130889. We thank Taimoor Ul Islam and Guoying Zu for their contribution during the experimental study.

REFERENCES

[1] J. Byers and M. Luby. 2020. Liquid Data Networking. In *Proceedings of the 7th ACM Conference on Information-Centric Networking* (Virtual Event, Canada) *(ICN '20)*. 129–135. https://doi.org/10.1145/3405656.3418710
[2] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang. 2015. FMTCP: A Fountain Code-Based Multipath Transmission Control Protocol. *IEEE/ACM Transactions on Networking* 23, 2 (Apr. 2015), 465–478. https://doi.org/10.1109/TNET.2014.2300140
[3] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch. 2020. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 8684. https://doi.org/10.17487/RFC8684
[4] T. Islam, T. Zhang, J. Boateng, E. Gossling, G. Zu, S. Babu, H. Zhang, and D. Qiao. 2023. AraMIMO: Programmable TVWS mMIMO Living Lab for Rural Wireless *(WiNTECH '23)*. https://doi.org/10.1145/3615453.3616512
[5] M. Li, A. Lukyanenko, S. Tarkoma, Y. Cui, and A. Ylä-Jääski. 2014. Tolerating path heterogeneity in multipath TCP with bounded receive buffers. *Computer Networks* 64 (May 2014), 1–14. https://doi.org/10.1016/j.comnet.2014.01.011
[6] M. Luby. 2002. LT codes. 271–280. https://doi.org/10.1109/SFCS.2002.1181950
[7] M. Luby. 2023. Enabling Immersive Experiences in Challenging Network Conditions *(MHV '23)*. https://doi.org/10.1145/3588444.3591018
[8] L. Minder, A. Shokrollahi, M. Watson, M. Luby, and T. Stockhammer. 2011. RaptorQ Forward Error Correction Scheme for Object Delivery. RFC 6330. https://doi.org/10.17487/RFC6330
[9] Y. Ni, Z. Zheng, X. Lin, F. Gao, X. Zeng, Y. Liu, T. Xu, H. Wang, Z. Zhang, S. Du, G. Yang, Y. Su, D. Cai, H. Liu, C. Xu, E. Zhai, and Y. Ma. 2023. CellFusion: Multipath Vehicle-to-Cloud Video Streaming with Network Coding in the Wild *(ACM SIGCOMM '23)*. https://doi.org/10.1145/3603269.3604832
[10] R. Pantos and W. May. 2017. HTTP Live Streaming. RFC 8216. https://doi.org/10.17487/RFC8216
[11] A. Shokrollahi and M. Luby. 2009. Raptor Codes. *Found. Trends Commun. Inf. Theory* (Mar. 2009), 213–322. https://doi.org/10.1561/0100000060
[12] Online Source. 2023. GStreamer: Open Source Multimedia Framework. https://gstreamer.freedesktop.org/. (Accessed on 12/01/2023).
[13] Online Source. 2023. Open Broadcaster Software OBS Studio. https://obsproject.com/. (Accessed on 12/01/2023).
[14] Online Source. 2023. RidgeRun/gst-perf: GStreamer element to measure framerate, bitrate and CPU usage. https://github.com/RidgeRun/gst-perf. (Accessed on 12/01/2023).

[15] J. Wu, R. Tan, and M. Wang. 2019. Streaming High-Definition Real-Time Video to Mobile Devices with Partially Reliable Transfer. *IEEE Transactions on Mobile Computing* 18, 2 (Feb. 2019), 458–472. https://doi.org/10.1109/TMC.2018.2836914

[16] H. Zhang, Y. Guan, A. Kamal, D. Qiao, M. Zheng, A. Arora, O. Boyraz, B. Cox, T. Daniels, M. Darr, D. Jacobson, A. Khokhar, S. Kim, J. Koltes, J. Liu, M. Luby, L. Nadolny, J. Peschel, P. Schnable, A. Sharma, A. Somani, and L. Tang. 2021. ARA: A Wireless Living Lab Vision for Smart and Connected Rural Communities *(WiNTECH '21)*. https://doi.org/10.1145/3477086.3480837