2018 10th International Conference on Communication Software and Networks

A Low-Latency Reliable Transport Solution for Network-Connected UAV

Sai Jiang, Qingqing Zhang, Aozhou Wu, Qingwen Liu, Jun Wu, Pengfei Xia
College of Electronics and Information Engineering
Tongji University
Shanghai, China
e-mail: sai.jiang@tongji.edu.cn, anne@tongji.edu.cn, 13146653740@163.com, qliu@tongji.edu.cn, wujun@tongji.edu.cn, pengfei.xia@gmail.com

***Abstract*—Existing Unmanned Aerial Vehicles (UAVs) mainly rely on the direct ground-to-UAV communications over the unlicensed spectrum, which can only operate within the visual line of sight. Integrating UAVs into cellular networks is a promising solution to extend the range but poses new challenges to the transport protocols. To ensure the flight safety, the transmission of UAV commands and mission-related data is required to be reliable with low latency. In this paper, we propose a Low-latency Reliable Transmission (LRT) protocol by exploiting the rateless and online properties of online network coding. We design the LRT, implement it, and evaluate its performance in terms of the in-order per message delay. Experimental results show that LRT can provide low-latency transmission service comparable to that of UDP while giving the same reliability of TCP, which makes it an attractive option for Network-Connected UAV applications.**

Keywords-TCP/IP; high throughput low latency communication; network coding; network-connected UAVs

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have found fast growing applications in recent years, such as for cargo delivery, sky monitoring, traffic control, rescue and search, communication relaying [1]. However, existing UAV systems mainly rely on the direct ground-to-UAV communications over the unlicensed spectrum, which is of limited data rate, unreliable, insecure, vulnerable to interference, and can only operate within the visual line of sight. As UAV applications increase explosively in the coming years, these shortcomings will be magnified. Thanks to the almost ubiquitous accessibility of cellular systems, it is natural to integrate UAVs into cellular networks, referred to as Network-Connected UAVs [2, 3].

By means of cellular networks, Network-Connected UAVs are expected to outperform the traditional ones significantly. However, many new challenges need to be addressed before realizing the promising visions. For instance, we must deliver control commands reliably with low latency to ensure the flight safety. Also, UAVs must transfer a rate-demanding live stream back for monitoring. The traditional transport protocols, such as TCP and UDP, fail to meet the complex requirements. UDP guarantees low latency and high throughput at the cost of no reliability. TCP indeed guarantees the reliability, but the retransmission latency is unacceptable when the end-to-end delay is large. Moreover, the congestion control mechanism in TCP may lead to throughput degradation mistakenly [4]. Instead, rateless codes are good options to meet those requirements [5-8].

As the term implies, rateless codes do not exhibit a fixed code rate, thereby enabling a reliable transmission approach. The sender sends encoding symbols endlessly. The receiver receives these symbols and tries to recover the original symbols. If the decoding operation fails, the receiver receives more symbols, retries the recovery and repeats the process until the original being recovered successfully. Once the source symbols are recovered, the receiver sends an acknowledgement (ACK) to the sender for confirmation. Then, the sender stops encoding, and the entire transmission is completed. Thus, even without the channel status information, rateless codes can complete self-adaptation and achieve reliable transmission, which also contributes to high throughput.

However, most rateless codes are essentially block codes. Block codes take a source block as the coding unit, which is awkward for time-critical applications. As pieces of the original data, source symbols can be sent individually without latency. However, if some symbol is lost during transmission, we still have to accumulate enough source symbols before encoding, which increases the latency. In the worst case, we must wait for the rest $(k\text{-}1)$ source symbols if the first symbol in a k-sized source block is lost. In other words, a large part of the latency comes from the accumulation of enough symbols for the source block. Thus, it is hard for block-based rateless codes to achieve a good balance between latency and throughput [9, 10].

Network coding shares the similar rateless property but offers more flexibility [11, 12]. In this paper, we propose an online network coding based Low-latency Reliable Transmission (LRT) protocol that facilitates Network-Connected UAV applications. This paper is organized as follows. In section II, we give an overview of online network coding. Then, we present the details of our proposed protocol in section III. In section IV, we present the performance evaluation of LRT. Finally, conclusions are given in section V. Fig. 1 illustrates a LRT-integrated Network-Connected UAV. Logically, LRT provides a high-throughput low-latency reliable tunnel for the data exchanged between a UAV and its controller.

978-1-5386-7223-5/18/$31.00 ©2018 IEEE

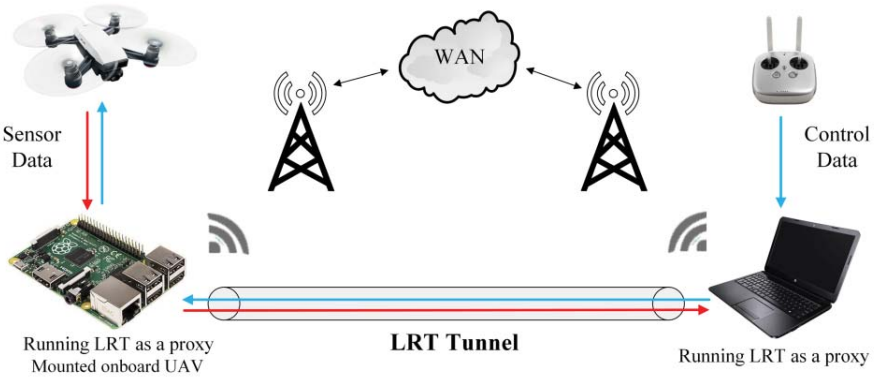

Figure 1. LRT-integrated Network-Connected UAVs

II. ONLINE NETWORK CODING

In principle, network coding generates encoding symbols as linear combinations of source symbols with coefficients selected from a finite field. The simple encoding principle makes it possible for network coding to meet various application demands. As a variant of network coding, online network coding is even capable of on-the-fly coding, which makes it free from the block size restriction. For online network coding, every source symbol can be added to the encoder on the fly. Repair symbols can be generated immediately from the existing source symbols. At the receiver, the fact that the i^{th} symbol is seen by the decoder means that 1) the i^{th} symbol is part of the linear combination given by a new encoding symbol, 2) all the previous $(i\text{-}1)$ symbols have been seen or decoded before, and 3) the new linear combination is linearly independent of the previous combinations [13]. As a result, all the source symbols involved in the encoding process can be recovered, once the number of encoding symbols received is equal to or slightly larger than the number of source symbols. Thus, we can recover the lost symbols with much lower latency, which makes online network coding the ideal alternative to the traditional block-based rateless codes. Fig. 2 gives a summary of the coding principles of both block-based rateless codes and online network coding.

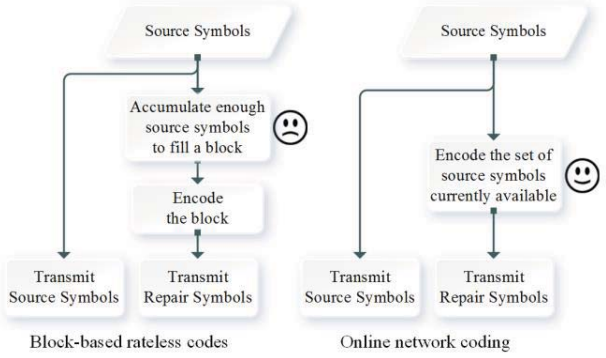

Figure 2. Block coding versus Online coding

Fig. 3 gives an example of using online network coding for transmission. At the beginning, the first source symbol P1 is lost during transmission, which is similar to the worst case of the traditional block codes. A repair symbol, which is a linear combination of P1 and P2, is allowed to be transferred after the successful delivery of P2. The reception of the repair symbol makes P1 being recovered immediately. Then, both P3 and P4 are lost during transmission, and the next repair symbol makes P3 being seen. Next, P6 is lost, and the following repair symbol makes P4 being seen. At last, both P7 and P8 are transferred successfully. The last repair symbol makes all the symbols being seen, thereby recovering P3, P4 and P6 successfully. As a result, the latency of online network coding only depends on the loss rate and the applied redundancy.

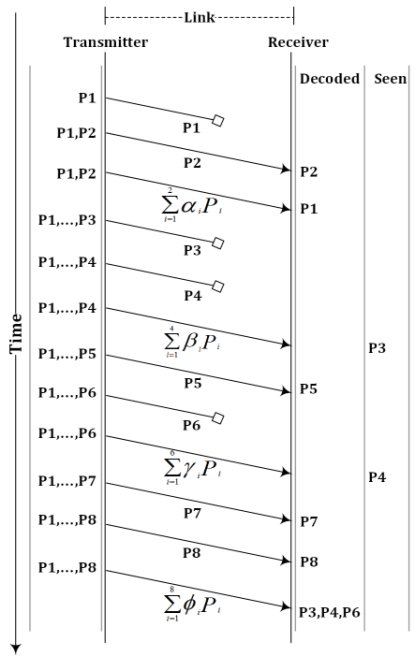

Figure 3. Online network coding example

III. THE LRT PROTOCOL

LRT is a UDP-based application-level reliable transport protocol for latency-sensitive applications over various networks. LRT provides low latency transmission service comparable to that of UDP while guaranteeing reliable in-order delivery like TCP. The time to transfer a message via LRT can be expressed as:

$$T = T_{div} + T_{enc} + T_{e2e} + T_{dec} + T_{rec} \quad (1)$$

where T_{div} is the time required to transform the original message into the equal-sized source symbols, T_{enc} is the encoding latency, T_{e2e} is the inherent end-to-end network delay, T_{dec} is the decoding latency, and T_{rec} is the time required to reconstruct the original message from source symbols. Equation (1) lists the five time-consuming processes of rateless transmission. The following parts are organized accordingly.

A. Preprocessing

In practice, messages are of various size and may not fit equal-sized source symbols perfectly. When we transform a message into one or several source symbols, the last symbol may be partially filled. Thus, the residual space must be handled carefully. A straightforward approach is to fill the residual space with zeros directly, which minimizes the latency but adds significant bandwidth overhead. To get a balance between efficiency and latency, we adopt the strategy that the residual space is filled with zeros when there is no more queued message. Otherwise, the residual space is filled with the data from the next message. To recover the original message, the length of the message is prefixed as part of the data that will be encoded.

B. Coding Scheme

As a stable and high-performance network coding library, Kodo is adopted to build a low-latency reliable transmission protocol in this paper [14]. The specific online network coding implementation provided by Kodo is built on block-based Random Linear Network Coding (RLNC) [15]. As a result, the specific implementation operates like block codes but with the online capability. To initialize the coder instances, both the maximum block size and the symbol size are specified. With the adding of source symbols, the block size expands dynamically until reaching the preset limit. In return, the maximum number of decoded symbols between two consecutive decoding events is limited, which gives an upper bound on the decoding latency.

C. ACK Scheme

For block codes, once the source block is successfully decoded at the receiver, an ACK is fed back indicating that the sender can stop encoding. However, the source block of online network coding expands with the appearance of new symbols, which makes that ACK scheme not applicable. Here we take the concept of matrix rank. For the source symbols that have been added to the encoder so far which form the knowledge space of the encoder, the term encoder rank is used to refer to the number of source symbols. Similarly, for the source symbols that have been seen by the decoder which form the knowledge space of the decoder, the term decoder rank is used to refer to the number of seen symbols. Thereafter, encoding symbols are required only when the encoder knowns more information than the decoder, i.e. when the encoder rank is larger than the decoder rank. The encoder keeps tracking the decoder rank via rank-update ACKs, which are fed back when the decoder sees new symbols.

D. Multiple Coders Support

As there exist multiple coders during transmission, the symbols generated by some encoder should be delivered to the correct decoder, and the rank-update ACKs generated by some decoder should be delivered to the correct encoder as well. During the transmission, a unique ID is given to each newly initialized encoder, which is carried by each encoding symbol later. At the receiver, a new decoder is initialized and assigned with the ID carried by the encoding symbol. Thus, the encoder and the paired decoder share the same ID. The encoding symbols are dispatched to the decoder sharing the same ID. The similar rule is applied to the rank-update ACKs as well.

On the other hand, the encoding symbols may arrive at the receiver after the source symbols being recovered and the paired decoder being released. Therefore, a variable for filtering the out-of-date symbols is maintained by the receiver. The received symbols of which the ID is less than the filtering value can be discarded directly. Unfortunately, the reception of out-of-date symbols suggests that the paired encoder is probably not released due to the lost rank-update ACK. Thus, an extra full-rank ACK is fed back to the encoder. We increase the filtering value by one only when all the source symbols sharing the same value are fully decoded.

E. Rate Control

To fully exploit the rateless property, congestion control mechanisms are omitted. Instead, rate limiting algorithms such as the token bucket algorithm are used to throttle the outgoing flow. Before generating a symbol, the rate limiting algorithm is checked first. If there exists an outgoing opportunity, a symbol is generated, packetized and sent. As source symbols are essentially the user data, they are always transferred as quickly as possible. Then, sending opportunities are distributed evenly among multiple encoders in a round-robin approach. Thus, the outgoing rate limitation should fall within this interval:

$$[src * (1 + r), Bw * \beta] \quad (2)$$

where src is the flow rate of the source symbols, r is the redundancy rate that matches the observed packet loss rate, Bw is the total available bandwidth, and β is a parameter varying between 0 and 1 which is used to reserve some bandwidth for other best-effort traffic.

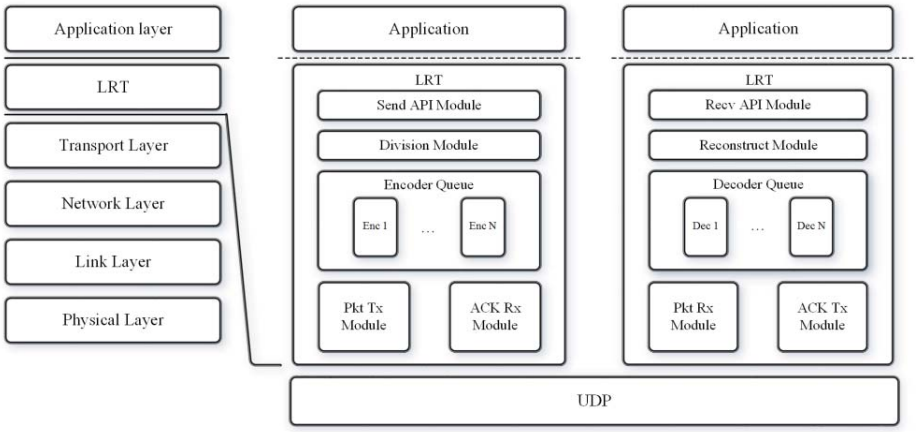

Figure 4. LRT Architecture

F. *Implementation Details*

We have implemented LRT following the architecture shown in Fig. 4. According to the data flow, LRT can be divided into four sublayers. At the top, Send API Module and Recv API Module provide a user interface for transmission. Then, Division Module and Reconstruct Module take the responsibility of transforming messages into equal-sized symbols and the reverse operation. At the middle, all the coder instances are placed in the queues for the convenience of scheduling. At the bottom, Pkt Module and ACK Module deal with the actual transmission using UDP.

IV. EVALUATION

Since high throughput is assured by the rateless property and latency is of utmost importance to the remote control of UAVs, we only evaluate the latency performance of LRT in this section. To reflect the latency, the in-order per message delay is used as our key metric, which captures the elapsed time between submitting the message and fetching the message. Actually, the in-order per message delay consists of two parts: the end-to-end network delay and the latency overhead of the reliability assured by the transport protocols. The end-to-end delay for a real network changes dramatically [16, 17]. To ensure the reproducibility, netem is used to simulate various network conditions [18]. With the constant end-to-end delay, the latency overhead can be measured accurately. According to the QoS specification of the current LTE networks, the simulated end-to-end delay is fixed at 50ms. For convenient, a random generator is used as our data source for transmission, which generates equal-sized messages of 1Kbytes at a constant interval. Moreover, each message contributes to exactly one source symbol.

A. *LRT versus TCP*

In this part, we evaluate the latency performance of both LRT and TCP. Fig. 5 shows the delay distribution of both LRT and TCP at the loss rates of 2%, 5% and 10%. As the loss rate increases, the Cumulative Distribution Function (CDF) curve of either LRT or TCP shifts to the lower right corner, which indicates the latency performance degradation. Since the gaps between adjacent curves are much smaller, LRT has a much better immunity to losses. We also observe that the CDF curves of TCP follow a different pattern from those of LRT. TCP adopts Automatic Repeat reQuest (ARQ) to guarantee reliable transmission. Before realizing the packet loss and being able to retransmit the packet, TCP has to wait for one RTT which is 100ms in this setting. Moreover, due to the requirements of in-order delivery, packets being retransmitted block the subsequent packets that have been successfully transferred, which adds the extra latency. As a result, over 70% of messages transferred by TCP have a delay between 50ms and 150ms. Also, less than 10% have an exact delay of 50ms at the loss rate of 10%. For LRT, the source flow is protected by as many repair symbols as possible, which avoids the retransmission completely. Even at the loss rate of 10%, over 70% of LRT messages still have an exact delay of 50ms. Also, over 99% have a delay between 50ms and 120ms, which outperforms TCP significantly.

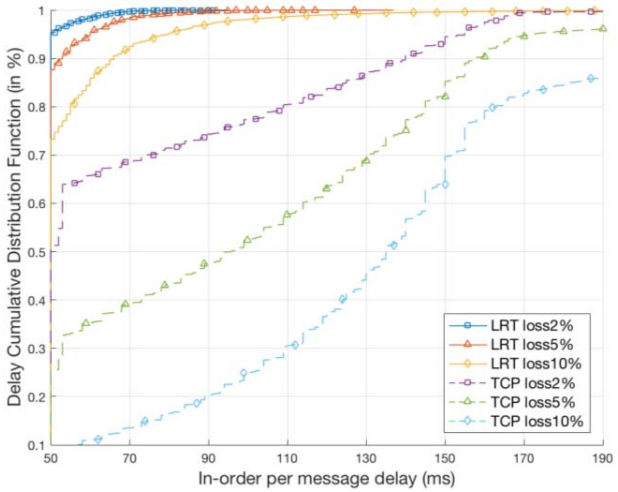

Figure 5. Delay CDF of both LRT and TCP at the loss rate of 2%, 5%, 10%

B. LRT versus Block-Based Scheme

In this part, we compare LRT with a similar scheme that is built on block-based RLNC. For block codes, the block size has a critical influence on the latency performance. To determine the size, we assemble the source symbols accumulated in a timing cycle as a source block. As a result, the size of each source block is dynamically determined according to the source flow rate and the timing cycle. In this case, the timing cycle is fixed at 10ms.

Fig. 6 shows the in-order per message delay of 50,000 messages at the loss rate of 5%. Since LRT is designed to be fully reliable, both LRT and the block-based scheme successfully deliver the separate 50,000 messages. We observe that most messages of the block-based scheme have a delay between 50ms and 80ms. By contrast, most messages of LRT have a delay between 50ms and 60ms, which reduces the maximum latency overhead from 30ms to 10ms significantly. We also observe that LRT has much lower jitter. For the block-based scheme, the delay outliers reach 120ms. However, the delay outliers of LRT are only around 70ms, which enables smoother transmission.

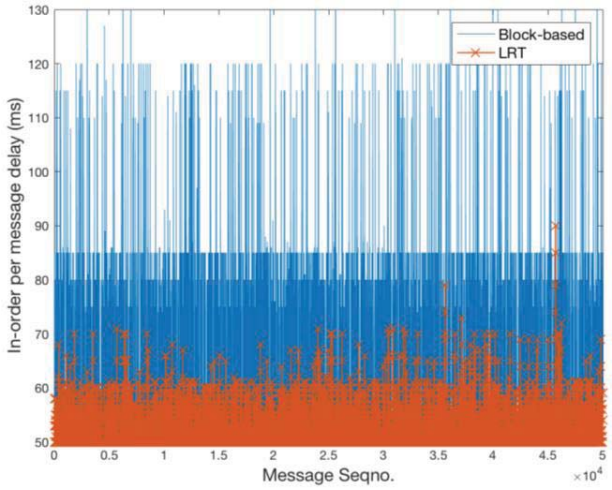

Figure 6. Delay comparison between LRT and Block-based Scheme at the loss rate of 5%

V. CONCLUSIONS

Considering the complex QoS requirements of Network-Connected UAVs, both TCP/UDP and block-based rateless codes fail to give a promising transport solution. In this paper, we propose an online network coding based Low-latency Reliable Transmission (LRT) protocol. Experimental results show that LRT can achieve significantly lower latency without losing the benefit of reliability and high throughput. To support the potential large-scale deployment of LRT, the evaluation of LRT will not be limited to the simulated environments in the feature research. The relevant issues, such as the competition among LRT flows and the interaction between LRT and the underlying networks, will be further studied.

REFERENCES

[1] H. C. Kim, C. S. Lim, C. S. Lee, and J. H. Choi, "Introduction of Real-Time Video Surveillance System Using UAV," Journal of Communications, vol. 11, no. 2, pp. 213-220, 2016.

[2] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," IEEE Communications Magazine, vol. 54, no. 5, pp. 36-42, 2016.

[3] X. Lin et al., "The Sky Is Not the Limit: LTE for Unmanned Aerial Vehicles," IEEE Communications Magazine, vol. 56, no. 4, pp. 204-210, 2018.

[4] V. Paxson, "End-to-end Internet packet dynamics," IEEE/ACM Transactions on Networking, vol. 7, no. 3, pp. 277-292, 1999.

[5] D. J. C. MacKay, "Fountain codes," IEEE Proceedings - Communications, vol. 152, no. 6, pp. 1062-1068, 2005.

[6] M. Luby, "LT codes," in The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings., 2002, pp. 271-280.

[7] A. Shokrollahi, "Raptor codes," IEEE Transactions on Information Theory, vol. 52, no. 6, pp. 2551-2567, 2006.

[8] M. Watson and T. S. M. L. A. Shokrallahi, "RFC 6330: RaptorQ Raptor Forward Error Correction Scheme for Object Delivery," IETF, RFC 6330, 2011.

[9] S. Nazir, D. Vukobratović, and V. Stanković, "Performance evaluation of Raptor and Random Linear Codes for H.264/AVC video transmission over DVB-H networks," in 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011, pp. 2328-2331.

[10] V. Roca, B. Teibi, C. Burdinat, T. Tran-Thai, and C. Thienot, "Block or convolutional AL-FEC codes? A performance comparison for robust low-latency communications," 2017.

[11] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: an instant primer," ACM SIGCOMM Computer Communication Review, vol. 36, no. 1, pp. 63-68, 2006.

[12] N. Thomos and P. Frossard, "Network Coding and Media Streaming," Journal of Communications, vol. 4, no. 9, pp. 628-639, 2009.

[13] J. K. Sundararajan, D. Shah, M. Médard, and P. Sadeghi, "Feedback-Based Online Network Coding," IEEE Transactions on Information Theory, vol. 63, no. 10, pp. 6628-6649, 2017.

[14] M. V. Pedersen, J. Heide, and F. H. Fitzek, "Kodo: An open and research oriented network coding library," in International Conference on Research in Networking, 2011, pp. 145-152.

[15] S. Wunderlich, F. Gabriel, S. Pandi, and F. H. P. Fitzek, "We don't need no generation - a practical approach to sliding window RLNC," in 2017 Wireless Days, 2017, pp. 218-223.

[16] Y. Zhong, T. Q. Quek, and X. Ge, "Heterogeneous cellular networks with spatio-temporal traffic: Delay analysis and scheduling," IEEE Journal on Selected Areas in Communications, vol. 35, no. 6, pp. 1373-1386, 2017.

[17] Y. Zhong, M. Haenggi, F. C. Zheng, W. Zhang, T. Q. S. Quek, and W. Nie, "Toward a Tractable Delay Analysis in Ultra-Dense Networks," IEEE Communications Magazine, vol. 55, no. 12, pp. 103-109, 2017.

[18] S. Hemminger, "Network emulation with NetEm," in Linux conf au, 2005, pp. 18-23.