Reliable Point-to-point Underwater Acoustic Data Transfer: To Juggle or Not to Juggle?

Mandar Chitre, *Senior Member, IEEE,* and Wee-Seng Soh, *Member, IEEE*

***Abstract*—Reliable data transfer speeds using underwater acoustic communication systems are limited by long propagation delays, small link data rates, and high bit error rates. We consider the practical problem of transferring a data file or data stream reliably from one half-duplex underwater node to another. In a typical ARQ approach, a node transmits one or more packets and waits for the corresponding acknowledgments (ACKs). With long propagation delay, the long waiting time for ACKs results in low average throughput. The long propagation delay, however, presents an opportunity for two nodes to simultaneously transmit data and ACKs towards each other in a juggling-like approach, potentially reducing the average waiting time for ACKs. The approach needs to satisfy certain timing constraints, and its performance is largely dependent on the network settings and chosen parameters. Through analytical and numerical studies, we provide key insights into appropriate choice of ARQ strategies and protocol parameters under different inter-nodal propagation delays. We show that the juggling-like ARQ provides good data streaming throughput but performs poorly for small file transfers. We propose a novel rate-less code based juggling-like ARQ protocol that overcomes this limitation and offers high data transfer speeds for small files in long propagation delay environments.**

***Index Terms*—Juggling-like ARQ, rate-less codes, file transfer, reliable data transfer, underwater acoustic networks.**

I. Introduction

THE problem of transferring a data file or data stream reliably from one node to another is commonly encountered in many applications. Although mechanisms such as Automatic Repeat Request (ARQ) can be used for such data transfer [1], their performance deteriorates when operating in half-duplex channels with long propagation delays. Such a scenario is commonly encountered in underwater communication applications. Unlike terrestrial wireless networks that mainly rely on radio waves for communications, underwater networks typically utilize acoustic waves, which result in significantly longer propagation delays and very limited data rates [2]. The variability of the channel and the high levels of ambient noise present in many ocean environments also lead to much higher bit error rate (BER) compared to terrestrial wireless networks. Although forward error correction (FEC) is commonly used to lower the packet error rate (PER) at the expense of the effective data transfer rate, it is often not practical to select an FEC that is able to correct all possible packet errors. Therefore, higher layer protocols are usually also tasked to provide reliable data transfer through mechanisms such as ARQ, erasure coding, etc. Depending on the particular network setting, these mechanisms may exhibit different relative performance.

Manuscript received November 2012, revised May 2013, November 2013.

M. Chitre (e-mail: mandar@arl.nus.edu.sg) and W-S. Soh (e-mail: weeseng@arl.nus.edu.sg) are with the Electrical & Computer Engineering Department and the ARL, National University of Singapore.

In this paper, we focus on this practical problem of reliable data transfer from one half-duplex node to another in a channel with long propagation delay. We compare the efficiencies of different reliable data transfer mechanisms under different network settings, in terms of the effective average data rate that each achieves. We consider two static nodes equipped with half-duplex modems, and assume that they are the only users of the channel in the geographical area and frequency band. Alternatively, in a multiuser case, the channel is assumed to be successfully reserved via a handshaking medium access control (MAC) protocol such as multiple access collision avoidance (MACA). Since our file transfer protocol operates above the physical layer, we do not have to concern ourselves with all the details of the signal propagation in the channel. Instead we model the channel as a half-duplex long propagation delay channel with non-zero packet loss. Since the nodes are static, we assume the channel to be time invariant over the duration of the data transfer. However, we allow for small propagation delay jitters that may arise from limited motion of nodes or changes in signal propagation speed. In the case of the underwater acoustic channel, such changes can occur due to motion of anchored nodes and internal waves.

As mentioned above, one possible solution to the reliable data transfer problem is via the use of ARQ. In an ARQ-based solution, when a data packet remains unacknowledged, retransmissions ensure that the data packet is eventually reliably delivered. Three basic ARQ techniques are stop-and-wait, go-back-N, and selective repeat (SR). Of these, SR-ARQ is known to be most efficient in a channel with long propagation delay [3]. When half-duplex modems are used, we may use a time-division duplexed (TDD) [4] version of SR-ARQ that transmits one or more data packets in a block (or batch), and then waits for the corresponding acknowledgment (ACK) packet. The ACK packet contains a bitmap that acknowledges which data packets in the previous block have been successfully received. Based on this information, the sender can then decide what data packets to include in the next block that it sends, which may include retransmitted packets. This transmission style is illustrated in Fig. 1(a). This time-duplexed version of SR-ARQ has some similarities to the stop-and-wait ARQ scheme, and is called "Stop-and-wait variation no. 2" in [5] and "S&W-2" in [6].

When the propagation delay is long, the above approach is quite inefficient as the sender always idles for one round-trip time while waiting for the receiver's ACK. Although it

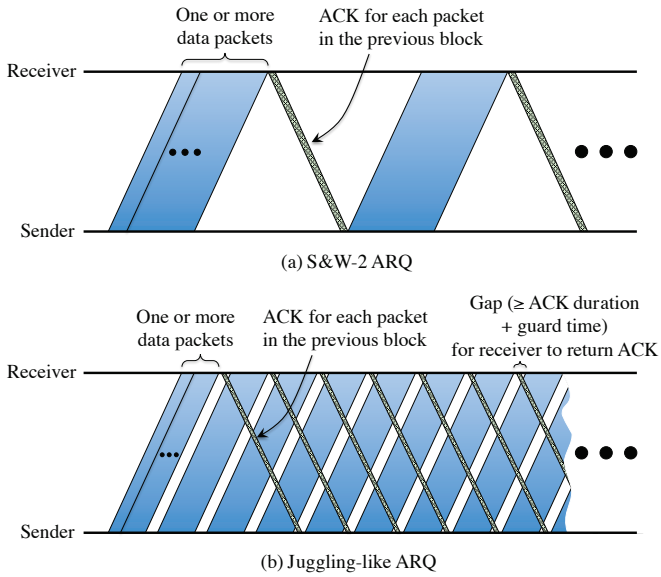

Fig. 1: An illustration of (a) S&W-2 ARQ [5], [6], and (b) juggling-like ARQ (J-ARQ), for half-duplex modems in the presence of long propagation delay.

is possible to improve the efficiency arbitrarily by choosing a very large block size so as to downplay the effect of the idling time, the average queuing delay resulting from such a strategy could become unacceptably large. In [7], a new transmission scheme based on TDD technique, known as the juggling-like stop-and-wait scheme, was proposed to improve the channel utilization of point-to-point data transfer between half-duplex acoustic modems. The juggling concept is illustrated in Fig. 1(b). As can be seen, after transmitting a block of data packets, the sender leaves a gap before transmitting another block of packets. The purpose of the gap is twofold. At the receiver side, it allows the receiver to return an ACK packet to the sender. At the sender side, it allows the sender to receive an ACK packet for an earlier (i.e., not the most recent) block of data packets. The concept is similar to the juggling of objects between two hands; although a hand cannot be throwing one object while receiving another concurrently, multiple objects can still be juggled between two hands. Likewise, although the half-duplex underwater acoustic modems cannot transmit and receive concurrently, multiple blocks of data and ACK packets can still be propagating towards each other simultaneously due to the long propagation delay. This is akin to SR-ARQ, but with some constraints imposed on the time of transmission of data and ACK packets to ensure successful time-duplexing using half-duplex modems. Like SR-ARQ, the ACK packet contains information on which packets in the block were successfully received. Only unsuccessful packets are repeated in subsequent blocks. For convenience, we shall refer to this juggling-like ARQ technique as "J-ARQ", and the time-duplexed SR-ARQ as "S&W-2" [6] in the rest of this paper.

Another TDD-like technique, known as underwater selective repeat (USR), is proposed in [4]. Unlike the J-ARQ, which is designed for transferring a data file or data stream between two static nodes (with minimal propagation delay jitters), the USR provides a solution for medium access control in static and mobile networks, and is suitable for both small (i.e., even a single packet) and large data transfer. It adapts to both sender-receiver distance as well as the number of packets to transfer, and can fall back to plain ARQ when the distance is below a certain threshold. However, its robustness comes with a price; a sender can only transmit at most two packets in a single round-trip time, unlike the J-ARQ.

An alternative solution to the point-to-point data file/stream transfer problem utilizes erasure codes [8]. The set of source data is expanded into a larger set of coded messages by an erasure code, and only the coded messages are transmitted to the receiver. The source data can be recovered at the receiving node from a subset of the coded messages that are successfully received, so long as the subset satisfies certain properties (such as the number of packets in the subset) that depend on the erasure code used. A special class of erasure codes, known as rate-less codes [9], [10] or fountain codes [11], [12], can generate a virtually infinite number of coded messages from the source data. This allows a file transfer protocol to be designed where the individual packets (rate-less code symbols) do not have to be acknowledged. The sender is only required to know when the receiver has received enough packets to reconstruct the source data (i.e., the file to be transferred). This reduces the need for acknowledgements dramatically and hence the protocol efficiency is less dependent on the propagation delay. Solutions based on erasure coding have been previously proposed for underwater data transfer [13], [14].

The performance of the three different reliable data transfer mechanisms described above, namely S&W-2 ARQ, juggling-like ARQ, and rate-less codes, is highly dependent on the network settings (e.g., the inter-nodal propagation delay, BER, etc.) as well as the chosen parameters (e.g., packet size, block size, etc.). Hence, it is not obvious which mechanism would be the most efficient one for a given network setting, and how should their parameters be chosen. For example, would a rate-less code based solution always outperform the other two ARQ-based solutions? As another example, if the inter-nodal propagation delay is short, would the S&W-2 approach outperform the juggling-like ARQ approach if we were to pick very large block size? To complicate the matter further, the best approach may also be different between transferring a finite-size data file versus transferring an infinite data stream, even if all other network settings were to remain the same. In this paper, we make two key contributions. Our first contribution is that, we provide more insights into these different mechanisms through both analytical and numerical studies. Given the propagation delay between the nodes, we propose algorithms to choose protocol parameters such as packet size, block length, etc. to maximize the throughput of S&W-2 and juggling-like ARQ protocols. While juggling-like ARQ outperforms S&W-2 for a virtually infinite data stream and long propagation delay, it performs poorly during small-sized file transfers. As our second contribution, we propose a novel protocol combining rate-less codes and juggling-like ARQ and show that this protocol is able to offer high throughput for the transfer of finite-sized data files. As rate-less codes are erasure FEC codes operating at a packet level, this protocol can be

considered to belong to a class of protocols known as hybrid ARQ (H-ARQ) [15].

The rest of the paper is organized as follows. We first define the problem formally in Section II, and describe the different mechanisms in greater details for both finite-size data file and infinite data stream scenarios. We then provide a detailed analysis and performance comparison for each of these cases, and discuss the implications of our findings in Sections III and IV. We finally summarize our findings in Section V.

II. Problem Definition

A. Data Transfer Problem

We consider data transfer between two half-duplex nodes with a propagation delay p between them. The nodes are the only users of the communication channel, i.e., there is no interference from other nodes. The nodes communicate with each other using packets, each of which consists of a detection preamble followed by a fixed length data payload. The nodes support two types of packets – highly robust low rate control packets (used for ACK), and high rate but less robust data packets. Although we allow the modems to support two distinct packet types, the results in this paper are equally valid for modems that only support a single packet type, by simply setting the parameters for both types of packets to be the same. Let α_{c} be the duration of the control packet overhead (detection preamble and headers), and β_{c} be the incremental duration per payload bit for a control packet. The total length of a control packet with n_{c} payload bits is therefore $\alpha_{\mathrm{c}} + n_{\mathrm{c}}\beta_{\mathrm{c}}$. Similarly, let α_{d} be the duration of the data packet overhead and β_{d} be the incremental duration per payload bit for data packets. A data packet always contains a fixed number of payload bits n_{d}. A packet may be lost if the receiver fails to detect it, or if it is detected but is corrupted due to bit errors after FEC decoding. It is assumed that an error detection mechanism such as a cyclic redundancy check (CRC) is used to detect errors with a negligible probability of wrongly accepting a corrupted packet. The overhead for the CRC is included in α_{c} and α_{d}. Let γ_{c} and γ_{d} be the detection probabilities, and e_{c} and e_{d} be the BERs (after FEC decoding) for the control and data packets respectively. The symbols used are summarized in Table I for easy reference. In this paper, we consider two specific data transfer problems:

- *Data stream:* A virtually infinite data stream is to be transferred reliably from one node to the other. We are interested in maximizing the average throughput S (average number of data bits transferred per unit time).
- *File transfer:* A finite sized file is to be transferred reliably from one node to the other. We are interested in minimizing the average time T it takes to transfer a file containing n_{f} bits of data. Since n_{f} is a constant for a given file, we define average throughput $S = n_{\mathrm{f}}/T$ as the performance metric of interest.

TABLE I: Notation

Notation	Description
p	Propagation delay
α_{d}	Data packet's overhead duration
β_{d}	Data packet's bit duration
e_{d}	Data packet's BER
γ_{d}	Data packet's detection probability
α_{c}	Control packet's overhead duration
β_{c}	Control packet's bit duration
e_{c}	Control packet's BER
γ_{c}	Control packet's detection probability
δ_{min}	Minimum guard time for juggling
n_{d}	Data packet's payload length in bits
m	Block size (number of packets)
S	Average throughput
n_{f}	File size in bits (for finite files)
T	File transfer time (for finite files)

B. Error Model

The probability P of successfully receiving a packet depends on the packet detection probability γ, BER e and the length n of the packet. Although we acknowledge that larger packet sizes potentially admit larger FEC block sizes leading to better error performance, we assume that the FEC code (and the BER) is unchanged when the packet size is varied. This assumption is justified as dynamic design of FEC codes with varying block size is computationally expensive, and therefore infeasible in most modems, including currently available underwater modems. The probability P also depends on the assumption of error model. To make our formulation general, we define $P = \gamma\Pi(e, n)$ where $\Pi(e, n)$ is the probability that all bits in a packet of length n are correctly received. The exact form of $\Pi(\cdot)$ depends on the error model. For example, a binary symmetric channel (BSC) model yields:

$$P = \gamma\Pi(e, n) = \gamma(1 - e)^n. \tag{1}$$

A generalized Pareto renewal (GPR) model has been reported to accurately model errors in telephone networks [16], [17], and more recently in underwater communication links [18]. Under this model, the distribution of interval between errors is modeled by generalized Pareto (GP) distribution. With this model, P decreases less rapidly with increasing n as compared to the BSC model. The probability P can be computed from the equilibrium distribution of the GPR process using renewal theory (see Appendix B for derivation):

$$P = \gamma\Pi(e, n) \quad = \quad \gamma\left(\frac{1-\theta}{1-\theta+en\theta}\right)^{-1+\frac{1}{\theta}} \tag{2}$$

where θ is the shape parameter of the GP distribution. In accordance with [18], we use $\theta = 0.12$ for relevant results presented later in this paper.

III. Data Stream Analysis

For the problem of transferring a virtually infinite data stream between two nodes, we consider two protocols. The first protocol uses the S&W-2 protocol with a fixed block size m. Analysis of this protocol is presented in Section III-A. The second protocol, J-ARQ, is similar to S&W-2 with block size m, but applies the concept of juggling to minimize the effect of propagation delay on the average throughput. The analysis for this protocol is presented in Section III-B.

A. Fixed Block S&W-2

To compute the average throughput S for S&W-2, we only need to consider the transfer of one block of data. As shown in Fig. 1(a), the operations for the transfer of a block of data are repeated throughout the data stream transfer. A data packet is considered to be successfully transferred when it is successfully received by the receiver and the corresponding ACK is successfully received by the transmitter. Since we use a single ACK packet per block, the loss of an ACK packet results in the loss of all packets in the block. Hence, we use robust control packets for the ACK packets, while we use high speed data packets for the transfer of the data in the block. Recall that this S&W-2 protocol is a time-duplexed version of the SR-ARQ where the ACK packet uses one acknowledgment bit for each data packet in the block, and hence has m bits in its payload. The probability P_{c} of successfully receiving the ACK packet is $\gamma_{\mathrm{c}}\Pi(e_{\mathrm{c}}, m)$. The probability P_{d} of successfully receiving a data packet is given by $\gamma_{\mathrm{d}}\Pi(e_{\mathrm{d}}, n_{\mathrm{d}})$. Suppose we have a data block whose ACK packet has been successfully received; for each successful data packet, we manage to transfer n_{d} data bits from the transmitter to the receiver. Hence, the expected number of data bits transferred during a block is given by

$$\mathbb{E}[n_{\mathrm{b}}] = P_{\mathrm{c}} \sum P_{\mathrm{d}} n_{\mathrm{d}} = m P_{\mathrm{c}} P_{\mathrm{d}} n_{\mathrm{d}}. \tag{3}$$

The time taken to transmit a data block and receive the corresponding ACK packet is

$$T_{\mathrm{b}} = m\left(\alpha_{\mathrm{d}} + n_{\mathrm{d}}\beta_{\mathrm{d}}\right) + \alpha_{\mathrm{c}} + m\beta_{\mathrm{c}} + 2p. \tag{4}$$

The throughput S is therefore given by

$$S = \frac{\mathbb{E}[n_{\mathrm{b}}]}{T_{\mathrm{b}}} \tag{5}$$

$$= \frac{\gamma_{\mathrm{c}}\gamma_{\mathrm{d}} n_{\mathrm{d}} \Pi(e_{\mathrm{c}}, m)\Pi(e_{\mathrm{d}}, n_{\mathrm{d}})}{\alpha_{\mathrm{d}} + n_{\mathrm{d}}\beta_{\mathrm{d}} + \beta_{\mathrm{c}} + \frac{\alpha_{\mathrm{c}}+2p}{m}}. \tag{6}$$

The optimal throughput S^* is obtained by varying the data packet length n_{d} and the block size m:

$$S^* = \max_{n_{\mathrm{d}}, m} \frac{\gamma_{\mathrm{c}}\gamma_{\mathrm{d}} n_{\mathrm{d}} \Pi(e_{\mathrm{c}}, m)\Pi(e_{\mathrm{d}}, n_{\mathrm{d}})}{\alpha_{\mathrm{d}} + n_{\mathrm{d}}\beta_{\mathrm{d}} + \beta_{\mathrm{c}} + \frac{\alpha_{\mathrm{c}}+2p}{m}}. \tag{7}$$

The maximization problem is easily solved using an iterative numerical technique such as steepest gradient descent, or simply by an exhaustive search over practical values of n_{d} and m.

B. Fixed Block J-ARQ

Next, we proceed to compute the average throughput for the J-ARQ protocol. Fig. 2 shows a more detailed illustration of some of its parameters. From the figure, it can be seen that the ACK for the i^{th} data block only reaches the sender after it has sent out the $(i+k)^{\mathrm{th}}$ block, where $k \geq 1$. Note that $k = 3$ for the example shown. As explained in Section I, after transmitting a data block, the sender needs to pause for a small time gap before transmitting the next block. The size of this gap depends on the particular network setting (such as the internodal propagation delay, data block size, etc.), but it must be at least as large as the ACK packet's duration plus any

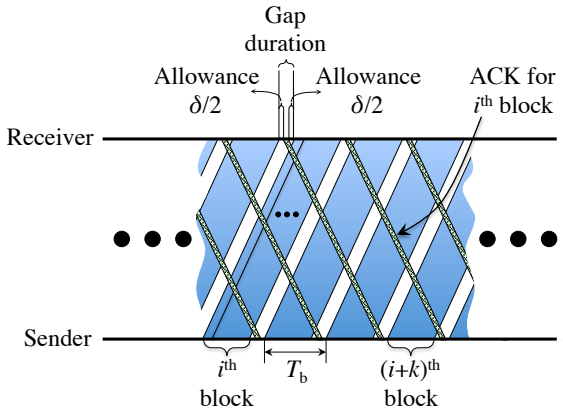

Fig. 2: A more detailed illustration of some of the J-ARQ parameters.

desired guard time or allowance δ_{min} to cater for propagation delay jitters arising from limited motion of nodes, internal waves or the modems.

Although the ACK packet that arrives at the sender does not acknowledge the most recently transmitted data block, it can be seen from Fig. 1(b) that the sender also undergoes a repetitive pattern of transmitting a data block and receiving an ACK packet, except for the initial few data blocks. Since we are considering a virtually infinite data stream, we can ignore the effect arising from the initial few data blocks, and compute the long term average throughput for J-ARQ by just considering the steady state shown in Fig. 2. Similar to the S&W-2 case, we can focus on the expected number of data bits transferred in a block ($\mathbb{E}[n_{\mathrm{b}}]$) over the duration of a basic repetitive pattern (T_{b}) for our computation. There is no change to the expression for $\mathbb{E}[n_{\mathrm{b}}]$, as previously derived in (3). However, the efficiency arising from the juggling mechanism allows T_{b} to be smaller than the case of S&W-2. For J-ARQ,

$$T_{\mathrm{b}} = \alpha_{\mathrm{c}} + m(\alpha_{\mathrm{d}} + \beta_{\mathrm{c}} + n_{\mathrm{d}}\beta_{\mathrm{d}}) + \delta, \tag{8}$$

where $\delta \geq \delta_{\mathrm{min}}$ represents the gap duration's allowance above the ACK packet's duration. Note that, ideally J-ARQ is most efficient when $\delta = \delta_{\mathrm{min}}$ (i.e., the gap duration is exactly equal to the ACK packet's duration plus the chosen guard time δ_{min}), but this is not always possible for any network setting. The reason why δ cannot be arbitrarily set to δ_{min} to maximize efficiency is because the juggling mechanism needs to satisfy a timing constraint arising from the half-duplex property of the acoustic modems, which is given by

$$kT_{\mathrm{b}} = 2p. \tag{9}$$

It is also important to note that juggling is not possible when the two nodes are too close to each other. Since $k \geq 1$, we can see from (9) that the largest possible value of T_{b} is $2p$. Knowing that $\delta \geq \delta_{\mathrm{min}}$, $m \geq 1$, and $n_{\mathrm{d}} \geq 1$, we obtain from (8) that the minimum propagation delay between any two nodes must satisfy the following condition, in order for juggling to be possible:

$$2p \geq \alpha_{\mathrm{c}} + \alpha_{\mathrm{d}} + \beta_{\mathrm{c}} + \beta_{\mathrm{d}} + \delta_{\mathrm{min}}. \tag{10}$$

Unlike the S&W-2 case, where the optimal throughput is obtained by varying only two parameters (n_d and m), J-ARQ requires the varying of three parameters, namely, n_d, m, and k. An efficient, iterative numerical technique described next is used to compute the optimal throughput. Combining (8) and (9), and using $\delta \geq \delta_{\min}$, we have

$$\begin{aligned} \frac{2p}{k} &\geq \alpha_\mathrm{c} + m(\alpha_\mathrm{d} + \beta_\mathrm{c} + n_\mathrm{d}\beta_\mathrm{d}) + \delta_{\min} \quad (11) \\ &\geq \alpha_\mathrm{c} + m(\alpha_\mathrm{d} + \beta_\mathrm{c} + \beta_\mathrm{d}) + \delta_{\min}. \quad (12) \end{aligned}$$

Hence,

$$k \leq \frac{2p}{\alpha_\mathrm{c} + m(\alpha_\mathrm{d} + \beta_\mathrm{c} + \beta_\mathrm{d}) + \delta_{\min}}. \quad (13)$$

Since $m \geq 1$, the maximum possible k can be obtained by setting $m = 1$, which gives

$$k_{\max} = \left\lfloor \frac{2p}{\alpha_\mathrm{c} + \alpha_\mathrm{d} + \beta_\mathrm{c} + \beta_\mathrm{d} + \delta_{\min}} \right\rfloor. \quad (14)$$

Using (12), for any given k, we have

$$m \leq \frac{\frac{2p}{k} - \alpha_\mathrm{c} - \delta_{\min}}{\alpha_\mathrm{d} + \beta_\mathrm{c} + \beta_\mathrm{d}}. \quad (15)$$

Hence, the maximum possible m for any given k can be obtained as

$$m_{\max}(k) = \left\lfloor \frac{\frac{2p}{k} - \alpha_\mathrm{c} - \delta_{\min}}{\alpha_\mathrm{d} + \beta_\mathrm{c} + \beta_\mathrm{d}} \right\rfloor. \quad (16)$$

Using (11), for any given pair of k and m, we have the following constraint:

$$n_\mathrm{d} \leq \frac{1}{\beta_\mathrm{d}} \left(\frac{\frac{2p}{k} - \alpha_\mathrm{c} - \delta_{\min}}{m} - \alpha_\mathrm{d} - \beta_\mathrm{c} \right). \quad (17)$$

The maximum possible n_d for any given pair of k and m is thus

$$n_{\mathrm{d,max}}(m,k) = \left\lfloor \frac{1}{\beta_\mathrm{d}} \left(\frac{\frac{2p}{k} - \alpha_\mathrm{c} - \delta_{\min}}{m} - \alpha_\mathrm{d} - \beta_\mathrm{c} \right) \right\rfloor. \quad (18)$$

Next, let us expand the expression for the throughput S by substituting (3) and (9) into (5):

$$\begin{aligned} S &= \frac{\mathbb{E}[n_\mathrm{b}]}{T_\mathrm{b}} = \frac{\mathbb{E}[n_\mathrm{b}]k}{2p} \quad (19) \\ &= \frac{k}{2p}\Pi(e_\mathrm{c}, m)\Pi(e_\mathrm{d}, n_\mathrm{d})mn_\mathrm{d}\gamma_\mathrm{c}\gamma_\mathrm{d}. \quad (20) \end{aligned}$$

Let n_d^* be the *optimally rounded*[1] value of n_d that yields $\partial_{n_\mathrm{d}} S = 0$. In the case of the BSC error model, this can be obtained analytically as

$$n_\mathrm{d}^* = \mathcal{N}_S \left[-\frac{1}{\ln(1 - e_\mathrm{d})} \right]. \quad (21)$$

In the case of the GPR error model, n_d^* can also be obtained analytically:

$$n_\mathrm{d}^* = \begin{cases} \mathcal{N}_S \left[\frac{1-\theta}{e_\mathrm{d}(1-2\theta)} \right] & \text{if } \theta < 0.5 \\ \infty & \text{if } \theta \geq 0.5 \end{cases} \quad (22)$$

[1]We define optimal rounding $\mathcal{N}_f[\tilde{n}] \rightarrow n$ such that $n = \arg\max_{\eta \in \{\lfloor \tilde{n} \rfloor, \lceil \tilde{n} \rceil\}} f(\eta)$. In effect $\tilde{n}$ is either rounded up or down to give n so that the cost function $f(n)$ is maximized.

Recall from (18) that, there is actually a maximum possible n_d for any given pair of k and m. Therefore, if $n_{\mathrm{d,max}}(m,k) \leq n_\mathrm{d}^*$, then the optimal n_d for any given pair of k and m is given by $n_\mathrm{d}^*(m,k) = n_{\mathrm{d,max}}(m,k)$. Otherwise, $n_\mathrm{d}^*(m,k) = n_\mathrm{d}^*$.

Now, we can proceed to find the optimal throughput S^* as follows. For $k = 1$ to $k_{\max}$, and for $m = 1$ to $m_{\max}(k)$, find the respective $n_\mathrm{d}^*(m,k)$ and the corresponding throughput $S(m,k,n_\mathrm{d}^*(m,k))$. Finally, among all the $S(m,k,n_\mathrm{d}^*(m,k))$ found, we pick the largest value to be the optimal throughput S^*. The optimal set of values for m, k, and n_d are also obtained accordingly.

C. Performance Comparison

Here, we compare the performance of fixed block S&W-2 and J-ARQ for data streaming application. We compute the optimal parameters at each value of propagation delay for each protocol and use them to estimate the performance of that protocol. Fig. 3 shows their average throughput performance as a function of the propagation delay p based on the setup given in Table II (in which we have considered both good and bad channel conditions) and the BSC error model. As expected, the average throughput of the S&W-2 scheme drops with increasing propagation delay, because the sender's idling time spent on waiting for the receiver's ACK also becomes longer. For the case of the poor channel, the S&W-2 scheme's throughput is not only lower than that of the good channel, but it also falls more steeply as the block duration is reduced in poor channels. The J-ARQ scheme starts off with very low average throughput when the propagation delay is low; this is because the delay is too short to allow large block size and long payload length in the presence of juggling time constraints, and hence the header overheads in both the ACK packets and the data packets become very significant when compared to the payload length. However, as the propagation delay increases, the above restriction quickly relaxes; the J-ARQ scheme's average throughput is seen to rise sharply with increasing propagation delay to quickly outperform the S&W-2 scheme, and then asymptotically approaches a maximum as the propagation delay increases further. From the above results, we note that the J-ARQ scheme is able to realize its full potential only when the propagation delay is sufficiently large, and when this happens, it is able to provide significantly higher throughput than the S&W-2 scheme. This trend is observed in both good and bad channel conditions; the cross-over point in Fig. 3 is at about 1.7 seconds of propagation delay (about 2.55 km of distance between nodes) for the good channel case, and at about 0.3 seconds of propagation delay (about 0.45 km of distance between nodes) for the poor channel case. Data transfer at ranges of several kilometers is common in underwater applications where J-ARQ may provide better performance.

We examined the effect of changing the error model to the GPR, while keeping all other settings the same and found that the throughput performance is almost identical for both the error models (BSC or GPR). We therefore use the GPR error model alone for performance evaluation in the rest of this paper.

TABLE II: System Parameters Assumed

Parameter	Good Channel	Poor Channel
α_{d}	40 ms	40 ms
β_{d}	1 ms	1 ms
e_{d}	10^{-4}	10^{-3}
γ_{d}	0.99	0.90
α_{c}	40 ms	40 ms
β_{c}	10 ms	10 ms
e_{c}	10^{-5}	10^{-4}
γ_{c}	0.99	0.90
δ_{min}	2 ms	2 ms

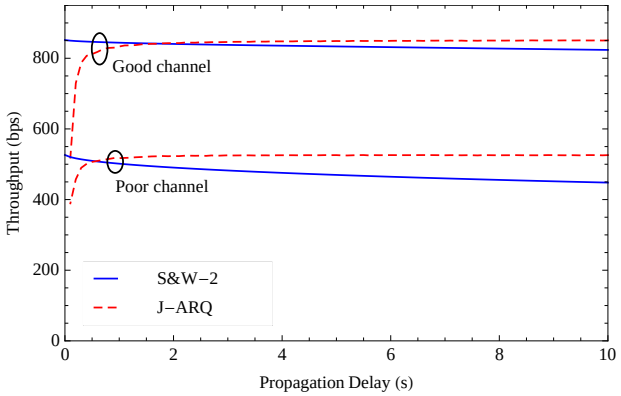

Fig. 3: Performance comparison of fixed block S&W-2 and J-ARQ for data streaming using the setup given in Table II and the BSC error model. The results for the GPR error model are almost identical.

IV. File Transfer Analysis

The problem of transferring a file between two nodes is somewhat more complicated. Since the file size is finite, towards the end of the file transfer, we may not have enough data for a given block size. Rather than transmit empty packets (or equivalently, simply not transmit during the time allotted for those packets), the S&W-2 protocol allows us to reduce the block size towards the end of the transmission. We term this protocol as variable block S&W-2 or simply "V-S&W-2". Due to the timing constraints on the J-ARQ protocol, it is not possible to change dynamically the duration of the basic repetitive pattern, T_{b}. Hence, when there are insufficient outstanding packets to fill up the originally allotted block duration towards the end of the file transfer, the J-ARQ protocol effectively wastes the available transmission opportunity. In this section, we compare the performance of the V-S&W-2 protocol with the J-ARQ protocol for the transfer of finite sized files.

The use of rate-less codes allows the generation of a virtually infinite number of packets from a finite sized file. When sufficient number of these packets are received, the receiving node is able to recover the original file. Such codes can be combined with the J-ARQ protocol to avoid the wastage of transmission opportunity towards the end. In this section, we also analyze a version of the fixed block J-ARQ protocol with rate-less codes.

A. Variable Block S&W-2

We divide a file of size n_{f} into n packets of optimal size n_{d} (from Section III-A). We assume that the file size is much larger than the packet size, and hence ignore the small loss in efficiency due to a partially filled terminal packet when n_{f} is not an integer multiple of n_{d}. We then have,

$$n = \left\lceil \frac{n_{\mathrm{f}}}{n_{\mathrm{d}}} \right\rceil . \tag{23}$$

The optimal block size m (from Section III-A) is used while sufficient data is available to fill all the m packets in a block. Once the number of outstanding packets to be transferred drops below m, the block size is reduced to the number of outstanding packets and the round-trip time for a block is effectively reduced.

To analyze the performance of this protocol, we consider a Markov chain with $n+1$ states σ_0 through σ_n. The system is said to be in state σ_j when j outstanding packets remain to be transferred from the source to the receiver. The system starts in an initial state σ_n, and the file transfer is completed when the system reaches the absorbing state σ_0. When the system is in state $\sigma_j \ \forall \ j \geq m$, a block size of m is used. Once the system reaches state $\sigma_j \ \forall \ j < m$, a block size of j is used. The state transition probabilities P_{ij} for transition from state σ_i to σ_j are given by:

$$P_{ij} = \begin{cases} 0 & i < j \\ 0 & i - j > m \\ 1 - \gamma_{\mathrm{c}} \Pi(e_{\mathrm{c}}, m) \left(1 - \bar{P}_{\mathrm{d}}^{m}\right) & i = j \geq m \\ 1 - \gamma_{\mathrm{c}} \Pi(e_{\mathrm{c}}, i) \left(1 - \bar{P}_{\mathrm{d}}^{i}\right) & i = j < m \\ \gamma_{\mathrm{c}} \Pi(e_{\mathrm{c}}, m) \binom{m}{i-j} P_{\mathrm{d}}^{i-j} \bar{P}_{\mathrm{d}}^{m-i+j} & 0 < i - j \leq m, i \geq m \\ \gamma_{\mathrm{c}} \Pi(e_{\mathrm{c}}, i) \binom{i}{i-j} P_{\mathrm{d}}^{i-j} \bar{P}_{\mathrm{d}}^{j} & \text{otherwise} \end{cases} \tag{24}$$

where $\bar{P}_{\mathrm{d}} = 1 - P_{\mathrm{d}}$. State σ_0 is an absorbing state since $P_{0,0} = 1$. We form matrix $\mathbf{Q}$ by dropping the row and column associated with the absorbing state from matrix $\mathbf{P}$, i.e., $\mathbf{Q} = [P_{ij} \ \forall \ i = 1..n, j = 1..n]$. The n^{th} row of the fundamental matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$ gives the expected number of times that the system spends in each state, given that it starts in state σ_n [19]. The expected value of the time taken to transfer the entire file is therefore,

$$T = \sum_{j=1}^{n} N_{nj} \left(\min(j, m) \left(\alpha_{\mathrm{d}} + n_{\mathrm{d}} \beta_{\mathrm{d}} + \beta_{\mathrm{c}} \right) + \alpha_{\mathrm{c}} + 2p \right) \tag{25}$$

and the average throughput $S = n_{\mathrm{f}} / T$.

B. Fixed Block J-ARQ

As discussed previously, the J-ARQ protocol cannot dynamically change its T_{b} due to its timing constraints. When there is not enough data to fill up the originally allotted block duration, a node will transmit empty packets such that the block size m is always fixed and therefore T_{b} is fixed. We calculate the optimal set of values for m, k and n_{d} using the numerical approach given in Section III-B. Just like the V-S&W-2 protocol, the file is divided into n packets, where n is given by (23). However, unlike the V-S&W-2 protocol,

towards the end of the file transfer the number of (non-empty) data packets transmitted within a block is not equal to the number of outstanding packets; rather, it is the number of outstanding packets that are not pending acknowledgement. To understand this better, we note that the ACK for the i^{th} block only reaches the sender after it has sent out the $(i+k)^{\text{th}}$ block. This implies that a node will have less than m data packets to form a complete block as soon as the number of outstanding packets in the file transfer is less than $m(k+1)$. It should also be noted that, while the number of outstanding packets decreases monotonically, the number of non-empty data packets transmitted in successive blocks may not.

Due to the behavior of the fixed block J-ARQ towards the end of the file transfer, its performance cannot be easily analyzed using the Markov chain approach similar to Section IV-A. Hence, we perform Monte Carlo simulation to estimate its performance using the pseudocode given in the Appendix A.

C. Fixed Block J-ARQ with Rate-less Codes

The fixed block J-ARQ is unable to fully utilize the allocated transmission time for blocks towards the end of the file transfer. The use of rate-less codes allows the generation of a virtually infinite number of packets from a finite sized file. These packets can be transmitted without the need to wait for acknowledgement. When sufficient number of these packets are received, the receiving node is able to recover the original file. The acknowledgement packets do not need to carry information about each packet in a block, but simply need to inform the transmitter when enough packets have been received to recover the file.

We consider a simple linear random erasure code, where the transmitter transmits a random linear combination (over a finite field $\mathbb{F}_q$)[2] of its source packets at each transmission opportunity. Given n source packets X_k for $k = 1..n$, an encoded packet $Y = \sum_{k=1}^{n} e_k X_k$ is formed by randomly choosing $e_k \in \mathbb{F}_q$ at each transmission opportunity, with a uniform distribution over $\mathbb{F}_q$. The transmitted packet needs to carry information on the random coefficients e_k used to generate the packet. Rather than transmit all the coefficients, the transmitter may simply transmit a pseudo-random number generator seed (n_{s} bits), which can be used at the receiver to generate the same set of coefficients as the transmitter. When the receiver successfully receives n linearly independent encoded packets, it is able to decode the entire file by solving the linear equations relating the source packets with the received encoded packets. Efficient techniques for solving such equations may be used in practical implementations [20], [21].

Like the fixed-block J-ARQ protocol, we calculate the optimal set of values for m, k and n_{d} using a numerical approach. However, the approach here deviates slightly from the one given previously in Section III-B, because now each ACK packet only needs to carry a 1-bit payload to inform the transmitter whether enough packets have been received to recover the file. Although this slight difference does not change the steps given in Section III-B, the expressions in (8), (10)–(18) need to be modified accordingly. Rather than verbosely presenting all these expressions, we note that the exact expressions for the current approach can be obtained by simply removing all β_{c} from the original equations, and then substituting α_{c} by $\alpha_{\text{c}}+\beta_{\text{c}}$. Since P_{c} is now given by $\gamma_{\text{c}}\Pi(e_{\text{c}}, 1)$ and the data carried in a packet is reduced by n_{s} due to the added overhead of the pseudo-random number generator seed, (20) also needs to be modified as

$$S = \frac{k}{2p}\Pi(e_{\text{c}}, 1)\Pi(e_{\text{d}}, n_{\text{d}})m(n_{\text{d}} - n_{\text{s}})\gamma_{\text{c}}\gamma_{\text{d}}. \tag{26}$$

Finally (21) and (22) also have to be modified to take the reduced data carried in the packet into consideration:

$$n^*_{\text{d}} = \mathcal{N}_S\left[n_{\text{s}} - \frac{1}{\ln(1-e_{\text{d}})}\right] \qquad \text{for BSC} \tag{27}$$

$$n^*_{\text{d}} = \mathcal{N}_S\left[n_{\text{s}} + \frac{1-\theta}{e_{\text{d}}(1-2\theta)}\right] \qquad \text{for GPR, } \theta < 0.5. \tag{28}$$

The file to be transferred is divided into n packets. Since n_{s} bits in each transmitted packet are reserved for the random number seed, we divide the file into packets of size $n_{\text{d}} - n_{\text{s}}$ bits. Hence,

$$n = \left\lceil \frac{n_{\text{f}}}{n_{\text{d}} - n_{\text{s}}} \right\rceil. \tag{29}$$

As in section IV-A, we analyze the performance of this protocol using a Markov chain with states $\sigma_0 \cdots \sigma_n$. The system is said to be in state σ_j when $n-j$ linearly independent packets have been successfully transferred from the source to the receiver. The system starts in σ_n and eventually reaches absorbing state σ_0. The state transition probabilities P_{ij} are modeled for each packet transmitted, and are given by:

$$P_{ij} = \begin{cases} 1 - P_{\text{u}}(i) & i = j \\ P_{\text{u}}(i) & i = j+1 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

where

$$P_{\text{u}}(i) = P_{\text{d}}\left(1 - \frac{q^{n-i}-1}{q^n - 1}\right) \tag{31}$$

is the probability of a packet being successfully received and not linearly dependent on the $n-i$ encoded packets already known at the receiver. State σ_0 is an absorbing state since $P_{0,0} = 1$. We form matrix $\mathbf{Q}$ by dropping the row and column associated with the absorbing state from matrix $\mathbf{P}$, i.e., $\mathbf{Q} = [P_{ij} \ \forall \ i = 1..n, j = 1..n]$. The n^{th} row of the fundamental matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$ gives the expected number of times that the system spends in each state, given that it starts in state σ_n [19]. The expected number of transmitted packets to reach the absorbing state is therefore given by $n' = \sum_{j=1}^{n} N_{nj}$. Given the lower bidiagonal structure of matrix $\mathbf{Q}$, we can compute n' explicitly:

$$n' = \sum_{j=1}^{n} \frac{1}{P_{\text{u}}(j)} = \frac{q^n - 1}{P_{\text{d}} q^n} \sum_{j=1}^{n} \frac{1}{1 - q^{-j}}. \tag{32}$$

[2]We have to ensure that a packet contains an integral number of symbols from the finite field $\mathbb{F}_q$. In the case of binary codes ($\mathbb{F}_2$) no special consideration is needed, but for larger finite fields, this puts a constraint on the choice of packet size n_{d}. The effect of this is small for practical finite field sizes and therefore we ignore it in this paper. Moreover, the results presented are for $\mathbb{F}_2$ and therefore accurate.

Since the transmissions are made in blocks of m packets, the receiver expects to acquire the entire file during block $\lceil n'/m \rceil$, and the ACK transmitted by the receiver after that block only reaches the transmitter when it has finished transmitting $\lceil n'/m \rceil + k$ blocks. Also, since the probability that the ACK is successfully received is $\gamma_{\mathrm{c}}\Pi(e_{\mathrm{c}}, 1)$, we expect an additional $\zeta = 1/(\gamma_{\mathrm{c}}\Pi(e_{\mathrm{c}}, 1)) - 1$ blocks to be transmitted before the transmitter successfully receives the ACK and stops transmission. Since an integral number of blocks are always transmitted, lower and upper bounds on the average file transfer time are given by

$$T \geq \frac{2p}{k}\left(\frac{\sum_{j=1}^{n} N_{nj}}{m} + k + \zeta\right) \tag{33}$$

$$T \leq \frac{2p}{k}\left(\left\lceil \frac{\sum_{j=1}^{n} N_{nj}}{m} + \zeta \right\rceil + k\right). \tag{34}$$

Since the file is divided into n packets, we can be sure that the file transfer will not be completed until at least n packets have been transmitted. Hence an ACK will not be transmitted for the first $\lceil n/m \rceil - 1$ blocks. We can therefore improve the throughput further by removing gaps between block transmissions for the first $\lceil n/m \rceil - 1$ blocks. The transfer time bounds given by (33) and (34) are then reduced:

$$T \geq \frac{2p}{k}\left(\frac{\sum_{j=1}^{n} N_{nj}}{m} + k + \zeta\right) - \Delta \tag{35}$$

$$T \leq \frac{2p}{k}\left(\left\lceil \frac{\sum_{j=1}^{n} N_{nj}}{m} + \zeta \right\rceil + k\right) - \Delta \tag{36}$$

where

$$\Delta = \left(\left\lceil \frac{n}{m} \right\rceil - 1\right)\left(\frac{2p}{k} - m\left(\alpha_{\mathrm{d}} + n_{\mathrm{d}}\beta_{\mathrm{d}}\right)\right). \tag{37}$$

The upper and lower bounds on the average throughput can be obtained by substituting the above bounds into $S = n_{\mathrm{f}}/T$.

In a lossy channel, the probability that the file transfer is successfully completed when $\lceil n/m \rceil - 1$ blocks are transmitted is typically small. Depending on the packet loss rate, at least a few extra blocks are required before an ACK is expected from the receiver. In [22], [23], the authors compute the optimal number of packets to transmit before waiting for an ACK in case of a time-duplexed ARQ protocol using erasure codes. Instead of using the ACK to simply indicate whether the data transfer is completed, the authors use the ACK to convey the additional linearly independent packets required for completion. This allows the transmitter to determine the number of packets to send before the next ACK for optimal mean data transfer time, energy or throughput. Although the analysis from [22], [23] cannot be directly applied to our problem due to the fixed block size constraint of J-ARQ, the throughput can be improved by delaying the first ACK until the probability of file transfer completion is sufficiently high. In other words, we can skip an ACK if the probability π of sufficient packets having reached the destination is small enough that the time taken for an extra block is justified as compared to the time savings from the skipped ACK. Specifically, we should skip the ACK if the expected time cost from an unnecessary block tranmission is less than the time cost of leaving a gap for the ACK:

$$\pi m(\alpha_d + n_d\beta_d) < \frac{2p}{k} - m(\alpha_d + n_d\beta_d), \tag{38}$$

i.e.,

$$\pi < \frac{2p}{km(\alpha_d + n_d\beta_d)} - 1. \tag{39}$$

The probability π at the end of x transmitted blocks can be computed from the Markov chain as $\pi = (\mathbf{P}^{xm})_{n,0}$. We find the maximum x for which (39) is met, and skip the ACKs until x blocks are transmitted. This is easily done iteratively by starting with $x = \lceil n/m \rceil - 1$ and increasing x by 1 until the condition is violated. We tested this and found only a marginal improvement in performance. Since the J-ARQ protocol does not incur the two-way propagation delay at each ACK, the cost of an ACK is much lesser as compared to other time-duplexed ARQ protocols. The time savings yielded by this additional optimization are negligible when compared with the total file transfer time.

D. Performance Comparison

Fig. 4 and Fig. 5 compare the average throughput performance of V-S&W-2, fixed block J-ARQ and fixed block J-ARQ with rate-less codes for a file transfer application as a function of the propagation delay p. We compute optimal parameters at each value of propagation delay for each protocol, and use them to estimate the performance of that protocol. The results presented in both figures are for a 16-bit random number generator seed (i.e., $n_{\mathrm{s}} = 16$), and again based on the setup given in Table II with both good and poor channel conditions. Only results using the GPR error model are presented here. Fig. 4 presents the results for a smaller file of 10 kB[3] while Fig. 5 presents similar results for a larger file of 20 kB. In both cases, and also for both good and bad channel conditions, it is observed that the performance of V-S&W-2 is adversely affected by increasing propagation delay, as the idling time spent waiting for the ACK increases. For the fixed block J-ARQ, while its performance has the same initial trend as the infinite stream case when the propagation delay is low, its performance actually starts to fall drastically as the propagation delay increases further, as opposed to the infinite stream case. This can be explained by the following two reasons. Firstly, towards the end of a finite-sized file transfer, the fixed block J-ARQ cannot reduce its block size even though there are insufficient packets to fill up the originally allotted block duration. Thus, it effectively wastes a large amount of available transmission opportunity. Secondly, towards the end of the file transfer, the fixed block J-ARQ still requires the sender to wait for the ACKs before it can determine what packets to retransmit. In the worst case, for every useful block that contains non-empty data packets, the sender may have to waste k subsequent blocks before it can transmit another useful block; this reduces its efficiency to somewhat resemble that of fixed block S&W-2, and there is no real benefit from juggling. This effect also becomes more pronounced as p increases.

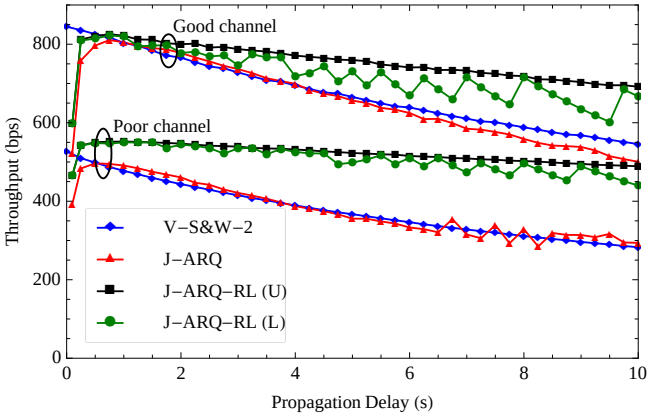

Fig. 4: Performance comparison of various protocols for a 10 kB file transfer for the setup in Table II and the GPR error model. The V-S&W-2 and J-ARQ curves denote the performance of the variable block S&W-2 and the fixed block J-ARQ protocols. The J-ARQ-RL (U) and J-ARQ-RL (L) curves denote the upper and lower bounds on the performance of the fixed block J-ARQ protocol with rate-less codes.

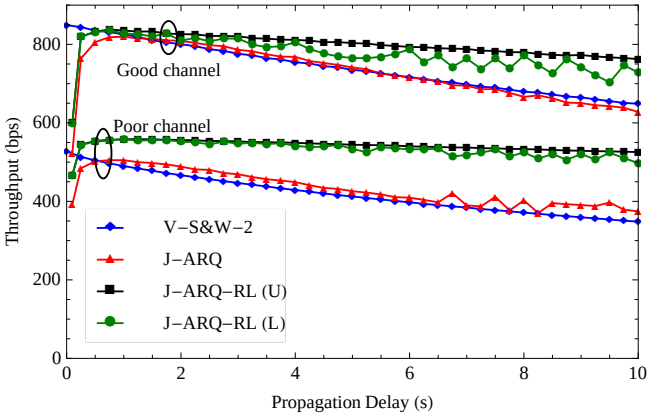

Fig. 5: Performance comparison of various protocols for a 20 kB file transfer for the setup in Table II and the GPR error model. The V-S&W-2 and J-ARQ curves denote the performance of the variable block S&W-2 and the fixed block J-ARQ protocols. The J-ARQ-RL (U) and J-ARQ-RL (L) curves denote the upper and lower bounds on the performance of the fixed block J-ARQ protocol with rate-less codes.

For the case of fixed block J-ARQ with rate-less codes, despite the fact that each data packet contains a 16-bit random number generator seed that does not count towards the throughput, its throughput performance is still much better than the other two schemes. When the propagation delay increases, it encounters a much smaller decrease in performance compared to V-S&W-2 and fixed block J-ARQ. This advantage is even more pronounced when the channel conditions are poor. With the use of rate-less codes, the sender can now fully utilize every block to transmit useful data packets that potentially contribute towards the recovery of the original file, thus avoiding the wastage of transmission opportunity towards the end of the file transfer.

Since J-ARQ with rate-less codes transmits some potentially redundant packets to fill blocks towards the end of the file transfer, its energy consumption may be somewhat higher than S&W-2 and J-ARQ without rate-less codes. This is partially offset by shorter ACK packets and possibly smaller number of blocks needed to complete the file transfer. Although a thorough analysis of energy consumption is beyond the scope of this paper, we observed less than 20% extra energy consumption for J-ARQ with rate-less codes as compared with other protocols in all our test scenarios. In a few scenarios (poor channel, medium propagation delay), J-ARQ with rate-less codes showed small savings in energy. In applications where throughput is of prime importance, the slightly higher energy consumption of J-ARQ with rate-less codes may be acceptable in order to achieve higher throughput.

[3]Here we adopt the convention that 1 kB = 1000 bytes = 8000 bits.

V. Conclusions

In this paper, we considered the practical problem of transferring a data file or data stream reliably for point-to-point underwater acoustic communications. Our key objective was to provide more insights into the average throughput performance of juggling versus non-juggling based ARQ strategies under different inter-nodal propagation delays.

We now summarize the findings. When the inter-nodal propagation delay is low, the S&W-2 approach works well for both data streaming and file transfer. In fact, for very low propagation delay, it performs even better than the J-ARQ protocols, since the latter suffer from high overhead-to-payload ratio imposed by their timing constraints. When the propagation delay is high, although fixed block J-ARQ works well for data streaming, it performs worse than V-S&W-2 for file transfer. This is because it wastes a lot of transmission opportunity towards the end of the file transfer due to its rigid block size imposed by its timing constraints. Hence, we cannot just blindly adopt a juggling approach. Nevertheless, when the propagation delay is high, our proposed use of rate-less codes with fixed block J-ARQ is shown to overcome the inefficiency of the latter for the case of file transfer, as it can now fully utilize all available transmission opportunities. This results in much higher throughput performance compared to the typical ARQ approach, and the performance only degrades moderately with increasing propagation delay.

In conclusion, the negative impact of high propagation delay on the throughput performance of reliable data transfer can be significantly mitigated through the use of J-ARQ for the case of data streaming, while for file transfer applications, its negative impact can be significantly mitigated through the use of rate-less codes in conjunction with J-ARQ.

Appendix A: Pseudocode for Estimating the Performance of Fixed Block J-ARQ

INPUT:

Propagation delay p, file size n_f, number of iterations N
Optimal m, k and n_d from Section III-B
n from (23), T_b from (9)

BEGIN

1: $sum \leftarrow 0$
2: **FOR** $i \leftarrow 1$ **TO** N
3: $availablePkts \leftarrow n$
4: $j \leftarrow 0$

5: ***numOfPktsTransmitted*** $\leftarrow \mathbf{0}_k$
6: **WHILE** (*availablePkts* > 0) or (***numOfPktsTransmitted*** $\neq \mathbf{0}$)
7: $j \leftarrow j + 1$
8: *numOfPktsInOldestBlock* $\leftarrow$ *numOfPktsTransmitted*$[1]$
9: *numOfPktsInNewBlock* $\leftarrow \min(m,$ *availablePkts*$)$
10: ***numOfPktsTransmitted*** $\leftarrow$ (*numOfPktsTransmitted*$[2..k]$, *numOfPktsInNewBlock*)
11: **IF** *Random*$(0,1) < P_{\mathrm{c}}$
12: *numOfFailedPkts* $\leftarrow$ random variable drawn from Binomial Distribution $B($*numOfPktsInOldestBlock*$, 1-P_{\mathrm{d}})$
13: **ELSE**
14: *numOfFailedPkts* $\leftarrow$ *numOfPktsInOldestBlock*
15: **END IF**
16: *availablePkts* $\leftarrow$ *availablePkts* $-$ *numOfPktsInNewBlock* $+$ *numOfFailedPkts*
17: **END WHILE**
18: *sum* $\leftarrow$ *sum* $+ jT_{\mathrm{b}}$
19: **END FOR**
20: *averageThroughput* $\leftarrow n_{\mathrm{f}}/($*sum*$/N)$
END

APPENDIX B: PACKET SUCCESS PROBABILITY FOR A GENERALIZED PARETO RENEWAL BIT ERROR PROCESS

Consider a renewal bit error process where the interval between errors is distributed according to a generalized Pareto distribution with shape parameter $\theta < 1$, scale parameter ψ and a location parameter of 0. The cumulative distribution function of the interval τ is given by

$$F(\tau) = 1 - \left(1 + \frac{\theta\tau}{\psi}\right)^{-\frac{1}{\theta}}. \quad (40)$$

The mean interval between errors is $\lambda = \psi/(1-\theta)$. Therefore the average BER $e = 1/\lambda = (1-\theta)/\psi$, or equivalently $\psi = (1-\theta)/e$. Starting at an arbitrary time t, the distribution of time τ to the next error is given by the *forward recurrence time* of the renewal process. As $t \to \infty$, the distribution of τ is given by the *equilibrium distribution* with probability density function $(1 - F(\tau))/\lambda$ [24]. Since no errors are allowed in a successful packet, the probability Π that a packet of length n is received successfully is given by

$$\Pi = 1 - \int_0^n \frac{1 - F(\tau)}{\lambda} d\tau = \left(\frac{1-\theta}{1-\theta+en\theta}\right)^{-1+\frac{1}{\theta}}. \quad (41)$$

REFERENCES

[1] L. Peterson and B. Davie, *Computer networks: a systems approach*. Morgan Kaufmann Pub, 2007.
[2] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *The Spring 2008 MTS Journal, "The State of Technology in 2008"*, vol. 42, no. 1, pp. 103–116, 2008.
[3] L. Badia, P. Casari, M. Levorato, and M. Zorzi, "Analysis of an Automatic Repeat Request Scheme Addressing Long Delay Channels," in *Advanced Information Networking and Applications*, 2009, pp. 1142–1147.
[4] S. Azad, P. Casari, F. Guerra, and M. Zorzi, "On ARQ strategies over random access protocols in underwater acoustic networks," in *OCEANS - Europe*, 2011.
[5] J. Morris, "Optimal block lengths for arq error control schemes," *IEEE Transactions on Communications*, vol. 27, no. 12, pp. 488–493, February 1979.
[6] M. Stojanovic, "Optimization of a data link protocol for an underwater acoustic channel," in *OCEANS 2005*, June 2005.
[7] M. Gao, W.-S. Soh, and M. Tao, "A transmission scheme for continuous ARQ protocols over underwater acoustic channels," in *Proc. IEEE ICC*, Dresden, Germany, June 2009.
[8] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.
[9] P. Maymounkov and D. Mazieres, "Rateless codes and big downloads," in *Proc. 2nd International Workshop on Peer-to-peer Systems (IPTPS)*, 2003, pp. 247–255.
[10] P. Maymounkov, "Online codes," New York University Technical Report TR2002-833, Tech. Rep., October 2002.
[11] M. Luby, "LT codes," in *Proc. IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2002, pp. 271–280.
[12] S. A, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, 2006.
[13] M. Chitre and M. Motani, "On the use of rate-less codes in underwater acoustic file transfers," in *OCEANS 2007 - Europe*, June 2007.
[14] O. Kebkal, "On the use of interwoven order of oncoming packets for reliable underwater acoustic data transfer," in *OCEANS 2009-EUROPE, 2009. OCEANS '09*, May 2009, pp. 1–7.
[15] S. Wicker, *Error control systems for digital communication and storage*. Prentice Hall, 1995.
[16] B. Mandelbrot, "Self-similar error clusters in communication systems and the concept of conditional stationarity," *Communication Technology, IEEE Transactions on*, vol. 13, no. 1, pp. 71–90, 1965.
[17] D. B. Levey and S. McLaughlin, "The statistical nature of impulse noise interarrival times in digital subscriber loop systems," *Signal Processing*, vol. 82, no. 3, pp. 329 – 351, 2002.
[18] M. Chitre, K. Pelekanakis, and M. Legg, "Statistical bit-error modeling of shallow water acoustic communication links," in *Proceedings of Underwater Communications: Channel Modelling & Validation*, Italy, September 2012.
[19] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*. Wiley, 1998.
[20] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly gaussian elimination for LT codes," *Communications Letters, IEEE*, vol. 13, no. 12, pp. 953 –955, December 2009.
[21] S. Kim, K. Ko, and S.-Y. Chung, "Incremental Gaussian elimination decoding of raptor codes over BEC," *Communications Letters, IEEE*, vol. 12, no. 4, pp. 307 –309, April 2008.
[22] D. E. Lucani, M. Stojanovic, and M. Medard, "Random linear network coding for time division duplexing: When to stop talking and start listening," in *INFOCOM 2009, IEEE*, 2009, pp. 1800–1808.
[23] D. E. Lucani, M. Medard, and M. Stojanovic, "On coding for delay–network coding for time-division duplexing," *Information Theory, IEEE Transactions on*, vol. 58, no. 4, pp. 2330–2348, 2012.
[24] D. R. Cox, *Renewal Theory*. Taylor & Francis, 1962, ch. 5.