Design of FEC for Low Delay in 5G

Mohammad Karzand[1], Douglas J. Leith[1], Jason Cloud[2], Muriel Medard[2]
[1]Trinity College Dublin, Ireland, [2]Massachusetts Institute of Technology, MA, USA.

***Abstract*—In this paper we consider the design of FEC tailored specifically for the low end-to-end latency requirements in 5G networks. We present experimental results that highlight a number of issues with conventional approaches and then introduce a new streaming code construction that achieves provides a superior throughput-delay trade-off. We analyse its performance both mathematically and experimentally. The mathematical analysis of throughput and delay requires the development of a number of novel analytic tools based on both queuing and coding theory. We implement the streaming code and evaluate its performance in an experimental testbed.**

I. Introduction

One of the most challenging requirements in 5G is the provision of connections with low end-to-end latency. In most use cases the target is for <100ms latency, while for some applications it is <10ms [1, Table 1]. In part, this reflects the fact that low delay is already coming to the fore in network services. For example, Amazon estimates that a 100ms increase in delay reduces its revenue by 1% [2], Google measured a 0.74% drop in web searches when delay was artificially increased by 400ms [3] while Bing saw a 1.2% reduction in per-user revenue when the service delay was increased by 500ms [4]. But the requirement for low latency also reflects the needs of next generation applications such as augmented reality and the tactile internet.

While much attention in 5G has been focussed on the physical and link layers, it is increasingly being realised that a wider redesign of network protocols is also needed in order to meet 5G requirements. Transport protocols are of particular relevance for end-to-end performance, including end-to-end latency. For example, ETSI have recently set up a working group to study next generation protocols for 5G [5]. The requirement for major upgrades to current transport protocols is also reflected in initiatives such as Google QUIC [6] and the Open Fast Path Alliance [7].

Internet paths almost always suffer from packet loss due, for example, to mobility, loss at wireless hops and queue overflow. In order to achieve 5G low end-to-end latency requirements the use of forward error correction (FEC) at the transport layer therefore seems essential since this is the only way to avoid the round-trip time (RTT) delay cost incurred by use of packet retransmission to recover from loss. In this paper we consider the design of FEC tailored specifically for achieving low end-to-end latency. We begin by presenting measurements from an experimental study on the latency induced in practice by use of retransmission for loss recovery and by use of FEC based on conventional block codes. Using the lessons learnt from this study we then introduce a novel code construction that achieves significantly improved delay performance and analyse its performance both mathematically and experimentally. The mathematical analysis of throughput and delay requires the development of a number of novel analytic tools based on both queuing and coding theory.

We note briefly that use of multipath connectivity is also on the 5G roadmap, primarily in the form of heterogenous radio access technologies (RATs) at the link layer but multipath extensions at the transport layer are also of much interest in view of their potential for flexible deployment, backward compatibility with legacy cellular systems etc. Latency induced by packet reordering due to path heterogeneity is a key issue, and the low delay FEC techniques studied here for single path connections may well also prove useful for multipath connections, although we leave this as future work.

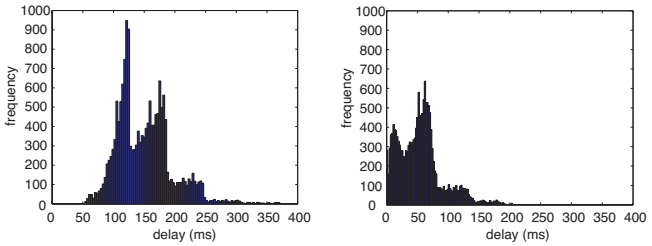
Work by MK and DL supported by Science Foundation Ireland grants 13/IF/I2781 and 11/PI/11771.

II. Preliminary Experimental Measurements

To explore the impact on latency when ARQ is used to recover from packet loss we implemented a reliable UDP-based transport in userspace where the sender maintains a bandwidth-delay product (BDP) worth of packets in flight and retransmits lost packets when informed of loss by ACKs from the receiver[1]. Packets are delivered in-order to the application layer at the receiver, with the receiver buffering packets on packet loss until arriving retransmissions allow recovery and in-order delivery to resume. The hardware consists of three commodity servers (Dell Poweredge 850, 3GHz Xeon, Intel 82571EB Gigabit NIC) connected via a router and gigabit switches. Both the server and client machines run a Linux 2.6.32.27 kernel, while the router run a FreeBSD 4.11 kernel. `ipfw-dummynet` was used on the router to configure various propagation delays T, packet loss rates p, queue sizes Q and link rates B.

Figure 1(a) plots the measured distribution of packet delay less the base RTT of the path, i.e. the distribution of the time that packets spend in the buffer at the receiver awaiting in-order delivery, which we refer to as the *in-order delivery delay*. In this example the path rate is 25Mbps

[1]In our implementation the receiver sends full state back to the sender, so the ARQ performance is not degraded by SACK limitations or the like.



(a) ARQ (b) FEC, block size 32 pkts

Fig. 1: Measured distribution of in-order delivery delay on a 25Mbps link with 60ms propagation delay and 20% packet loss rate (25MB file transfer). Data is shown when ARQ is used for loss recovery and when augmented with FEC using a standard block code. The packet delay shown is less the RTT, so ideally should be clustered around zero.

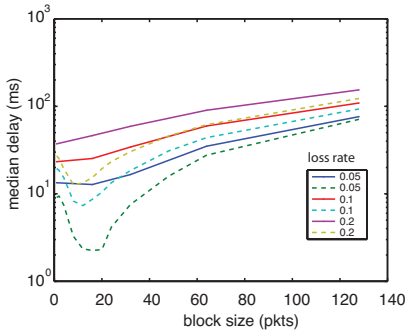


Fig. 2: Measured in-order delivery delay on a 25Mbps link with 60ms propagation delay and 20% packet loss rate (25MB file transfer). Solid lines are when ARQ is used for loss recovery, the dashed lines when this is augmented with FEC using a block code.

and the base RTT (i.e. the RTT without queueing delay) is 60ms, both of which are not unusual values for internet connections. The packet loss rate of 20% is on the high side for an internet connection, but is selected to help provide insight into the mechanism by which loss impacts on latency. Since the in-order delivery delay shown in Figure 1(a) is the end-to-end delay with the base RTT subtracted, ideally we would like the values to be clustered around zero. Unfortunately, it can be seen instead that the in-order delivery delays are rarely below 50ms and are often in the range 100-250ms i.e. the in-order delivery delay can be 2–4 times the path RTT.

The fact that using ARQ for loss recovery incurs a delay overhead equivalen to one or more RTTs is, of course, unsurprising. Responding by using FEC to recover from most, if not all, of the packet losses in order to reduce delay is also natural. With a view to reducing latency over lossy links we therefore also implemented an FEC scheme based on a linear block code with random linear coefficients in GF(256), a modern high performance code for packet erasure channels. We selected the coding rate equal to the path packet loss rate and since feedback is available from the receiver we use this to send additional coded packets when a decoding failure occurs, i.e. a form of ARQ is used as a safety net should there be too many losses for the FEC by itself to allow recovery. Figure 1(b) plots the measured distribution of in-order delivery delay when this FEC is used. The block size used is 32 packets, and we will return to this choice shortly. It can be seen from Figure 1(b) that the delay is much reduced, with almost half of packets now having delay close to zero and few having delay greater than 50ms, despite the high loss rate on the path.

This is encouraging, but it is important to also observe that there is a complex trade-off between delay and block size. This is illustrated in Figure 2, which plots the in-order delivery delay vs the block sized used and, for comparison, also shows the delay with pure ARQ is used for loss recovery. It can be seen that the delay is strictly reduced when ARQ is augmented with FEC (the dashed lines are strictly below the corresponding solid lines). However, the delay with FEC also shows a clear minimum as the block size is varied. What is happening is that when the block size is small then the number of packet losses in each block often exceeds the number of coded packets and so loss recovery essentially falls back to ARQ with associated high delay. When the block size is large, fall back to ARQ is rare (due to statistical multiplexing the number of losses in a block tends to concentrate around the mean as the block size increases) but the delay rises since recovery from loss cannot occur until a complete block has been received and so delay is roughly proportional to block size. Between these two regimes, there is a sweet spot where delay is minimised.

It can be seen that the location of this minimum varies with the path loss rate, and it also varies with the path RTT although we do not show the data here. The delay is highly sensitive to sub-optimality in the choice of block size – observe that a log scale is used in the y-axis in Figure 2 – and so adaptation of the block size is mandated to achieve the low latencies required by 5G specifications. Yet it is extremely hard to analytically predict the location of this minimum (recall that it involves an interaction between the distribution of losses in a block, decoding behaviour and delayed ARQ feedback), thus it is hard to perform accurate adaptation and so hard to achieve minimum latency. In addition, it may also be that an alternative code construction could achieve still lower latency than that of a block with optimised block size. It is these two observations, namely the need for complex adaptation of block codes and the potential to avoid this complexity while at the same time achieving still lower delay by use of alternative code constructions, that motivate the work in the rest of this paper.

III. Streaming Codes for Low Delay FEC

The lesson from the experimental measurements in the previous section is that design of FEC schemes that are capable of achieving the low latencies required in 5G is challenging and that standard block codes are not really

up to the job. The basic difficulty is that standard code constructions, such as block codes, focus on maximising throughput (i.e. achieving capacity). In contrast, in many 5G use cases the focus is very much on delay. Indeed, we are willing to sacrifice throughput (e.g. by sending extra redundant coded packets) if that is necessary to meet the low delay requirements, since bandwidth is often relatively plentiful whereas delay is a harder constraint.

These observations motivate us to seek alternative code constructions that avoid the need for compex adaptation while achieving at least as good, if not better, rate-delay trade-off than standard codes. That is, which are both simple and achieve significantly lower delay for a given level of redundancy than standard codes.

A. Low Delay Code Construction

With block codes, the reason that in-order delivery delay increases with block size is that decoding after loss cannot take place until the end of the block. Hence, delay is roughly proportional to the block size. On the other hand, a large block size must be used in order to achieve a high rate since (i) the coding overhead is amortised across more information packets and (ii) statistical multiplexing of packet losses means that the number of losses in a block tends to concentrate around the mean and so become more predictable for large block sizes. The behaviour of standard convolutional codes is similar, delay for loss recovery scaling with constraint length.

We therefore consider the following code construction which avoids the use of blocks, with the aim of achieving a better rate-delay trade-off. The construction is straightforward: coded packets are inserted at regular intervals within the stream of information packets, with each coded packet being constructed as a random linear combination of all not-yet-acknowledged information packets. Since it avoids the use of blocks we refer to it as a streaming code.

Intuitively, for a given coding rate r we can expect this to lower the delay of loss recovery. Namely, whereas for a block code of size n we typically have to wait for $nr/2$ subsequent information packets to be received before a coded packet arrives and loss recovery can be attempted, for the proposed streaming code we instead only have to wait for $1/(1-r)$ information packets to be received before a coded packet arrives. For example, when $r = 0.9$ (corresponding to a path loss rate of 10%) then $1/(1-r) = 10$ packets while $nr/2 = 45$ packets. However, this intuition ignores the impact of multiple losses within a span of $1/(1-r)$ information packets and the associated knock-on effects for decoding success and delay. In the following sections we therefore develop a rigorous rate and delay analysis for the proposed streaming code and compare with block codes. However, before proceeding we first define the streaming code construction more carefully.

1) Encoder: Assume that time is slotted and each slot is indexed as $t = 1, 2, \ldots$. Within each slot, a single packet can be transmitted. The code is constructed by interleaving information packets (i.e., uncoded packets) u_j, $j = 1, 2, \ldots$ with coded packets c_i, $i = 1, 2, \ldots$. One coded packet is inserted after every $l-1$ information packets and transmitted over the network or channel. This results in a code of rate $(l-1)/l$. Each coded packet is generated by taking random linear combinations from a span of previously transmitted information packets $\{u_L, \ldots, u_U\}$. This span is referred to as the *coding window*. We will assume in the analysis that $L = 1$ and U is the index of the information packet immediately proceeding the generated coded packet (i.e., $U = (l-1)\,i$ assuming the generated coded packet is c_i). Therefore, coded packet c_i is generated as follows:

$$c_i = f_i(u_1, u_2, \ldots, u_{(l-1)i}) := \sum_{j=1}^{(l-1)i} w_{ij} u_j, \qquad (1)$$

where each packet u_j is treated as a vector in finite field $\mathbb{F}_Q$ of size Q and each coefficient $w_{ij} \in \mathbb{F}_Q$ is chosen randomly from an i.i.d. uniform distribution. The receiver is informed of these random coefficients by, for example, including in the coded packet header the seed for a pseudo random number generator used to generate the coefficients.

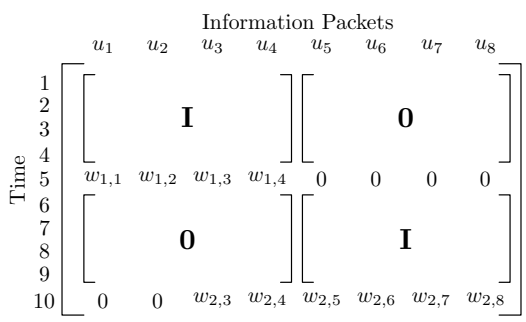


Fig. 3: Example generator matrix showing the coefficients used to produce each packet. In this example, the transmitter has obtained knowledge from the receiver by time 10 indicating that it has successfully received/decoded packets u_1 and u_2 allowing it to adjust the lower edge of the coding window to exclude them from packet c_2.

It should be noted that in practice the lower edge L of the coding window can be adapted in a variety of ways. The only constraint is that the coding window must be large enough to ensure intermediate decoding opportunities at the receiver. For example, suppose that the receiver has received or decoded all information packets up to and including packet u_j. Feedback can be used to communicate this to the transmitter allowing it to adjust the coding window so that the lower edge of the window is $L = j + 1$ for all subsequent coded packets. The generator matrix shown in Figure 3 illustrates this sliding window approach, where the columns indicate the information packets that need to be sent and the rows indicate the composition of the packet transmitted at any given time.

2) Channel: For our analysis we assume that packets are transmitted through an erasure packet channel and that the probability of packet erasure, ϵ, is known (e.g. estimated using receiver feedback) at the transmitter.

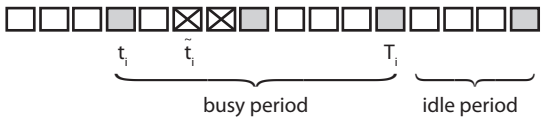


Fig. 4: Illustrating notation used. Clear rectangles indicate information packets, shaded rectangles coded packets, crosses indicate erasures, coded packets are inserted every $l = 4$ slots. t_i is the coded packet slot immediately preceding the information packet erasure at slot $\tilde{t}_i$ which pauses in-order delivery at the receiver, T_i the coded packet slot at which in-order delivery resumes. The information packet at slot t_i+1 is delivered without delay, but any information packets in slots $\{\tilde{t}_i, \cdots, T_i\}$ are delayed.

3) Decoder: The receiver decodes on-the-fly once enough packets/degrees of freedom have been received. In more detail, the receiver maintains a generator matrix G_t at time t which is similar to that shown in Figure 3 except that it is composed only of the coefficients obtained from received packets. If G_t is full rank, Gaussian elimination is used to recover from any packet erasures that may have occurred during transit.

B. Analysis of In-Order Delivery Delay

When using the proposed streaming code, at the receiver information packets are delivered in-order until an erasure of an information packet occurs. Upon erasure, in-order delivery is paused (arriving packets are buffered) until the decoder receives as many coded packets as the number of erasures, at which point in-order delivery resumes. This buffering delay is the in-order delivery delay, which for each packet is equal to the number of time slots a packet waits during the decoding process before it is delivered to the application. Note that this delay includes the delay caused by inserting coded packets into the information packet stream.

Let $\{\tilde{t}_i\}$ denote the sequence of slot times at which erasure of an information packet pauses in-order delivery and $\{T_i\}$ the corresponding sequence of times at which in-order delivery resumes. Note that the T_i must be a slot at which a coded packet is transmitted. Letting $t_i = \lfloor \tilde{t}_i/l \rfloor l$ be the coded packet slot immediately preceding slot $\tilde{t}_i$, we can then define the sequence of coded packet slots $\{t_1, T_1, t_2, T_2, \cdots\}$. See Figure 4 for a schematic illustration. Slots $\{t_i+1, t_i+2, \cdots, T_i\}$ contain information packets delayed by the i'th pause, plus perhaps non-delayed packets $\{t_i+1, \cdots, \tilde{t}_i\}$ and this set of slots is referred to as the i'th "busy" period. Slots $\{T_i+1, \cdots, t_{i+1}\}$ can be partitioned into intervals $\{T_i+1, T_i+l\}$, $\{T_i+l+1, T_i+2l\}$, *etc.* each of size l slots and ending with a coded packet slot (since T_i and t_i are both coded packet slots). Each of these intervals of l slots is referred to as an "idle" period.

The busy/idle period terminology is analogous with a queueing system operating in embedded time corresponding to the coded packet slots. Information packet erasures can be thought of as queue arrivals and reception of coded packets as queue service. Pauses in in-order delivery then correspond to periods when the queue size is non-zero.

Index the busy/idle periods by $j = 1, 2, \cdots$ and let $i(j)$ be the index of the pause corresponding to the j'th busy period (i.e., the j'th busy period consists of slots $\{t_{i(j)}, \cdots, T_{i(j)}\}$). With the j'th period we associate a random variable S_j, with $S_j = 0$ for an idle period and $S_j = (T_{i(j)} - t_{i(j)})/l$ for a busy period (i.e., S_j equals to how many coded packets transmitted before delivery resumes). Since packet erasures are i.i.d., the busy/idle periods form a renewal process and the $\{S_j\}$ are i.i.d. Letting $S \sim S_j$ the following theorem completely characterises the probability distribution of the busy time S and is one of our main results.

Theorem 1 (Busy Time)**.** *In a packet erasure channel with erasure probability ϵ, suppose we insert a coded packet in between every $l-1$ information packets. Assume that each coded packet can help us to recover from one erasure. We have:*

I. For all values of ϵ and l such that $l\epsilon < 1$, the mean of the probability distribution of S exists and is finite.

II.

$$p_S(s) = \begin{cases} (1-\epsilon)^{l-1} & \text{for } s = 0 \\ (l-1)\,\epsilon\,(1-\epsilon)^{l-1} & \text{for } s = 1 \\ \frac{l-1}{s}\epsilon^s\,(1-\epsilon)^{s(l-1)}\binom{(s-1)l}{s-1} & \text{for } s > 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

III.

$$E(S) = \frac{(l-1)\,\epsilon\,(1-\epsilon)^{l-1}}{1-l\epsilon} \tag{3}$$

$$E\left(S^2\right) = E(S) + \frac{l\,(l-1)\,\epsilon^2\,(1-\epsilon)^l}{(1-l\epsilon)^3} \tag{4}$$

Proof. See Appendix. □

Observe that the requirement that $l\epsilon < 1$ for S to have finite mean is a natural one. The rate of the coding scheme is $R = \frac{l-1}{l} = 1 - \frac{1}{l}$. Since this rate of transmission should be less than the channel capacity, we require $R < 1 - \epsilon$, and so $l\epsilon < 1$.

We also emphasise that the in-order delivery delay expressions in Theorem 1 are exact (they are not bounds) and have an easy to evaluate closed-form (they are not combinatorial in nature). This is notably different from previous analysis of in-order delivery delay and derives from the favourable structure of the low-delay coding scheme.

To continue the analysis, we introduce the random variable $S^+ = \max\{S, 1\}$. S^+ helps us to count the number of intervals in the communication interval T. It is straightforward to compute the probability distribution of S^+ as follows:

Corollary 1. *Let $S^+ = \max\{S, 1\}$. We have:*

I. For all values of ϵ and l such that $l\epsilon < 1$, the mean of the probability distribution of S^+ exists and is finite.

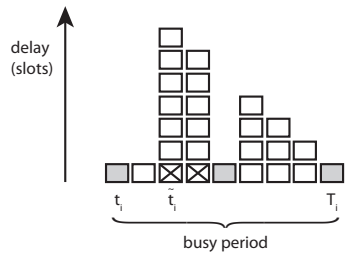


Fig. 5: Illustrating delay introduced by erasures. In this example the information packet at slot $\tilde{t}_i$ is delayed by 6 slots, the information packet at slot $\tilde{t}_i + 1$ by 5 slots, the information packet at $\tilde{t}_i + 3$ by 3 slots and so on. It can be seen that the sum-delay is the area under a triangle of base $T_i - t_i$ slots and height $T_i - t_i$ slots, less the area associated with any coded packets.

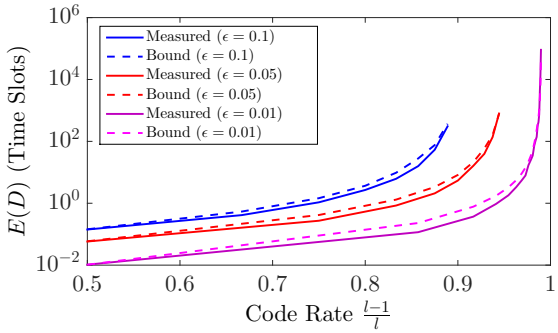


Fig. 6: Measured mean in-order delivery delay and upper bound versus code rate $^{l-1}/_{l}$ for different i.i.d. packet erasure rates ϵ.

II.

$$P_{S^+}(s) = \begin{cases} (l\epsilon + 1 - \epsilon)(1-\epsilon)^{l-1} & \text{for } s = 1 \\ \frac{l-1}{s}\epsilon^s (1-\epsilon)^{s(l-1)} \binom{(s-1)l}{s-1} & \text{for } s > 1 \\ 0 & \text{otherwise.} \end{cases}$$

III.

$$E\left(S^+\right) = \frac{(1-\epsilon)^l}{1 - l\epsilon}. \tag{5}$$

Combining Theorem 1 and Corollary 1 with the following result allows us to obtain a simple closed-form bound on the mean in-order delivery delay:

Theorem 2 (In-Order Delivery Delay)**.** *At the receiver, the asymptotic mean in-order delivery delay for information packets is upper bounded by* $\frac{E[S^2](l-1)}{2E[S^+]}$ *slots.*

Proof. See Appendix. □

The comparison of this upper bound for in-order delay with simulated results is provided in Fig. 6. It can be seen that the upper bound is tight at both low and high coding rates. Furthermore, it is reasonably tight at intermediate coding rates. Despite its simple form, it is therefore quite powerful.

C. Throughput and Rate

We can expect that the improved delay performance of the low delay coding scheme carries a throughput price. However, it turns out that this price is a small one. For a stream of N packets, a decoding error may occur since it is possible that a burst of errors near the end of the stream may not allow for sufficient time to transmit the necessary coded packets to recover the lost information packets. That is, a number of information packets at the end of a transmission may be lost. Define the good throughput GT as the ratio of the number of information packets delivered to the receiver and the number of packets transmitted by the transmitter. The good throughput is a random variable and its behavior is characterized in the following theorem:

Theorem 3 (Throughput)**.** *Consider the transmission of a coded stream of length N over an erasure channel and that $l\epsilon < 1$. For any $R_0 < 1 - \frac{1}{l}$ and $\delta > 0$, there exist an N large enough such that*

$$P(GT > R_0) > 1 - \delta. \tag{6}$$

Proof. See Appendix. □

Recall that the rate of transmission of the low delay coding scheme is $\frac{l-1}{l} = 1 - \frac{1}{l}$ and that the capacity of the erasure channel is $1 - \epsilon$, so the condition $l\epsilon < 1$ allows all coding rates up to the channel capacity. Theorem 3 therefore tells us that the low delay code is asymptotically capacity achieving as $N \to \infty$.

In other words, the fraction of information packets lost can be made arbitrarily small for sufficiently long transfers. This is because the length of the possible failure at the end of a transmission is independent of the length of the transmission. Therefore, the sacrifice in the good throughput becomes negligible as the length of the transmission grows. Of course, losing any information packet is undesirable. However, use of feedback or sending a small number of additional coded packets at the end of a transfer allows any straggling information packet erasures to be recovered.

D. Encoding and Decoding Complexity

The encoding and decoding complexity is dependent on the management of the coding window. As already noted, to limit the complexity we can use a sliding window approach that keeps track of the decoding process at the receiver. This results in a complexity that is polynomial in $E(S)$ and is considered in more detail below. Alternatively, the encoding and decoding might be constrained to the kl preceding information packets. This will result in a small probability of decoding failure $\Pr(S > k)$, but this will go to zero exponentially as k grows (see Theorem 1).

The complexity of the sliding window scheme depends on the process S. Assuming that $S = k$, Gaussian elimination can be performed at each step requiring approximately $\frac{2k^3}{3}$ arithmetic operations.Using the elementary renewal theorem[8], we have:

Theorem 4. *The total number of arithmetic operation C_d in a stream of length N_t which consists of $N_i = {}^{(l-1)N_t}/_{l}$ information packets, satisfies the following:*

$$\lim_{N_i\to\infty}\frac{1}{N_i}C_d = \frac{3}{2}\frac{1-l\epsilon}{(l-1)(1-\epsilon)^l}E\left(S^3\right), \tag{7}$$

where

$$E\left(S^3\right) = \frac{l(l-1)\epsilon^2(1-\epsilon)^l\left(2-2\epsilon-2l\epsilon^2+l\epsilon+l^2\epsilon^3\right)}{(1-l\epsilon)^5} + E\left(S^2\right). \tag{8}$$

Proof. The computation of $E(S^3)$ is similar to the computation of $E(S^2)$ in Theorem 1. □

While the number of arithmetic operations per information packet becomes large as the code rate approaches the capacity, the complexity in the low delay regime (which is the regime in which the low delay code is of interest) is perfectly manageable, see Table I and also the experimental results later.

TABLE I: The average number of arithmetic operations performed in the decoder per information packet

		$l = {}^{x}/_{\epsilon}$				
	x	0.5	0.6	0.7	0.8	0.9
ϵ	0.02	0.67	1.93	7.11	41.74	769.58
	0.1	3.13	8.87	32.56	190.96	3525

IV. Low Delay Performance Evaluation

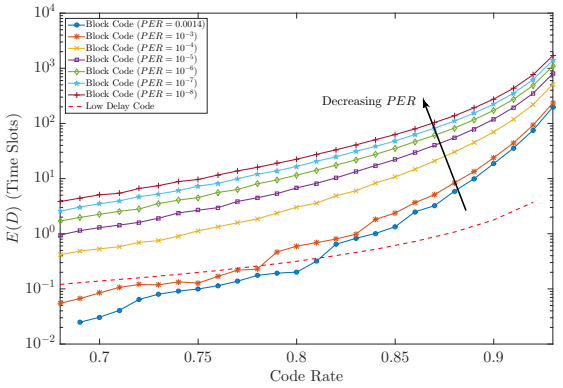
A. Simulation Results

We begin by carrying out simulations for i.i.d. erasures with packet erasure probability ϵ and for correlated packet erasures described by a two state Markov chain. The Markov chain has a "good" state (i.e., state G) with packet erasure rate $\epsilon = 0$, a "bad" state (i.e., state B) with packet erasure rate $\epsilon = 1$, and the transition probability matrix

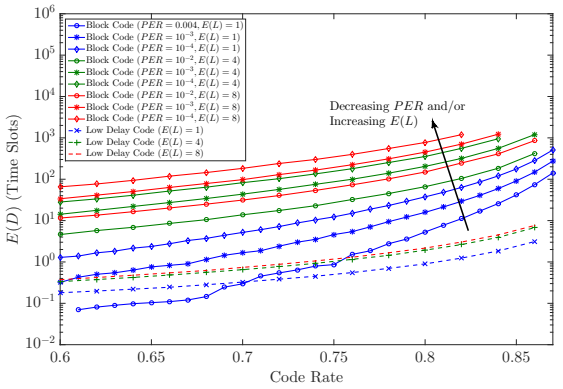
$$P_{GC} = \begin{bmatrix} 1-\gamma & \gamma \\ \beta & 1-\beta \end{bmatrix}. \tag{9}$$

Two parameters are used to generate the transition probabilities β and γ: the steady-state probability of state B, $\pi_B = {}^{\gamma}/_{\gamma+\beta}$; and the expected burst length, $E(L) = {}^{1}/_{\beta}$. Within the figures presented in this section, we will refer to the i.i.d. model by referencing either erasure rate ϵ or the 2-tuple $(\pi_B, E(L) = 1)$ (note that π_B equals the erasure rate). When the correlated loss model is assumed, the 2-tuple $(\pi_B, E(L) > 1)$ will be used.

We first consider the open-loop case where feedback communicating the successful reception of a packet is unavailable. In this case there is always non-zero probability of decoding failure for the block code, corresponding to the event that the number of packet losses over a block exceeds the number of codes packets in the block. Therefore, there remains a non-zero packet erasure rate (PER) after coding is applied. For the low-delay block code this probability is close to zero for sufficiently large connections. A comparison of the two codes is shown in



(a) I.I.D. Packet Losses with $\epsilon = 0.05$

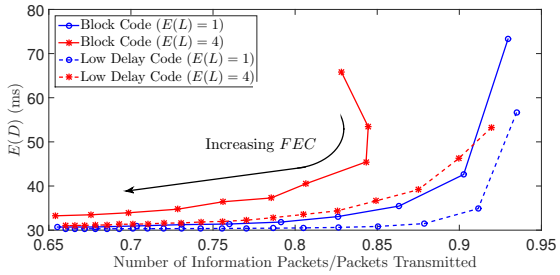


(b) Correlated Packet Losses with $\pi_B = 0.1$

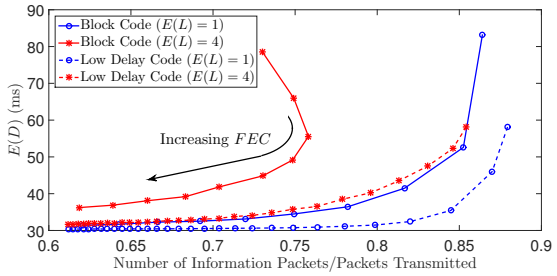
Fig. 7: Open-loop mean in-order delivery delay, $E(D)$, versus code rate for a systematic block code and the low delay code ($E(L)$ is the expected packet erasure burst duration).

Figure 7(a) where the mean in-order delivery delay $E(D)$ is plotted as a function of the coding rate for $\pi_B = 0.05$ and $\pi_B = 0.1$ when $E(L) = 1$ (i.e., the packet erasures are i.i.d.). A similar comparison is shown in Figure 7(b) for correlated packets losses where $E(L) \geq 1$. Each solid line shows the delay of the block code vs the coding rate when the block size is adjusted to hold the packet erasure rate constant (as the coding rate increases the block size must also grow to achieve the same PER). The mean in-order delivery delay for the low delay code is shown as a dotted line.

It can be seen that the low delay code achieves a smaller in-order delivery delay than block codes for the cases that are of the most interest. The reduction in delay is substantial, being on the order of a magnitude or more for any given code rate. The regime where this is not the case is when the rate of decoding failures for the block codes is large (e.g. $PER \geq 10^{-3}$ when compared to a channel packet erasure rate of 0.05) and the coding rate is small (e.g., $c \leq 0.8$). Recall that the low-delay code has a $PER \approx 0$, so the delay comparison is not really fair in within this regime. Also note that the block code in this regime typically has a block size of only 1 to 3 packets, which is far smaller than is usual for block codes. As both



(a) $\epsilon = 0.05$



(b) $\epsilon = 0.1$

Fig. 8: Mean in-order delivery delay, $E(D)$, versus code rate on a 25 $Mbps$ link with an RTT of 60 ms. Both a systematic block code and the low delay code are shown where feedback is used to signal retransmissions and packet erasures are correlated ($E(L)$ is the expected packet erasure burst duration). The non-uniqueness in the abscissa for the block code when $E(L) = 4$ occurs when the probability of decoding each generation or block without needing to retransmit additional degrees of freedom is very low.

the code rate and block size are increased, quantization due to these small block sizes results in the fluctuations shown in the delay-rate curves within Figure 7.

It can also be seen from Figure 7(b) that the block code's in-order delivery delay is much more sensitive to correlated losses than the low delay code's delay. This is a result of the low-delay code removing the requirement to partition the packet stream into blocks or generations prior to coding.

Simulation results for the closed-loop case are shown in Figure 8. The major difference between this and Figure 7 is that feedback is used to help communicate the receiver's need for additional degrees of freedom. When considering the block code, feedback is used to initiate retransmissions in the form of coded packets if a block cannot be decoded. These retransmissions occur until every block can be decoded and delivered. When considering the low delay code, feedback is used to adjust the code rate to ensure frequent decoding opportunities.

While the gain in delay is not as pronounced as the open-loop case, the low delay code achieves a lower in-order delivery delay than the block code over the entire range of code rates (measured as the total number of information packets divided by the total number of both transmitted information and coded packets). Furthermore, the figure highlights the inability of block codes to re-

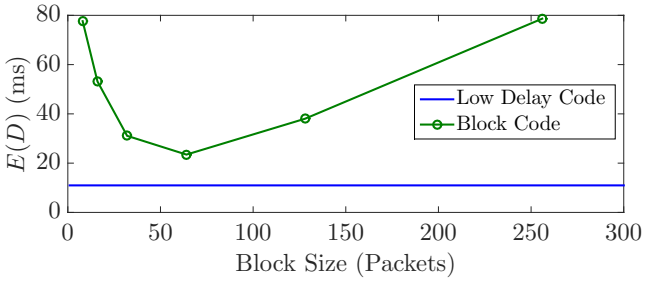


Fig. 9: Mean in-order packet delivery delay, $E(D)$, versus the block size for a random linear block code and low delay code. A link rate of 25 $Mbps$, RTT of 60 ms, packet erasure rate of 10%, redundancy of 15%.

cover from correlated losses. This is shown by the non-uniqueness in the abscissa. Each curve is generated by increasing the forward error correction (FEC) code rate. For smaller FEC rates where a decoding error occurs, retransmissions are necessary resulting in larger delays. As the FEC is increased, the probability of requiring retransmissions to decode a block decreases resulting in lower delay.

B. Experimental Results

We implemented FEC using the streaming code within the reliable UDP transport used in Section II. The implementation is actually somewhat simpler than for a block code since there is no need for book-keeping of generations/blocks. Finite field operations are implemented using fast SIMD instructions and on the commodity hardware described in Section II rates of 600-800Mbps are readily achieved for loss rates up to 20% while on a low end mobile handset rates of 100-200Mbps are achieved.

Revisiting the example in Section II, Figure 9 compares the measured mean in-order packet delivery delay, $E(D)$, for the low delay code and the systematic block code with different block sizes using the same coding rate. Since feedback is used, there are no decoding failures for either type of code. As already noted, there is an "optimal" block size where the block code achieves the lowest delay. However, it can be seen that the low delay code achieves a mean delay that is about half of of this value. The low delay code achieves this improved delay performance while also avoiding the extra complexity that would be needed to adapt block size to the path characateristics, as already discussed.

Figure 10 presents experimental measurements as both the block size and coding rate are varied. The data is plotted as mean delay vs effective coding rate (accounting for coded packets sent via both FEC and retransmissions) so that the trade-off between delay and rate discussed earlier can be assessed. It can be seen that the delay vs coding lower boundary is achieved for block sizes between $k = 32$ and $k = 64$ and a coding rate. Notable is that for any given coding rate the low delay code achieves a smaller delay than the best block code. Recall that the block code used for comparison is a modern, high performance code,

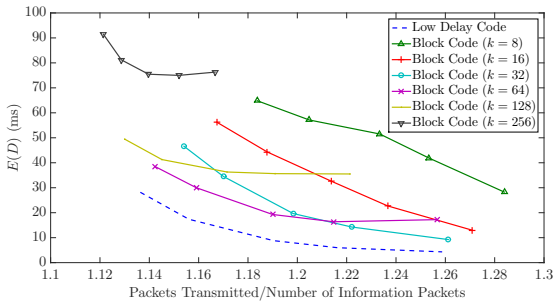


Fig. 10: Mean in-order packet delivery delay, $E(D)$, versus the number of packets transmitted divided by the number of information packets for a systematic random linear block code and low delay code. A link rate of 25 $Mbps$, RTT of 60 ms, loss rate of 10%.

and we expect similar behaviours when other types of block codes are used. While not shown here due to lack of space, similar results were obtained for a wide range of network bandwidths and RTTs.

V. Related Work

A. Streaming Codes

We use the term *streaming code* to refer to codes which do not require a bit/packet stream to be partitioned into blocks or generations before coding operations can occur. Probably the most common examples are convolutional codes. Most work on the performance of convolutional codes has focused on their error-correcting capabilities, particularly to bursty errors, and only a few have investigated delay. For example, Martinian et al. [9] and Hehn et al. [10] investigate the decoding delay of convolutional codes specifically designed for channels with burst errors; and Lee et al. [11] investigate the delay performance of orchard codes (a class of convolutional codes). Badr et al. [12] consider an adversarial erasure channel and propose a streaming code that performs when the erasures in the channel are bursty.

Classical convolutional codes do not lend themselves to recoding at intermediate nodes within a network. This has motivated work on convolutional network codes (proposed in [13] and studied extensively by [14], [15], [16], [17]). Delay bounds for convolutional network codes are provided by Guo et al. [18]. The delay of other codes of this type, such as those proposed by Joshi et al. [19], have also shown promising results. However, limitations in the models used within this work make it unclear whether or not their results can be extended to the scenarios in which we are interested. We note that Tömösközi et al. [20] introduce a non-systematic code similar to our low delay code. Experimental results show significant delay gains over Reed-Solomon codes, but no analysis is provided.

B. Block Codes

Block codes require a bit/packet stream to be partitioned into generations or blocks, each generation or block being treated independently from the rest. For example, assume that a message of size N information packets u_m, $m = 1, \ldots, N$, is partitioned into blocks or generations of size k packets. Coded packets $c_{i,j}$, $i = 1, \ldots, \lceil N/k \rceil$, $j = 1, 2, \ldots$, are then generated separately for each block. If the code is systematic, the information packets in each block/generation are transmitted first with the coded packets transmitted after to help recover from any errors or erasures. If the code is not systematic, only the coded packets are transmitted. Previous work has primarily focused on the decoding delay of non-systematic constructions [21], [22], [23], [24], [25], [26], [27], [28], and shown that the decoding delay is essentially proportional to the block size. The in-order delivery delay of systematic block codes is lower than that of non-systematic codes but remains essentially proportional to the block size (with a smaller pre-factor than for non-systematic codes), see Cloud et al. [29] and references therein.

C. Baseline Block Code

We use the following systematic random linear block code as a baseline for performance comparison. This block code is similar to that considered in [29] and is constructed as follows. We generate $n - k$ coded packets, $c_{i,j}$, $i = 1, \ldots, \lceil N/k \rceil$, $j = 1, \ldots, n - k$, from each block of k information packets, which results in a code of rate k/n. Each coded packet is a weighted random linear combination of the information packets within its block, i.e.,

$$c_{i,j} := \sum_{m=(i-1)k+1}^{ik} w_{i,j,m} u_m, \tag{10}$$

where each information packet u_m is treated as a vector in an appropriate finite field $\mathbb{F}$ of size Q and the coefficients $w_{i,j,m}$ are drawn i.i.d. uniformly at random from $\mathbb{F}$. It should be noted that calculations in field $\mathbb{F}$ are always carried out over symbols of size Q rather than the packets themselves. A systematic code is obtained by first transmitting the k information packets followed by the $n - k$ coded packets.

Maximum likelihood decoding is used *i.e.* Gaussian elimination. Should an erasure occur, any coded packet can be used to help reconstruct/decode the missing information packet. For sufficiently large field size Q and no more than $n - k$ erasures, any combination of k packets in each block can be used to recover from the erasures with high probability. If more than $n - k$ erasures occurs, a decoding failure occurs and some of the information packets within the block may be unrecoverable. Otherwise, information packets will be recovered with high probability when at least k packets have been successfully received.

This code is asymptotically capacity achieving over erasure channels as block size $n \to \infty$ [30]. It also provides a good baseline for comparison since it is a modern, high performance code that has a number of optimality properties (*i.e.,* it is representative of the best possible block code performance). In particular, this code minimises the probability of a decoding failure for any given coding rate

over a large class of block codes in addition to minimizing the decoding delay [30].

VI. Conclusions

Motivated by 5G low latency requirements we introduce a class of streaming codes for FEC on packet erasure channels that is capacity achieving and provides a superior throughput-delay trade-off compared to block codes. The mathematical analysis of throughput and delay is one of the primary contributions of the paper, requiring the development of a number of novel analytic tools based on both queuing and coding theory. Simulation and experimental results are also presented and demonstrate that a superior rate-delay trade-off is indeed achieved in practice across a wide range of realistic network conditions.

References

[1] *5G White Paper*. Next Generation Mobile Networks (NGMN) Alliance, 2015. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

[2] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: the virtue of gentle aggression," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 159–170, 2013.

[3] W. Zhou, Q. Li, M. Caesar, and P. Godfrey, "ASAP: A low-latency transport layer," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 2011, p. 20.

[4] S.Souders, "Velocity and the bottom line," in *Velocity (Web Performance and Operations Conference)*, 2009. [Online]. Available: http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html

[5] *Next Generation Protocols – Market Drivers and Key Scenarios*. European Telecommunications Standards Institute (ETSI), 2016. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp17_Next_Generation_Protocols_v01.pdf

[6] J. Iyengar and I.Swett, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," *IETF Internet Draft*, 2015. [Online]. Available: https://tools.ietf.org/html/draft-tsvwg-quic-protocol-00

[7] *Open Fast Path*, 2016. [Online]. Available: http://www.openfastpath.org/

[8] R. G. Gallager, *Stochastic Processes, Theory for Applications*. Cambridge University Press, 2013.

[9] E. Martinian and C. E. W. Sundberg, "Burst erasure correction codes with low decoding delay," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2494–2502, Oct 2004.

[10] T. Hehn and J. B. Huber, "Ldpc codes and convolutional codes with equal structural delay: a comparison," *IEEE Transactions on Communications*, vol. 57, no. 6, pp. 1683–1692, June 2009.

[11] J. Lee and A. Shiozaki, "Decoding delays of orchard codes," *IEE Proceedings I - Communications, Speech and Vision*, vol. 139, no. 4, pp. 413–417, Aug 1992.

[12] A. Badr, A. Khisti, W. t. Tan, and J. Apostolopoulos, "Streaming codes with partial recovery over channels with burst and isolated erasures," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 3, pp. 501–516, April 2015.

[13] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct 2003.

[14] E. Erez and M. Feder, "Convolutional network codes," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, June 2004, pp. 146–.

[15] ——, "Efficient network codes for cyclic networks," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, Sept 2005, pp. 1982–1986.

[16] S. y. R. Li and R. W. Yeung, "On convolutional network coding," in *2006 IEEE International Symposium on Information Theory*, July 2006, pp. 1743–1747.

[17] E. Erez and M. Feder, "Efficient network code design for cyclic networks," *IEEE Transactions on Information Theory*, vol. 56, no. 8, pp. 3862–3878, Aug 2010.

[18] W. Guo, X. Shi, N. Cai, and M. Medard, "Localized dimension growth: A convolutional random network coding approach to managing memory and decoding delay," *IEEE Transactions on Communications*, vol. 61, no. 9, pp. 3894–3905, September 2013.

[19] G. Joshi, Y. Kochman, and G. W. Wornell, "On playback delay in streaming communication," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, July 2012, pp. 2856–2860.

[20] M. Toemoeskoezi, F. H. P. Fitzek, F. H. P. Fitzek, D. E. Lucani, M. V. Pedersen, and P. Seeling, "On the delay characteristics for point-to-point links using random linear network coding with on-the-fly coding capabilities," in *European Wireless 2014; 20th European Wireless Conference; Proceedings of*, May 2014, pp. 1–6.

[21] A. Heidarzadeh, "Design and analysis of random linear network coding schemes: Dense codes, chunked codes and overlapped chunked codes," Ph.D. dissertation, Carleton University, Ottawa, Canada, 2012.

[22] D. E. Lucani, M. Medard, and M. Stojanovic, "Broadcasting in time-division duplexing: A random linear network coding approach," in *2009 Workshop on Network Coding, Theory, and Applications*, June 2009, pp. 62–67.

[23] ——, "Online network coding for time-division duplexing," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010, pp. 1–6.

[24] 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*, April 2009, pp. 1800–1808.

[25] M. Nistor, J. Barros, F. Vieira, T. T. V. Vinhoza, and J. Widmer, "Network coding delay: A brute-force analysis," in *Information Theory and Applications Workshop (ITA), 2010*, Jan 2010, pp. 1–5.

[26] J. K. Sundararajan, P. Sadeghi, and M. Medard, "A feedback-based adaptive broadcast coding scheme for reducing in-order delivery delay," in *2009 Workshop on Network Coding, Theory, and Applications*, June 2009, pp. 1–6.

[27] W. Zeng, C. T. K. Ng, and M. Médard, "Joint coding and scheduling optimization in wireless systems with varying delay sensitivities," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, June 2012, pp. 416–424.

[28] K. Premkumar, X. Chen, and D. J. Leith, "Proportional fair coding for wireless mesh networks," *IEEE/ACM Transactions on Networking*, vol. 23, no. 1, pp. 269–281, Feb 2015.

[29] J. Cloud, D. Leith, and M. Médard, "A coded generalization of selective repeat arq," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 2155–2163.

[30] V. G. Subramanian and D. J. Leith, "On a class of optimal rateless codes," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, Sept 2008, pp. 418–425.

[31] J. C. Tanner, "A derivation of the borel distribution," *Biometrika*, vol. 48, no. 1/2, pp. 222–224, 1961. [Online]. Available: http://www.jstor.org/stable/2333154

[32] R. L. Graham, D. E. Knuth, and O. Patashnik., *Concrete Mathematics*. Addison-Wesley, 1994.

Appendix I: Proof of Theorem 1

The following lemma corresponds to a result of Tanner [31] for the busy time distribution of a G/D/1 queue.

Lemma 5 ([31]). *Consider a decoding procedure at time $t = 0$ with r erasures already occurred in $t < 0$. Suppose that the decoder performs a successful decoding procedure at time $t = lk$ when it receives the coded packet $S = k$ and decodes all the erasurd packets up to that point. The*

followings are necessary and sufficient conditions for a decoding period to contain just k decoding packets given that the decoding process starts with r erasures in the previous interval:

(i) precisely $k-r$ erasures happen in the period of lk;

(ii) the patterns of these erasures shall be admissible*. The term* admissible *is used here to mean that at least one erasure should happen during a time $t=(r+1)l$, at least two within $(r+2)l$, and so on, so that the decoding process will remain activated for the whole of the time $t=kl$.*

The probability of (i) is

$$P(\mathcal{B}(\epsilon,kl)=k-r)=\binom{kl}{k-r}\epsilon^{k-r}(1-\epsilon)^{k(l-1)+r} \quad (11)$$

and the probability of any pattern of the $k-r$ erasures being admissible is r/k.

We need the following lemmas:

Lemma 6 (Pascal's Rule, Vandermonde's identity)**.** *[32]*

$$\binom{n}{k}=\binom{n-1}{k-1}+\binom{n-1}{k}$$
$$\binom{m+n}{r}=\sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k},\quad m,n,r\in\mathbb{N}_0$$
$$k\binom{n}{k}=\frac{n}{2}(\binom{n}{k}-\binom{n-1}{k}+\binom{n-1}{k-1})$$

Lemma 7.

$$\sum_{r=2}^{\min(k,l)}(r-1)\binom{l}{r}\binom{(k-1)l}{k-r}=\binom{(k-1)l}{k} \quad (12)$$

Proof. We use the following identities that follow from Vandermonde's identity:

$$\sum_{r=2}^{\min(k,l)}\binom{l}{r}\binom{(k-1)l}{k-r}=\binom{kl}{k}-\binom{(k-1)l}{k}-l\binom{(k-1)l}{k-1} \quad (13)$$

$$\sum_{r=2}^{\min(k,l)}\binom{l-1}{r-1}\binom{(k-1)l}{k-r}=\binom{kl-1}{k-1}-\binom{(k-1)l}{k-1} \quad (14)$$

$$\sum_{r=2}^{\min(k,l)}\binom{l-1}{r}\binom{(k-1)l}{k-r}=\binom{kl-1}{k}-\binom{(k-1)l}{k}-(l-1)\binom{(k-1)l}{k-1} \quad (15)$$

Hence, $\sum_{r=2}^{\min(k,l)}(r-1)\binom{l}{r}\binom{l(k-1)}{k-r}=\frac{l}{2}\sum_{r=2}^{\min(k,l)}\big[(1-\frac{2}{l})+\binom{l-1}{r-1}+\binom{l-1}{r}\big]\binom{l(k-1)}{k-r}\overset{(13),(14),(15)}{=}\binom{(k-1)l}{k}$. □

Proof of Theorem 1, Part I. We know that for the decoding process to go beyond k we need at least k erasures in the first kl interval. This means that there are cases with more than k erasures in the first kl interval that the decoding process stops before k but for it to be greater than k we should have at least k erasures. Suppose E_{kl} denotes the number of erasures in the interval kl. We have $P(S>k|E_{kl}\le k)=0$. Formally speaking: $P(S>k)=P(E_{kl}>k)P(S>k|E_{kl}>k)+P(E_{kl}\le k)P(S>k|E_{kl}\le k)=P(E_{kl}>k)P(S>k|E_{kl}>k)<P(E_{kl}>k)=P(\mathcal{B}(\epsilon,kl)>k)<Q(\sqrt{k}\frac{1-l\epsilon}{\sqrt{l\epsilon(1-\epsilon)}})<\frac{1}{2}e^{-k\frac{(1-l\epsilon)^2}{l\epsilon(1-\epsilon)}}$. The Q-function is the tail probability of the standard normal distribution, $Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^\infty\exp\left(-\frac{u^2}{2}\right)du$. In the last part we used the Chernov bound and the Normal distribution $(\epsilon kl, kl\epsilon(1-\epsilon))$ as an approximation of the binomial distribution for large values of k. This bound only holds if $l\epsilon<1$. We know $E(S)=\sum_k P(S>k)$. Putting these two together we can deduce that $E(S)$ is finite if $l\epsilon<1$.

Alternatively, if $l\epsilon<1$, using the Chernov bound we have: $P(\mathcal{B}(\epsilon,kl)>k)<e^{-k\frac{(1-l\epsilon)^2}{l\epsilon}}$. □

Proof of Theorem 1, Part II. The goal is to determine $P(S=k)$. We know that $S=0$ if there is no erasure in the first l-interval or only the coding packet is erased, so $P(S=0)=(1-\epsilon)^l+\epsilon(1-\epsilon)^{l-1}=(1-\epsilon)^{l-1}$. We also know that $P(S=1)$ if one and only one of the information packets is lost in the first l-interval, so $P(S=1)=(l-1)\epsilon(1-\epsilon)^{l-1}$. We know that for $k>1$ we need at least $r=2$ erasures in the first l-interval which has the probability of $P(\mathcal{B}(\epsilon,l)=r)=\binom{l}{r}\epsilon^r(1-\epsilon)^{l-r}$ for $2\le r\le l$. Starting from the second decoding l-interval, there are $r-1$ erasures to be taken care of. The reason is that either the first coded packet is erased, which means $r-1$ information packets are erased, or the first coded packet is not erased and it will eventually help us to decode one of the r erasures. Using Lemma 5, knowing that there have been r erasures in the first l-interval, we have the following for $k>1$, $P(S=k|r)=\frac{r-1}{k-1}P(\mathcal{B}(\epsilon,l(k-1))=k-r)$. This means that we can allow $k-r$ erasures in the $l(k-1)$-interval. Putting these two together we have

$$\begin{aligned}
P(S=k)&=\sum_{r=2}^{\min(k,l)}P(\mathcal{B}(\epsilon,l)=r)\frac{r-1}{k-1}P(\mathcal{B}(\epsilon,l(k-1))=k-r)\\
&=\sum_{r=2}^{\min(k,l)}\binom{l}{r}\epsilon^r(1-\epsilon)^{l-r}\frac{r-1}{k-1}\binom{l(k-1)}{k-r}\epsilon^{k-r}(1-\epsilon)^{k(l-1)-l+r}\\
&=\sum_{r=2}^{\min(k,l)}\frac{r-1}{k-1}\binom{l}{r}\binom{l(k-1)}{k-r}\epsilon^k(1-\epsilon)^{k(l-1)}\\
&=\frac{1}{k-1}\epsilon^k(1-\epsilon)^{k(l-1)}\sum_{r=2}^{\min(k,l)}(r-1)\binom{l}{r}\binom{l(k-1)}{k-r}.
\end{aligned}$$

Using Lemma 7, we have

$$\begin{aligned}
P(S=k)&=\frac{1}{k-1}\epsilon^k(1-\epsilon)^{k(l-1)}\binom{(k-1)l}{k}\\
&=\frac{1}{k-1}\epsilon^k(1-\epsilon)^{k(l-1)}\frac{((k-1)l)!}{k!((k-1)l-k)!}\\
&=\frac{((k-1)l-k+1}{(k-1)k}\epsilon^k(1-\epsilon)^{k(l-1)}\frac{((k-1)l)!}{(k-1)!((k-1)l-k+1)!}\\
&=\frac{(k-1)(l-1)}{(k-1)k}\epsilon^k(1-\epsilon)^{k(l-1)}\binom{(k-1)l}{k-1}\\
&=\frac{l-1}{k}\epsilon^k(1-\epsilon)^{k(l-1)}\binom{(k-1)l}{k-1}
\end{aligned}$$

□

Proof of Theorem 1, Part III. We know that for $l\epsilon < 1$ the mean of probability distribution of S exists, we have $\sum_k P(S=k) = 1$, which means $\sum_{k=1}^{\infty} \frac{l-1}{k}\epsilon^k(1-\epsilon)^{k(l-1)}\binom{(k-1)l}{k-1} = 1-(1-\epsilon)^{l-1}$. By taking derivatives with respect to ϵ, we have $(1-\epsilon)^{l-2} = \sum_{k=1}^{\infty}\epsilon^{k-1}(1-\epsilon)^{k(l-1)}\binom{(k-1)l}{k-1} - (l-1)\epsilon^k(1-\epsilon)^{k(l-1)-1}\binom{(k-1)l}{k-1}$. That is, $\frac{E(S)}{(l-1)\epsilon} - \frac{E(S)}{1-\epsilon} = (1-\epsilon)^{l-2}$ and by re-arranging, we have $E(S) = \frac{(l-1)\epsilon(1-\epsilon)^{l-1}}{1-l\epsilon}$. By taking derivatives with respect to ϵ, we have $\frac{E(S^2)}{\epsilon} - \frac{E(S^2)(l-1)}{1-\epsilon} = \frac{\partial}{\partial\epsilon}E(S)$ which means $E(S^2) = \frac{(l-1)\epsilon(1-\epsilon)^{l-1}((1-l\epsilon)^2+l\epsilon(1-\epsilon))}{(1-l\epsilon)^3} = E(S) + \frac{l(l-1)\epsilon^2(1-\epsilon)^l}{(1-l\epsilon)^3}$. □

Appendix II: Proof of Theorem 2

Proof. Consider a transmission of a stream with length N_t a multiple of l at time t. This translates to $\frac{N_t}{l}$ l-intervals in time t and assume that the decoding process is consisted of l-intervals of length $s_1, s_2, \ldots, s_n, \ldots$. Remember that $S_1, S_2, \ldots, S_n$ is sequence of positive independent identically distributed random variables. Consider the S^+ process, and define J_n as $J_n = \sum_{i=1}^{n} S_i^+, \qquad n > 0$ and the renewal interval $[J_n, J_{n+1}]$ is a decoding l-interval. Then the random variable $(N_t)_{t\geq 0}$ given by $X_t = \sum_{n=1}^{\infty}\mathbb{I}_{\{J_n\leq t\}} = \sup\{n : J_n \leq t\}$ (where $\mathbb{I}$ is the indicator function) represents the number of decoding l-intervals that have occurred by time t, and is a renewal process.

Let $W_1, W_2, \ldots$ be a sequence of *i.i.d.* random variables denoting the sum of in-order delivery delay in each decoding l-interval. In other words, random variables S_js are the length of a decoding interval and random variables W_is are the sum of in order delivery for all packets during a decoding interval. We have two cases to consider. Case (i): suppose the j'th period is an idle period. Then $S_j = 0$ and the information packets are delivered in-order with no delay. Case (ii): suppose the j'th period is a busy period and the information packet erasure that initiated the busy period started in the first slot $t_{i(j)}+1$. Then the first information packet is delayed by $S_j l$ slots, the second by $S_j l - 1$ slots and so on. The sum-delay over all of the information packets in the busy period is therefore $\sum_{k=1}^{S_j l} k - \sum_{k=0}^{S_j-1} kl < \frac{S_j^2 l(l-1)}{2}$.

The random variable $Y_t = \sum_{i=1}^{X_t} W_i$ is a renewal-reward process and its expectation is the sum of in order delivery delay over the time-span of t. Based on the elementary renewal theorem for renewal-reward processes, we have $\lim_{t\to\infty}\frac{1}{t}\mathbb{E}[Y_t] = \frac{\mathbb{E}(W_1)}{\mathbb{E}(S_1^+)}$. Based on the construction of W_i, we have $\mathbb{E}[W_1] = \sum_{i=1}^{\infty}\mathbb{E}[W_1|S_1^+ = i]\Pr(S_1^+ = i) = \sum_{i=0}^{\infty}\mathbb{E}[W_1|S_1 = i]\Pr(S_1 = i) = \frac{l(l-1)}{2}E[S^2]$. We have $\lim_{t\to\infty}\frac{1}{t}\mathbb{E}[Y_t] \leq \frac{1-l\epsilon}{(1-\epsilon)^l}\frac{l(l-1)}{2}E[S^2]$ Since we assumed that $N_t = tl$, we have: $\lim_{N_t\to\infty}\frac{1}{N_t}\mathbb{E}[Y_t] \leq \frac{(l-1)(1-l\epsilon)}{2(1-\epsilon)^l}E[S^2]$

□

Appendix III: Proof of Theorem 3

Proof of Theorem 3 . Assume that the decoding process consists of intervals of lengths $s_1, s_2, s_3, \ldots, s_k, s_{k+1}, \ldots$, and without the loss of generality, assume that

$$\#\{s_j = 0\}_{j\leq k} + \sum_{j=1}^{k} s_j < \frac{N}{l} < \#\{s_j = 0\}_{j\leq k+1} + \sum_{j=1}^{k+1} s_j. \quad (16)$$

Note that if $S = 0$, then no erasure has occurred in the $(l-1)$-interval and all information packets are received without delay. The throughput of this scheme is $GT = (l-1)(\#\{s_j = 0\}_{j\leq k} + \sum_{j=1}^{k} s_j)/N = (l-1)(\#\{s_j = 0\}_{j\leq k+1} + \sum_{j=1}^{k+1} s_j - s_{k+1} + \mathbb{I}(s_{k+1} = 0))/N$, where $\mathbb{I}(s_{k+1}=0)$ is the indicator function and is equal to one if $s_{k+1} = 0$ and otherwise it is zero. From (16), we know

$$GT > \frac{l-1}{l} - \frac{(l-1)s_{k+1} + (l-1)\mathbb{I}(s_{k+1}=0)}{N}. \quad (17)$$

From Theorem 1, we know the probability distribution of S_{k+1}, so $P(GT > \frac{l-1}{l} - \frac{(l-1)f}{N}) = \Pr(S_{k+1} \leq f)$. In particular, using the Chernov bound for $\Pr(S > f)$, we have $P(GT > \frac{l-1}{l} - \frac{(l-1)f}{N}) > 1 - \frac{1}{2}e^{-f\frac{(1-l\epsilon)^2}{l\epsilon(1-\epsilon)}}$. Setting $\delta = \frac{1}{2}e^{-f\frac{(1-l\epsilon)^2}{l\epsilon(1-\epsilon)}}$, we can solve for f_δ. Assume that $R_0 = \frac{l-1}{l} - \gamma$, for any $N > \frac{(l-1)f_\delta}{\gamma}$, we have $P(GT > R_0) > 1-\delta$. □

Mohammad Karzand obtained the Electrical Engineering Degree at the University of Tehran in 2005. Then, he received his M.Sc. (2007) and Ph.D. (2013) from the School of Computer and Communication Sciences in Ecole Polytechnique Fédérale de Lausanne (EPFL). Since 2013, he has been working as a senior research fellow at the Hamilton Institute and Trinity College Dublin.

Doug Leith graduated from the University of Glasgow in 1986 and was awarded his PhD, also from the University of Glasgow, in 1989. Prof. Leith moved to the National University of Ireland, Maynooth in 2001 to establish the Hamilton Institute of which he was founding Director from 2001-2014. In 2014, Prof Leith moved to Trinity College Dublin to take up the Chair in Computer Systems in the School of Computer Science and Statistics.

PLACE PHOTO HERE

Jason Cloud received a Ph.D. degree in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT) and a B.S. degree in Engineering with a specialty in Electrical Engineering in 2002 from Colorado School of Mines and a M.S. degree in Electrical Engineering and Computer Science in 2011 from MIT. From 2003 to 2009, he was an officer and developmental engineer in the U.S. Air Force.

Muriel Médard