



## King's Research Portal

DOI:  
[10.1109/TCOMM.2018.2869791](https://doi.org/10.1109/TCOMM.2018.2869791)

[Link to publication record in King's Research Portal](#)

*Citation for published version (APA):*

Aliasgari, M., Kliewer, J., & Simeone, O. (2019). Coded Computation Against Processing Delays for Virtualized Cloud-Based Channel Decoding. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 67(1), 28-39. Article 8463544. <https://doi.org/10.1109/TCOMM.2018.2869791>

### **Citing this paper**

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

### **General rights**

Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

### **Take down policy**

If you believe that this document breaches copyright please contact [librarypure@kcl.ac.uk](mailto:librarypure@kcl.ac.uk) providing details, and we will remove access to the work immediately and investigate your claim.

# Coded Computation Against Processing Delays for Virtualized Cloud-Based Channel Decoding

Malihe Aliasgari\*, *Member, IEEE*, Jörg Kliewer\*, *Senior Member, IEEE*, and Osvaldo Simeone† *Fellow, IEEE*,

**Abstract**—The uplink of a cloud radio access network architecture is studied in which decoding at the cloud takes place via network function virtualization on commercial off-the-shelf servers. In order to mitigate the impact of straggling decoders in this platform, a novel coding strategy is proposed, whereby the cloud re-encodes the received frames via a linear code before distributing them to the decoding processors. Transmission of a single frame is considered first, and upper bounds on the resulting frame unavailability probability as a function of the decoding latency are derived by assuming a binary symmetric channel for uplink communications. Then, the analysis is extended to account for random frame arrival times. In this case, the trade-off between average decoding latency and the frame error rate is studied for two different queuing policies, whereby the servers carry out per-frame decoding or continuous decoding, respectively. Numerical examples demonstrate that the bounds are useful tools for code design and that coding is instrumental in obtaining a desirable compromise between decoding latency and reliability.

**Index Terms**—Coded computation, network function virtualization, cloud radio access network, large deviation, queuing.

## I. INTRODUCTION

Promoted by the European Telecommunications Standards Institute (ETSI), network function virtualization (NFV) has become a cornerstone of the envisaged architecture for 5G systems [2]. NFV leverages virtualization technologies in order to implement network functionalities on commercial off-the-shelf (COTS) programmable hardware, such as general purpose servers, potentially reducing both capital and operating costs. An important challenge in the deployment of NFV is ensuring carrier grade performance while relying on COTS components. Such components may be subject to temporary unavailability due to malfunctioning, and are generally characterized by randomness in their execution runtimes. The typical solution to these problems involves replicating the virtual machines that execute given network functions on multiple processors, e.g., cores or servers [3]–[6].

Among the key applications of NFV is the implementation of centralized radio access functionalities in a cloud radio

access network (C-RAN) [7], [8]. As shown in Fig. 1, each remote radio head (RRH) of a C-RAN architecture is connected to a cloud processor by means of a fronthaul (FH) link. Baseband functionalities are carried out on a distributed computing platform in the cloud, which can be conveniently programmed and reconfigured using NFV. The most expensive baseband function in terms of latency to be carried out at the cloud is uplink channel decoding [7], [9], [10].

The implementation of channel decoding in the cloud by means of NFV is faced not only with the challenge of providing reliable operation despite the unreliability of COTS servers, but also with the latency constraints imposed by retransmission protocols. In particular, keeping decoding latency at a minimum is a major challenge in the implementation of C-RAN owing to timing constraints from the link-layer retransmission protocols [11]–[13]. In fact, positive or negative feedback signals need to be sent to the users within a strict deadline in order to ensure the proper operation of the protocol. In [14], [15] it is argued that exploiting parallelism across multiple cores in the cloud can reduce the decoding latency by enabling decoding as soon as one can has computed its task. However, parallel processing does not address the unreliability of COTS hardware. A different solution is needed in order to address both unreliability and delays associated with cloud decoding.

The problem of straggling processors, that is, of processors lagging behind in the execution of a certain orchestrated function, has been well studied in the context of distributed computing [16]–[21]. Recently, it has been pointed out that, for the important case of linear functions, it is possible to improve over repetition strategies in terms of the trade-off between performance and latency by carrying out linear precoding of the data prior to processing, e.g., [22]–[30]. The key idea is that, by employing suitable linear (erasure) block codes operating over fractions of size  $1/K$  of the original data, a function may be completed as soon as any  $K$  or more processors, depending on the minimum distance of the code, have completed their operations. Coding has also been found to be useful addressing the straggler problem in the context of coded distributed storage and computing systems, see, e.g., [31]–[35].

In this paper, we explore the use of coded computing to enable reliable and timely channel decoding in a C-RAN architecture based on distributed unreliable processors. Specifically, we formally and systematically address the analysis of coded NFV for C-RAN uplink decoding. The only prior work on coded computing for NFV is [36], which provides numerical results concerning a toy example with three processors in

This work was supported in part by U.S. NSF grants CNS-1526547, CNS-1815322, CCF-1525629, and by the European Research Council (ERC) under the European Union Horizon 2020 research and innovative programme (grant agreement No 725731).

Part of the material in this paper was presented at IEEE International Symposium on Information Theory (ISIT), 2018 [1].

M. Aliasgari and J. Kliewer are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, New Jersey 07102-1982 USA (email: ma839@njit.edu; jkliewer@njit.edu)

O. Simeone is with the Department of Informatics, King's College London, London, UK (email: osvaldo.simeone@kcl.ac.uk)

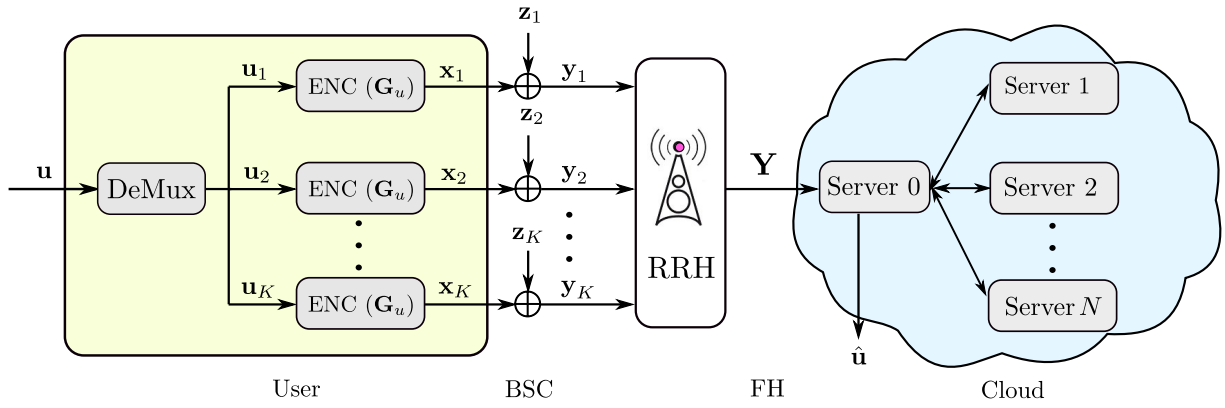


Fig. 1: NfV model for uplink channel decoding. The input information frame  $\mathbf{u}$  is divided into packets, which are encoded with a linear code  $\mathcal{C}_u$  with generator matrix  $\mathbf{G}_u$ . The packets are received by the RRH through a BSC and forwarded to the cloud. Server 0 in the cloud re-encodes the received packet with a linear code  $\mathcal{C}_c$  in order to enhance the robustness against potentially straggling Servers  $1, \dots, N$ .

which a processor in the cloud is either on or off. Unlike [36], in this work, we derive analytical performance bounds for a general scenario with any number of servers, random computing runtimes, and random packet arrivals. Specific novel contributions are as follows.

- We first consider the transmission of an isolated frame, and develop analytical upper bounds on the frame unavailability probability (FUP) as a function of the allowed decoding delay. The FUP measures the probability that a frame is correctly decoded within a tolerated delay constraint. The FUP bounds leverage large deviation results for correlated variables [37] and depend on the properties of both the uplink linear channel code adopted at the user and the NFV linear code applied at the cloud;
- As a byproduct of the analysis we introduce the dependency graph of a linear code and its chromatic number as novel relevant parameters of a linear code beside minimum distance, blocklength, and rate;
- We extend the analysis to account for random frame arrival times, and investigate the trade-off between average decoding latency and frame error rate (FER) for two different queuing policies, whereby the servers carry out either per-frame or continuous decoding;
- We provide extensive numerical results that demonstrate the usefulness of the derived analytical bounds in both predicting the system performance and enabling the design of NFV codes.

The rest of the paper is organized as follows. In Section II, we present the system model focusing, as in [36], on a binary symmetric channel (BSC) for uplink communications. Section III presents the two proposed upper bounds on the FUP as a function of latency. In Section IV we study the proposed system with random frame arrival times, and Section V provides numerical results.

## II. SYSTEM MODEL

As illustrated in Fig. 1, we consider the uplink of a C-RAN system in which a user communicates with the cloud via a remote radio head (RRH). The user is connected to the RRH via a BSC with cross error probability  $\delta$ , while the RRH-to-cloud link, typically referred to as fronthaul, is assumed to be noiseless. Note that the BSC is a simple model for the uplink

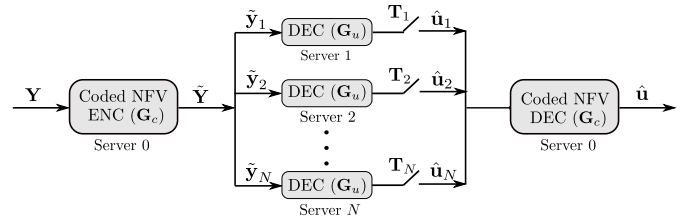


Fig. 2: Coded NFV at the cloud: Server 0 re-encodes the received packets in  $\mathbf{Y}$  by a linear NFV code  $\mathcal{C}_c$  with generator  $\mathbf{G}_c$ . Each encoded packet  $\tilde{\mathbf{y}}_i$  is then conveyed to Server  $i$  for decoding.

channel, while the noiseless fronthaul accounts for a typical deployment with higher capacity fiber optic cables. As we briefly discuss in Section VI, the analysis can be generalized to other additive noise channel, such as Gaussian channels.

The cloud contains a master server, or Server 0, and  $N$  slave servers, i.e., Servers  $1, \dots, N$ . The slave servers are characterized by random computing delays as in related works on coded computation [22], [23], [27]. Note that we use here the term “server” to refer to a decoding processor, although, in a practical implementation, this may correspond to a core of the cloud computing platform [14], [15].

In the first part of this paper, we consider transmission of a single information frame  $\mathbf{u}$ , while Section IV focuses on random frame arrival times and queuing effect delays. The user encodes an information frame  $\mathbf{u}$  consisting of  $L$  bits. Before encoding, the information frame is divided into  $K$  blocks  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K \in \{0, 1\}^{L/K}$  of equal size, each of them containing  $L/K$  bits. As shown in Fig. 1, in order to combat noise on the BSC, the  $L/K$  blocks are encoded by an  $(n, k)$  binary linear code  $\mathcal{C}_u$  of rate  $r = k/n$  defined by generator matrix  $\mathbf{G}_u \in \mathbb{F}_2^{n \times k}$ , where  $n = L/(rK)$  and  $k = L/K$ . Let  $\mathbf{x}_j \in \{0, 1\}^n$  with  $j \in \{1, \dots, K\}$  be the  $K$  transmitted packets of length  $n$ . At the output of the BSC, the length- $n$  received vector for the  $j$ th packet at the RRH is given as

$$\mathbf{y}_j = \mathbf{x}_j \oplus \mathbf{z}_j, \quad (1)$$

where  $\mathbf{z}_j$  is a vector of i.i.d. Bern( $\delta$ ) random variables (rvs). The  $K$  received packets  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$  by the RRH are transmitted to the cloud via the fronthaul link, and the cloud performs decoding. Specifically, as detailed next, we assume

that each Server  $1, \dots, N$  performs decoding of a single packet of length  $n$  bits while Server 0 acts as coordinator.

Assuming  $N \geq K$ , we adopt the idea of NFV coding proposed in [36]. Accordingly, as seen in Fig. 2, the  $K$  packets are first linearly encoded by Server 0 into  $N \geq K$  coded blocks of the same length  $n$  bits, each forwarded to a different server for decoding. This form of encoding is meant to mitigate the effect of straggling servers in a manner similar to [22], [23], [27]. Using an  $(N, K)$  binary linear NFV code  $\mathcal{C}_c$  with  $K \times N$  generator matrix  $\mathbf{G}_c \in \mathbb{F}_2^{N \times K}$ , the encoded packets are obtained as

$$\tilde{\mathbf{Y}} = \mathbf{Y}\mathbf{G}_c, \quad (2)$$

where  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_K]$  is the  $n \times K$  matrix obtained by including the received signal  $\mathbf{y}_j$  as the  $j$ th column and  $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_N]$  is the  $n \times N$  matrix whose  $i$ th column  $\tilde{\mathbf{y}}_i$  is the input to Server  $i$ , where  $i \in \{1, \dots, N\}$ . From (1), this vector can be written as

$$\tilde{\mathbf{y}}_i = \sum_{j=1}^K \mathbf{y}_j g_{c,ji} = \sum_{j=1}^K \mathbf{x}_j g_{c,ji} + \sum_{j=1}^K \mathbf{z}_j g_{c,ji}, \quad (3)$$

where  $g_{c,ji}$  is the  $(j, i)$ th entry of matrix  $\mathbf{G}_c$ .

The signal part  $\sum_{j=1}^K \mathbf{x}_j g_{c,ji}$  in (3) is a linear combination of  $d_i$  codewords for the rate- $r$  binary code with generator matrix  $\mathbf{G}_u$ , and hence it is a codeword of the same code. The parameter  $d_i$ ,  $i \in \{1, \dots, N\}$ , denotes the Hamming weight of the  $i$ th column of matrix  $\mathbf{G}_c$ , where  $0 \leq d_i \leq K$ . Each server  $i$  receives as input  $\tilde{\mathbf{y}}_i$  from which it can decode the codeword  $\sum_{i=1}^K \mathbf{x}_i g_{c,ji}$ . This decoding operation is affected by the noise vector  $\sum_{j=1}^K \mathbf{z}_j g_{c,ji}$  in (3), which has i.i.d. Bern( $\gamma_i$ ) elements. Here,  $\gamma_i$  is obtained as the first row and second column's entry of the matrix  $\mathbf{Q}^{d_i}$ , with  $\mathbf{Q}$  being the transition matrix of the BSC with cross over probability  $\delta$ , i.e.,

$$\mathbf{Q} = \begin{bmatrix} 1 - \delta & \delta \\ \delta & 1 - \delta \end{bmatrix}. \quad (4)$$

As an example,  $d_i = 2$ , implies a bit flipping probability of  $\gamma_i = 2\delta(1 - \delta)$ . Note that a larger value of  $d_i$  yields a larger bit probability  $\gamma_i$ . We define as  $P_{n,k}(\gamma_i)$  the decoding error probability of the  $(n, k)$  linear user code at Server  $i$ , which can be upper bounded by using [38, Theorem 33].

Server  $i$  requires a random time  $T_i = T_{1,i} + T_{2,i}$  to complete decoding, which is modeled as the sum of a component  $T_{1,i}$  that is independent of the workload and a component  $T_{2,i}$  that instead grows with the size  $n$  of the packet processed at each server, respectively. The first component accounts, e.g., for processor unavailability periods, while the second models the execution runtime from the start of the computation. The first variable  $T_{1,i}$  is assumed to have an exponential probability density function (pdf)  $f_1(t)$  with mean  $1/\mu_1$ , while the variable  $T_{2,i}$  has a shifted exponential distribution with cumulative distribution function (cdf) [39]

$$F_2(t) = 1 - \exp\left(-\frac{rK\mu_2}{L}\left(t - a\frac{L}{rK}\right)\right), \quad (5)$$

for  $t \geq aL/(rK)$  and  $F_2(t) = 0$  otherwise. The parameter  $a$  represents the minimum processing time per input bit, while  $1/\mu_2$  is the average additional time needed to process one bit. As argued in [22], [39], the shifted exponential model provides

a good fit for the distribution of computation times over cloud computing environments such as Amazon EC2 clusters. The cdf of the time  $T_i$  can hence be written as the integral  $F(t) = \int_0^t f_1(\tau)F_2(t - \tau)d\tau$ . We also assume that the runtime rvs  $\{T_i\}_{i=1}^N$  are mutually independent. Due to (5), the probability that a given set of  $l$  out of  $N$  servers has finished decoding by time  $t$  is given as

$$a_l(t) = \binom{N}{l} F(t)^l (1 - F(t))^{N-l}. \quad (6)$$

Let  $d_{\min}$  be the minimum distance of the NFV code  $\mathcal{C}_c$ . Due to (3), Server 0 in the cloud is able to decode the message  $\mathbf{u}$  or equivalently the  $K$  packets  $\mathbf{u}_j$  for  $j \in \{1, \dots, K\}$ , as soon as  $N - d_{\min} + 1$  servers have decoded successfully. Let  $\hat{\mathbf{u}}_i$  be the output of the  $i$ th server in the cloud upon decoding. We assume that an error detection mechanism, such as a cyclic redundancy check (CRC), is in place so that Server 0 outputs

$$\hat{\mathbf{u}}_i = \begin{cases} \mathbf{u}_i, & \text{for correct decoding,} \\ \emptyset, & \text{otherwise.} \end{cases}$$

The output  $\hat{\mathbf{u}}(t)$  of the decoder at Server 0 at time  $t$  is then a function of the vectors  $\hat{\mathbf{u}}_i(t)$  for  $i \in \{1, \dots, N\}$ , where

$$\hat{\mathbf{u}}_i(t) = \begin{cases} \mathbf{u}_i, & \text{if } T_i \leq t, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Finally, the frame unavailability probability (FUP) at time  $t$  is defined as the probability

$$P_u(t) = \Pr[\hat{\mathbf{u}}(t) \neq \mathbf{u}]. \quad (7)$$

The event  $\{\hat{\mathbf{u}}(t) \neq \mathbf{u}\}$  occurs when either not enough servers have completed decoding or many servers have completed but failed decoding by time  $t$ . We also define the FER as

$$P_e = \lim_{t \rightarrow \infty} P_u(t). \quad (8)$$

The FER measures the probability that, when all servers have completed decoding, a sufficiently large number, namely larger than  $N - d_{\min}$ , has decoded successfully.

### III. BOUNDS ON THE FRAME UNAVAILABILITY PROBABILITY

In this section we derive analytical bounds on the FUP  $P_u(t)$  in (7) as a function of the decoding latency  $t$ .

#### A. Preliminaries

Each server  $i$  with  $i \in \{1, \dots, N\}$  decodes successfully its assigned packet  $\tilde{\mathbf{y}}_i$  if: (i) the server completes decoding by time  $t$ ; (ii) the decoder at the server is able to correct the errors caused by the BSC. Furthermore as discussed, an error at Server 0 occurs at time  $t$  if the number of servers that have successfully decoded by time  $t$  is smaller than  $N - d_{\min} + 1$ .

To evaluate the FUP, we hence define the indicator variables  $C_i(t) = \mathbb{1}\{T_i \leq t\}$  and  $D_i$  which are equal to 1 if the events (i) and (ii) described above occur, respectively, and zero otherwise. Based on these definitions, the FUP is equal to

$$P_u(t) = \Pr\left[\sum_{i=1}^N C_i(t)D_i \leq N - d_{\min}\right]. \quad (9)$$

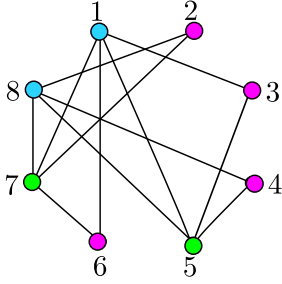


Fig. 3: Dependency graph associated with the (8,4) NRV code  $\mathcal{C}_c$  in Example 1.

The indicator variables  $C_i(t)$  are independent Bernoulli rvs across the servers  $i \in \{1, \dots, N\}$ , due to the independence assumption on the rvs  $T_i$ . However, the indicator variable  $D_i$  are dependent Bernoulli rvs. The dependence of the variables  $D_i$  is caused by the fact that the noise terms  $\sum_{i=1}^K \mathbf{z}_j g_{c,ji}$  in (3) generally have common terms. In particular, if two columns  $i$  and  $j$  of the generator matrix  $\mathbf{G}_c$  have at least a 1 in the same row, then the decoding indicators  $D_i$  and  $D_j$  are correlated. This complicates the evaluation of bounds on the FUP (9).

### B. Dependency Graph and Chromatic Number of a Linear Code

To capture the correlation among the indicator variables  $D_i$ , we introduce here the notion of the *dependency graph* and its chromatic number for a linear code. These appear to be novel properties of a linear code, and we will argue below that they determine the performance of the NRV code  $\mathcal{C}_c$  for the application at hand.

**Definition 1.** Let  $\mathbf{G} \in \mathbb{F}_2^{K' \times N'}$  be a generator matrix of a linear code. The dependency graph  $\mathcal{G}(\mathbf{G}) = (\mathcal{V}, \mathcal{E})$  comprises a set  $\mathcal{V}$  of  $N'$  vertices and a set  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  of edges, where edge  $(i, j) \in \mathcal{E}$  is included if both the  $i$ th and  $j$ th columns of  $\mathbf{G}$  have at least a 1 in the same row.

**Example 1.** For an (8,4) NRV code  $\mathcal{C}_c$  with the following generator matrix

$$\mathbf{G}_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad (10)$$

the resulting dependency graph  $\mathcal{G}(\mathbf{G}_c)$  is shown in Fig. 3.

The chromatic number  $\mathcal{X}(\mathbf{G})$  of the graph  $\mathcal{G}(\mathbf{G})$  will play an important role in the analysis. We recall that the chromatic number is the smallest number of colors needed to color the vertices of  $\mathcal{G}(\mathbf{G})$ , such that no two adjacent vertices share the same color (see the example in Fig. 3). Generally, finding the

chromatic number of a graph is NP-hard [40]. However, a simple upper bound on  $\mathcal{X}(\mathbf{G})$  is given as [41]

$$\mathcal{X}(\mathbf{G}) \leq \Delta(\mathbf{G}) + 1, \quad (12)$$

where  $\Delta(\mathbf{G})$  is the maximum degree of a graph  $\mathcal{G}(\mathbf{G})$ . A consequence of (12) is the following.

**Lemma 1.** Let  $\mathbf{G}$  be a  $K' \times N'$  matrix, where  $\alpha_r$  and  $\alpha_c$  are the maximum Hamming weights of the rows and columns in  $\mathbf{G}$ , respectively. Then the chromatic number of the corresponding dependency graph  $\mathcal{G}(\mathbf{G})$  is upper bounded as

$$\mathcal{X}(\mathbf{G}) \leq \min\{N, \alpha_c(\alpha_r - 1) + 1\}. \quad (13)$$

*Proof.* According to Definition 1 we have the upper bound  $\Delta(\mathbf{G}) \leq \alpha_c(\alpha_r - 1)$  and hence (13) follows directly from (12).  $\square$

### C. Large Deviation Upper Bound

In this subsection, we derive an upper bound on the FUP. The bound is based on the large deviation result in [37] for the tail probabilities of rvs  $X = \sum_{i=1}^M X_i$ , where the rvs  $X_i$  are generally dependent. We refer to this bound as the large deviation bound (LDB). The correlation of rvs  $\{X_i\}$  is described in [37] by a dependency graph. This is defined as any graph  $\mathcal{G}(X)$  with  $X_i$  as vertices, such that, if a vertex  $i \in \{1, \dots, M\} \setminus \{i\}$  is not connected to any vertex in a subset  $\mathcal{J} \subset \{1, \dots, M\}$ , then  $X_i$  is independent of  $\{X_j\}_{j \in \mathcal{J}}$ .

**Lemma 2** ([37]). Let  $X = \sum_{i=1}^M X_i$ , where  $X_i \sim \text{Bern}(p_i)$  and  $p_i \in (0, 1)$  are generally dependent. For any  $b \geq 0$ , such that the inequality  $X_i - \mathbb{E}(X_i) \geq -b$  holds for all  $i \in \{1, \dots, M\}$  with probability one, and for any  $\tau \geq 0$  we have

$$\Pr[X \leq \mathbb{E}(X) - \tau] \leq \exp\left(-\frac{S}{b^2 \mathcal{X}(\mathcal{G}(X))} \varphi\left(\frac{4b\tau}{5S}\right)\right), \quad (15)$$

where  $S \triangleq \sum_{i=1}^N \text{Var}(X_i)$  and  $\varphi(x) \triangleq (1+x) \ln(1+x) - x$ . The same bound (15) holds for  $\Pr(X \geq \mathbb{E}(X) + \tau)$ , where  $X_i - \mathbb{E}(X_i) \leq b$  with probability one.

The following theorem uses Lemma 2 to derive a bound on the FUP.

**Theorem 1.** Let  $P_{n,k}^{\min} = \min_i \{P_{n,k}(\gamma_i)\}_{i=1}^N$ . For all

$$t \geq F^{-1}\left(\frac{N - d_{\min}}{N - \sum_{i=1}^N P_{n,k}(\gamma_i)}\right), \quad (16)$$

the FUP is upper bounded by in (11), shown at the bottom of the page, where  $b(t) \triangleq F(t) \left(1 - P_{n,k}^{\min}\right)$  and  $S(t) \triangleq \sum_{i=1}^N F(t) (1 - P_{n,k}(\gamma_i)) (1 - F(t)(1 - P_{n,k}(\gamma_i)))$ . The upper bound (11) on the FUP captures the dependency of the

$$P_u(t) \leq \exp\left(-\frac{S(t)}{b^2(t) \mathcal{X}(\mathbf{G}_c)} \varphi\left(\frac{4b(t) \left(NF(t) - F(t) \sum_{i=1}^N P_{n,k}(\gamma_i) - N + d_{\min}\right)}{5S(t)}\right)\right), \quad (11)$$

FUP on both the channel and the NFV code. In particular, the bound is an increasing function of the error probabilities  $P_{n,k}(\gamma_i)$ , which depend on both codes. It also depends on the NFV code through parameters  $d_{\min}$  and  $\mathcal{X}(\mathbf{G}_c)$ .

*Proof.* Let  $X_i(t) \triangleq C_i(t)D_i$  and  $X(t) = \sum_{i=1}^N X_i(t)$ , where  $X_i(t)$  are dependent Bernoulli rvs with probability  $\mathbb{E}[X_i(t)] = \Pr[X_i(t) = 1] = F(t)(1 - P_{n,k}(\gamma_i))$ . It can be seen that a valid dependency graph  $\mathcal{G}(X)$  for the variables  $\{X_i\}$  is the dependency graph  $\mathcal{G}(\mathbf{G}_c)$  defined above. This is due to the fact that, as discussed in Section III-C, the rvs  $X_i$  and  $X_j$  are dependent if and only if the  $i$ th and  $j$ th column of  $\mathbf{G}_c$  have at least a 1 in a common row. We can hence apply Lemma 2 for every time  $t$  by selecting  $\tau = \mathbb{E}(X) - N + d_{\min}$ , and  $b(t)$  as defined above. Note that this choice of  $b(t)$  meets the constraint for  $b$  in Lemma 2. For  $1/\mu_1 = 0$ , (16) can be simplified as follows:

$$t \geq n \left( a - \frac{1}{\mu} \ln \left( \frac{d_{\min} - \sum_{i=1}^N P_{n,k}(\gamma_i)}{N - \sum_{i=1}^N P_{n,k}(\gamma_i)} \right) \right). \quad (17)$$

**Remark 1.** When  $t \rightarrow \infty$ , we have the limit  $\lim_{t \rightarrow \infty} F(t) = 1$ , which implies that eventually all servers complete decoding. Letting  $d^{\max} \triangleq \max\{d_i\}_{i=1}^N$  and  $\gamma \triangleq \mathbf{Q}^{d^{\max}}(1, 2)$ , the first row and second column's entry of the matrix  $\mathbf{Q}^{d^{\max}}$ , the bound (11) reduces to

$$\lim_{t \rightarrow \infty} P_u(t) \leq \exp \left( \frac{-NP_{n,k}(\gamma)}{(1 - P_{n,k}(\gamma))\mathcal{X}(\mathbf{G}_c)} \varphi \left( \frac{4(d_{\min}/N - P_{n,k}(\gamma))}{5P_{n,k}(\gamma)} \right) \right). \quad (18)$$

This expression demonstrates the dependence of the FUP bound (11) on the number of servers  $N$ , the decoding error probability  $P_{n,k}(\gamma)$  for each server, the chromatic number  $\mathcal{X}(\mathbf{G}_c)$ , and minimum distance  $d_{\min}$  of the NFV code. In particular, it can be seen that the FUP upper bound (18) is a decreasing function of  $d_{\min}$ , while it increases with the chromatic number,  $P_{n,k}(\gamma)$  and with  $d^{\max}$ .

#### D. Union Bound

As indicated in Theorem 1, the large deviation based bound in (14) is only valid for large enough  $t$ , as can be observed from (17). Furthermore, it may generally not be tight, since it neglects the independence of the indicator variables  $C_i$ . In this subsection, a generally tighter but more complex union bound (UB) is derived that is valid for all times  $t$ .

**Theorem 2.** For any subset  $\mathcal{A} \subseteq \{1, \dots, N\}$ , define

$$P_{n,k}^{\min(\mathcal{A})} \triangleq \min\{P_{n,k}(\gamma_i)\}_{i \in \mathcal{A}} \quad \text{and} \quad P_{n,k}^{\mathcal{A}} \triangleq \sum_{i \in \mathcal{A}} P_{n,k}(\gamma_i),$$

and let  $\mathbf{G}_{\mathcal{A}}$  be the  $K \times |\mathcal{A}|$ , submatrix of  $\mathbf{G}_c$ , with column indices in the subset  $\mathcal{A}$ . Then, the FUP is upper bounded

$$P_u(t) \leq 1 - \frac{1}{\binom{N}{l}} \sum_{l=N-d_{\min}+1}^N a_l(t) \sum_{\substack{\mathcal{A} \subseteq \{1, \dots, N\}: \\ |\mathcal{A}|=l}} \left( 1 - \exp \left( - \frac{S_{\mathcal{A}}}{b_{\mathcal{A}}^2 \mathcal{X}(\mathbf{G}_{\mathcal{A}})} \varphi \left( \frac{4b_{\mathcal{A}}(l - N + d_{\min} - P_{n,k}^{\mathcal{A}})}{5S_{\mathcal{A}}} \right) \right) \right). \quad (14)$$

by (14), shown at the bottom of the page, where  $S_{\mathcal{A}} \triangleq \sum_{i \in \mathcal{A}} P_{n,k}(\gamma_i)(1 - P_{n,k}(\gamma_i))$  and  $b_{\mathcal{A}} \triangleq 1 - P_{n,k}^{\min(\mathcal{A})}$ .

*Proof.* Let  $I_i = 1 - D_i$  be the indicator variable which equals 1 if Server  $i$  fails decoding. Accordingly, we have  $I_i \sim \text{Bern}(P_{n,k}(\gamma_i))$ . For each subset  $\mathcal{A} \subseteq \{1, \dots, N\}$ , let  $I_{\mathcal{A}} = \sum_{i \in \mathcal{A}} I_i$ . The complement of the FUP  $P_s(t) = 1 - P_u(t)$  can hence be written as

$$\begin{aligned} P_s(t) &= \Pr \left[ \sum_{i=1}^N C_i(t)D_i > N - d_{\min} \right] \\ &= \frac{1}{\binom{N}{l}} \sum_{l=N-d_{\min}+1}^N a_l(t) \sum_{\substack{\mathcal{A} \subseteq \{1, \dots, N\}: \\ |\mathcal{A}|=l}} \\ &\quad \cdot \sum_{j=N-d_{\min}+1}^l \Pr \left[ \begin{array}{l} j \text{ servers from } \mathcal{A} \text{ decode successfully} \\ \text{and} \\ l-j \text{ servers from } \mathcal{A} \text{ fail to decode} \end{array} \right] \\ &= \frac{1}{\binom{N}{l}} \sum_{l=N-d_{\min}+1}^N a_l(t) \sum_{\substack{\mathcal{A} \subseteq \{1, \dots, N\}: \\ |\mathcal{A}|=l}} (1 - \Pr[I_{\mathcal{A}} \geq l - N + d_{\min}]). \end{aligned} \quad (19)$$

We can now apply Lemma 2 to the probability in (21) by noting that  $\mathcal{G}(\mathbf{G}_{\mathcal{A}})$  is a valid dependency graph for the variables  $\{I_i\}$ ,  $i \in \mathcal{A}$ . In particular, we apply Lemma 2 by setting  $\tau_{\mathcal{A}} = l - N + d_{\min} - \mathbb{E}(I_{\mathcal{A}})$ ,  $b_{\mathcal{A}} \geq I_i - \mathbb{E}[I_i]$ , and  $S_{\mathcal{A}} = \sum_{i \in \mathcal{A}} \text{Var}(I_i)$ , leading to

$$\begin{aligned} &\Pr[I_{\mathcal{A}} \geq l - N + d_{\min}] \leq \\ &\exp \left( - \frac{S_{\mathcal{A}}}{b_{\mathcal{A}}^2 \mathcal{X}(\mathbf{G}_{\mathcal{A}})} \varphi \left( \frac{4b_{\mathcal{A}}(l - N + d_{\min} - P_{n,k}^{\mathcal{A}})}{5S_{\mathcal{A}}} \right) \right). \end{aligned} \quad (22)$$

By substituting (22) into (21), the proof is completed.  $\square$

## IV. RANDOM ARRIVALS AND QUEUING

In this section we extend our analysis from one to multiple frames transmitted by the users. To this end, we study the system illustrated in Fig. 4 with random frame arrival times and queuing at the servers. We specifically focus on the analysis of the trade-off between average latency and FER.

#### A. System Model

As illustrated in Fig. 4, we assume that the arrival times of the received frames are random and distributed according to a Poisson process with a rate of  $\lambda$  frames per second. Upon arrival, Server 0 applies an NFV code to any received frame  $\mathbf{y}^r$  for  $r = 1, 2, \dots$ , as described in Section II and sends each resulting coded packet  $\tilde{\mathbf{y}}_i^r$  to Server  $i$ , for  $i = 1, \dots, N$ . At Server  $i$ , each packet  $\tilde{\mathbf{y}}_i^r$  enters a first-come-first-serve



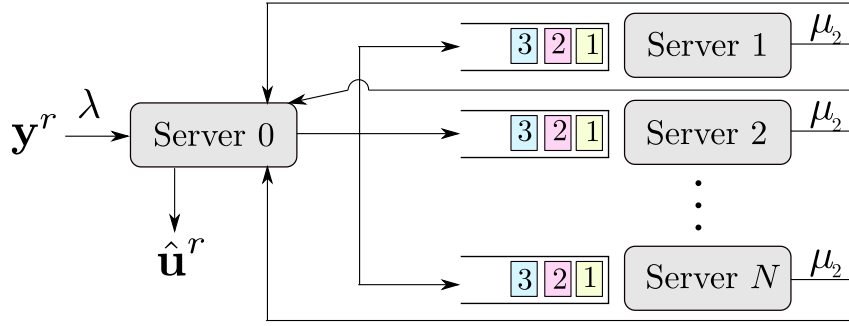


Fig. 4: In the model studied in Section IV, frames arrive at the receiver according to a Poisson process with parameter  $\lambda$ . Server 0 in the cloud encodes the received frames using an NFV code and forwards the encoded packets to servers  $1, \dots, N$  for decoding.

queue. After arriving at the head of the queue, each packet  $\tilde{\mathbf{y}}_i^r$  requires a random time  $T_i$  to be decoded by Server  $i$ . Here, we assume that  $T_i$  is distributed according to an exponential distribution in (5) with an average processing time of  $1/\mu_2$  per bit. Furthermore, the average time to process a frame of  $n$  bits is denoted as  $1/\mu$ . Also, the random variables  $T_i$  are i.i.d. across servers.

If the NFV code has minimum distance  $d_{\min}$ , as soon as  $N - d_{\min} + 1$  servers decode successfully their respective packets derived from frame  $\mathbf{y}^r$ , the information frame  $\mathbf{u}^r$  can be decoded at Server 0. We denote as  $T$  the average overall latency for decoding frame  $\mathbf{u}^r$ , which includes both queuing and processing.

Using (8), (9) and the fact that all servers complete decoding almost surely as  $t \rightarrow \infty$ , that is  $C_i(t) \rightarrow 1$  as  $t \rightarrow \infty$ , the FER is equal to

$$P_e = \Pr \left[ \sum_{i=1}^N I_i \geq d_{\min} \right], \quad (23)$$

where  $I_i$  is the indicator variable that equals 1 if decoding at Server  $i$  fails. This probability can be upper bounded by the following corollary of Theorem 1.

**Corollary 1.** *The FER defined in (23) is upper bounded by*

$$P_e \leq \exp \left( \frac{-S}{b^2 \mathcal{X}(\mathbf{G}_C)} \varphi \left( \frac{4b(d_{\min} - \sum_{i=1}^N P_{n,k}(\gamma_i))}{5S} \right) \right), \quad (24)$$

where  $S \triangleq \sum_{i=1}^N P_{n,k}(\gamma_i) (1 - P_{n,k}(\gamma_i))$  and  $b \triangleq 1 - P_{n,k}^{\min}$ .

*Proof.* The result follows from Theorem 1 by selecting  $\tau = d_{\min} - \sum_{i=1}^N P_{n,k}(\gamma_i)$ .  $\square$

We now discuss the computation of the average delay  $T$  for different queueing management policies.

### B. Per-Frame Decoding

We first study the system under a queue management policy whereby only one frame  $\mathbf{y}^r$  is decoded at any time. Therefore, all servers wait until at least  $N - d_{\min} + 1$  servers have completed decoding of their respective packets  $\tilde{\mathbf{y}}_i^r$  before moving to the next frame  $r+1$ , if this is currently available in the queues. Furthermore, as soon as Server 0 decodes a frame, the corresponding packets still being present in the servers' queues are evicted.

As a result, the overall system can be described an M/G/1 queue with arrival time  $\lambda$  and service time distributed according to the  $(N - d_{\min} + 1)$ th order statistic of the exponential distribution [42]. The latter has the pdf [43] (25), shown at the bottom of the page, where  $F_T(t)$  and  $f_T(t)$  are the cdf and pdf of rv  $T_i$ , respectively. This queueing system was also studied in the context of distributed storage systems.

Using the Pollaczek-Khinchin formula [44], the average delay of an M/G/1 queue can be obtained as (26), shown at the bottom of the page, where  $H_N$  and  $H_{N^2}$  are generalized harmonic numbers, defined by  $H_N = \sum_{i=1}^N \frac{1}{i}$  and  $H_{N^2} = \sum_{i=1}^N \frac{1}{i^2}$  [42]. Note that the queue is stable, and hence the average delay (26) is finite, if the inequality  $n\lambda(H_N - H_{d_{\min}-1}) < \mu(N - d_{\min} + 1)$  holds. We refer to the described queue management scheme as per-frame decoding (pfd). This set-up is equivalent to the fork-join system studied in [42].

### C. Continuous Decoding

As an alternative queue management policy, as soon as any Server  $i$  decodes its packet  $\tilde{\mathbf{y}}_i^r$ , it starts decoding the next packet  $\tilde{\mathbf{y}}_i^{r+1}$  in its queue, if this is currently available. Furthermore, as above, as soon as Server 0 decodes a frame  $\mathbf{y}^r$ , all corresponding packets  $\tilde{\mathbf{y}}_i^r$  still in the servers' queues are evicted. We refer this queue management policy as continuous decoding (cd).

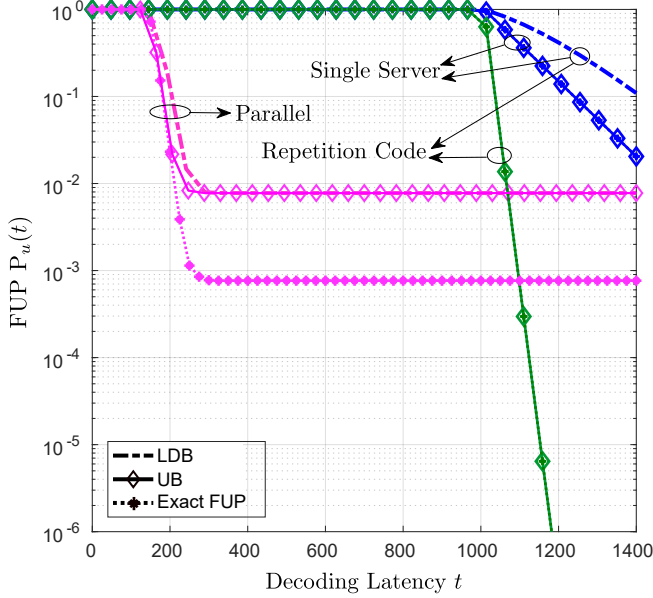
The average delay (26) of per-frame decoding is an upper bound for the average delay of continuous decoding, i.e., we have  $T_{\text{cd}} \leq T_{\text{pfd}}$  [42]. This is because, with per-frame decoding, all  $N$  servers are blocked until  $N - d_{\min} + 1$  servers decode their designed packets. We evaluate the performance of continuous decoding using Monte Carlo methods in the next section.

## V. SIMULATION RESULTS

In this section we provide numerical results to provide additional insights into the performance trade-off for the system shown in Fig. 1. We first consider individual frame transmission as studied in Section II and Section III, and then we study random arrivals as investigated in Section IV.

### A. Single Frame Transmission

We first consider single frame transmission. The main goals are to validate the usefulness of the two bounds presented in



(a) Parallel, single server and repetition code.

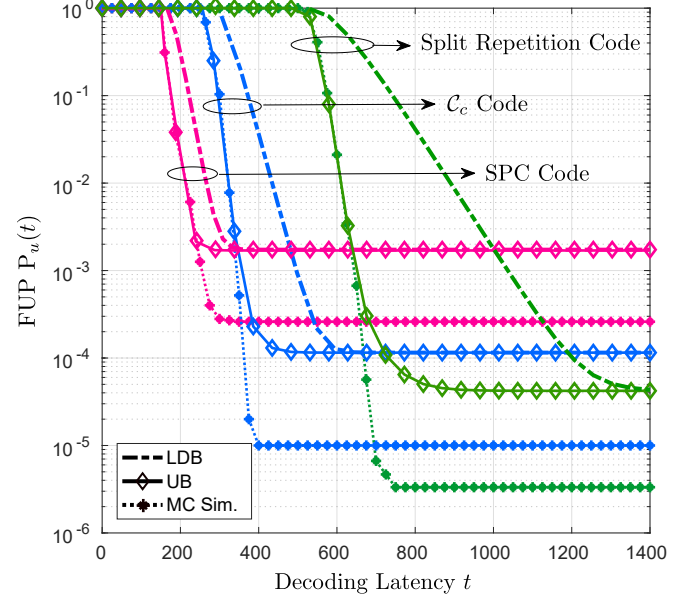
(b) Split repetition code, SPC code and  $\mathcal{C}_c$  code.

Fig. 5: Decoding latency versus FUP for  $L = 504$ ,  $N = 8$ ,  $1/\mu_1 = 0$ ,  $\mu_2 = 10$ ,  $a = 1$ ,  $\delta = 0.01$ ,  $r = 0.5$ : (a) LDB, UB and Exact FUP for the parallel, single-server, and repetition coding; (b) LDB, UB and Monte Carlo simulation (“MC Sim.”) results for split repetition code, SPC code, and the NFV code  $\mathcal{C}_c$  defined in (10).

Theorems 1 and 2 as design tools and to assess the importance of coding in obtaining desirable trade-offs between decoding latency and FUP. We employ a frame length of  $L = 504$  and  $N = 8$  servers. The user code  $\mathcal{C}_u$  is selected to be a randomly designed (3, 6) regular (Gallager-type) LDPC code with  $r = 0.5$ , which is decoded via belief propagation.

We compare the performance of the following solutions: (i) *Standard single-server decoding*, whereby we assume, as a benchmark, the use of a single server, that is  $N = 1$ , that decodes the entire frame ( $K = 1$ ); (ii) *Repetition coding*, whereby the entire frame ( $K = 1$ ) is replicated at all servers; (iii) *Parallel processing*, whereby the frame is divided into  $K = N$  disjoint parts processed by different servers; (iv) *Split repetition coding*, whereby the frame is split into two parts, which are each replicated at  $N/2$  servers. The code has hence  $K = 2$ ,  $d_{\min} = N/2$ ,  $\mathcal{X}(\mathbf{G}_c) = N/2$ , which can be thought of as an intermediate choice between repetition coding and the parallel scheme; (v) *Single parity check code (SPC)*, with  $N = K + 1$ , whereby, in addition to the servers used by parallel decoding, an additional server decodes the binary sum of all other  $K$  received packets; and (vi) an NFV code  $\mathcal{C}_c$  with the generator matrix  $\mathbf{G}_c$  defined in (10), which is characterized by  $K = 4$ . Note that, with both single-server decoding and

repetition coding, we have a blocklength of  $n = 1008$  for the channel code. Single-server decoding is trivially characterized by  $\mathcal{X}(\mathbf{G}_c) = d_{\min} = 1$ , while repetition coding is such that the equalities  $\mathcal{X}(\mathbf{G}_c) = d_{\min} = 8$  hold. Furthermore, the parallel approach is characterized by  $n = 126$ ,  $d_{\min} = 1$  and  $\mathcal{X}(\mathbf{G}_c) = 1$ ; the split repetition code is characterized by  $n = 504$ ,  $d_{\min} = 4$  and  $\mathcal{X}(\mathbf{G}_c) = 4$ ; the SPC code has  $n = 144$ ,  $d_{\min} = 2$  and  $\mathcal{X}(\mathbf{G}_c) = 2$ ; and the NFV code  $\mathcal{C}_c$  has  $n = 252$ ,  $d_{\min} = 3$  and  $\mathcal{X}(\mathbf{G}_c) = 3$ . The exact FUP for a given function  $P_{n,k}(\cdot)$  can easily be computed for cases (i)-(iii). In particular, for single server decoding, the FUP equals

$$P_u(t) = 1 - a_1(t)(1 - P_{L/r,L}(\delta)); \quad (27)$$

for the repetition code, the FUP is

$$P_u(t) = 1 - \sum_{i=1}^N a_i(t)(1 - P_{L/r,L}(\delta)); \quad (28)$$

and for the parallel approach, we have

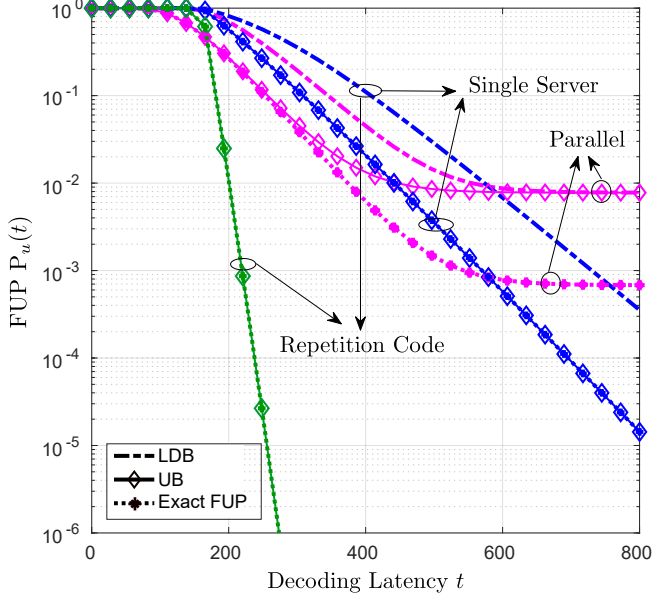
$$P_u(t) = 1 - a_N(t)(1 - P_{L/(rN),L/N}(\delta))^N. \quad (29)$$

In contrast, the exact FUPs for the SPC and code  $\mathcal{C}_c$  are difficult to compute, due to the discussed correlation among the decoding outcomes at the servers.

$$f_{T_{N-d_{\min}+1:N}}(t) = \frac{N!}{(N-d_{\min})!(d_{\min}-l)!} f_T(t) F_T(t)^{N-d_{\min}} (1-F_T(t))^{d_{\min}-1}, \quad (25)$$

$$T_{\text{pdf}} = \frac{n(H_N - H_{d_{\min}-1})}{(N - d_{\min} + 1)\mu} + \frac{\lambda n^2 [(H_N - H_{d_{\min}-1})^2 + (H_{N^2} - H_{(d_{\min}-1)^2})]}{2(N - d_{\min} + 1)^2 \mu^2 [1 - \lambda n \mu^{-1} (N - d_{\min} + 1)^{-1} (H_N - H_{d_{\min}-1})]}, \quad (26)$$





(a) Parallel, single server and repetition code.

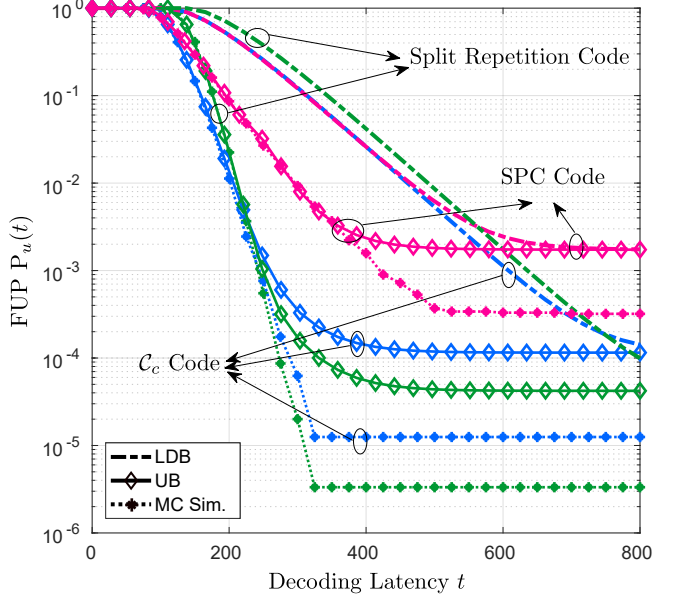
(b) Split repetition code, SPC code and  $C_c$  code.

Fig. 6: Decoding latency versus FUP for ( $L = 504, N = 8, 1/\mu_1 = 50, \mu_2 = 20, a = 0.1, \delta = 0.01, r = 0.5$ ): (a) LDB, UB and Exact FUP for the parallel, single-server, and repetition coding; (b) LDB, UB and Monte Carlo simulation (“MC Sim.”) results for split repetition code, SPC code, and the NfV code  $C_c$  defined in (10).

Fig. 5a shows decoding latency versus FUP for the LDB in Theorem 1, the UB in Theorem 2, and the exact error (27), (28), (29), for the first three schemes (i)-(iii), and Fig. 5b shows the LDB in Theorem 1, the UB in Theorem 2, as well as Monte Carlo simulation results for schemes (iv), (v), and (vi). Here, we assume that the latency contribution that is independent of the workload, is negligible, i.e.,  $1/\mu_1 = 0$ . We also set  $a = 1$  and  $\mu_2 = 10$ . As a first observation, Fig. 5 confirms that the UB bound is tighter than the LDB.

Leveraging multiple servers in parallel for decoding is seen to yield significant gains in terms of the trade-off between latency and FUP as argued also in [14] by using experimental results. In particular, the parallel scheme is observed to be preferred for lower latencies. This is due to the shorter blocklength  $n$ , which entails a smaller average decoding latency. However, the error floor of the parallel scheme is large due to the higher error probability for short blocklengths. In this case, other forms of NfV coding are beneficial. To elaborate, repetition coding requires a larger latency in order to obtain acceptable FUP performance owing to the larger blocklength  $n$ , but it achieves a significantly lower error floor. For intermediate latencies, the SPC code, and at larger latencies also both the NfV code  $C_c$ , and the split repetition code provide a lower FUP. This demonstrates the effectiveness of NfV encoding in obtaining a desirable trade-off between latency and FUP.

In order to validate the conclusion obtained using the bounds, Fig. 5 also shows the exact FUP for the schemes (i)-(iii), as well as Monte Carlo simulation results for schemes (iv)-(vi), respectively. While the absolute numerical values of the bounds in Fig. 5a and 5b are not uniformly tight with respect to the actual performance, the relative performance of

the coding schemes are well matched by the analytical bounds. This provides evidence of the usefulness of the derived bounds as a tool for code design in NfV systems.

Fig. 6 is obtained in the same way as Fig. 5, except for the parameters  $\mu_1 = 0.02$ ,  $\mu_2 = 20$ , and  $a = 0.1$ . Unlike Fig. 5, here latency may be dominated by effects that are independent of the blocklength  $n$  since we have  $1/\mu_1 > 0$ . The key difference with respect to Fig. 5 is that, for this choice of parameters, repetition coding tends to outperform both the parallel case, and the NfV code  $C_c$ , apart from very small latencies. This is because repetition coding has the maximum resilience to the unavailability of the servers, while not being excessively penalized by the larger blocklength  $n$ . This is not the case, however, for very small latency levels, where the NfV code  $C_c$  provides the smallest FUP given its shorter blocklength as compared to repetition coding and its larger  $d_{\min}$ , with respect to the parallel scheme.

Fig. 7 shows the exact FUP for the extreme cases of parallel and repetition coding for different number of servers  $N \in \{3, 6, 12\}$ . The figure confirms that, for both schemes, the latency decreases for a larger number of servers  $N$ . However, by increasing  $N$ , the error floor of the parallel scheme grows due to the higher channel error probability for shorter blocklengths.

## B. Random Frame Transmission

We now consider the queueing system described in Section IV, and present numerical results that provide insights into the performance of both per-frame and continuous decoding in terms of FER versus average latency (23). As above, the decoding error probability is upper bounded by using [38,

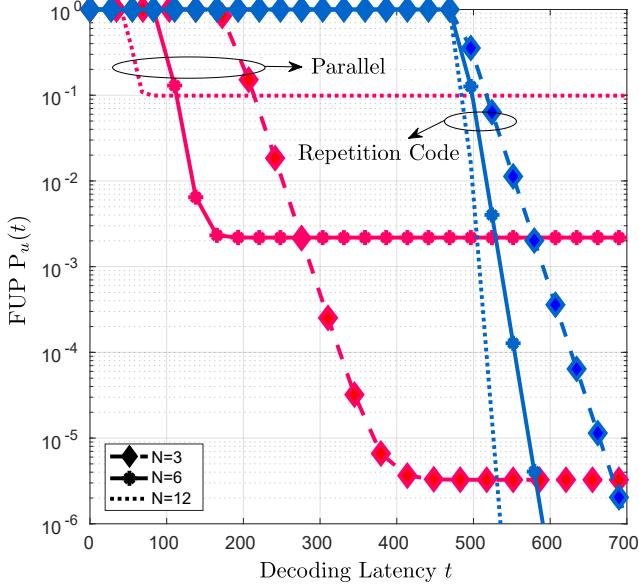


Fig. 7: Decoding latency versus exact FUP for parallel and repetition coding for different number of servers  $N \in \{3, 6, 12\}$  and ( $L = 240, 1/\mu_1 = 0, \mu_2 = 10, a = 1, \delta = 0.03, r = 0.5$ )

Theorem 33]. Both FER and average latency are a function of the user code rate  $r$ . We hence vary  $r \in \{1/2, \dots, 1/5\}$  to parametrize a trade-off curve between FER and latency. We assume a frame length of  $L = 112$  bits with  $N = 8$  servers, and adopt the same user code  $\mathcal{C}_c$  as in the previous subsection. The average delay  $T_{\text{pdf}}$  is computed from (26), and  $T_{\text{cd}}$  is obtained via Monte Carlo simulations.

Figs. 8a and 8b compare the performance of repetition coding, the NFV code  $\mathcal{C}_c$  with the generator matrix (10), and the parallel approach as defined above. Fig. 8a considers a lightly loaded system with  $\lambda = 0.1$  frames per second and  $\mu = 500$  frames per second, while Fig. 8b shows a highly loaded system with both  $\lambda = 1$  frames per second and  $\mu = 50$  frames per second.

First, by comparing the two figures we observe that per-frame decoding and continuous decoding have a similar performance when the system is lightly loaded (see Fig. 8a), while continuous decoding yields a smaller average latency than per-frame decoding when the system is heavily loaded (see Fig. 8b). This is because, in the former case, it is likely that a frame is decoded successfully before the next one arrives. This is in contrast to heavily loaded systems in which the average latency becomes dominated by queuing delays. We also note that, for repetition coding, the performance of per-frame decoding and continuous decoding coincides in both lightly or heavily loaded systems, since decoding is complete as soon as one server decodes successfully.

Also, by comparing the performance of different codes, we recover some of the main insights obtained from the study of the isolated frame transmission. In particular, the parallel approach outperforms all other schemes for low average delays

due to its shorter block length  $n$ . In contrast, repetition coding outperforms all other schemes in FER for large average delay because of its large block length  $n$  and consequently low probability of decoding error (not shown). Furthermore, we observe that split repetition coding is to be preferred for small values of FER.

Finally, Fig. 9 demonstrates the behavior of the average latency as the arrival rate  $\lambda$  increases and the system becomes more heavily loaded. We observe that, for a lightly loaded system, the latencies of per frame and continuous decoding are similar, while continuous decoding is preferable for a large number of  $\lambda$ . This is because per-frame decoding requires all servers to wait until at least  $N - d_{\text{min}} + 1$  servers have completed decoding of their respective packets before moving on to the next frame.

## VI. CONCLUSIONS

In this paper, we analyzed the performance of a novel coded NFV approach for the uplink of a C-RAN system in which decoding takes place at a multi-server cloud processor. The approach is based on the linear combination of the received packets prior to their distribution to the servers or cores, and on the exploitation of the algebraic properties of linear channel codes. The method can be thought of as an application of the emerging principle of coded computing to NFV. In addition, we obtain novel upper bounds on the FUP as a function of the decoding latency based on evaluating tail probabilities for Bernoulli dependent rvs. By extending the analysis from isolated frame transmission to random frame arrival times, the trade-off between average decoding latency and FER for two different policies are derived. Analysis and simulation results demonstrate the benefits that linear coding of received packets, or NFV coding, can yield in terms of trade-off between decoding latency and reliability. In particular, a prescribed decoding latency or reliability can be obtained by selecting an NFV code with a specific minimum distance and chromatic number, where the two extremes are parallel NFV-based processing and repetition coding. The former scheme obtains the smallest latency but the lowest reliability, whereas the latter scheme yields the largest latency, but the highest reliability. All other linear NFV codes operate between these two extreme cases.

Among interesting open problems, we mention the design of optimal NFV codes and the extension of the principle of NFV coding to other channels. Note that the approach proposed here applies directly to other additive noise channels in which the user code is an additive group. A key example is the additive Gaussian channel with lattice codes at the user, which will be studied in future work.

## REFERENCES

- [1] M. Aliasgari, J. Kliewer, and O. Simeone, "Coded computation against straggling decoders for network function virtualization," *Proc. IEEE International Symposium on Information Theory*, pp. 711–715, Jun., 2018.
- [2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

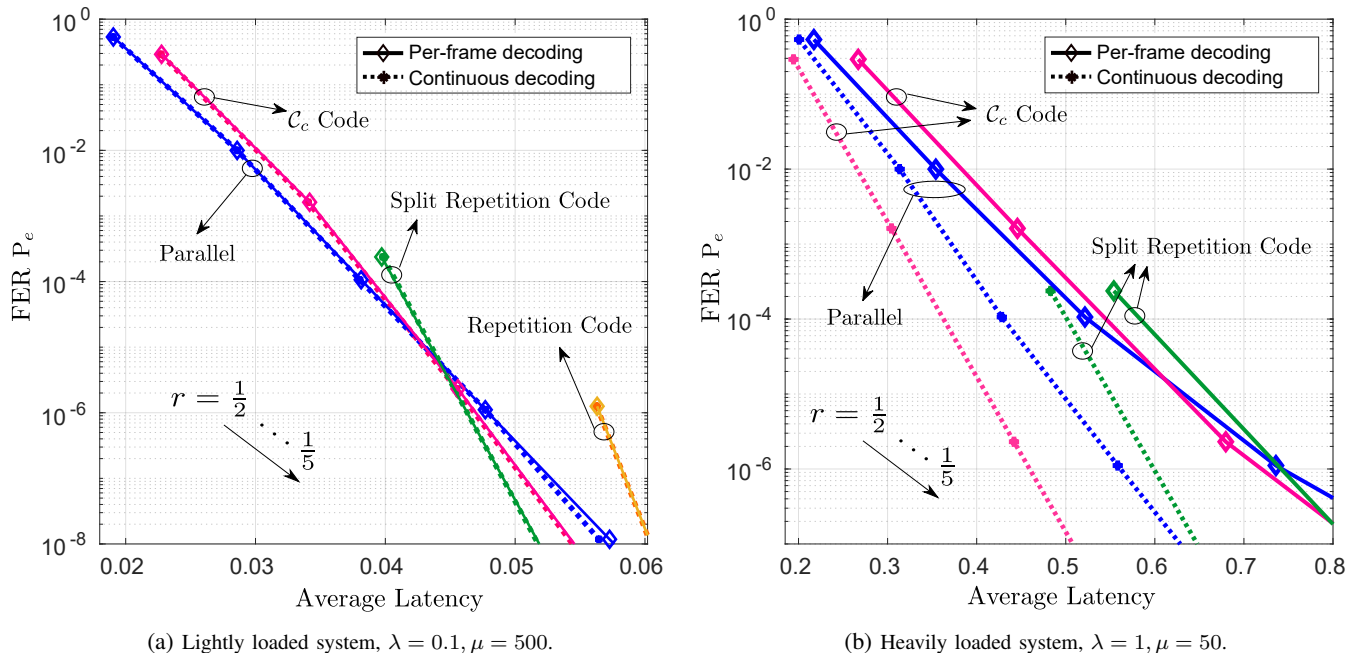


Fig. 8: Average latency versus FER with different values of the user code rate  $r$  and for different coding schemes when the system is (a) lightly loaded and (b) heavily loaded, respectively ( $L = 112, N = 8, \delta = 0.03$ ).

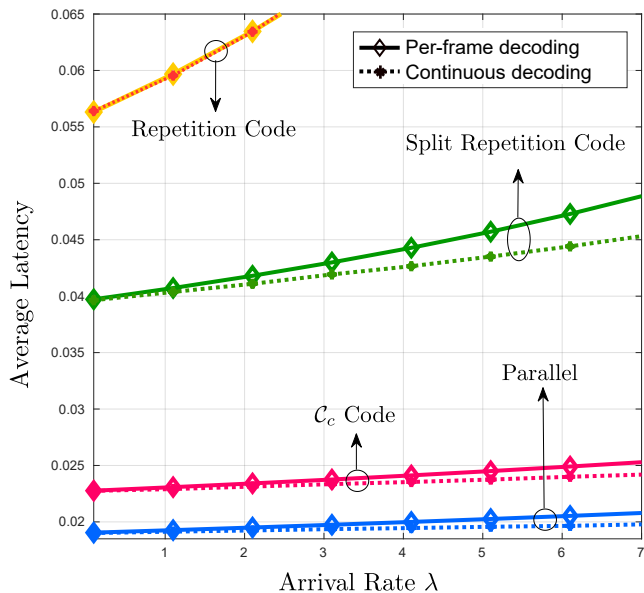


Fig. 9: Average latency versus arrival rate  $\lambda$  ( $L = 112, N = 8, r = 0.5, \mu = 500$ ).

- [3] European Telecommunications Standards Institute, "Network function virtualisation (NFV); report on models and features for end-to-end reliability," Technical Report GS NFV-REL 003, Apr., 2016.
- [4] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," *IEEE Wireless Communications*, vol. 23, no. 3, pp. 90–96, 2016.
- [5] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.
- [6] J. Kang, O. Simeone, and J. Kang, "On the trade-off between computational load and reliability for network function virtualization," *IEEE Communications Letters*, vol. 21, pp. 1767–1770, 2017.

- [7] N. Nikaiein, "Processing radio access network functions in the cloud: Critical issues and modeling," in *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, ACM, Apr., 2015, pp. 36–43.
- [8] European Telecommunications Standards Institute, "Cloud RAN and MEC: A perfect pairing," ISBN No. 979-10-92620-17-7, Feb., 2018.
- [9] I. Alyafawi, E. Schiller, T. Braun, D. Dimitrova, A. Gomes, and N. Nikaiein, "Critical issues of centralized and cloudified LTE-FDD radio access networks," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, Jun., 2015, pp. 5523–5528.
- [10] N. Nikaiein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, "OpenAirInterface: an open LTE network in a PC," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, Sep., 2014, pp. 305–308.
- [11] U. Dötsch, M. Doll, H.-P. Mayer, F. Schaich, J. Segel, and P. Sehier, "Quantitative analysis of split base station processing and determination of advantageous architectures for LTE," *Bell Labs Technical Journal*, vol. 18, no. 1, pp. 105–128, 2013.
- [12] P. Rost and A. Prasad, "Opportunistic hybrid architecture of centralized-RAN over nonideal backhaul," *IEEE Wireless Communications Letters*, vol. 3, no. 5, pp. 481–484, 2014.
- [13] S. Khalili and O. Simeone, "Uplink HARQ for cloud RAN via separation of control and data planes," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4005–4016, 2017.
- [14] V. Q. Rodriguez and F. Guillemin, "Towards the deployment of a fully centralized cloud-RAN architecture," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, Valencia, Spain, Jun., 2017, pp. 1055–1060.
- [15] —, "Cloud-ran modeling based on parallel processing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 457–468, 2018.
- [16] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [17] G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in Map-Reduce clusters using mantri," in *OSDI*, vol. 10, no. 1, Oct., 2010, p. 24.
- [18] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *Proceeding of the 2nd USENIX conference on Hot topics in cloud computing*, pp. 10–10, 2010.
- [19] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded distributed computing: Straggling servers and multistage dataflows," in *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*. IEEE, Oct., 2016, pp. 164–171.

- [20] —, “Coded MapReduce,” in *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE, Oct., 2015, pp. 964–971.
- [21] —, “A unified coding framework for distributed computing with straggling servers,” in *Globecom Workshops (GC Wkshps), 2016 IEEE*. IEEE, Dec., 2016, pp. 1–6.
- [22] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” *Proc. IEEE International Symposium on Information Theory*, pp. 1143–1147, Jul., 2016.
- [23] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.
- [24] Y. Yang, P. Grover, and S. Kar, “Computing linear transformations with unreliable components,” *IEEE Transactions on Information Theory*, 2017.
- [25] R. Tandon, Q. Lei, A. Dimakis, and N. Karampatziakis, “Gradient coding: Avoiding stragglers in synchronous gradient descent,” [Online] [www.arxiv.org](http://www.arxiv.org) arXiv:1612.03301 [cs.IT], 2016.
- [26] S. Dutta, V. Cadambe, and P. Grover, “Short-dot: Computing large linear transforms distributedly using coded short dot products,” *Advances In Neural Information Processing Systems*, pp. 2092–2100, 2016.
- [27] A. Severinson, A. Graell i Amat, and E. Rosnes, “Block-diagonal coding for distributed computing with straggling servers,” in *Information Theory Workshop (ITW)*, Nov., 2017, pp. 464–468.
- [28] Q. Yu, M. Maddah-Ali, and S. Avestimehr, “Polynomial codes: An optimal design for high-dimensional coded matrix multiplication,” in *Advances in Neural Information Processing Systems*, 2017, pp. 4403–4413.
- [29] A. Mallick, M. Chaudhari, and G. Joshi, “Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication,” *arXiv preprint arXiv:1804.10331*, 2018.
- [30] J. Kosaian, K. Rashmi, and S. Venkataraman, “Learning a code: Machine learning for approximate non-linear coded computation,” *arXiv preprint arXiv:1806.01259*, 2018.
- [31] D. Wang, G. Joshi, and G. Wornell, “Using straggler replication to reduce latency in large-scale parallel computing,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 3, pp. 7–11, 2015.
- [32] G. Joshi, E. Soljanin, and G. Wornell, “Efficient redundancy techniques for latency reduction in cloud systems,” *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 2, no. 2, p. 12, 2017.
- [33] G. Anantharayanan, A. Ghodsi, S. Shenker, and I. Stoica, “Effective straggler mitigation: Attack of the clones,” in *NSDI*, vol. 13, Apr., 2013, pp. 185–198.
- [34] Y. Yang, M. Chaudhari, P. Grover, and S. Kar, “Coded iterative computing using substitute decoding,” *arXiv preprint arXiv:1805.06046*, 2018.
- [35] M. F. Aktas, P. Peng, and E. Soljanin, “Effective straggler mitigation: Which clones should attack and when?” *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, no. 2, pp. 12–14, 2017.
- [36] A. Al-Shuwaili, O. Simeone, J. Kliewer, and P. Popovski, “Coded network function virtualization: Fault tolerance via in-network coding,” *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 644–647, 2016.
- [37] S. Janson, “Large deviations for sums of partly dependent random variables,” *Random Structures & Algorithms*, vol. 24, no. 3, pp. 234–248, 2004.
- [38] Y. Polyanskiy, H. V. Poor, and S. Verdú, “Channel coding rate in the finite blocklength regime,” *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.
- [39] A. Reiszadehmobarakeh, S. Prakash, R. Pedarsani, and S. Avestimehr, “Coded computation over heterogeneous clusters,” [Online] [www.arxiv.org](http://www.arxiv.org), arXiv:1701.05973 [cs.IT], 2017.
- [40] A. Sánchez-Arroyo, “Determining the total colouring number is NP-hard,” *Discrete Mathematics*, vol. 78, no. 3, pp. 315–319, 1989.
- [41] R. L. Brooks, “On colouring the nodes of a network,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 37, no. 02, pp. 194–197, 1941.
- [42] G. Joshi, Y. Liu, and E. Soljanin, “On the delay-storage trade-off in content download from coded distributed storage systems,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 989–997, 2014.
- [43] S. M. Ross, *Introduction to Probability Models*. Academic Press, 2014.
- [44] H. C. Tijms, *A First Course in Stochastic Models*. John Wiley and Sons, 2003.