Caching at the Edge with LT Codes

Estefanía Recayte, Francisco Lázaro, Gianluigi Liva
*Institute of Communications and Navigation of DLR (German Aerospace Center),
Wessling, Germany. Email: {Estefania.Recayte, Francisco.LazaroBlasco,Gianluigi.Liva}@dlr.de

***Abstract*—We study the performance of caching schemes based on LT under peeling (iterative) decoding algorithm. We assume that users ask for downloading content to multiple cache-aided transmitters. Transmitters are connected through a backhaul link to a master node while no direct link exists between users and the master node. Each content is fragmented and coded with LT code. Cache placement at each transmitter is optimized such that transmissions over the backhaul link is minimized. We derive a closed form expression for the calculation of the backhaul transmission rate. We compare the performance of a caching scheme based on LT with respect to a caching scheme based on maximum distance separable codes. Finally, we show that caching with LT codes behave as good as caching with maximum distance separable codes.**

I. Introduction

Caching multimedia contents at the network edge is a promising solution in order to mitigate traffic congestion in the backhaul link. Numerous works have investigated the potential benefits of caching schemes based on coded content applied in heterogeneous networks [1]–[5]. In literature, caching schemes based on maximum distance separable (MDS) codes have been studied for future wireless networks for maximizing the energy efficiency, minimizing the expected downloading time [3] or reducing the amount of data that has to be sent from the core of the network [4]. The technique of storing coded fragments of the files significantly outperforms the uncoded approach. Although MDS codes represent a benchmark for coding and caching schemes, their practical application is limited by the use of moderate field size due to the high encoding and decoding complexity. A further disadvantage of MDS codes is that their rate is fixed before the encoding takes place.

In cache-aided networks a feasible substitute of optimal codes might be *rateless* codes. Rateless codes are characterized by the fact that the encoder can generate an unlimited amount of coded symbols. In [6] it was shown that linear random fountain code (LRFC) caching schemes can perform very close to a system based on MDS codes.

In this paper, we analyse the performance of a caching scheme based on LT codes. In particular, our scheme considers LT codes under peeling (interative) decoding, which are characterized by a low decoding complexity. We derive a simple closed form expression for the calculation of the average backhaul transmission rate. The backhaul transmission rate is defined as the number of extra output symbols sent by the master node (mNode) to a transmitter to serve a user request. We show that caching at the edge with low complexity coding schemes does not significantly penalize the performance in terms of backhaul transmission rate.

The remainder of this paper is organized as follows. The system model considered is introduced in Section II. In Section III the caching scheme based on LT codes is presented. The achievable backhaul rate is derived in Section IV while the LT placement optimization problem is formulated in Section V. In Section VI the numerical results are presented. At last, conclusions are drawn in Section VII.

This work has been accepted for publication at IEEE ISTC 2018.

II. System Model

Let us consider the cache-aided network depicted in Fig. 1 where users are interested in downloading files from a mNode. Transmitters equipped with a local cache are deployed in the area in order to serve users requests. We assume that each transmitter is connected to the mNode through a wireless backhaul link while no direct link exists between users and the mNode.

Users send request to download files belonging to a library of n of equal size files $\mathcal{F} = \{\mathsf{f}_1 \ldots, \mathsf{f}_n\}$. A file f_j is requested with probability θ_j, following a Zipf distribution with shape parameter α such that

$$\theta_j = \frac{1/j^{\alpha}}{\sum_{i=1}^{n} 1/i^{\alpha}} \quad j = 1, ..., n.$$

Each transmitter can store in its local cache M files. A user can be served by the cached content of multiple transmitters according to his own location. We denote by γ_h the probability that a user is connected to h transmitters.

We consider a caching scheme based on LT codes. The mNode, who has access to the whole library $\mathcal{F}$, encodes each file using an LT code. In particular, each file is fragmented into k input symbols and an LT code is used to generate output (coded) symbols. During the placement phase, each transmitter stores the same number w_j of output coded symbols for file f_j. Due to the LT construction, each output symbol is generated independently from all other output symbols. During the delivery phase, users send request for contents belonging to $\mathcal{F}$. A user initially receives coded content of the requested file directly from the caches of the transmitters. If the user cannot decode the file requested from the cached content then the mNode sends supplementary output symbols until the decoding process is declared successful. Extra output symbols are sent via the backhaul link from the mNode to one of the transmitters which is serving the users and finally that transmitter forwards the symbols to the user. For simplicity we assume that all transmissions are error-free.

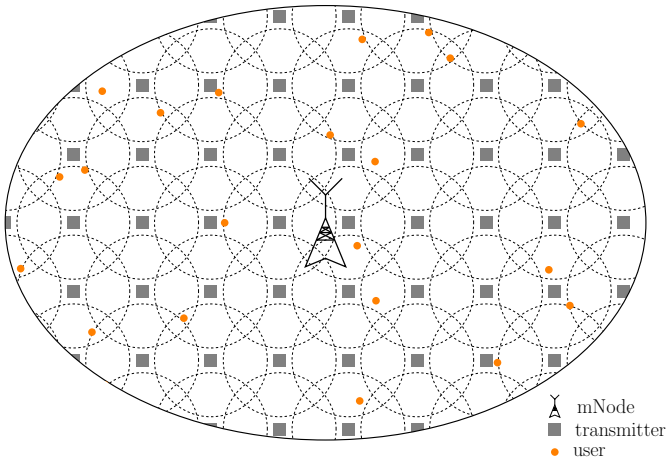

Fig. 1. System model

III. LT Codes

We consider the use of binary LT codes [7] under peeling (iterative) decoding for the delivery of the different files in the library. Each file is fragmented into k input symbols, $(u_1, u_2, \ldots, u_k)$, which are fed into the LT encoder. The LT encoder generates output symbols $\mathbf{c} = (c_1, c_2, \ldots, c_\ell)$, where the number of outputs symbols ℓ can grow indefinitely. Each output symbol c_i is constructed by first sampling a degree d from a degree distribution $\Omega = (\Omega_1, \Omega_2, \Omega_3, \ldots, \Omega_{d_{\text{max}}})$, where d_{max} is the maximum output degree. Next, d distinct symbols are selected uniformly at random among the k source symbols, and their x-or is computed to generate the output symbol.

For fixed ℓ, LT encoding can be expressed as a vector matrix multiplication

$$\mathbf{c} = \mathbf{u}\mathbf{G}$$

where $\mathbf{u}$ is the vector of input symbols and $\mathbf{G}$ is the generator matrix of the LT code, being each column of $\mathbf{G}$ associated to an output symbol.

In order to download a file, a user must collect a set of $m \geq k$ output symbols $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. If we denote by $\mathcal{I} = (i_1, i_2, \ldots, i_m)$ the set of indices corresponding to the m output symbols collected by the receiver we have

$$y_r = c_{i_r}.$$

The user attempts decoding by solving the system of equations

$$\mathbf{y} = \mathbf{u}\tilde{\mathbf{G}}.$$

where $\tilde{\mathbf{G}}$ is the matrix obtained by taking the columns of $\mathbf{G}$ with indices in $\mathcal{I}$. In this paper we shall assume that the peeling decoder proposed in [7] is used, which is suboptimal but shows a good performance if k is large and the degree distribution is suitably chosen. The peeling decoder is better explained based on a bipartite representation of the LT code. Decoding consists of initially marking all input symbols as unresolved and then carrying out k decoding stages. At every decoding stage the decoder looks for a degree one output symbol. If a degree one output symbol y is found, its only neighbor u is resolved, and all edges attached to u are erased from the bipartite graph, which reduces the degree of the neighbors of u. If no degree one output symbol is found a decoding failure is declared. After k decoding stages without a failure all k input symbols are recovered and decoding succeeds.

The performance of the peeling decoder was analyzed in [8], [9] using dynamic programming. In the following we shall make use of the notation used in [10] to describe the peeling decoder. Let us define the output ripple (or simply ripple) as the set of output symbols of degree 1, denoted $\mathscr{R}$, and let us define the cloud as the set of output symbols of degree $d \geq 2$, denoted $\mathscr{C}$. Furthermore, we shall denote the random variables associated to the cardinality of the cloud and ripple by and $\mathtt{C}$ and $\mathtt{R}$, respectively, and by $\mathtt{c}$ and $\mathtt{r}$ their realizations. Moreover, we shall use the subscript u to represent the number of unresolved input symbols, so that $\mathtt{C}_u$, for example, represents the cardinality of the cloud when u input symbols are unresolved. In [8], [9] the LT decoder is modelled as a finite state machine. In particular, given the absolute receiver overhead $\delta = m - k$, the decoder is modelled as a finite state machine with state

$$\mathtt{S}_u := (\mathtt{C}_u, \mathtt{R}_u).$$

Based on this model, a recursion is derived that allows to obtain $\Pr\{\mathtt{S}_{u-1} = (\mathtt{c}_{u-1}, \mathtt{r}_{u-1})\}$ as a function of $\Pr\{\mathtt{S}_u = (\mathtt{c}_u, \mathtt{r}_u)\}$. This recursion allows deriving the exact probability of the decoder being at state $\mathtt{S}_u := (\mathtt{c}_u, \mathtt{r}_u)$ for $u = k, k-1, \ldots, 1$. By observing that decoding fails whenever the ripple is empty ($\mathtt{r}_u = 0$), the probability of decoding failure can be obtained as

$$P_F(\delta) = \sum_{u=1}^{k} \sum_{\mathtt{c}_u} \Pr\{\mathtt{S}_u = (\mathtt{c}_u, 0)\}$$

where we observe that $P_F(\delta) = 1$ for $\delta < 0$.

IV. Average Backhaul Rate

A. Overhead Average

Let Δ be the random variable associated to the number of LT output symbols in excess to k that a user needs to successfully decode the requested content. We denote with δ a realization of Δ. We can calculate the expected value of Δ, i.e. the average overhead, as follows

$$\begin{aligned} \mathbb{E}[\Delta] &= \sum_{\delta=1}^{\infty} \delta \cdot \Big[P_F(\delta - 1) - P_F(\delta)\Big] \qquad (1) \\ &= \sum_{\delta=0}^{\infty} (\delta + 1) \cdot P_F(\delta) - \sum_{\delta=0}^{\infty} \delta \cdot P_F(\delta) \\ &= \sum_{\delta=0}^{\infty} P_F(\delta). \end{aligned}$$

B. Backhaul Transmission Rate

The average backhaul transmission rate is calculated as proposed in [6].

The total number of LT cached symbols of the file requested that a user receives from the transmitters is modelled by the random variable (r.v.) Z, denote by z the realization of Z. Let also denote by H the r.v. associated to the number

of transmitters connected to the user of interest, and by h its realization. The index of the file requested by a user is modelled by the r.v. J and with j we indicate its realization. We have

$$P_{Z|J,H}(z|j,h) = \begin{cases} 1 & \text{if } z = w_j\, h \\ 0 & \text{otherwise} \end{cases}$$

where we recall that w_j stands for the number of coded symbols from file j stored in every transmitter. The probability mass function of Z is

$$\begin{aligned} P_Z(z) &= \sum_j \sum_h P_{Z|J,H}(z|j,h) \cdot P_J(j) \cdot P_H(h) \\ &= \sum_j \sum_h \theta_j\, \gamma_h\, P_{Z|J,H}(z|j,h). \end{aligned}$$

We are interested in deriving the distribution of the number of output symbols which have to be sent over the backhaul link to serve the request of a user, which we denote by random variable T. If we condition T to Z, it is easy to see how the probability of $T = t$ corresponds to the probability that decoding succeeds when the user has received exactly t output symbols from the backhaul link in excess to the z output symbols it received from the transmitters through local links, that is, when $m = z + t$.
In order to derive $P_{T|Z}(t|z)$ we shall distinguish two cases.

If $z \leq k$, then

$$P_{T|Z}(t|z) = \begin{cases} P_F(z-k+t-1) - P_F(z-k+t) & \text{if } t \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

If $z > k$, then

$$P_{T|Z}(t|z) = \begin{cases} 1 - P_F(z-k) & \text{if } t = 0, \\ P_F(z-k+t-1) - P_F(z-k+t) & \text{if t} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

As in [6], the expectation $\mathbb{E}[T]$ is given by

$$\begin{aligned} \mathbb{E}[T] = \sum_t t \Bigg(& \sum_{z=0}^{k} P_{T|Z}(t|z) P_Z(z) \\ & + \sum_{z=k+1}^{\infty} P_{T|Z}(t|z) P_Z(z) \Bigg). \end{aligned}$$

Let us define $\bar{T}_1$ and $\bar{T}_2$ as

$$\begin{aligned} \bar{T}_1 &:= \sum_t t \sum_{z=0}^{k} P_{T|Z}(t|z) P_Z(z) \\ \bar{T}_2 &:= \sum_t t \sum_{z=k+1}^{\infty} P_{T|Z}(t|z) P_Z(z) \end{aligned}$$

so that

$$\mathbb{E}[T] = \bar{T}_1 + \bar{T}_2. \tag{2}$$

If we introduce the variable change $\delta = z - k + t$ in the expression of $\bar{T}_1$, we obtain

$$\begin{aligned} \bar{T}_1 &= \sum_{z=0}^{k} P_Z(z) \sum_{\delta=z-k}^{\infty} (\delta - z + k) \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &\overset{(a)}{=} \sum_{z=0}^{k} P_Z(z) \sum_{\delta=0}^{\infty} (\delta - z + k) \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &= \sum_{z=0}^{k} P_Z(z) \Bigg(\sum_{\delta=0}^{\infty} \delta \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &\quad + (k-z) \sum_{\delta=0}^{\infty} \Big[P_F(\delta-1) - P_F(\delta) \Big] \Bigg) \\ &\overset{(b)}{=} \sum_{z=0}^{k} P_Z(z) \Big(\mathbb{E}[\Delta] + k - z \Big) \\ &= (\mathbb{E}[\Delta] + k) \Pr\{Z \leq k\} - \sum_{z=0}^{k} z P_Z(z) \end{aligned} \tag{3}$$

where equality (a) is due to $[P_F(\delta-1) - P_F(\delta)] = 0$ for $\delta < 0$ and equality (b) is due to

$$\sum_{\delta=0}^{\infty} [P_F(\delta-1) - P_F(\delta)] = 1.$$

Introducing the same variable change in the expression of $\bar{T}_2$ we have

$$\begin{aligned} \bar{T}_2 &= \sum_{z=k+1}^{\infty} P_Z(z) \sum_{\delta=z-k}^{\infty} (\delta - z + k) \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &= \sum_{z=k+1}^{\infty} P_Z(z) \Bigg(\sum_{\delta=z-k}^{\infty} \delta \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &\quad + \sum_{\delta=z-k}^{\infty} (k-z) \Big[P_F(\delta-1) - P_F(\delta) \Big] \Bigg). \end{aligned} \tag{4}$$

Let us rewrite (4) as follow

$$\bar{T}_2 = \sum_{z=k+1}^{\infty} P_Z(z) \Big(\bar{T}_{21}(z) + \bar{T}_{22}(z) \Big) \tag{5}$$

where

$$\begin{aligned} \bar{T}_{21}(z) &= \sum_{\delta=z-k}^{\infty} \delta \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &= \sum_{\delta=0}^{\infty} \delta \Big[P_F(\delta-1) - P_F(\delta) \Big] - \sum_{\delta=0}^{z-k-1} \delta \Big[P_F(\delta-1) - P_F(\delta) \Big] \\ &= \sum_{\delta=0}^{\infty} P_F(\delta) - \Bigg[\sum_{\delta=0}^{z-k-1} P_F(\delta) - (z-k) P_F(z-k-1) \Bigg] \\ &= \mathbb{E}[\Delta] - \sum_{\delta=0}^{z-k-1} P_F(\delta) + (z-k) P_F(z-k-1) \end{aligned} \tag{6}$$

and

$$\begin{aligned}\bar{T}_{22}(z) &= \sum_{\delta=z-k}^{\infty}(k-z)\Big[P_F(\delta-1)-P_F(\delta)\Big]\\ &=(k-z)\Bigg\{\sum_{\delta=0}^{\infty}\Big[P_F(\delta-1)-P_F(\delta)\Big]\\ &\quad-\sum_{\delta=0}^{z-k-1}\Big[P_F(\delta-1)-P_F(\delta)\Big]\Bigg\}\\ &=(k-z)\Big\{1-\Big[1-P_F(z-k-1)\Big]\Big\}\\ &=(k-z)P_F(z-k-1). \end{aligned} \tag{7}$$

By inserting (6) and (7) in (5) and sum we obtain

$$\bar{T}_2 = \sum_{z=k+1}^{\infty} P_Z(z)\Big[\mathbb{E}[\Delta]-\sum_{\delta=0}^{z-k-1}P_F(\delta)\Big]. \tag{8}$$

Finally, if we replace (3) and (8) in (2), the expression of the average backhaul transmission rate becomes

$$\begin{aligned}\mathbb{E}[T] &= \Big(\mathbb{E}[\Delta]+k\Big)\Pr\{Z\le k\}-\sum_{z=0}^{k}zP_Z(z) \qquad (9)\\ &+\sum_{z=k+1}^{\infty}P_Z(z)\Big[\mathbb{E}[\Delta]-\sum_{\delta=0}^{z-k-1}P_F(\delta)\Big]\\ &=\mathbb{E}[\Delta]+\sum_{z=0}^{k}(k-z)P_Z(z)-\sum_{z=k+1}^{\infty}P_Z(z)\Bigg[\sum_{\delta=0}^{z-k-1}P_F(\delta)\Bigg].\end{aligned}$$

The last term in (9) is always non negative. Hence, we can upper bound (9) as

$$T_{\text{UP}} = \mathbb{E}[\Delta]+\sum_{z=0}^{k}(k-z)P_Z(z)\ge\mathbb{E}[T].$$

V. LT Placement Optimization

The goal of the LT placement optimization problem is to define the number of output coded symbols per file that has to be stored in the transmitters' caches such that during the delivery phase the average backhaul link is minimized.

The LT placement optimization problem can be formulated as follows

$$\min_{w_1,\dots,w_n} \quad T_{\text{UP}} \tag{10}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j = Mk \tag{11}$$

$$w_j\in\mathbb{N} \quad j=1,\dots,n. \tag{12}$$

where (11) limits total number of stored LT coded symbols to the size cache capacity, while (12) accounts for the discrete nature of the optimization variable.

The evaluation on the exact expression $\mathbb{E}[\Delta]$ is more complex than the evaluation of the upper bound. We hence opt for the minimization of the upper bound, comforted by the numerical results presented next.

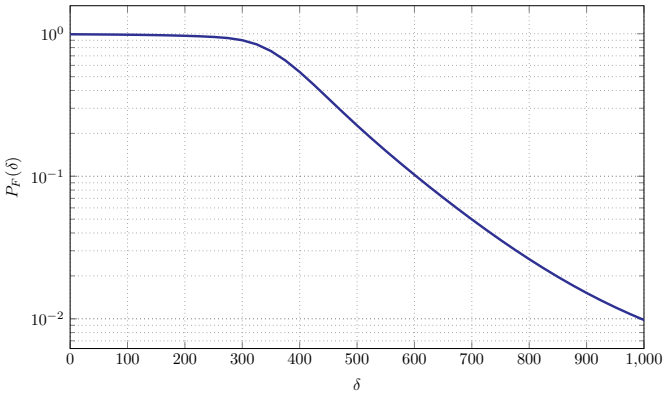

Fig. 2. Failure probability in function of overhead δ for a robust soliton distribution with parameters $c = 0.05$ and $d = 3$ for k = 10000 under peeling iterative decoding.

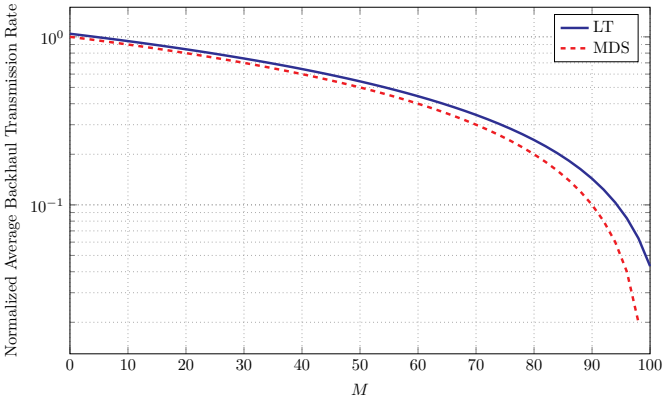

Fig. 3. Normalized average backhaul transmission rate as a function of memory size M for LT and MDS codes given $n = 100$, $k = 10000$, $\alpha = 0$ and $\gamma_1 = 1$.

A suboptimal solution of the problem in (10)-(12) can be obtained by using linear programming methods when the constraint (12) is relaxed.

VI. Results

We are interested in evaluating the normalized backhaul transmission rate defined as $\mathbb{E}[T]/k$, where k the number of fragments of a file. Given the connectivity γ_h and the requested θ_j probability distribution for each scenario, we optimize the number of fragments w_j to be cached during the placement phase by solving the optimization problem (10)-(12). Finally, given w_j we calculate $\mathbb{E}[T]/k$ from (9).

We consider the system described in Section II which is illustrated in Fig. 1. We assume users to be uniformly distributed and transmitters have an area of coverage of $r = 60$ m. We further assume that each transmitter is deployed according to a uniform two dimensional grid, with spacing $d = 45$ m. A user can be served by multiples caches due to the fact that the coverage area of transmitters partially overlap. In particular, a user can be served by h caches with probability γ_h. With simple geometrical calculations we can derive the following connectivity distribution

$$\gamma_1 = 0.2907,\ \gamma_2 = 0.6591,\ \gamma_3 = 0.0430,\ \gamma_4 = 0.0072. \tag{13}$$

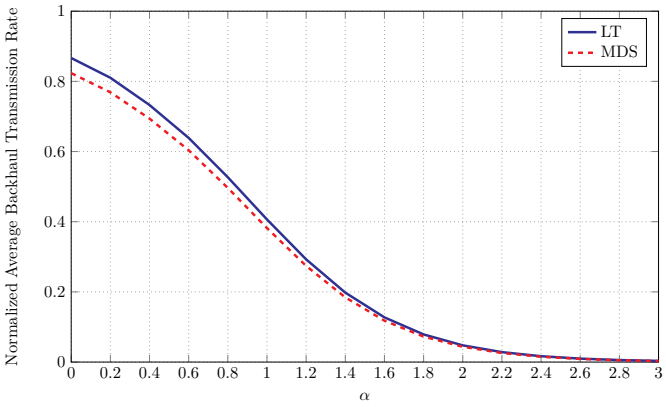

Fig. 5. Normalized average backhaul transmission rate as a function of the file parameter distribution α for LT and MDS given $n = 100$, $k = 10000$, $M = 10$ and $\gamma_1 = 0.2907$, $\gamma_2 = 0.6591$, $\gamma_3 = 0.0430$, $\gamma_4 = 0.0072$.

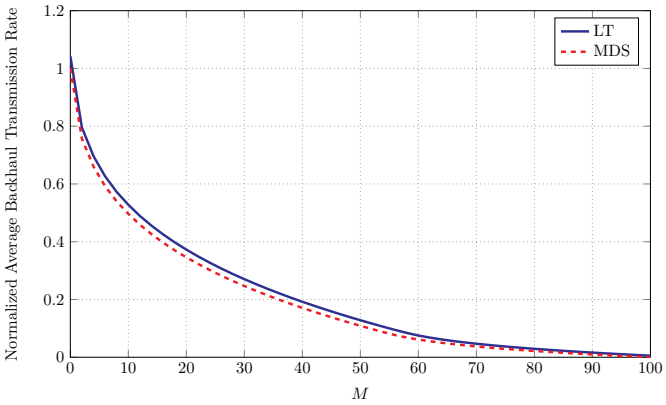

Fig. 4. Normalized average backhaul transmission rate as a function of memory size M for LT and MDS codes given $n = 100$, $k = 10000$, $\alpha = 0.8$ and $\gamma_1 = 0.2907, \gamma_2 = 0.6591, \gamma_3 = 0.0430$ and $\gamma_4 = 0.0072$.

In all setups, we consider that each file is fragmented in $k = 10000$ input symbols. We assume that the mNode implements an LT code characterized by a robust soliton distribution (RSD) [7] with parameter $c = 0.05$ and $d = 3$, where c and d have been chosen so that the average overhead is minimized. The probability of decoding failure $P_F(\delta)$ was derived as explained in Sec. III.

In Fig. 2 the probability of decoding failure as a function of δ for RSD under iterative peeling decoding is plotted. From (1), we obtained that the overhead average is $\mathbb{E}[\Delta] = 431,95$.

In our first setup, we study the normalized average backhaul transmission rate as a function of the cache size M when the library cardinality is $n = 100$. We consider that each file has the same probability to be requested, i.e. Zipf distribution with $\alpha = 0$. We assume that each user can be connected to only one transmitter, i.e. $\gamma_1 = 1$. In Fig. 3, we can observe that the LT coded caching scheme performs close to the MDS coded caching scheme. The cost in terms of average transmission rate for using a LT code is only the 4.32% which coincides with the average overhead of the LT code.

In our second setup, we consider the connectivity distribution given in (13). We also assume that files are requested according to a Zipf distribution with $\alpha = 0.8$. In Fig. 4 the normalized average backhaul transmission rate is shown as a function of the cache size M. We can see that the LT caching scheme is comparable to the benchmark MDS scheme. Furthermore, as the cache size increases, the difference between the two approaches becomes negligible.

In our last case, we consider the same connectivity γ_h as the previous setup. We assume to have a fixed memory size $M = 10$ and library size $n = 100$ files. In Fig. 5 the normalized average backhaul transmission rate is shown as a function of the shape parameter α. As expected, for $\alpha = 0$, i.e. when files are equiprobable, the MDS scheme outperforms the scheme based on LT codes. However as α increases, the performance of the LT scheme approaches that of the MDS scheme.

VII. Conclusions

We considered a scheme were coded content is cached at the edge in a heterogeneous network, with the aim of minimizing the number of transmission from the core of the network during the delivery phase. In particular, we considered the use of LT codes under peeling (iterative) decoding. We delivered an analytical expression of the average backhaul transmission rate. We also formulated the optimization problem related to the placement phase of the LT code based caching scheme. Finally, we compared the performance of the LT scheme with that of an optimal but impractical MDS scheme. Our simulation results indicate that the performance of LT codes approaches that of the optimal scheme, but exhibiting a much lower decoding complexity.

References

[1] J. Liao, K. K. Wong, M. R. A. Khandaker, and Z. Zheng, "Optimizing cache placement for heterogeneous small cell networks," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 120–123, Jan. 2017.

[2] E. Ozfatura and D. Gündüz, "Mobility and popularity-aware coded small-cell caching," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 288–291, Feb. 2018.

[3] A. Piemontese and A. G. i. Amat, "MDS-coded distributed storage for low delay wireless content delivery," in *Int. Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Brest, France, Sep. 2016.

[4] V. Bioglio, F. Gabry, and I. Land, "Optimizing mds codes for caching at the edge," in *Proc. IEEE Globecomm*, San Diego, U.S.A., Dec. 2015.

[5] K. Shanmugam, N. Golrezaim, A. Dimakis, A. Malisch, and G. Caire, "FemptoCaching: wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 61, no. 12, pp. 6815 – 6832, Dec. 2015.

[6] E. Recayte, F. Lázaro, and G. Liva, "Caching at the edge with fountain codes," in *Submitted to the 9th Advanced Satellite Mobile Systems Conference*, Berlin, Germany, Sep. 2018.

[7] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symp. on Foundations of Computer Science*, Vancouver, Canada, Nov. 2002, pp. 271–282.

[8] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proc. 2004 IEEE International Symp. on Inf. Theory*, Chicago, Illinois, US, Jun. 2004.

[9] A. Shokrollahi, "Theory and applications of Raptor codes," *Mathknow*, vol. 3, pp. 59–89, 2009.

[10] F. Lázaro, G. Liva, and G. Bauch, "Inactivation decoding of LT and Raptor codes: Analysis and code design," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, Oct. 2017.