Coding for Caching in 5G Networks

Yasser Fadlallah, Antonia Maria Tulino, Dario Barone, Giuseppe Vettigli, Jaime Llorca, Jean-Marie Gorce

Coding for Caching in 5G Networks

Yasser Fadlallah[1,3], *Member, IEEE,* Antonia M. Tulino[2,4], *Fellow, IEEE,* Dario Barone[2], *Student Member, IEEE,* Giuseppe Vettigli[2], *Student Member, IEEE,* Jaime Llorca[4], *Member, IEEE,* and Jean-Marie Gorce[1], *Senior, IEEE,*

[1] INSA-Lyon, CITI-INRIA, F-69621, Villeurbanne, France.

[2] DIETI Department, University of Naples Federico II, Italy. Email: antoniamaria.tulino@unina.it

[3] CS Department, University of Sciences and Arts in Lebanon, Lebanon. Email: y.fadlallah@usal.edu.lb

[4] Nokia Bell Labs, NJ. Email: jaime.llorca@nokia.com

Abstract

One of the major goals of the 5G technology roadmap is to create disruptive innovation for the efficient use of the radio spectrum to enable rapid access to bandwidth-intensive multimedia services over wireless networks. The biggest challenge towards this goal lies on the difficulty in exploiting the multicast nature of the wireless channel in the presence of wireless users that rarely access the same content at the same time. Recently, the combined use of wireless edge caching and coded multicasting has been shown to be a promising approach to simultaneously serve multiple unicast demands via coded multicast transmissions, leading to order-of-magnitude bandwidth efficiency gains. However, a crucial open question is how these theoretically-proven throughput gains translate in the context of a practical implementation that accounts for all the required coding and protocol overheads. In this article, we first provide an overview of the emerging caching-aided coded multicast technique, including state of art schemes and their theoretical performance. We then focus on the most competitive scheme proposed to date and describe a fully working prototype implementation in CorteXlab, one of the few experimental facilities where wireless multiuser communication scenarios can be evaluated in a reproducible environment. We use our prototype implementation to evaluate the experimental performance of state-of-the-art caching-aided coded multicast schemes compared to state-of-the-art uncoded schemes, with special focus on the impact of coding computation and communication overhead on the overall bandwidth efficiency performance. Our experimental results show that coding overhead does not significantly affect the promising performance gains of coded multicasting in small-scale real-world scenarios, practically validating its potential to become a key next generation 5G technology.

I. Introduction

Along with the internet revolution, IP traffic is growing at a tremendous pace and it is expected to reach two zettabytes per year by 2019. Mobile data networks are envisioned to support up to 14% of this global data traffic coming from a plethora of different market segments. Among these segments, multimedia streaming is the service with the highest penetration rate, having the major impact on the overall traffic increase. On the other hand, current mobile network generations cannot cope with this explosive traffic growth due to the capacity limitations of radio access, backhaul, and core mobile networks, and the increasingly unicast and on-demand nature of users' content demands. In order to support this traffic expansion, the fifth generation (5G) of mobile networks is under

preparation. Among the key performance challenges that 5G needs to address are: throughput, latency, and energy efficiency. That is, 5G is expected to provide 1000x higher throughput, sub-millisecond service latencies, and up to 90 percent overall energy savings [1]. Despite the myriad of technological advances at the physical (PHY) and medium access control (MAC) layers (e.g., inter-cell interference coordination (ICIC), massive multiple-input-multiple-output (MIMO), carrier aggregation), targeted data rates are still significantly out of reach. To this end, 5G envisions novel architectural components for the next generation radio access network (RAN), including small cell densification, efficient wireless backhauling, and network self-organization [1]. In this context, the use of inexpensive storage resources within the RAN is emerging as a promising approach to reduce network load, and effectively increase network capacity.

A. Prominence of Wireless Caching in 5G

Wireless caching, i.e., caching content within the wireless access network is gaining interest, specially in ultra-dense networks where many connected devices try to access various network services under latency, energy efficiency, and/or bandwidth limitation constraints [1]. Proactively caching content items at the network edge (e.g., at the RAN) helps in relieving backhaul congestion and meeting peak traffic demands with lower service latency as Fig. 1 illustrates. For maximum benefits, network operators can intelligently exploit users' context information, classify content by popularity, and improve predictability of future demands to proactively cache the most popular content before being requested by end users. Such a strategy is able to fulfill the quality of service (QoS) requirements while significantly reducing the use of bandwidth resources and its associated energy consumption. Content items can be cached at different locations of the mobile network. Within the RAN, base stations (or small base stations), user equipment (UE) devices, and access points (AP) can be enhanced with additional memory for content caching. While caching can also happen within the evolved packet core (EPC), the main benefit of caching at the EPC is to reduce peering traffic between internet service providers (ISP). It is the additional deployment of cache memories within the RAN that can crucially help minimizing intra-ISP traffic, relieving backhaul load, and reducing service latencies [2].

B. From Uncoded to Coded Content Distribution

A substantial amount of recent studies have analyzed the use of wireless caching as a promising solution for 5G. Among these studies, [3] introduced the idea of femtocaching and addressed the question of which files should be assigned to which helper nodes (femtocell-like base stations), while [4, and reference therein] considered the improvement in caching efficiency that can be obtained by dynamically learning content popularity and updating cache contents at the network edge. Despite considerable interest, such studies focus on the data placement problem in isolation, assuming the use of unicasting or naive (uncoded)[1] multicasting during transmission, and hence ignoring the potential benefits of joint placement and transmission code design.

[1]The term uncoded is used to refer to a scheme in which, at each use of the channel, the transmission is composed of packets that belong to the same file, while the term coded refers to a scheme in which transmissions can be composed of a mixture of packets from different files.

In [5], the data placement problem is generalized to the coded content distribution problem where the goal is to jointly determine the placement and routing of (possibly coded) information over the network, showing that joint code design significantly increases multicast efficiency, leading to substantial improvements in reducing network load and access latencies. A number of information-theoretic studies have then characterized the order-optimal performance of a caching network of special practical interest, the shared link caching network, formed by a single source node (e.g., base station) with access to a library of content files connected via a shared multicast link to multiple user nodes (e.g., end devices or access points), each with caching capabilities. In this context, the work in [6] showed that under worst-case demands, caching portions of each file uniformly at random and using index coding (IC) [8] during transmission yields an overall load reduction that is proportional to the aggregate cache size. In [9], the authors analyzed the case in which user demands follow a Zipf popularity distribution, designing order-optimal achievable schemes[2] that adjust the caching distribution as a function of the system parameters to balance the gains from local cache hits and coded multicasting. Shortly after, [10] showed that the gains achieved by these schemes require a number of packets per requested item that grows exponentially with the number of caches in the system, leading to codes of exponential complexity that compromise their theoretical gains. Efficient polynomial-time schemes (e.g., [11]) have then been proposed to recover a significant part of the promising multiplicative caching gain.

In terms of practical implementations, the work in [4] provided a big data platform where learning algorithms can be used to predict content popularity and drive caching decisions, but the benefit of the learning techniques on improving caching efficiency is evaluated via numerical simulations. In addition, only conventional uncoded schemes are considered, and aspects related to advanced coding techniques such as caching-aided coded multicasting that can potentially provide much larger gains are largely overlooked. It is also important to note that, so far, only information-theoretic studies have shown the potential gains of such schemes, and the emulation work in [7] only considers $2-4$ users and 3 files, a very limited scenario that does not allow showing the real impact of computational complexity and coding overhead. Moreover, it is unclear whether existing schemes meet the requirements of current technologies, thus leaving plenty of open questions regarding practical performance benefits.

C. The Need for Experimental Validation

This article aims at bridging the gap between theory and practice in order to validate the benefits of caching-aided coded multicasting by designing a fully working prototype implementation and testing it in a large network testbed. Such testbed and prototype implementation provide a cornerstone for the evaluation of future schemes with more advanced wireless caching protocols and cache-enabled PHY layer techniques such as joint source-channel coding. We first provide an overview of the caching and coded multicasting framework and discuss the key concepts behind the ability to provide load reductions that are proportional to the aggregate cache size. We then introduce a new frame structure that includes specific fields to account for all the practical aspects required for a fully working real-world

[2]An achievable scheme is said to be order-optimal if, as the file size goes to infinity, the number of transmissions needed to satisfy the user demands scales as the information theoretic optimal number of transmissions needed to satisfy the user demands; i.e the ratio between the achievable and optimal number of transmissions is upper bounded by a constant independent of all the system parameters.

implementation. The primary role of the newly designed frame structure is to allow decoding of coded data at each receiver. Our MAC layer frame design is combined with an orthogonal frequency division multiplexing (OFDM) PHY layer, which makes it compatible with long term evolution (LTE) advanced mobile networks or further PHY layer standards. The resulting fully working prototype is implemented in a large-scale testbed facility, CorteXlab [13], composed of tens of highly flexible radio nodes deployed in a controlled and reproducible environment. We present experimental results in the context of key 5G challenges related to transmission delay, bandwidth usage, and energy efficiency. Our experimentation validates the fact that memory can be effectively turned into bandwidth leading to substantial network throughput gains.

II. Caching-aided coded multicasting

As previously stated, the use of caching together with smart offloading strategies in a RAN composed of evolved NodeBs (eNBs), AP (e.g., WiFi), and UEs, can significantly reduce the backhaul traffic and service latency. In this context, a shared link caching network (SLCN) topology can be identified at different levels of the mobile network. Indeed, a radio cell constitutes a SLCN where the eNB acts as the source node connected to the UEs via a shared multicast link. In addition, a SLCN can also be formed by a core network (CN) server (source node) connected to a set of eNBS via a shared wireless backhaul. In both cases, user nodes are equipped with storage resources for content caching. Accordingly, we focus on the analysis and implementation of a SLCN composed of a source node, with access to a library $\mathcal{F}$ of m binary files, connected to n user nodes via a shared multicast link. Each user node is equipped with a cache of storage capacity equivalent to M files, and can make up to L file requests according to a Zipf demand distribution. A multicast link is a shared channel in which any transmission can be overheard by all receivers.

A caching-aided coded multicast scheme is performed over two phases: i) the caching phase, where the source node populates the user caches with appropriate functions of the content library, and ii) the delivery phase, where the source forms a multicast codeword to be transmitted over the shared link in order to meet the users' content demands. These phases are generic for both coded and uncoded schemes, but naively performed in the uncoded case. In fact, when relying on uncoded or naive multicasting during the delivery phase, it is well known that the optimal caching strategy is to cache the top M most popular files at each user cache. This is however, in general, far from optimal when coding can be used in the delivery phase [9]. In the following, we discuss the potential of caching-aided code design and illustrate its major advantages compared to the optimal caching policy under uncoded (naive) multicasting.

A. Random Fractional Caching

Each binary file $f \in \mathcal{F}$ is divided into B_f equal-size packets or chunks. Given the *caching distribution* $\{p_f\}$, with $\sum_{f=1}^{m} p_f = 1$, each user caches chunks of file f with probability p_f. That is, each user caches a number of chunks $p_f M B_f$ ($p_f \leq 1/M$) of file f chosen uniformly at random. It is important to note that the randomized nature of the selection process allows users to cache different sets of chunks of the same file, shown to be key in creating coded multicast opportunities during the delivery phase. In [9], the authors showed that the optimal caching

distribution can be approximated by a truncated uniform distribution $p_f = 1/\tilde{m}, \forall f \leq \tilde{m}$ and $p_f = 0, \forall f > \tilde{m}$, without affecting order-optimality[3], and referred to this caching policy as random least frequently used (RLFU). Compared to the least frequently used (LFU) caching policy (best option under naive multicasting) where the same most popular files are entirely cached at each user, RLFU maximizes the amount of distinct packets collectively cached by the network.

B. Coded multicasting

A simple example in Fig. 2 illustrates the key benefits of coded multicasting during the delivery phase. The network is composed of a source and 3 user nodes requesting files from a library of $m = 4$ binary files $\mathcal{F} = \{A, B, C, D\}$. Each file (e.g., video segment) is divided into 2 chunks, yielding a library of chunks $\mathcal{C} = \{A_1, A_2, B_1, B_2, C_1, C_2, D_1, D_2\}$. During the caching phase, users 1, 2, and 3 randomly fill their caches with chunks $\{A_1, B_2, D_2\}$, $\{A_2, B_1, D_2\}$, and $\{A_2, B_2, D_1\}$, respectively. During the delivery phase, at a given request round, users 1, 2, and 3 make requests for video segments A, B, and D, respectively. Under an uncoded naive multicasting transmission scheme, the source needs to transmit the missing chunks A_2, B_2, and D_2 over the shared multicast link using 3 time slots. In contrast, by employing coded multicasting, the source can mix the three chunks A_2, B_2 and D_2 via a XOR operation (binary addition) and multicast the coded chunk $A_2 \oplus B_2 \oplus D_2$ using only one time slot. Clearly, in this case, coded multicasting reduces the number of transmissions (and hence the number of delivery time slots) by a factor of three.

As illustrated in the above example, a given user is able to decode its requested chunk from a mixture of combined chunks if and only if it has knowledge of all other combined chunks. Such a problem can be seen as an IC problem [8], and can be described by what is referred to as the *conflict graph* [9]. The conflict graph is constructed such that each graph vertex corresponds to one requested chunk, and an edge between two vertices is created if: i) they correspond to different requested chunks and ii) for each vertex, the associated chunk is not included in the cache of the user requesting the chunk associated with the other vertex. Notice that an edge between two vertices indicates that their associated chunks must be separately transmitted, while non-connected vertices can be modulo summed via XOR operation [8]. The goal is to find the best chunk combinations such that the total number of transmissions is minimized. A common approach, referred to as chromatic index coding (CIC) [9], is to compute a minimum *graph coloring* of the IC conflict graph, where the goal is to find an assignment of colors to the vertices of the graph such that no two connected vertices have the same color, and the total number of colors is minimized. The multicast codeword is constructed by generating sub-codewords obtained XORing the chunks with the same color, and then concatenating the resulting sub-codewords. The conflict graph of the example given in Fig. 2 is illustrated in the top left corner of the figure. The graph consists of 3 vertices corresponding to the three requested packets A_2, B_2, and D_2. There are no edges between the vertices of the graph since, for each vertex, the associated chunk is included in the cache of the users associated with the other vertices. Therefore, all vertices can be assigned the same color and binary added into a single coded transmission, as shown in 2.

[3]Details about the selection of the optimal $\tilde{m}$ are given in [9] (see section IV).

The work in [9] showed that the combined use of RLFU caching and CIC coded multicasting is order-optimal[4] under any Zipf demand distribution, and that RLFU-CIC provides multiplicative caching gains, that is, the per-user throughput scales linearly or super-linearly with the cache size. In order to prove this result, the authors resort to a polynomial-time approximation of CIC, referred to as greedy constrained coloring (GCC). While GCC exhibits polynomial complexity in the number of users and packets, both CIC and GCC can only guarantee the promising multiplicative caching gain when the number of packets per file grows exponentially with the number of users, significantly limiting their practical performance [10]. Subsequently, the works in [11] and [12] extended the RLFU-CIC and RLFU-GCC schemes to the non-homogeneous SLCN and proposed two improved coded multicasting algorithms: i) the greedy randomized algorithm search procedure (GRASP) based on a greedy randomized approach, and ii) the hierarchical greedy coloring (HGC). These algorithms have been shown to recover a significant part of the multiplicative caching gain, while incurring a complexity at most quadratic in the number of requested packets.

C. Decoding phase

From the observation of the received multicast codeword and its cached content, each user has to decode its intended chunks via its own decoding function. In order to guarantee decoding, the receiver needs to be informed (e.g., via a packet header that carries all necessary information, as shown in Fig. 3.a) of the sub-codewords in the concatenated multicast codeword that contain any of its intended chunks. For each of the identified sub-codewords, the receiver obtains its intended chunks by performing the simple binary addition.

In the next section, we describe a fully working prototype implementation that includes the design of the required packet header to ensure full decodability.

III. Implementation of caching-aided coded multicasting

While section II describes state of the art wireless caching and transmission code design, the impact of real protocol overheads on the multiplicative caching gain remains an open question that we address via a real prototype implementation in the following.

Our prototype implementation is based on the following components: i) a simplified application layer for generating and combining the requested chunks, ii) a MAC layer extended with additional header fields to allow decoding of coded packets, and iii) a PHY layer compliant with LTE standards. Our basic MAC layer frame implementation does not account for a complete standardized frame structure and the generated data is not encapsulated through the protocol stack, since our main goal is a proof-of-concept of caching-aided coded multicasting and its real-time feasibility. In the following, we describe in detail the MAC layer frame structure.

A. Frame structure

For a clear understanding of the implementation process, the basic frame structure is given in Fig. 3. Every accumulated packet is composed of two parts: header and payload. The payload represents the coded packet (divided

[4]Order-optimal in the sense that the number of transmissions needed to satisfy the user demands scales (in number of users, number of files, and memory size) as the optimal scheme.

as: $payload_1, \cdots, payload_K$); a mixture of original data chunks with elements in the Galois Field of order two GF(2), making it easy to encode and decode with a simple XOR operation. The header illustrated in Fig. 3.c contains the minimal information required for a successful extraction of each individual chunk. It carries the combined chunks identities and consists of the following information:

- Header Length: This is the first element of the header, and its size is fixed to one byte. It carries the number of header bytes.
- File IDs: These are the IDs of the files to which the combined chunks (payload) belong. Each ID requires a multiple of one byte, depending on the number of content files in the library.
- Chunk IDs: These are the IDs of the combined chunks within the transmitted packet. Each ID requires a multiple of one byte, depending on how many chunks a file is partitioned into.
- Chunk sizes: These are the sizes of the combined chunks, and are encoded with a multiple of four bytes to make the receiver able to recognize the size of each chunk.

In a practical network scenario, it is unusual to have a header length exceeding one byte since the number of requests and the number of simultaneously served users is generally limited. Notice that the uncoded design will necessitate the same header structure but only for one target user (unless the same packet is destined to multiple users) because no packet combination is performed. This means that in an uncoded scheme each packet is separately transmitted with its associated header, without the need for additional overhead information. This is due to the fact that we are assuming multicast transmission over the downlink shared channel (DL-SCH). This LTE physical layer transport channel is the main channel for downlink data transfer and it is used by many logical channels. The fact that the header information in the coded scheme depends on the number of served users implies a variable header length. An example of the header decomposition is illustrated in Fig. 3.c, where the number of files and chunks are assumed to not exceed one byte each, and the maximum size of a chunk is limited to four bytes. The payload length of a coded packet is equal to the largest chunk's length among the combined ones. Before being transmitted, the coded packet is partitioned into small packets and numbered such that the receiver can rebuilt the original coded packet. Each coded packet (see Fig. 2) is dedicated to users with IDs indicated in the header information. Aiming at decreasing the packet error probability (PER), the first small packet will be limited to the header data, and the others are charged with the payload. A $32-$bit cyclic redundancy check (CRC) is appended to each small packet for error detection. In the header information, if the CRC detects some errors the whole coded packet is lost and the user drops all related small packets. Otherwise, each user checks whether concerned or not. If so, the user proceeds to the small packet decoding, based on its cached data and the reported files and chunks IDs. Conversely, if the user is not concerned the packet is dropped and the user waits for the next header to check out its affiliation. In case of channel erasure, the small packets are replaced with dummy bytes.

B. CorteXlab platform

The resulting fully working prototype is implemented in a large-scale testbed facility, CorteXlab [13] which is a testbed for cutting edge radio experimentation, composed of a mix of radio nodes, including SISO and MIMO software defined radio (SDR) nodes. The testbed shown in Fig. 4 is installed in a large ($180\ m^2$) shielded room

partly covered with electromagnetic wave absorbing material. User nodes are placed over a regular grid with an inter-node distance of 1.8 meters, and accept any PHY layer implementations on both hardware and software. A unified server is available for starting, coordinating, and collecting the results of experiments. As a development tool, the GNU Radio software is employed for real-time experimentation.

IV. End-to-end performance results and perspectives

A. Setup Environment

Our SLN experimentation consists of one radio source node and $n = 10$ radio user nodes. Each user requests $L = 10$ files from a library $\mathcal{F}$ of $m = 20$ binary files, each of size 2.8 Mb. A cache of size M files is deployed at every user. Such a scenario can be seen as if the users are APs carrying multiple requests from different UEs, and the source is the eNB having access to the content library. The file request distribution is drawn from a Zipf distribution with Zipf parameter α: $\alpha = 0$ returns a uniform request distribution; the higher the Zipf parameter α, the more skewed is the request distribution. The binary files are partitioned into equally sized $B = 100$ chunks yielding a library of $m_b = 2000$ chunks. Both RLFU with $\tilde{m}$ optimized as in [9] and LFU caching policies are adopted for the coded and naive multicasting delivery schemes, respectively.

The output of multicast encoder goes into an OFDM modulator with the following transmission parameters. Each PHY frame is decomposed into small packets of size 100 bytes to which a CRC-32 and an OFDM header are appended for error detection and OFDM demodulation, respectively. The OFDM header and payload data are mapped into a binary phase shift keying (BPSK) and a quadrature PSK (QPSK), respectively, and each symbol is transmitted over a sample duration of $T_s = 1\mu$s. The data is carried over $L_f = 48$ subcarriers spaced by $\Delta f = 15$KHz and the central frequency is set to 2.45 GHz.

B. Experimentation results

The focus herein is on the gain at the MAC layer that is based on counting the total number of required bytes to serve all UEs. Assuming the same number of requests L from all users, the normalized minimum rate (NMR) is defined as $R_t/(L \times \text{file size})$, where R_t is the total number of required bytes at the MAC layer to satisfy all user demands. Note that NMR is in general a non-decreasing function of the number of users, and a decreasing function of cache size, M; in particular, for $M = 0$, the NMR is equal to the total number of distinct user requests. In the following, we provide a numerical validation of the prototype performance in terms of NMR. Specifically, we analyze the performance of our prototype solutions prot-HGC and prot-GRASP in terms of NMR compared with: i) HGC and GRASP for finite file packetization simulated in Matlab environment without taking into account implementation overhead ii) Naive multicasting with LFU caching policy at the rate of the worse channel receiver, and iii) the benchmark upper bound GCC when $B = \infty$ (see [9]). The trend in terms of NMR demonstrated by the prototype confirms the gains predicted by the theory. Figs. 5.a and 5.b show the NMR as function of the cache size and the Zipf parameter α respectively. This metric is specially illustrative of the amount of bandwidth resources the wireless operator needs to provide in order to meet the receiver demands. In Fig. 5.a, we assume a Zipf parameter $\alpha = 0$. Observe first the performance of naive-multicast. As expected, the load reduces approximately linearly with

the cache size M. Observe, now, how the significant multiplicative caching gains (w.r.t. naive-multicast) quantified by the upper bound (RLFU-GCC with $B = \infty$) are remarkably preserved by the prototype solutions (prot-HGC and prot-GRASP) which achieve an NMR almost indistinguishable from the corresponding schemes implemented in Matlab environment without taking into account the encoding and decoding overheard. Fig. 5.a clearly shows the effectiveness of the proposed implementation in allowing receivers to decode their requested files at an NMR very close to the theoretically optimal NMR. From Fig. 5.a, it is also apparent that the two coloring algorithms have similar performance for $\alpha = 0$. The effectiveness of coded multicasting is highly influenced by the Zipf parameter α, as illustrated in Fig. 5.b for cache size $M = 2$ files. Observe how the reduction in NMR enabled via coded multicasting is much more attractive in the region of $\alpha \leq 1$.

In order to illustrate the behavior of the multiplicative gains, Figs. 5.c and 5.d show the prototype NMR gains as a function of the cache size M and the Zipf parameter α, respectively. The gain of a given scheme is defined as the ratio between the NMR achieved by naive multicasting with LFU caching policy and the NMR achieved by that scheme. In particular, when the scheme is a prototype implementation, then the NMR of naive multicasting is computed with its associated overhead. From Fig 5.c, we can observe that the gain is a monotonic non-decreasing function of the cache size. Note that we do not plot the point at $M = 20$ since it is well known that the NMR is zero for all the schemes, and hence the gain is given by an indeterminate form of type 0/0. Fig 5.c shows that the gains achieved by prot-GRASP and prot-HgC are very close to the gains achieved by the corresponding MATLAB simulated schemes, confirming the little impact of the implementation overhead on the overall performance. Furthermore, it is worth noticing that due to the reduced number of transmitted coded packets compared to the number of uncoded packets transmitted by naive multicasting, the total overhead size is also smaller. That is, even though each packet header length is larger, the total number of header bytes over all transmissions is also reduced.

In terms of the Zipf parameter α, Fig. 5.d shows that for $M = 2$ files, a gain close to 1.3 is obtained under uniform popularity ($\alpha = 0$), and this gain tends to 1 as the popularity distribution becomes more skewed.

C. Turning memory into bandwidth

In this section, we evaluate the physical layer bandwidth gains enabled by coded multicasting. To do so, we assume a fixed video transmission delay (e.g., according to users' QoS) and evaluate the bandwidth required to serve the video segment requests of all users. Fig. 6 illustrates the bandwidth gain (BG) evolution at the PHY layer with respect to the number of users for different cache sizes. We define the PHY BG as the bandwidth required to serve all requests via naive multicasting over the bandwidth required via the use of coded multicasting. The increase in BG can be clearly observed with respect to both the cache size and the number of users. For instance, assuming a cache size $M = 10\%$ of the library size, the gain starts with a value around 1.1 for 5 users and goes up to 1.31 for 40 users. Similarly, at $M = 30\%$, the gain increases from around 1.4 for 5 users and reaches around 1.68 for 40 users. The increase of the BG with respect to the number of users is specially relevant, as it illustrates the scalability benefits of coded multicasting.

V. Future directions

In the above discussion, coding overhead and computational complexity have been proven not to limit the performance gain of wireless caching for coded multicasting. However, several open problems related to PHY layer protocols are currently under investigation, among which we cite the following:

- Variation of the channel characteristics: Regarding the variations of channel statistics across users (e.g., different SNRs), the work in [14] provided a theoretical analysis that takes into account the wireless channel characteristics in the presence of any combination of unicast/multicast transmission and wireless edge caching. They proposed a channel-aware caching-aided coded multicast scheme based on joint source-channel coding with side information. Such scheme is able to guarantee a rate to each receiver, within a constant factor of the optimal rate, had the remaining users experienced its same channel conditions. The scheme preserves the cache-enabled multiplicative throughput gains by completely avoiding throughput penalization from the presence of receivers experiencing worse propagation conditions. The implementation of this scheme in CorteXlab is part of our next steps for future work. As opposed to network emulation platforms such as in [7], CorteXlab will allow us to properly test user mobility and realistic channel degradation across wireless end points.
- Combination with MIMO schemes: The use of MIMO schemes is an interesting topic with significant active research. Undergoing studies such as [15] have shown that in a MIMO setting, coded multicasting is indeed complementary to MIMO, and the combination of both provides cumulative gains in most practical scenarios. This is also object of future work, and again, CorteXlab represents a key advantage in order to easily include next generation radio technologies.
- Dynamic scenarios: Our current implementation setting is limited to static scenarios with respect to file popularity and number of users. Ideas related to cache adaptation with respect to dynamic popularity distributions and varying number of users are of interest for future work and currently under investigation.

VI. Conclusion

This article discusses the potential of caching-aided coded multicasting for improving bandwidth efficiency in next generation wireless access networks. A real-time implementation for performance evaluation in real environments has been presented. On the way from theory to practical evaluation, a complete frame structure for the transmitter and the receiver has been proposed. Interestingly, the additional coding overhead does not compromise performance and leads to an overall positive multicasting gain, reducing bandwidth requirements and transmission delay when compared to the best uncoded schemes. We have integrated the coded multicast design in an OFDM based PHY layer, and deployed the scenario in CorteXlab, a shielded experimentation room, using radio nodes. Our work also shows the potential of such facility to validate new concepts relative advanced radio technologies for 5G networks. Finally, we have briefly described interesting open problems related to PHY layer protocols that are currently under investigation.

REFERENCES

[1] A. Osseiran *et al.*, "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," in *IEEE Commun. Mag.,* vol. 52, no. 5, pp. 26-35, May 2014

[2] X Wang *et al.*, "Cache in the air: exploiting content caching and delivery techniques for 5G systems." in *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131-139, February 2014.

[3] N. Golrezaei *et al.*, "Femtocaching: Wireless video content delivery through distributed caching helpers." in *Proc. of IEEE INFOCOM*, Orlando, FL, 2012, pp. 1107-1115.

[4] E. Zeydan *et al.*, "Big data caching for networking: Moving from cloud to edge", in *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36-42, September 2016.

[5] J. Llorca *et al.*, "Network-coded caching-aided multicast for efficient content delivery," in *Proc. IEEE ICC* Budapest, 2013, pp. 3557-3562.

[6] M. Maddah-Ali, and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," in *IEEE/ACM Trans. Netw.* , vol. 23, no 4, p. 1029-1040, 2015.

[7] U. Niesen, and M.A. Maddah-Ali, "Coded caching for delay-sensitive content," in *Proc. IEEE ICC*, London, 2015, pp. 5559-5564.

[8] A. Blasiak, R. Kleinberg, and E. Lubetzky, "Index coding via linear programming," *preprint arXiv:1004.1379*, 2010 ; http://arxiv.org/abs/1004.1379.

[9] M. Ji *et al.*, "Order-optimal rate of caching and coded multicasting with random demands," *preprint arXiv:1502.03124,* Feb. 2015 ; https://arxiv.org/abs/1502.03124.

[10] K. Shanmugam *et al.*, "Finite Length Analysis of Caching-Aided Coded Multicasting," in *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5524-5537, Oct. 2016.

[11] G. Vettigli *et al.*, "An efficient coded multicasting scheme preserving the multiplicative caching gain," in *Proc. IEEE INFOCOM WKSHPS*, Hong Kong, 2015, pp. 251-256.

[12] M. Ji *et al.*, "An efficient multiple-groupcast coded multicasting scheme for finite fractional caching," in *Proc. IEEE ICC*, London, 2015, p. 3801-3806.

[13] L. S. Cardoso *et al.*, "CorteXlab: A facility for testing cognitive radio networks in a reproducible environment," in *Proc. EAI CROWNCOM,* Oulu, 2014, pp. 503-507.

[14] A.S. Cacciapuoti *et al.*, M Caleffi, M. Ji, J. Llorca, A. Tulino, "Speeding up Future Video Distribution via Channel-Aware Caching-Aided Coded Multicast", in *IEEE JSAC*, vol. 34, no. 8, pp. 2207-2218, Aug. 2016.

[15] S. Yang, K. Ngo, and M. Kobayashi, "Content Delivery with Coded Caching and Massive MIMO in 5G," in *Proc. IEEE ISTC*, Brest, France, 2016, pp. 370-374.

PLACE PHOTO HERE

Yasser Fadlallah (S10 – M' 14) received the Telecommunication Engineering Diploma from the Faculty of Engineering, Lebanese University, Lebanon, in 2009, the M.S. degree from the Universit de Bretagne Occidentale, France, in 2010, and the Ph.D. degree from Tlcom Bretagne, France, in 2013. In 2012, he was a visiting PhD student at the Coding and Signal Transmission Laboratory, University of Waterloo, Canada. Between 2013 and 2014 he was an R&D engineer position at Orange Labs, Paris. In 2015, he held a post-doctoral research position at INRIA until Sept. 2016, and then joined the University of sciences and arts in Lebanon (USAL) where he hold currently an assistant professor position. His research interests lie in the wireless communications area, focusing on interference management, advanced and low-complexity receivers, wireless caching, and multiple antennas systems.

PLACE PHOTO HERE

Antonia M. Tulino (S '00 – M '03 – SM '05 – F '13) received the Ph.D. degree in Electrical Engineering from Seconda Universita' degli Studi di Napoli, Italy, in 1999. She held research positions at Princeton University, at the Center for Wireless Communications, Oulu, Finland and at Universita' degli Studi del Sannio, Benevento, Italy. From 2002 she has joined the Faculty of the Universita' degli Studi di Napoli Federico II, and in 2009 she joined Bell Labs. From 2011, Dr. Tulino is Member of the Editorial Board of the IEEE Transactions on Information Theory and in 2013, she was elevated to IEEE Fellow. She has received several paper awards and among the others the 2009 Stephen O. Rice Prize in the Field of Communications Theory for the best paper published in the IEEE TRANSACTION ON COMMUNICATION in 2008. She has been principal investigator of several research projects sponsored by the European Union and the Italian National Council, and was selected by the National Academy of Engineering for the Frontiers of Engineering program in 2013. Her research interests lie in the area of communication systems approached with the complementary tools provided by signal processing, information theory and random matrix theory.

PLACE PHOTO HERE

Dario Barone received the B.E. in Telecommunications Engineering from Universita' degli Studi di Napoli Federico II, Italy, in 2016. He developed his thesis working on this project where he makes his first contribution in research and development. He is currently studying for his Master of Telecommunications Engineering at the same university.

PLACE PHOTO HERE

Giuseppe Vettigli received a Master degree in Computer Science from the Universit Federico II di Napoli in 2014. He is now a PhD student in Computer Science in the same university.

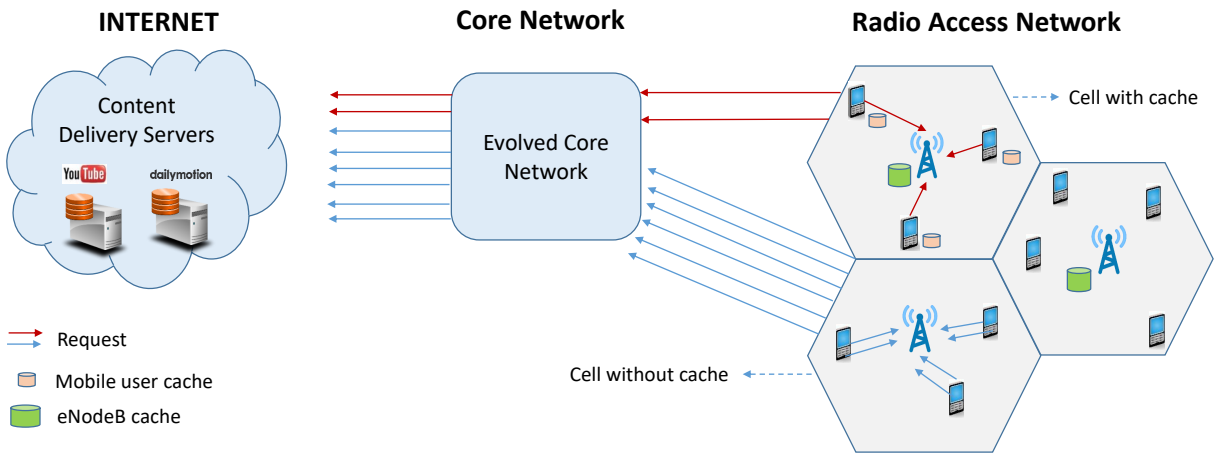

Fig. 1. Caching within the radio access network: impact on network load and traffic congestion.

PLACE PHOTO HERE

Jaime Llorca (S '03 – M '09) received the B.E. degree in Electrical Engineering from the Universidad Politecnica de Catalunya, Barcelona, Spain, in 2001, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Maryland, College Park, MD, USA, in 2003 and 2008, respectively. He held a post-doctoral position at the Center for Networking of Infrastructure Sensors (CNIS), College Park, MD, USA, from 2008 to 2010. He joined Nokia Bell Labs at Holmdel, NJ, USA, in 2010, where he is currently a Research Scientist in the Network Algorithms Group. His research interests include energy efficient networks, distributed cloud networking, content distribution, resource allocation, network information theory, and network optimization. He is a recipient of the 2007 Best Paper Award at the IEEE International Conference on Sensors, Sensor Networks and Information Processing (ISSNIP), the 2016 Best Paper Award at the IEEE International Conference on Communications (ICC), and the 2015 Jimmy H.C. Lin Award for Innovation.

PLACE PHOTO HERE

Jean-Marie Gorce (M'07, SM'14), is the director of the Telecommunications department of INSA Lyon and the holder of the SPIE ICS industrial chair on Internet of Things. He is a member of the scientific committee of the joint lab INRIA-Nokia Bell Labs. He received the PhD degree in Electrical Engineering from the National Institute of Applied Sciences (INSA), Lyon, France, in 1998. He held a research position at Bracco Research, S.A. Geneva and was recruited at INSA Lyon in 1999. Prof. JM Gorce was a co-founder of the CITI (Centre for Innovation in Telecommunications and Integration of Services) laboratory of whom he was the director from 2009 to 2014. He has been a researcher associated to INRIA since 2003 and he was visiting scholar at Princeton University from Sept. 2013 to Aug.2014. He was the principal investigator of several research projects sponsored by the French government or the European Union. He is an associate editor if the Eurasip journal of Wireless Communications and Networking (Springer). His research interests lie in wireless networking, focusing on realistic modeling, wireless system optimization and performance assessment considering both infrastructure-based and ad-hoc networks. He is the scientific coordinator of the experimental facility FIT-CorteXlab.

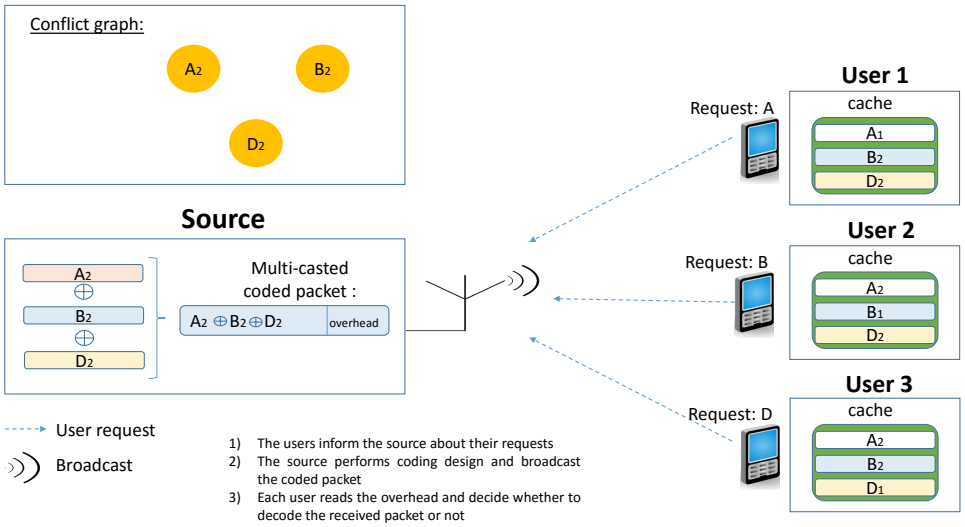

Fig. 2. Joint design of caching-aided coded multicasting in a 3-user SLN where each user is equipped with a storage capacity of 1.5 file and requests one file from a library with 4 binary files.

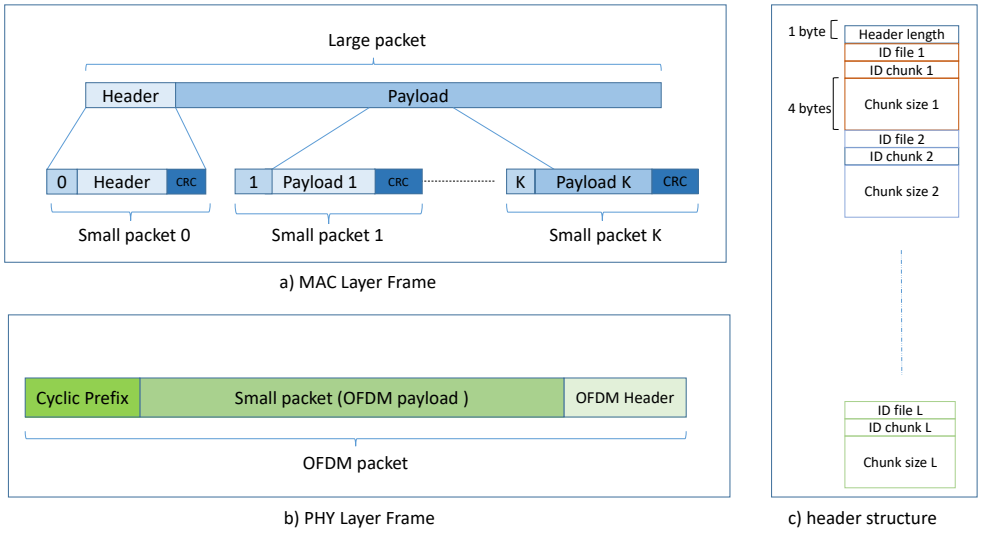

Fig. 3. An example of the proposed frame structure.

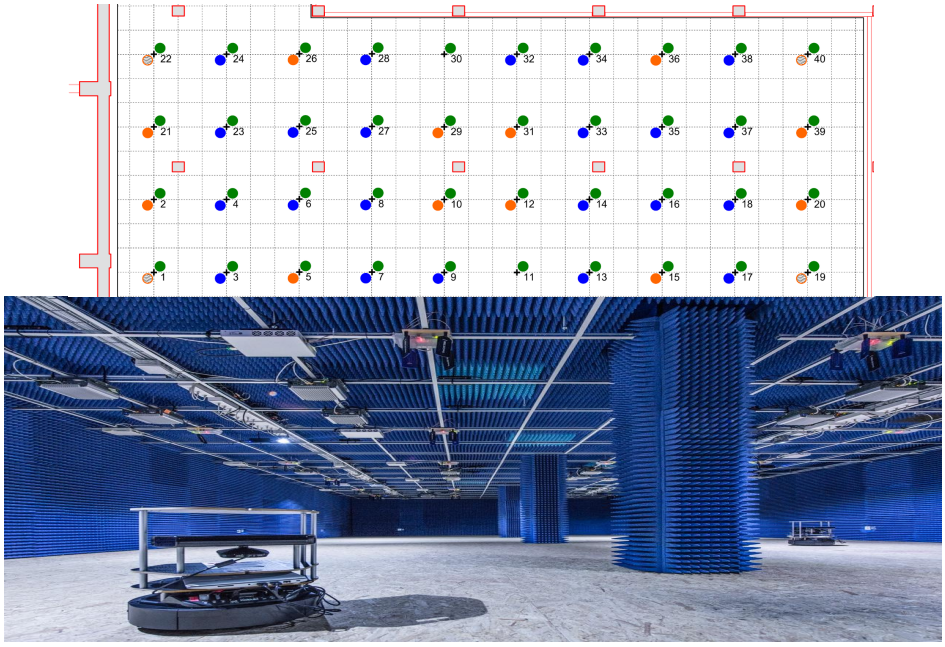

Fig. 4. CorteXlab platform and the nodes placement map.

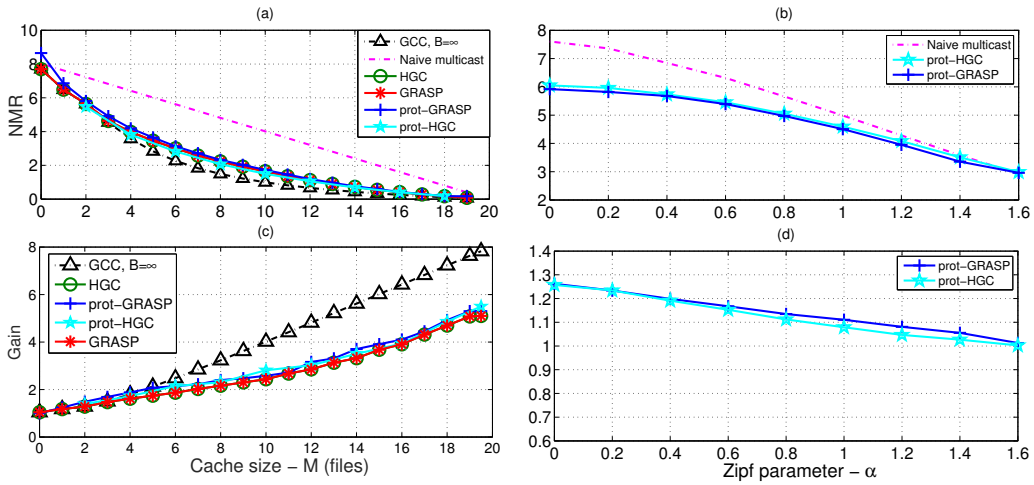

Fig. 5. Comparison of the total minimum rate normalized by the 1-user network minimum rate with respect to : a) the cache size with $\alpha = 0$, and b) the Zipf α parameter with $M = 2$ files, and the gain of the coded over the uncoded scheme with respect to: c) the cache size with $\alpha = 0$, and d) the Zipf α parameter with $M = 2$ files.

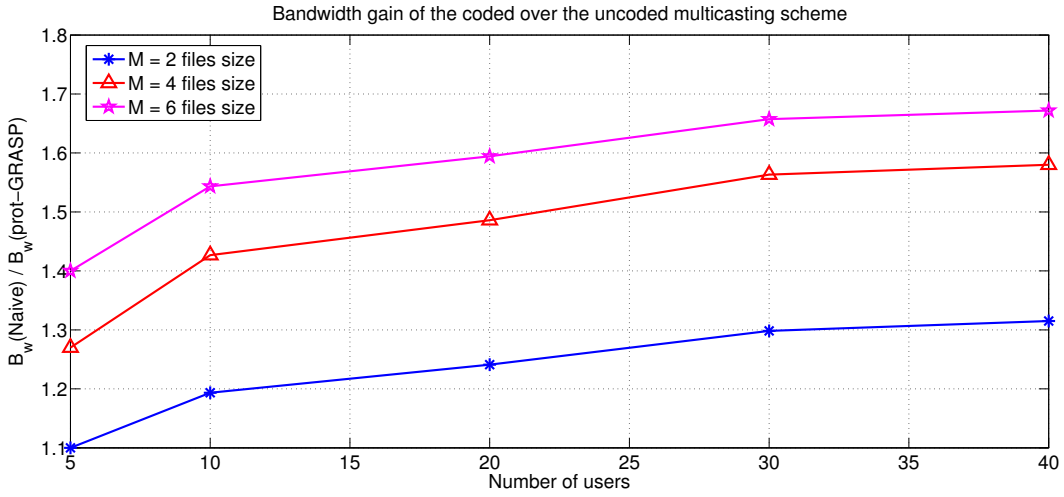

Fig. 6. The bandwidth gain of coded multicasting over naive multicasting for different memory cache sizes.