Mobility-Aware Edge Caching for Connected Cars

A. Mahmood†, C. Casetti†, C.F. Chiasserini†, P. Giaccone†, J. Härri‡
† Dept. Electronics and Telecommunications, Politecnico di Torino, Italy
‡EURECOM, 450 route des Chappes, 06410 Sophia Antipolis, France
E-mails:{casetti,chiasserini,giaccone}@polito.it, jerome.haerri@eurecom.fr

***Abstract*—Content caching on the edge of 5G networks is an emerging and critical feature to support the thirst for content of future connected cars. Yet, the compactization of 5G cells, the finite edge storage capacity and the need for content availability while driving motivate the need to develop smart edge caching strategies adapted to the mobility characteristics of connected cars. In this paper, we propose a Mobility-Aware Probabilistic (MAP) scheme, which optimally caches content at edge nodes where connected vehicles mostly require it. Unlike blind popularity decisions, the probabilistic caching used by MAP considers vehicular trajectory predictions as well as content service time by edge nodes. We evaluate our approach on realistic mobility datasets and against popularity-based edge approaches. Our MAP edge caching scheme provides up to 40% enhanced content availability, 70% increased cache throughput, and 40% reduced backhaul overhead compared to popularity-based strategies.**

I. Introduction

Connected cars are considered by drivers as a projection of their home on the road, and the same connectivity and content access services are expected. Accordingly, any future large-scale deployment of connected cars will require a significant redesign of the architecture of communication networks in order to support the required connectivity and capacity for cloud-based content and applications. Content providers are gradually migrating their content items from the cloud to the edge of communication networks to bring them as close as possible to connected cars in order to reduce delay and network overhead.

A major limitation of this approach is that edge nodes do not have the same storage flexibility as the cloud, and efficient strategies have to be developed to store the right content at an Edge Node (EN) (e.g., cellular base station, AP, roadside unit) required by the users under its coverage. Moreover, the thirst for wireless capacity has led to a reduced coverage size of ENs, which requires content to be replicated in multiple ENs to sustain user demands. The design of caching policies has therefore been widely investigated in the past [1], [2], [3], [4], [5] since caching is considered one of the most efficient ways to decrease the access delay to the content and to decrease the congestion in the network. Indeed, effects are beneficial both for the users, who have a better quality of experience, and the network operator, which can better exploit network resources.

Yet, connected cars add further challenges to edge caching strategies. They are, after all, highly-mobile vehicles and such mobility requires storage strategies to be optimized not to the current content popularity, but, instead, to the expected content popularity among future users about to enter the coverage of ENs. Furthermore, the dynamics of connected cars augmented by the limited coverage size of ENs require content to be stored where connected cars have a chance to actually download it, say in slow-motion areas, such as congested intersections. All these aspects create a challenging triumvirate for edge caching for connected cars: low coverage, low storage capacity and high mobility. Tackling such a triumvirate requires edge caching strategies to be adapted to car mobility and connectivity.

In the past, mobility-based caching policies have been investigated, such as in [6], [7], but they require a full knowledge of the trajectory of each car, rising concerns about drivers' privacy. Also, most of caching policies (as the ones in [1], [8], [9]) are not tailored to in-sequence delivery of content, typically required by future on-board streaming applications. A major design challenge for caching policies is therefore to rely only on coarse mobility information (sequences of waypoints, dwell time, etc.), while supporting in-sequence content delivery.

In this paper, we propose a Mobility-Aware Probabilistic (MAP) edge caching strategy specifically adapted to highly dynamic environments with a coarse knowledge of car trajectories. Our approach is based on the knowledge of the sequence of travel waypoints, and some aggregate statistics about the distribution of the dwell time under each EN. Note that this is in accordance with the current trend in 5G systems [10], which foresees a dynamic caching system to proactively store content in the ENs, based on future demand estimation obtained by user's context information such as direction and speed. With respect to classical works on caching, the peculiarities of the scenario addressed in our work are two. First, we consider a data streaming application in which the content is divided into fixed size chunks, thus the download process is strongly correlated at each EN, due to the in-sequence chunk delivery. Second, mobility introduces another level of correlation in the request process among different ENs. We remark that the instantaneous popularity of a chunk at an EN depends on the actual temporal and spatial trajectory of all the cars interested in the corresponding content. The goal of our strategy is to cache in advance the chunks in the sequence of ENs traversed by the car, by choosing the chunks that will be most likely downloaded at each specific EN. Thus our solution maximizes the cache hit probability, with beneficial effects on the backhaul traffic and content access delay.

Our contributions are manyfold: i) we introduce a split content caching architecture, with an Area Controller (AC) located in the backhaul and content caches located on ENs; ii) we describe a simple analytical model capable of predicting the probability for content to be required at specific ENs; iii) we leverage the previous model to develop a mobility-aware edge caching strategy; iv) we evaluate our solution under a realistic

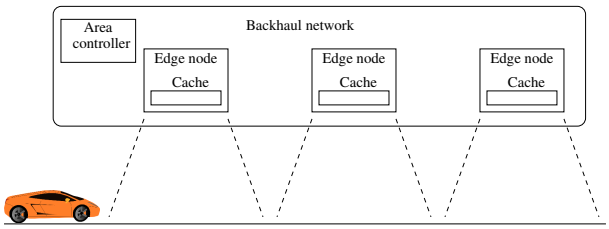

Fig. 1: Caching scheme for a car traversing 3 edge nodes.

urban traffic dataset of the city of Bologna. Our proposed MAP architecture yields a 30% improvement in hit probability and overall throughput, and reduces the backhaul traffic by 35%, compared to pure popularity-based caching.

The rest of the paper is organized as follows. In Section II, we describe our scenario and edge caching system. Section III introduces the proposed MAP edge caching strategy, while Section IV provides performance evaluations of MAP. We discuss related work in Section V. Finally, in Section VI we conclude the paper and shed light on future directions in edge caching for connected cars.

II. System scenario

We consider an urban environment where cars can connect to Edge Nodes (ENs) in order to download contents. ENs are equipped with caches where content can be stored so as to ensure a swift service to passing-by car users. In particular, in this work we focus on streaming content applications run by the car user for which in-order packet delivery should be ensured. We therefore consider that each content item is composed of K chunks, which, for the sake of simplicity, we assume to be of equal size. We also assume each chunk to be identified by a sequence number, k, with $1 \leq k \leq K$. For simplicity, each car is supposed to receive a single data stream.

As shown in Fig. 1, a group of nearby ENs are controlled by an Area Controller (AC), which resides in the network backhaul. We assume that the AC knows or can predict the ENs that a car will traverse in the next few minutes, and knows the distribution of the dwell time under each of them (e.g., based on past measurements). The AC is also aware of the available room in the caches of ENs and of content requests made by the passing-by users. Based on such information, the AC can define *which chunks* of *which content* each EN should cache and instruct ENs accordingly.

Upon entering the coverage area of an EN, the streaming application of a car user requests a new batch of chunks, indicating the content item it is receiving and the chunk number s from which it expects the streaming to resume. The EN checks whether it caches chunk number s or it needs to download it from the backhaul, an operation which is obviously costly in terms of bandwidth and latency. When the chunk is delivered, the EN proceeds to send the next chunk, resorting once more to the backhaul if its own cache does not store it.

We are interested in establishing an efficient strategy to store content chunks at ENs so as to ensure that cars receive prompt, high-throughput content transfers and that the requested data is fetched directly from ENs with high probability, rather than from the backhaul.

III. Mobility-Aware Probabilistic (MAP) caching

As a first step, we theoretically evaluate the probability that a specific chunk is downloaded from an EN by a tagged car. We then leverage this probability to define our caching scheme. Indeed, our approach consists in letting each EN store those chunks whose probability to be downloaded by a car is above a given threshold.

A. Chunk download probability

Let us focus on one car and one specific piece of content that the vehicular user wishes to download from the ENs it will pass by. We define $\mathcal{E}$ as the set of ENs that should serve vehicular user and $|\mathcal{E}|$ its cardinality. We also consider that $\mathcal{E}$ is ordered according to which EN will be visited first by the tagged car.

We start by defining the chunk delivery process under the idealistic assumption that all ENs can cache the whole content, i.e., any chunk can be available at any EN $\in \mathcal{E}$. This assumption of unlimited cache size is aimed at devising a simple model that will be tailored to finite cache sizes in Section III-C.

Let Y_i be the random variable representing the last chunk received from EN $i \geq 1$, and let $Y_0 = 0$ by definition. Then the set of chunks downloaded from EN i is given by: $\{k | k \in (Y_{i-1}, Y_i]\}$. Thus the probability that chunk k is downloaded from EN $i \geq 1$ is, for any $k \in \{1, \ldots, K\}$,

$$\phi_i(k) = \mathbb{P}\left(k \in (Y_{i-1}, Y_i]\right) = \mathbb{P}\left(Y_i \geq k \ \wedge \ Y_{i-1} < k\right) \quad (1)$$

Let X_i be the random variable representing the total number of chunks downloaded from EN i by the tagged car. The probability density function (pdf) of X_i depends mainly on two factors: (1) the mobility of the car, since, e.g., longer dwell times under the EN coverage typically imply larger amounts of download data, and (2) the actual throughput obtained by the vehicular user when connected to the EN, which in its turn depends on the wireless data rate and on channel contention. In Section IV-A, we will describe how to compute the pdf of X_i in the reference scenario under study.

Based on our definitions, it is easy to see that for $i \geq 1$,

$$Y_i = Y_{i-1} + X_i = \sum_{j=1}^{i} X_j \, . \quad (2)$$

The following theorem relates the download probability $\phi_i(k)$ to X_i.

Theorem 1: Given a car traversing a sequence of ENs, $\mathcal{E}$, the probability to download a specific chunk k from EN i, with $1 \leq i \leq |\mathcal{E}|$, can be expressed as

$$\phi_i(k) = \sum_{n=1}^{k-1} \mathbb{P}\left(X_i \geq k - n | Y_{i-1} = n\right) \mathbb{P}\left(Y_{i-1} = n\right) \, . \quad (3)$$

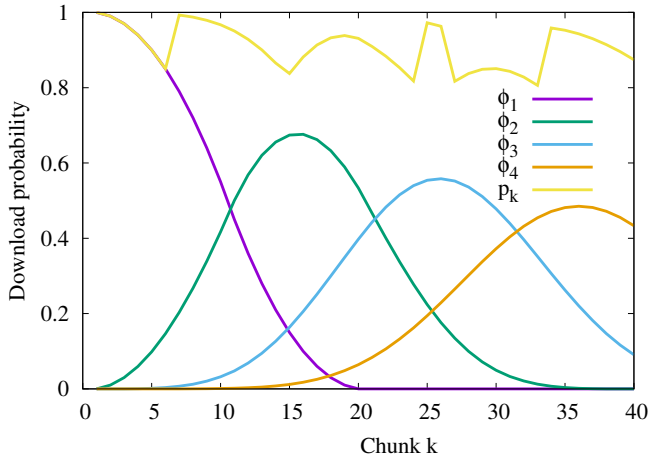

Fig. 2: Download probability for each EN and overall download probability, p_k, for the MAP caching policy, given a toy example scenario with $\tau = 0.8$ and $\mathbb{E}[X_i] = 10$ chunks.

Proof: Given (2), we can write (1), for any $i \geq 1$, as:

$$\begin{aligned}\phi_i(k) &= \mathbb{P}\left(Y_{i-1} + X_i \geq k \ \wedge \ Y_{i-1} < k\right) \\ &= \sum_{n=1}^{k-1} \mathbb{P}\left(X_i \geq k - Y_{i-1} | Y_{i-1} = n\right) \mathbb{P}\left(Y_{i-1} = n\right) \\ &= \sum_{n=1}^{k-1} \mathbb{P}\left(X_i \geq k - n | Y_{i-1} = n\right) \mathbb{P}\left(Y_{i-1} = n\right) .\end{aligned}$$

■

Note that, as expected, for $i = 1$ the expression in (3) becomes $\phi_1(k) = \mathbb{P}(X_1 \geq k)$ since the tagged car will download chunk k from the first EN only if the total amount of downloaded chunks from EN 1 is greater than k. Also, we remark that $\phi_i(k)$ is not a discrete probability density function. Indeed, thanks to the well-known property of the expectation of non-negative integer random variables, we have:

Property 1: $\sum_{k=1}^{K} \phi_i(k) = \mathbb{E}[X_i]$.

The following corollary holds in the special case when the car dwell times under the ENs are i.i.d. random variables. Note that this case is addressed here for completeness, as well as to provide a more explicit expression of the ϕ's, however our approach does not require such an assumption.

Corollary 1: Let the random variables X_i's be i.i.d. and defined on a positive support. Let $f_X(k)$ be their discrete pdf. Then, for any $i \geq 1$, we have:

$$\phi_i(k) = \left(f_X * \phi_{i-1}\right)(k) \tag{4}$$

where $*$ is the convolution operator.

The proof is reported in Appendix A.

B. Download probability in a toy scenario

To better clarify the behavior of the download probability at each EN, consider a toy scenario in which a car traverses four ENs. As an example, we assume i.i.d. X_i's with a symmetric triangular distribution and mean value equal to 10 chunks, i.e., the average number of chunks downloaded at each EN is 10. Fig. 2 shows the download probabilities $\phi_i(k)$ at each EN for each chunk k. From Fig. 2 we can observe that, at the first EN, $\phi_1(k)$ decreases as k increases since the randomness in the mobility reduces the probability of downloading farther chunks. Due to the limited support of the distribution of X_1, $\phi_1(k)$ becomes zero for $k \geq 20$ chunks. At the second EN, $\phi_2(k)$ is now bell-shaped, since values of k close to zero correspond to the case in which X_1 takes very small values (which is unlikely), i.e., the car speed is very high under the coverage of the first EN, hence the car does not have enough time to download any chunk. The maximum is obtained around 15, which is reasonable since in the case of deterministic mobility with $X_i = 10$ chunks for any i, the chunks to be downloaded would be exactly in the interval $[10, 20]$, which is symmetric around 15. The chunk download probability from the following ENs ($i > 2$) still exhibits a bell-shaped behavior, but with an expanded support. This is due to the larger uncertainty on downloading a tagged chunk from a given EN, which, in its turn, is due to the increased randomness in the number of previously delivered chunks.

C. Finite cache size

The above model can be refined to consider the actual room that will be available in the cache for the data that an EN should store. Let M_i be the maximum amount of data that can be stored in the cache of EN i for the considered content. Then we can introduce a discrete random variable, $\widehat{X}_i$, which represents the total number of cached chunks downloaded from EN i by the tagged car, *given the available room in the cache of EN* i. The pdf of $\widehat{X}_i$ is given by:

$$\mathbb{P}\left(\widehat{X}_i = x\right) = \begin{cases} \mathbb{P}(X_i = x) & \forall\, 0 \leq x < M_i \\ \mathbb{P}(X_i \geq x) & x = M_i \\ 0 & \forall\, x > M_i\,. \end{cases}$$

Indeed, it is not possible to download more than M_i cached chunks, and the events corresponding to a number of downloaded chunks larger than M_i in the original model with unlimited cache size, now correspond to downloading all the M_i available chunks. The tagged car can of course download more than M_i chunks from EN i, provided that the chunks in excess are actually fetched by EN i from the backhaul (i.e., they were not available in the cache).

Given the above distribution, the probabilities $\phi_i(k)$ can be computed as in (3), or as in (5) when $\widehat{X}_i$'s are i.i.d..

D. Caching scheme

We now define our caching scheme, i.e., for each chunk we determine which ENs should store it. The goal of our scheme is to ensure that the probability with which a tagged car can download a chunk from one of the EN caches is greater than a given threshold τ, so as to reduce the need to fetch chunks from the backhaul. For each chunk, our scheme thus identifies the *minimum* set of ENs that should cache it so that such probability exceeds threshold τ.

To this end, for each chunk k we first order the probabilities $\phi_i(k)$ in decreasing order. Then, chunk k should be stored at the EN i corresponding to the highest $\phi_i(k)$ value. If this

value is already greater than τ, no other EN should cache k. Otherwise, the EN corresponding to the second top value of $\phi_i(k)$ should store the chunk too. Eventually, chunk k will be cached at as many ENs, associated to the top $\phi_i(k)$ values, as necessary so that the sum of their $\phi_i(k)$ exceeds τ. The scheme is reported in Algorithm 1, which, for each chunk k, returns the set of ENs, $\mathcal{S}(k)$, that should store k.

Algorithm 1 MAP caching algorithm

Require: τ, $\{\phi_i(k)\}_{i,k}$
1: **for** $k = 1, \ldots, K$ **do**
2: $\mathcal{S}(k) = \emptyset$, $p_k = 0$
3: $\mathcal{F}(k) \leftarrow \{\phi_i(k)\}_i$
4: **while** $\mathcal{F}(k) \neq \emptyset$ and $p_k < \tau$ **do**
5: $p_{top} \leftarrow$ remove the highest value from $\mathcal{F}(k)$
6: $i_{top} \leftarrow$ EN index corresponding to p_{top} probability
7: $p_k = p_k + p_{top}$
8: $\mathcal{S}(k) = \mathcal{S}(k) \cup \{i_{top}\}$
9: **end while**
10: **end for**
11: **return** $\{\mathcal{S}(k)\}_k$

At the end of procedure, p_k represents the estimated download probability based on the actual number of copies for chunk k; by construction, we have $p_k \geq \tau$. In the example of Fig. 2, we show the p_k obtained in our toy scenario and assuming a simple triangular distribution for X_i. Note that the non-monotonic behavior is due to the different number of copies that are stored at the ENs for different chunks. Intuitively, as k increases, the uncertainty about the possibility to download chunk k increases, thus the MAP caching algorithm compensates by creating a higher number of copies. Any time the number of copies increases, we observe an upward spike in p_k.

Whenever a car enters the coverage area of an EN requesting a new content item, the EN contacts the AC. Based on the predicted car mobility, the controller determines the set $\mathcal{E}$ and then, using the above MAP algorithm, computes the set $\mathcal{S}(k)$ for each chunk of the requested content. It thus instructs the ENs in $\mathcal{E}$ accordingly, about which chunks they should store for the new user. As the car proceeds along its route, the AC predicts the new sets of ENs that will be traversed by the car and notifies them about the corresponding new $\mathcal{S}(k)$ (i.e., the chunks to store), until the content downloading is completed.

Next, consider the more general scenario where an EN should serve multiple cars. Since ENs have finite cache sizes, an eviction policy is needed to determine which chunks should be removed in the case where the cache is full and new chunks should be inserted. The chunks to be removed at higher priority are the ones that have been already delivered, i.e., the chunks destined to cars not anymore under coverage. Among these chunks, we evict at higher priority the chunks with the lowest download probability. In this way, our caching policy is able to exploit the cache under both space and time varying chunk demands.

IV. Performance Evaluation

We now introduce the scenario and real-world vehicular traces that we have used to assess the performance of our solution. Then we compare the MAP scheme against a popularity-based solution in terms of some relevant performance metrics.

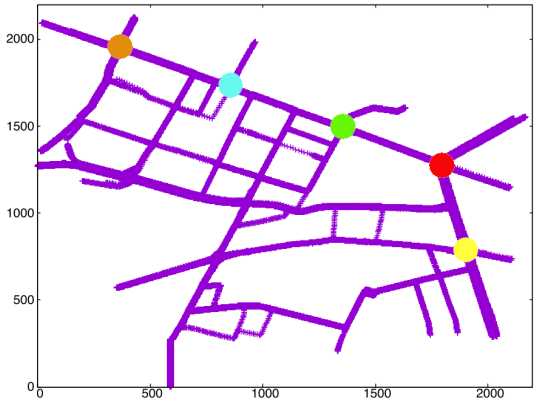

Fig. 3: Reference scenario: road topology in the city of Bologna. Circles represent RSU locations. Distances are expressed in meters.

TABLE I: Number of vehicles stopping on red light and going through green light at each RSU

RSU1		RSU2		RSU3		RSU4		RSU5	
Green	Red	Green	Red	Green	Red	Green	Red	Green	Red
423	1036	1084	704	281	323	137	274	18	118

TABLE II: Average number of users under each RSU

RSU1	RSU2	RSU3	RSU4	RSU5
23.75	15.26	7.28	6.94	3.18

A. Reference scenario

We consider an urban environment where Internet connectivity to cars is provided by WAVE roadside units (RSUs). Each RSU is equipped with a cache and acts as an EN. In order to represent real-world conditions, we take as reference scenario a 2 km $\times$ 2 km urban section of the Italian city of Bologna, illustrated in Fig. 3. The vehicular mobility traces correspond to a dataset adjusted to real traffic by Bieker et al. [11], using real Origin-Destination matrices and traffic detectors at intersections. The total trace duration is 78.6 minutes comprising 11,079 vehicles (approximately, 950 are simultaneously on the map) and representing 120 minutes of the morning rush hours, under quite stationary traffic conditions.

In this scenario, we select the east-north corridor, corresponding to a major traffic artery of this section of Bologna, and place five RSUs along it (represented as circles in Fig. 3). All RSUs are placed in correspondence of intersections regulated by a traffic light and have the same cache size and radio range. Since our objective is to have a sequence of waypoints that are not fully connected given the reference scenario, we set the radio range to 100 m. The overall number of cars that were observed to enter the coverage of at least two RSUs were 2,199 and our investigation focuses on such subset of cars.

Given the vehicular traces, for each RSU we derived the pdfs of car dwell times, conditioned to the car stopping on

red light and to the car going through green light. We then used such pdfs to obtain the predicted dwell time at the AC, for a car under a given RSU. Table I reports the number of vehicles that, according to our mobility traces, stop on red light and go through green light at the intersections covered by the deployed RSUs. The average number of vehicles under each RSU is presented in Table II.

Furthermore, we consider that RSUs serve the car users under coverage by devoting the same amount of time to each of them. Indeed, according to the WAVE standard, RSUs adopt the IEEE 802.11p MAC protocol, which provides temporal fairness to traffic flows belonging to the same access category. Thus, in order to determine X_i for each connected car, we scale the car dwell time distribution dividing it by the average number of cars served by RSU i and multiplying it by the average data rate that a car experiences while being under coverage of RSU i. Finally, we evaluate $\widehat{X}_i$ to take into account the finite cache, according to the formula discussed in Section III-C.

Figs. 4 and 5 depict the distribution corresponding to $\hat{X}_1$ and $\hat{X}_3$ in the considered traces, when the cache size is set equal to 600 chunks. For the first RSU, Fig. 4 shows that "red cars" (i.e., stopping on red light) and "green cars" (i.e., going through green light) can download, on average, around 200 chunks and 80 chunks, respectively. The difference between these values is clearly due to the different average speed of the cars when under coverage. Observing the average number of cars in Table II, RSU 1 is expected to be located at a very congested intersection. Indeed, only very few red cars experience a large enough dwell time and a small channel contention that would allow them to download all the 600 chunks stored in the cache. RSU 3, instead, appears to be located at a much less congested intersection, according to Table II. This implies a lower channel contention, hence, a higher download capability, as shown in Fig. 5. Indeed, under RSU 3, green cars are able to download at least 140 chunks, with an average around 200, whereas most of the red cars download around 600 chunks, i.e., the maximum number of chunks available in the cache. For the sake of brevity, we omit the distribution of the number of chunks downloaded from the other RSUs, but we mention that $\widehat{X}_2$ behaves very similarly to $\widehat{X}_1$, while $\widehat{X}_4$ and $\widehat{X}_5$ are very similar to $\widehat{X}_3$. This is due to their similar levels of congestion, coherently with the values reported in Table II.

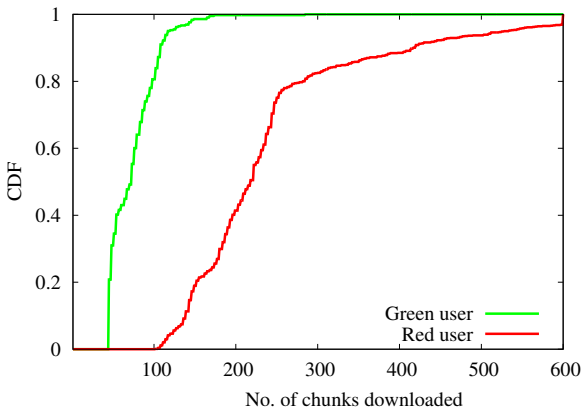

Fig. 4: Cumulative density function of the number of downloaded chunks $\widehat{X}_1$ measured in the trace, based on the red/green classification at RSU 1.

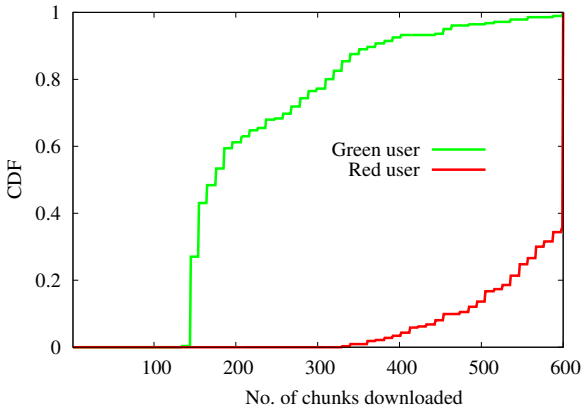

Fig. 5: Cumulative density function of the number of downloaded chunks $\widehat{X}_3$ measured in the trace, based on the red/green classification at RSU 3.

B. Methodology

We have investigated the performance of our MAP caching scheme by simulating the vehicular traffic according to the mobility trace of Bologna. In our discrete-event simulator, developed using OMNeT++ [12] libraries, we have modeled the backhaul network with fixed propagation delays to access the content server, whereas the wireless access network has been modeled in terms of communication data rates and channel contention. RSUs use IEEE 802.11p at the MAC and physical layer, and operate on a 10 MHz-wide frequency channel.

Regarding the content request process, each time a new car enters the coverage of an RSU for the first time, it generates a request for a content item. We assume that such item is chosen at random according to a Zipf's distribution with exponent $\alpha = 0.5$. The size of content item is 600 chunks, with each chunk being 10 kbytes large. The normalized cache size is defined as the cache size divided by the total size of all content items in the catalog.

In our simulations, we compare the performance of the MAP caching scheme to that of the classical policy (denoted by POP in the following). According to POP, the cache stores the most popular content items. In our streaming scenario, where content items are divided into chunks, we assume that the POP policy stores the chunks of the most popular item in sequence; thus, if no room is available for the whole item, only the first chunks of it are stored until the cache is full. We have considered the POP policy since, under a stationary content request process feeding a single cache, it is well known to be the optimal in terms of hit probability.

The performance metrics we consider are as follows:

- cache throughput: amount of data received by the car and directly downloaded from the cache, i.e., in the

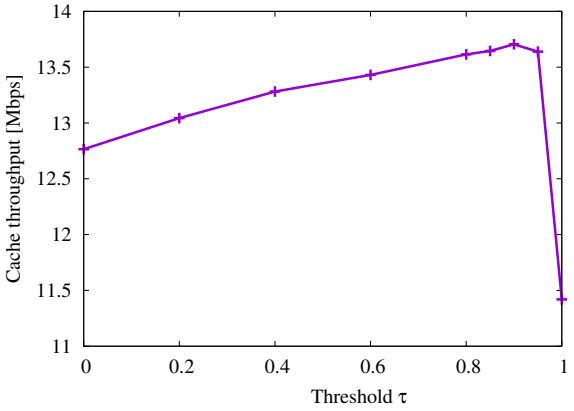

Fig. 6: Cache throughput vs. threshold τ.

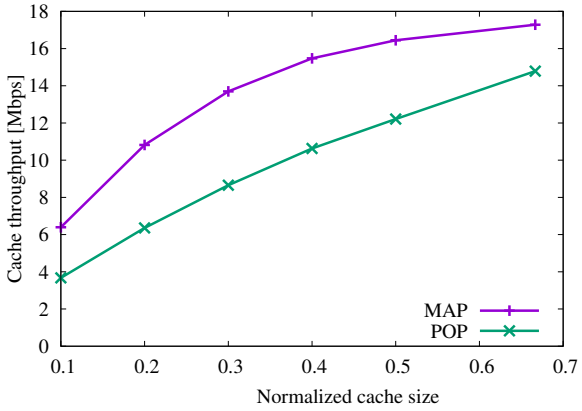

Fig. 7: Cache throughput.

event of a cache hit;

- cache hit probability: fraction of chunk requests that are satisfied by directly downloading the chunk from the cache;
- backhaul bandwidth: amount of data downloaded from the server in the backhaul per time unit, i.e., in the event of a cache miss;
- total throughput: total amount of data received by cars per time unit, obtained as the sum of cache throughput and backhaul bandwidth.

C. Simulation results

As a preliminary step, we have investigated the effect of the threshold τ adopted in the definition of our MAP policy. Fig. 6 shows the average cache throughput for different values of τ given a cache size equal to 1800 chunks. Intuitively, one would expect that, in a cache system with infinite cache size, large values of τ should imply that every chunk is stored across all caches. However, the finite size of caches limits the effectiveness of this approach (almost oblivious of car mobility). Thus, when τ is close to one, MAP, which is very sensitive of the mobility, instructs the RSUs to store just few copies for the first chunks (for which the effect of mobility can be estimated with higher confidence), and an increasing number of copies for the remaining chunks, so as to satisfy the minimum download probability τ. From Fig. 6, it can be seen that in our scenario the best value of τ is equal to 0.9; this value appears to be optimal also for all the other cache sizes we considered in our simulations. Thus, we use this value in all of the following results.

Fig. 7 depicts the cache throughput as a function of the normalized cache size, for the POP and the MAP caching schemes. As expected, larger values of cache size improve the performance of both caching schemes. However, MAP outperforms POP by 15% up to 70%, depending on the cache size. This is due to the higher effectiveness of the MAP policy, which tends to store chunks only in those RSUs from where they can be downloaded with high probability, taking into account the channel contention and the dwell time statistics. Indeed, Fig. 8 shows that the cache hit probability on the cache

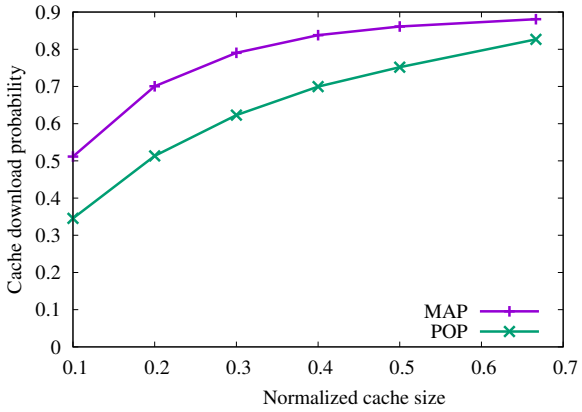

Fig. 8: Cache hit probability.

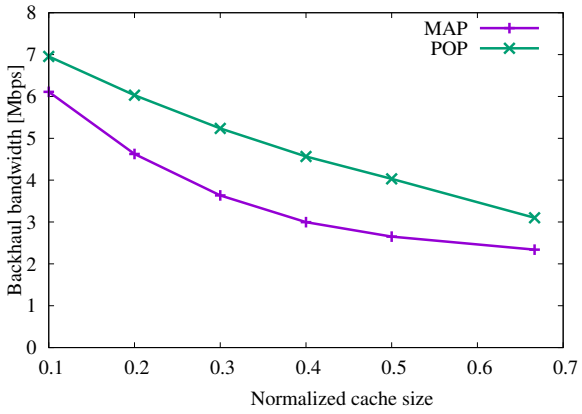

Fig. 9: Backhaul bandwidth.

for MAP is 30%-40% higher than POP, which is oblivious of mobility, for most of the values of cache size.

The performance gain provided by MAP over POP can be observed also in terms of backhaul bandwidth, since the higher hit probability of MAP implies a lower probability to access the server and retrieve the content from there. Fig. 9 shows that the reduction of the used backhaul bandwidth obtained

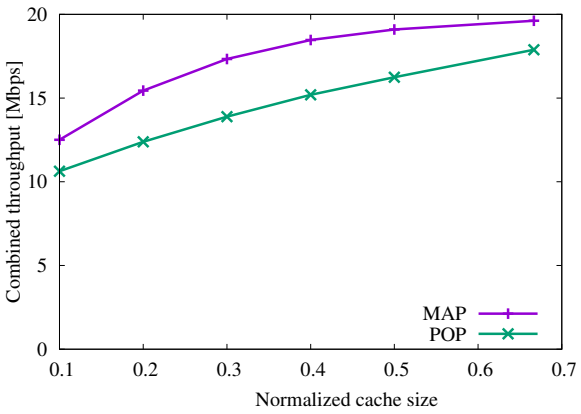

Fig. 10: Total throughput.

when MAP is applied is between 25% and 40%. Clearly, such reduction leads to a lower backhaul congestion. Finally, Fig. 10 presents the overall system throughput, accounting for both the chunks downloaded from a cache and from the backhaul. This value must be compared with the maximum throughput achievable in the overall network, which is equal to 28.7 Mbit/s (i.e., around 5.7 Mbit/s, which is the average data rate we observed at the application layer, times 5 RSUs). This value would be achieved for infinite cache sizes, independently of content popularity or car mobility.

We remark that, while deriving our results, we considered that the AC has just a rough estimate of car mobility, since it predicts only whether a car will stop on red or will go through green light, and then leverages the pdf of the dwell time under each RSU conditioned to the status of the traffic light. Still, MAP significantly outperforms the popularity-based solution. We expect that greater improvements can be obtained if a more accurate mobility prediction is available at the AC.

V. Related work

In the context of cellular networks, [8] devises an optimal geographic caching assuming that a user is covered by multiple base stations; this case is different from our work, since we do not consider overlapping coverage areas among ENs. Furthermore, [8] proposes an optimal probabilistic content placement policy that maximizes the total hit probability for random network topologies, based on content popularity. Thus, the policy there is oblivious of the actual mobility pattern of users, differently from our MAP policy. In a hybrid scenario comprising MANET and cellular networks, [9] proposes an optimal caching and routing policies. Each node estimates locally the content popularity and stores the content in the cache based on its popularity. This scheme can be considered as a distributed implementation of the POP policy that we use for comparison in our work.

For the specific case of cellular backhaul networks, [13] investigates the effect of different criteria to identify the web content adopted when accessing the caching system. The main idea is to avoid duplicated content items in the caches, since those same items could appear with different identifiers at application level. In our scenario, instead, we specifically consider the streaming of content through a chunk-based approach, for which we assume that each chunk and each content item are univocally identified.

We remark that all the above cited caching schemes are oblivious of user mobility. Instead, we assume to know the sequence of waypoints of the temporal and spatial trajectory followed by the cars. This information can be deduced from car navigation systems or it can be easily predicted. For example, studies such as [14] and [15] suggest that people usually drive on familiar routes (drive to work, school, etc.) and this can be exploited to develop quite accurate prediction models. On this regard, [16] proposes a mobility prediction scheme, based on the previous history of users, which improves content distribution in vehicular networks through simpler handover procedures.

Few works have investigated caching schemes specifically taking into account user mobility. The authors in [6] consider a scenario very similar to ours, based on an architecture denoted as “MobilityFirst”, introduced in [17], which ensures seamless mobile content delivery when users move across the network. Thanks to a global identifier associated to each user, the user mobility is recorded at each node. In terms of caching, each node is equipped with two distinct buffers. The first one caches the most popular content items, exactly as the POP policy considered in our work. The second buffer is instead devoted to store the content based on a prefetching policy leveraging the predicted sequence of nodes traversed by each particular user. A similar prefetching policy is proposed by [7] in a cellular network scenario. Similarly to our work, the content is delivered to users by base stations using a chunk-based approach. The specific mobility of each user is considered in order to identify the chunks to prefetch in the caches along the user path. Unlike our work, however, both [6] and [7] assume that the caching policy knows or predicts the spatial and temporal trajectory of each user, in order to estimate the time intervals in which the user will be covered by each base station. Our approach instead requires to know just the distribution of the dwell times under each edge node at aggregate level. This distribution can be estimated locally by each EN and does not require at all the precise knowledge of the car trajectory: only the sequence of ENs is needed. This simplifies the prediction process and goes a long way toward preserving the privacy of users.

VI. Conclusions

In this paper we have studied the problem of efficiently providing connected cars with streaming data as they drive along a road covered by wireless edge nodes. Using a Mobility-Aware Probabilistic (MAP) edge caching strategy, we let a central controller determine the content of edge node caches by predicting the probability for content to be required at each edge node. The controller can base such prediction upon the achievable data rates and the distribution of dwell times of cars under the coverage of edge nodes. Our scheme was designed to allow an operator to select a minimum set of edge nodes that should cache the content achieving a desired delivery probability without having to fetch it from the backhaul. We have tested the MAP strategy with real traces from the city of Bologna dataset and registered significant improvements in content availability, throughput and backhaul overhead with respect to popularity-based strategies.

ACKNOWLEDGMENTS

This work has been partially funded by the EU H2020 5GCrosshaul project (grant no. 671598), and the EU H2020 HIGHTS project (grant no. 636537). EURECOM acknowledges the support of its industrial members, namely, BMW Group Research and Technology, IABG, Monaco Telecom, Orange, SAP, ST Microelectronics, and Symantec.

REFERENCES

[1] C. C. M. Fiore, C. Casetti, "Caching strategies based on information density estimation in wireless ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 5, pp. 2194–2208, 2014.

[2] N. L. Scouarnec, C. Neumann, and G. Straub, "Cache policies for cloud-based systems: To keep or not to keep," *IEEE CLOUD*, 2014.

[3] A. G. Li Zhe, G. Simon, "Caching policies for in-network caching," *IEEE ICCCN*, 2012.

[4] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MPC: Popularity-based caching strategy for content centric networks," *IEEE International Conference on Communications (ICC)*, 2012.

[5] S. Tewari and L. Kleinrock, "Proportional replication in peer-to-peer networks," *IEEE INFOCOM*, 2006.

[6] F. Zhang, C. Xu, Y. Zhang, K. Ramakrishnan, S. Mukherjee, R. Yates, and T. Nguyen, "EdgeBuffer: Caching and prefetching content at the edge in the MobilityFirst future internet architecture," in *WoWMoM*. IEEE, 2015, pp. 1–9.

[7] S. K. Dandapat, S. Pradhan, N. Ganguly, and R. Roy Choudhury, "Sprinkler: distributed content storage for just-in-time streaming," in *Proceeding of the 2013 workshop on Cellular networks: operations, challenges, and future design*. ACM, 2013, pp. 19–24.

[8] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *IEEE ICC*, June 2015, pp. 3358–3363.

[9] M. Dehghan, A. Seetharamz, T. He, T. Salonidis, J. Kurose, and D. Towsley, "Optimal caching and routing in hybrid networks," in *Military Communications Conference (MILCOM)*. IEEE, 2014, pp. 1072–1078.

[10] "5G: Challenges, research priorities, and recommendations," Joint white paper, NetWorld 2000, Sept. 2014. [Online]. Available: http://networld2020.eu/wp-content/uploads/2015/01/Joint-Whitepaper-V12-clean-after-consultation.pdf

[11] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic simulation for all: a real world traffic scenario from the city of Bologna," in *Modeling Mobility with Open Data*. Springer, 2015, pp. 47–60.

[12] "OMNet++ discrete event simulator." [Online]. Available: http://www.omnetpp.org/

[13] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 319–332.

[14] J. Froehlich and J. Krumm, "Route prediction from trip observations," SAE Technical Paper, Tech. Rep., 2008.

[15] J. Krumm, "A Markov model for driver turn prediction," SAE Technical Paper, Tech. Rep., 2008.

[16] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das, "Predictive methods for improved vehicular WiFi access," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 263–276.

[17] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst: a robust and trustworthy mobility-centric architecture for the future Internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

APPENDIX A
PROOF OF COROLLARY 1

Proof: When X_i's are i.i.d.,

$$\phi_i(k) = \sum_{n=1}^{k-1} \mathbb{P}\left(X \geq k-n\right) \mathbb{P}\left(Y_{i-1} = n\right).$$

Thus, (3) can be rewritten as:

$$\begin{aligned}
\phi_i(k) = & \sum_{n=1}^{k-1} \mathbb{P}\left(X \geq k-n\right) \sum_{t=1}^{n} \mathbb{P}\left(X = t | Y_{i-2} = n-t\right) \cdot \\
& \mathbb{P}\left(Y_{i-2} = n-t\right)) \\
\overset{z=n-t}{=} & \sum_{z=1}^{k-1} \mathbb{P}\left(X \geq (k-t)-z\right) \cdot \\
& \sum_{t=1}^{k-1} \mathbb{P}\left(X = t | Y_{i-2} = z\right) \mathbb{P}\left(Y_{i-2} = z\right) \\
= & \sum_{t=1}^{k-1} \mathbb{P}\left(X = t\right) \cdot \\
& \sum_{z=1}^{k-1} \mathbb{P}\left(X \geq (k-t)-z\right) \mathbb{P}\left(Y_{i-2} = z\right) \\
= & \left(f_X * \phi_{i-1}\right)(k). \qquad (5)
\end{aligned}$$

■