PCach: The Case for Pre-Caching your Mobile Data

Katia Jaffrès-Runser and Gentian Jakllari
Institut de Recherche en Informatique de Toulouse
University of Toulouse, Toulouse INP, Toulouse, France
{kjr, jakllari}- at - enseeiht.fr

***Abstract*—We present PCach, a smartphone application designed to relieve the congestion in cellular networks resulting from the exponential growth of mobile data traffic. The basic idea underlying PCach is simple: use Wi-Fi to proactively cache content on the smartphone's memory, which otherwise would have been delivered through the cellular network during the next Wi-Fi coverage time gap. However, it leads to several challenging questions, including how much mobile data actually flows through cellular networks, how much data can be pre-cached, and when and what to pre-cache. By analysing the extensive MACACO measurement dataset, our analysis shows that the median smartphone user transfers 15% of her data via the cellular network and that up to 80% of it could be pre-cached via Wi-Fi. From our empirical observations, we introduce an algorithm that can run stand-alone on off-the-shelf smartphones and predict with good accuracy when and what to pre-cache.**

I. Introduction

Recently, we marked the 10th anniversary of the launch of the first iPhone, which sparked the smartphone revolution and has had a big impact on how we generate and consume content. It has been a boon for cellular providers but also an extreme challenge. Cisco projects that by 2021 [2], 5G will represent only 0.2 percent of connections and 1.5 percent of total traffic. Solutions that can be deployed immediately and serve as bridge to the 5G roll-out are sorely needed. To this end, we introduce PCach, a user-centric approach that can be deployed on off-the-shelf smartphones by simply downloading an application and help relieve congestion in cellular networks. To accomplish this, PCach uses Wi-Fi connectivity to proactively cache (pre-cache) content users are likely to need in the immediate future and otherwise would have downloaded through the cellular connection. The concept of pre-caching (or prefetching) has been introduced in the early ages of web browsing to gain in user-perceived download time. The key idea is to identify the webpage the user is highly likely to request in the near future to start its download before the actual download request takes place [7]. In our case, we have to predict both the dates at which Wi-Fi stops and resumes, and the data the user is expected to need during Wi-Fi outage.

The concept of offloading content from a metered network (e.g. cellular) to an un-metered network (e.g. Wi-Fi) has been thoroughly studied in the past. The potential of offloading the content generated by the mobile (i.e. in upload) has been investigated in [6]. Authors show that offloading delay-tolerant data is particularly beneficial for the overall system performance. Contrary to the work in [6], we don't postpone the transfer of 3G data to the Wi-Fi resume date but study the solution where 3G data is prefetched before Wi-Fi outage happens. In the context of vehicular networks where Wi-Fi coverage is scarcer and often of lower quality, authors of [1] propose to predict Wi-Fi availability and decide whether to transmit data on cellular or Wi-Fi depending on flow timing constraints. We address as well in this work the problem of predicting Wi-Fi coverage, but with a finer granularity as we have to predict the Wi-Fi cut and resume dates.

Our main contributions may be summarized as follows:

- Using a dataset crowdsourced from a multi-year, multi-country deployment of MACACO-app (section II), we establish a case for pre-caching (section III) by showing that i) a significant amount of mobile traffic is delivered through the cellular network, ii) there are non-trivial gaps in the Wi-Fi connectivity.
- We introduce PCach (section IV), a user-centric approach that can run as an ordinary app on off-the-shelf smartphones and pre-cache via Wi-Fi content that otherwise would have been delivered through the cellular network.

II. The crowdsourced MACACO data

To make the case for pre-caching data on smartphones, we have designed and deployed MACACO-app, a mobile Android application to crowdsource fine-grained statistics on networking content and context. It has been deployed on a set of 45 smartphones for an extended period of time, offering a rich dataset on which we base our analysis. The architecture of MACACO-app is described in [5].

A. Collected data

MACACO-app collects the information listed in [5] every 5 minutes, grouped into *context and content* features:

a) Context: defines in our case the mobile user's environment. For this study, we leverage the following context items: the list of visible Wi-Fi networks; whether the smartphone's active network is Wi-Fi or cellular; and the name of the Wi-Fi network the user is currently connected to if Wi-Fi.

b) Content: relates to the nature of the data that is either pushed onto or pulled from the Internet by the smartphone user. As we do not want to break Android's privacy rules, we capture for each measurement the following content features: the list of applications currently running; the volume of data uploaded per application to the Internet; and the volume of data downloaded per application.

B. Dataset

MACACO-app has been installed on 162 devices between July 2014 and July 2017, creating a dataset of 2.64 million measurement samples, representing 220k hours of measurements. Volunteers originate from 5 different countries of different profiles, including students, full-time employees in academia and industry. References [4], [5] offer detailed statistics on the data collection periods of all phones, of the top 20 applications in terms of download volume (top) and upload volume (bottom).

1) Data used in this study: For this study, 45 smartphones totaling at least 5000 measurements (i.e. ∼17 days) since May 2015 have been selected, representing a total of 1.635.641 measurements (∼136k) hours of measurements. Volunteers have joined and left, but over 40% of the phones reported data for over 100 days, with some doing so for over 500 days.

2) Cellular and Wi-Fi Traffic: The total traffic generated by 33 smartphones during the data collection period is plotted in Fig. 1[1]. Traffic volume is split between traffic delivered through cellular and Wi-Fi. Both upload and download traffic are merged in these statistics. Here, the download volume is 4.26 times the upload volume. Interestingly, Wi-Fi traffic is an order of magnitude higher than the cellular traffic.

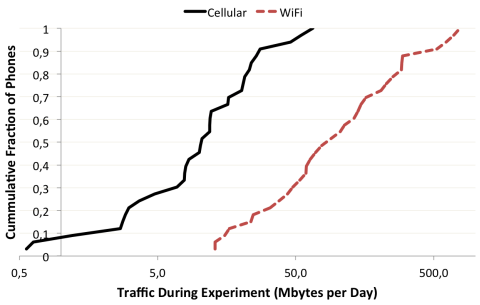

Fig. 1. Distribution of cellular and Wi-Fi traffic in Mbytes/day for 33 phones.

III. Mining for PCach

In this section, using the dataset collected by MACACO-app, we address what we consider to be the key questions regarding the feasibility of pre-caching as a strategy for reducing peak-hour congestion in cellular networks. Namely, how much traffic is delivered through cellular networks, what are the gaps in Wi-Fi connectivity that would justify pre-caching, and finally, what is the ceiling of pre-caching.

A. 3G Traffic and Wi-Fi gaps

A significant part of smartphone traffic already flows through Wi-Fi. Our measurements show that, while the proportion of cellular traffic is reduced compared to Wi-Fi, it still represents 15% of the total mobile data traffic. The challenge we address in this work is whether this 15% of cellular data can be pre-cached, and if yes, if is it worth the effort. Therefore, it is important to understand how Wi-Fi outage periods are experienced by MACACO participants.

[1]The traces of 12 phones have been discarded as these volunteers have generated too little Internet traffic

We define a *Wi-Fi gap* as the time period between a Wi-Fi cut event and a Wi-Fi resume event. A Wi-Fi cut event is identified on measurement sample x if: Wi-Fi is the active connection for sample $x-1$, cellular is the active connection for sample x and the time elapsed between x and $x-1$ is no longer than 10 minutes. Conversely, a Wi-Fi resume event is identified on measurement sample x if the previous sample had a cellular active connection while the current sample shows a Wi-Fi connection. users disabling MACACO-app for an extended time. The cumulative distribution of the Wi-Fi gap duration is given in [4]. Up to 80% of the Wi-Fi gaps last no more than one hour and a half while 65% of them last at most 30 minutes. By taking a closer look at the Wi-Fi cut and resume dates in [4], we notice a surge in cuts from 6:00 to 7:00 and from 15:00 to 16:30. These peaks in cuts are followed, around an hour later, by a surge in Wi-Fi resume events. These statistics underline that Wi-Fi disconnections are very likely to occur at commute times.

B. Pre-caching potential.

In this section, we establish a bound on the volume of 3G data that could be pre-cached knowing the geometry of Wi-Fi outage periods for the users in our dataset. To compute this bound, we first assume that we perfectly predict Wi-Fi cut and resume dates, and the content the user will request during Wi-Fi outage (predictability is discussed in section IV).

To capture the time-dependent relevance of data on the potential of pre-caching, we have defined a data validity period called the *horizon*. The horizon, expressed in minutes, represents the time for which the pre-cached data is meaningful to the user experiencing a Wi-Fi gap. For the 33 users in our dataset, we have calculated the amount of data that can be pre-cached for different horizon values. At each Wi-Fi cut event, we have accumulated the traffic sent over cellular for at most the duration of the horizon or until Wi-Fi resumes. Fig. 2 shows the fraction of cellular data that can be prefetched for different horizon values. We have seen that around 80% of the Wi-Fi gaps last less than one hour and a half. With a 2-hour horizon we could thus pre-cache at most to 80% of the cellular data. The data analysis in this section shows the potential of pre-caching since a significant percentage of mobile traffic is

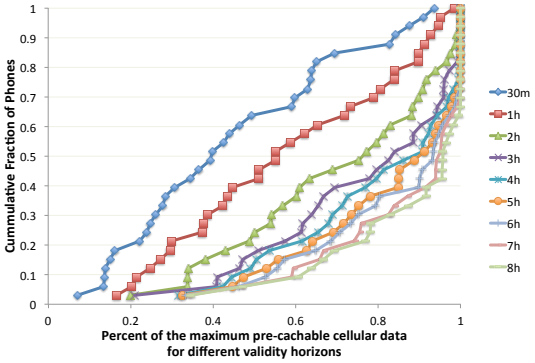

Fig. 2. Percent of cellular data that can be pre-cached for different validity horizons (in hours).

delivered through cellular at commuting hours during Wi-Fi gaps of at most 2 hours.

IV. PCACH: A USER-CENTERED PRE-CACHING STRATEGY

In this section, we introduce PCach, a pre-caching strategy whose design is driven by the analysis of the measurements dataset. It consists of predicting the future occurrence of Wi-Fi cut and resume dates, together with the data she is likely to need during the gap.

A. The PCach approach

PCach can be implemented as a standalone mobile application. Thus, it only leverages information that can be accessed through the native Application Programming Interface of the operating system. Our design aims at protecting user privacy: Exploitation and storage of sensed data, prediction processing and results are all made exclusively on the smartphone.

Algorithm 1: PCACH

Input : current time slot, $cSlt$; list of pre-cachable apps, $sApps$; number of apps to pre-cache, K
Output : List of apps to pre-cache, $PCachApps$

begin
  PastDB ← Update history($cSlt$) ;
  cut ← predictNextWiFiCut(**PastDB**, $cSlt$);
  if $cut == true$ **then**
    $rSlt$ ← predictWiFiResumeSlot(**PastDB**, $cSlt$);
    $PCachApps$ ← predictTopKApps($sApps$, K, $cSlt + 1$, $rSlt$);
  Return $PCachApps$;

Once the PCach app is installed on a smartphone, it triggers Algorithm 1 periodically. To predict Wi-Fi gaps, PCach divides a 24-hour period into time slots. In any given time slot, $cSlt$, it first predicts if a Wi-Fi cut event is going to occur in the next time slot, $cSlt + 1$. If that is the case, it predicts the identifier $rSlt$ of the future slot where Wi-Fi is supposed to resume, giving PCach the information necessary to know when a Wi-Fi gap occurs and its duration. If the duration is non-zero, it predicts the top K from a list of applications, $sApps$, considered to be *pre-cachable* (more on this in section IV-B). All predictions rely on the smartphone usage history, stored in a local database, $PastDB$, which is updated continuously. The efficiency of PCach obviously depends on the efficiency of the algorithms used for predicting Wi-Fi gaps and the content of applications to pre-cache.

In terms of implementation, MACACO-app derives from our crowdsensing app as it measures periodically the same data as the one in our MACACO data set. This measurement step is easily done in a standalone app. To trigger prefetching for a given app, P-Cach simply requests the operating system to launch the app for example by pushing it to the foreground for a short period of time. The app is thus likely to update its content in a standalone manner. The consequence is that prefetching only happens if the app is configured to update its content in running status. In terms of memory, the context and content data we store is really light: a day of measurement generates 158 bytes of raw data, in average. This can be further reduced in volume as our prediction schemes build on very simple quantitative information extracted from these measurements at each measurement date [4]. The prefetched content will add to it, but since we let the app prefetch its content we rely on its own memory management policy. In terms of energy footprint, it is the periodic data collection that is energy-hungry. P-Cach uses the same data collection operations as our crowdsensing app. This app has been designed with care to minimize energy consumption. Main design choices are clearly explained in [5].

Next, we show how simple learning strategies for Wi-Fi gap and top app prediction already perform well on our dataset.

B. Top app prediction

This prediction step selects the K mobile apps that are the most likely to send or receive data during the next Wi-Fi gap. This problem is closely related to the one addressed in [8]. Next, we identify from our dataset the actual number of apps K to be prefetched and the set $sApps$ of apps that are worth pre-caching.

1) Choice of K: In [5], we show that only a few apps trigger most of the downloaded data volume. The median phone only runs about 5 apps (resp. 10 for a 1h slot), an order of magnitude lower than the number of apps installed. As a result, it seems reasonable to set $K \leq 10$ for the 15-minute slot and $K \leq 17$ for the 1h-long slot.

2) Pre-cachable application set: Among the set of applications that send or receive data, we have identified categories of applications that have good properties in terms of data validity. We say that these applications are *PCachable*. Regular messaging applications, downloads, Internet browser, radio or music streaming applications are typically not PCachable. On the other hand, social media apps such as Facebook, Twitter or Instagram are PCachable as it is possible to pre-cache the latest posts. Of course, news, weather forecast and sports apps are PCachable as well. We have also included video streaming apps such as YouTube or Twitch as they have personalized channels whose latest data could be pre-cached. We refer the reader to [4] for details.

3) History-based prediction: We define here a simple history-based predictor to select the K apps to prefetch for the next slot. We show in [4] that there is a strong correlation between the number of times the user checks an app and the proportion of data being sent. This is not surprising but it gives us a very simple feature to leverage. We implement thus a prediction algorithm that creates, for each PCachable app, a histogram H that counts the number of times an app has been called in each time slot over time. The predicted K apps are the ones with top histogram values.

4) Results: Our app-prediction algorithm has been tested on our dataset. Initial histograms are built with the records of the first week. Since for each subsequent time slot we know exactly the set of apps that have sent data over cellular, we can compare our predictions to the ground truth. The histogram is constantly updated: prediction of slot x is made with the timeline ending in slot $x-1$. The value K has been varied and

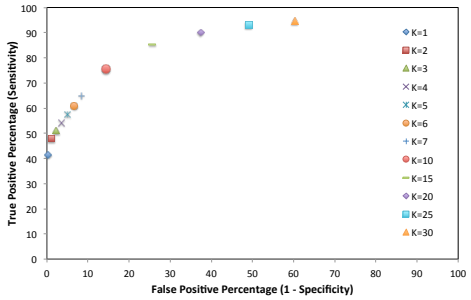

Fig. 3. Mean percentage of TPR and FPR over all phones for different K values.

TABLE I
FEATURES IMPLEMENTED FOR ADABOOST LEARNING.

	Data type	Description
1	boolean	Covered by home Wi-Fi network from 8pm to 8am
2	boolean	Covered by work Wi-Fi network from 8am to 8pm
3	boolean	It is a weekday or a weekend day
4	integer	The number of seen Wi-Fi networks
5	boolean	Top 1 Preferred Wi-Fi is in the list of seen Wi-Fi
6	boolean	Top 2 Preferred Wi-Fi is in the list of seen Wi-Fi
7	boolean	Top 3 Preferred Wi-Fi is in the list of seen Wi-Fi
8	integer	Index of current slot
9	float	Probability of cut / resume for this specific slot

for each phone, the true positive rate TPR and false positive rate FPR are derived. Figure 3 shows the performance of our algorithm for different K values. Each point represents average (TPR, FPR) calculated over all phones. As K increases, the TPR increases but at the cost of more false positives. False positives trigger unnecessary pre-caching and have thus to be kept low. The data shows that a good compromise is to select K lower than 10. In this case, the overall prediction quality is very good with less than 20% false positive rate.

We show as well in [4] that the prediction quality is best if setting K to 10 for 15-minutes slots and to 15 for 1-hour slots. There values are close to the average number of apps creating traffic for both slot sizes.

C. Wi-Fi gap prediction

A central step of Algorithm 1 is the Wi-Fi gap length prediction. It is decomposed into two challenges: predicting a Wi-Fi cut event and predicting the slot in which Wi-Fi resumes. It is very important for the Wi-Fi cut predictions to keep the false positive rate low to limit the number of unnecessary pre-cache operations. What makes this criterion particularly important is that our data shows that the proportion of cuts is low compared to the proportion of no-cuts. For 15-minute slots, on average, only 2% of slots contain a Wi-Fi-cut event. As a result, a 40% false positive rate would predict a wrong cut almost 38 times per day. Two prediction algorithms have been tested: The first is similar to the history-based app prediction. The second is based on AdaBoost [3], a state-of-the-art machine learning algorithm.

1) History-based predictions: We have developed a history-based algorithm that predicts whether Wi-Fi is likely to be cut or to resume in the next slot. This algorithm creates a histogram

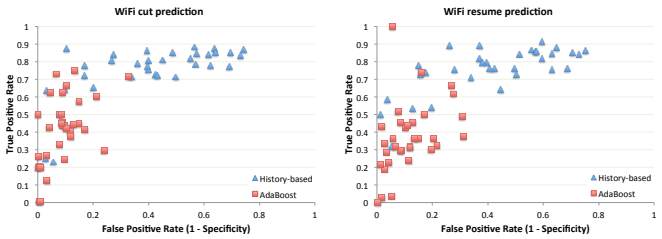

Fig. 4. Wi-Fi cut (left) and resume (right) prediction results: true positive rate as a function of false positive rate.

by counting cuts for all slots of the day. At prediction time, a random value between 0 and 1 is thrown N times. We count the number of times X this value is lower than the cut probability p_{cut} of the current slot stored in the histogram. If X belongs to the interval $[(1-\delta) * p_{cut}, (1+\delta) * p_{cut}]$, a Wi-Fi cut is predicted. Resumes are predicted the same way. Different values of N and δ have been tested to maximize the prediction quality metric (section IV-B4). The results in Fig. 4 are plotted for $N = 10000$ and $\delta = 0.1$.

2) AdaBoost prediction: AdaBoost (Adaptive Boosting) [3], an ensemble-learning method, is considered here. To apply it on Wi-Fi gap prediction, the set of features of Table I has been defined. These features are a result of our effort to define the user's context based on the Wi-Fi networks seen. Training has been performed on a per phone basis, each one using half of its available data. The obtained model has been used for predicting the other half. The results given in Fig. 4 show that Adaboost is the best solution for PCach: it delivers a good recall rate with a low false positive rate.

ACKNOWLEDGMENT

This work is supported in part by CHIST-ERA MACACO project, ANR-13-CHR2-0002-06.

REFERENCES

[1] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi. In *Proceedings of MobiSys*, pages 209–222, 2010.
[2] CISCO. Cisco visual networking index: Global mobile data traffic forecast update, 2015-2020. Technical report, White Paper, February 2016.
[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.
[4] K. Jaffrès-Runser and G. Jakllari. PCach: The Case for Pre-Caching your Mobile Data. *ArXiv e-prints*, July 2018.
[5] K. Jaffrès-Runser, G. Jakllari, T. Peng, and V. Nitu. Crowdsensing mobile content and context data: Lessons learned in the wild. In *Proceeding of PerCom Workshops*, pages 311–315, March 2017.
[6] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong. Mobile Data Offloading: How Much Can WiFi Deliver? *IEEE/ACM Transactions on Networking*, 21(2):536–550, April 2013.
[7] V. N. Padmanabhan and J. C. Mogul. Using Predictive Prefetching to Improve World Wide Web Latency. *SIGCOMM Comput. Commun. Rev.*, 26(3):22–36, July 1996.
[8] A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin. Practical Prediction and Prefetch for Faster Access to Applications on Mobile Phones. In *Proceedings of UbiComp '13*, pages 275–284, 2013.