Proactive Scheduling for Content Pre-fetching in Mobile Networks

O. Shoukry[1,2], M. Abd ElMohsen[1,2], J. Tadrous[3], H. El Gamal[3], T. ElBatt[1,2], N. Wanas[4], Y. Elnakieb[1] and M. Khairy[2,5]

[1]Nile University, [2]Cairo University, [4]Inmobly, [5]Smartec Group, Giza, Egypt
[3]Ohio State University, Columbus, Ohio, USA

[1] ***Abstract*—The global adoption of smart phones has raised major concerns about a potential surge in the wireless traffic due to the excessive demand on multimedia services. This ever increasing demand is projected to cause significant congestions and degrade the quality of service for network users. In this paper, we develop a proactive caching framework that utilizes the predictability of the mobile user behavior to offload predictable traffic through the WiFi networks ahead of time. First, we formulate the proactive scheduling problem with the objective of maximizing the user-content hit ratio subject to constrains stemming from the user behavioral models. Second, we propose a quadratic-complexity (in the number of slots per day) greedy, yet, high performance heuristic algorithm that pinpoints the best download slot for each content item to attain maximal hit ratio. We confirm the merits of the proposed scheme based on the traces of a real dataset leveraging a large number of smart phone users who consistently utilized our framework for two months.**

***Index Terms*—Content pre-fetching, behavioral models, scheduling, traffic offloading, smart phone user traces**

I. Introduction

Recently, we witness an ever increasing demand for the wireless spectrum resulting from the exponential growth of mobile data traffic due to the increasing penetration of smart phones and the adoption of bandwidth intensive multimedia applications that cater to diverse users' needs. This trend has been recognized as the major cause of cellular networks congestion, forcing leading wireless operators around the world to consider significant additional investments in the cellular infrastructure. This trend is also growing with the increasing number of active smart phone devices. Moreover, the demand for mobile data will soon globally outpace the existing network capacity. According to [1], data and voice services in North America had similar network loads until May 2007. This ratio has dramatically changed ever since and it is suggested by recent reports that mobile data traffic will increase 18-fold from 2011 to 2016 dwarfing voice services [2]. This, in turn, gives rise to severe network congestion degrading the quality of service (QoS) perceived by mobile users.

Handa [1] suggested three alternative approaches to solve the cellular congestion problem, namely, (i) Scaling, (ii) Optimization and (iii) Network offloading. Scaling to 4G/LTE networks may help overcome the congestion problem in the short-term, however, this solution remains a temporary measure, at best. Network optimization, on the other hand, may help relieve the congestion problem through efficient resource utilization, yet, it is limited in its potential. Moreover, this approach is faced with serious issues including isolation of heavy data users, privacy preservation and policing users' traffic [3]. Offloading to secondary infrastructure provides an alternate path to wireless delivery. This approach has been widely accepted and, according to [4], the majority of traffic (63%) generated by smart phones, tablets and feature phones will transfer onto the fixed network via WiFi by 2015. Since a high percentage of mobile data consumption occurs while indoors or in motion, data traffic can be offloaded onto complementary fixed networks via WiFi.

This paper addresses the traffic offloading approach from a fundamentally different perspective that is based on the novel framework of proactive resource allocation pioneered by El Gamal et al. [5]. This framework holds the promise of a significant increase in the wireless network capacity without requiring additional investments in costly infrastructure. The basic idea is to exploit the inherent predictability of user behavior to schedule and pre-fetch content (i.e. before demand) to ease congestion and enhance the user experience. There has been ample evidence in the literature pointing to the predictability of mobile user behavior [6], [7]. In [6], it is argued that a large portion of the Internet users have repetitive content access patterns over time (i.e. days), whereas, in [7], it is shown that the human mobility is up to 93% predictable.

When contrasted to traditional "reactive" resource allocation schemes, proactive resource allocation is proven to attain considerable cost reduction gains for network operators, service providers, and content distribution networks [8]-[10]. These gains ultimately attribute to the smoothed out loads and the potentially improved QoS experienced by users. Yet, these works consider only proactive service through cellular network resources, and do not address the potential gains under WiFi offloading.

In this paper, we give a particular attention to proactive service through WiFi networks. Thus, multimedia content will be offloaded through WiFi which yields no payment at the user end, and more relaxed network conditions at the cellular operator. This process requires careful selection and scheduling of new content to best fit the WiFi connectivity patterns of the users while maintaining an acceptable degree of content freshness, and battery usage.

Among popular techniques used in the literature to solve such problems are multi-objective Evolutionary Al-

[1]This paper was supported by a grant from the Egyptian National Telecommunications Regulatory Authority (NTRA) and the Information Technology Industry Development Agency (ITIDA).

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

gorithms [11] and multi-objective meta-heuristics [12]. The results obtained in [11] show that evolutionary algorithms can be effectively applied to the intrinsically multi-objective scheduling problem of large-scale space network. Multi-objective flow shop scheduling using meta-heuristics is discussed by Kumar [12]. The objective therein is to minimize the weighted sum of the total weighted squared tardiness, make span, total weighted squared earliness and the number of tardy jobs, concurrently. Genetic algorithms and simulated annealing were also employed to solve this problem, and a hybrid approach has been proposed for enhanced performance. Unfortunately, both approaches exhibit high computational complexity for resource-limited mobile devices.

Our contribution in this paper is two-fold. Firstly, we introduce an operative architecture for Proactive User-centric Automatic Loading, coined PAUL, for proactive content caching. Such a system is tested on a multitude of users and is proven to result in considerable savings to the cellular networks, which dominantly happen during the peak demand hour.

Secondly, we propose and detail a greedy proactive scheduler leveraging the developed stochastic models for the user behavior processes of interest. The objective of such an algorithm is to maximize the hit ratio of cached content. It features quadratic complexity in the number of time slots per day, which renders it practically viable.

The rest of this paper is organized as follows. Section II overviews the overall system architecture, PAUL. Afterwards, we present the system model, formulate and solve the proactive scheduling problem via a greedy algorithm in Section III. In Section IV, we conduct a performance evaluation study with the aid of extensive simulations using real-life smart phone traces. Finally, Section V concludes the work and point out potential directions for future research.

II. PAUL System Architecture

The ultimate objective of PAUL is to find a schedule for pre-fetching content items, over the course of a day, before actual user demand. The entire system hinges on developing trace-based stochastic models (profiles) for the mobile user behavioral processes of interest, namely content consumption, WiFi and battery state. Given the profiles, an estimate of the average download data rate is computed to estimate the success probability for content caching at a given time slot. Using the calculated schedule, user content can be pre-fetched and made available ahead of demand.

As illustrated in Fig. 1, PAUL consists of three major building blocks, namely: (i) user behavior loggers residing on the mobile device, as part of the Mobile App, (ii) trace-based stochastic profilers modeling the user behavior and (iii) proactive scheduler which retrieves "predicted" content of interest before demand. The loggers are responsible for capturing the behavior of the mobile user and device and sending them to the cloud. The logged data, pertaining to the demand side (content consumption) and resource side (average data rate and battery state), are then

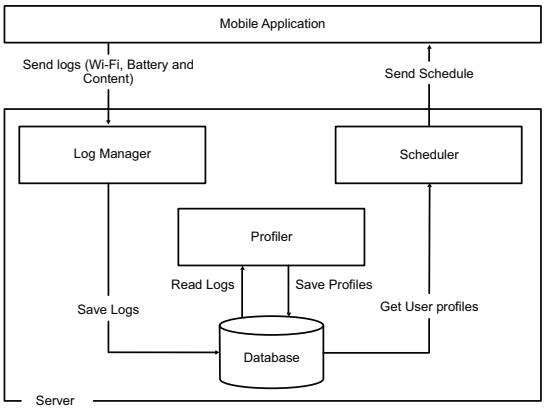


Fig. 1: PAUL System Architecture

used to create representative users' profiles. Ultimately, the scheduler leverages the user profiles to build schedules for content pre-fetching. The profiler and scheduler may reside on the mobile device at the expense of consuming computational and storage resources and battery. Hence, they were moved to the cloud with abundant resources.

A. Logging User Behavior

This module resides on the user's mobile phone with the responsibility of logging the user WiFi connectivity and data rate, the phone battery state and the content usage behavior. The WiFi connectivity model aims at capturing the visited networks, the residence time in each network and the average download data rate throughout each visit. Active Probing is employed to estimate the average downlink data rate over the residence duration within an access point (AP) [13], [14], [15]. We adopt active probing to avoid wasting much bandwidth using small (100 bytes) packets in order to minimize the incurred overhead.

The Battery State Logger is responsible for monitoring the battery via logging the following battery events: (i) Battery charging, (ii) Battery low and (iii) Battery OK.

The Content Usage Logger is responsible for logging the user content consumption behavior for a number of popular application categories. The logged data contains application name, application category and time of request.

B. Probabilistic User Behavioral Models (Profiles)

Here aim to build a probabilistic user behavioral model for each day of the week, based on the logged data. Therefore, we divide the day into a number of scheduling periods (called segments) where each segment is divided into a number of time slots with equal duration (system parameter). We use the data from the logs (battery usage, average data rate and accessed content) to generate representative histograms, which constitute the user profile. In order to build a probabilistic model for a physical phenomena, it is well known from stochastic processes [16] that the sufficient statistics for a process involves computing the joint probability of events at different time instants in order to capture the temporal correlations. This statement is generally true for all three processes at hand. However, to reduce the model complexity, we assume that the marginal probability in a specific time slot is a sufficient statistic for the data

rate and battery state processes. This implies no temporal correlations are captured for these two processes. On the other hand, the content usage process is assumed to capture short-term temporal correlations, manifested through the successive download states from the same category over a sequence of time slots. This short-term correlation is intuitive since a typical user goes through a sequence of download events upon visiting a content category of interest.

Furthermore, the above mentioned processes generally exhibit correlations among themselves. Based on intuition, we assume that the battery state and data rate processes are correlated. This can be illustrated with the aid of an example. A user on the go, with limited or no available WiFi access, will most probably have a Low Battery state. On the other hand, a charging, or a fully charged battery state, would have higher probability of having a stable WiFi connectivity. This correlation is the prime motivation for introducing a joint WiFi and Battery state model in this paper. In order to simplify the model, the content usage process is assumed to be uncorrelated to the data rate and battery state processes.

The joint data rate-battery model captures the user's resource availability. As illustrated earlier, the battery logger periodically logs the state of the battery and lists it in terms of the following three states: Charging, OK and Low. On the other hand, the WiFi model provides a breakdown of the probability of a certain WiFi data rate during the slot. We assume that the available download rate can be divided into the following five data rate bins in kbps e.g., 0-20, 20-40, 40-60, 60-80, $>$ 80. Thus, the modeler creates histograms for the battery and bandwidth with 15 different states (3 battery states * 5 data rate states). For each slot, the probability that the state is ("Charging", "OK", "Low"x"0-20", "20-40", "40-60", "60-80", "$>$ 80") is computed from the logs. The probability of a given state s is simply the ratio of time spent in that state over the entire segment duration.

The Content Usage Profile models the user's content demand. The content usage logger periodically logs the content (application) requests and lists them in terms of the Application Categories. Afterwards, the content usage modeling generates a histogram profile for a given user. We assume that the system can retrieve one content item per slot and that we have M applications categories. The content usage logs are used to determine the probability of consuming any of the M categories accessed by a given user. Generally, different users may have different values of M, however, M is fixed for a given user in a single time slot. The content usage profiler computes the probability of requesting each application category (Content State) within a specific time slot by dividing the number of requests for each content item by the total number retrievals over a pre-specified period of time.

C. Proactive Scheduler

The proposed proactive scheduling algorithm is the centerpiece of PAUL and constitutes a major contribution of this paper and, hence, is presented in details in the next section.

III. Proactive Content Scheduler

Given the probabilistic, trace-based user behavioral profiles constructed in Section II, the proposed proactive content scheduler leverages the inherent predictability of mobile users behaviors, along with the increasing capabilities of smart phones, to meet the ever growing demand for the limited, non-renewable wireless spectrum. The resulting scheduler is envisioned to deliver content before demand that allows not only off-peak data offloading but also enhanced user experience. It also improves network resource utilization by offloading across time and/or secondary networks (WiFi).

A. System Model

The scheduling scope is specified as one day as we assume one segment per day throughout the paper. For any day of the week, a schedule is generated based on the constructed probabilistic model for that day. r_i and b_i denote the average download rate and battery state in slot i, respectively. The system accommodates M different types of content items (application categories). A day is divided into a number of slots with fixed duration, denoted, D. Slot duration is the same, and fixed, throughout constructing the behavioral models and scheduling. A single day is split into N slots where $N = \frac{24*60}{D(mins)}$. We assume that a single content item, at maximum, can be retrieved in a single slot. Notice that no content items are retrieved, in a particular slot, due to the lack of connectivity and/or battery resources or violation of content freshness constraints. A scheduling policy, i.e. the slot-content item assignment, for the N slots is modeled by the vector $\vec{S}$ as follows

$$\vec{S} = [C_i^1, C_j^2 .... C_k^N]$$

where C_i^n represents retrieving content item of type i in slot n and $1 \leq i,j,k \leq M$.

B. Problem Formulation and Complexity

Our prime objective in this paper is to schedule content item downloads, over the course of a day, in order to maximize the probability of the user having the content cached before demand, subject to data rate, battery and content freshness constraints.

Given a number of content items $m \leq N$ that are consumed by an arbitrary mobile user over the course of a day at time instants, c_j, where $1 \leq j \leq m$, the optimal proactive scheduler aims at maximizing the content hit ratio (H) subject to the average download data rate and battery state profiles, along with freshness, constraints as follows

$$\max_{\vec{S}} H \tag{1}$$

$$\text{s.t.} \quad K_i \in \{0,1\}, \qquad 1 \leq i \leq N$$

$$K_i = 1 - [1(P(r_i < \rho) \geq \alpha) \vee 1(P(b_i = Low) \geq \beta)]$$

$$c_j - n_j \leq f, \qquad 1 \leq j \leq m$$

where $\vee$ is the logical OR operator and H is the content item hit ratio for one segment defined as the percentage of content items retrieved before the user consumption time, where item j is retrieved in slot n_j, that is,

$$H = \sum_{j=1}^{m} \frac{1(n_j < \lfloor c_j \rfloor)}{m} \leq 1 \quad (2)$$

and $\vec{S}$ represents the content item-slot assignment policy over the span of one segment, b_i is the battery state in slot i, r_i is the average download data rate probed in slot i, ρ is a system parameter indicating the minimum data rate that allows downloading a single item per slot, K_i is the number of items that can be retrieved per slot and f is a freshness threshold, measured in number of slots. Evidently, ρ is dictated by the maximum item size and slot duration. The first constraint captures the case of no retrieval in slot i due to the lack of data rate and/or battery resources. α and β are probability thresholds that can be tuned for each user and application consumption portfolio. Per the system model, the second constraint asserts that one item can be retrieved per slot at maximum. The third is a content freshness constraint.

Evidently, the optimization problem in (1) is combinatorial and, hence, the optimal scheduler has to examine all $|\vec{S}| = (M+1)^N$ combinations where M is the number of application categories of interest and N is the number of slots per day. This, in turn, gives rise to exponential complexity of the optimal policy with the number of slots, which makes the optimal prohibitively complex and practically infeasible. This motivates us to explore a simple, greedy proactive scheduler, with only quadratic complexity. Interestingly, it considerably outperforms the baseline simulated annealing scheduler.

C. Greedy, Quadratic-Complexity Scheduler

The proposed greedy, proactive scheduler hinges on two important notions, namely *Reward* and *Utility*. The Reward captures the benefit of assigning a content item (Application Category) to a given slot and the Utility captures the feasibility of retrieving this content item in a slot based on the availability of battery and WiFi connectivity resources. Next, we give a detailed description of the greedy proactive scheduler.

As indicated earlier, the day is divided into a number of slots with fixed slot duration, D, which is the same time unit used for building the user models in Section II.B. The proposed scheduler proceeds through N rounds, where a content item is assigned to one of the N slots in each round. Each round consists of two major steps, one for computing the Reward and another for the Utility. First, the proactive scheduler assigns the maximum Reward content item to each slot of the day. The Reward for each content item in a given slot depends on the probability of consuming this item over successive slots within a freshness window, W. In essence, we assign the content item to a slot based on the highest probability of being requested over the next W slots. The content item with maximum Reward is selected as a candidate for this slot. Accordingly, the Reward, R, of

Algorithm 1 Greedy Proactive Scheduler

```
READ Content Profiles
READ Battery and WiFi Profiles
while UnscheduledSlots > 0 do
  i = 0
  while i < N do
    j = 0
    while j < M do
      CALCULATE Application Content Reward
      SAVE Application with maximum Reward
      j++
    end while
    CALCULATE Slot Utility
    SAVE Slot with maximum Utility
    i++
  end while
  Schedule Slot with maximum Utility
  UPDATE Content Profile
  UnscheduledSlots--
end while
```

retrieving a content item, C_j, in a specific slot, n, is given by

$$R(C_j, n) = \sum_{i=n+1}^{W} F(i, C_j), \qquad 1 \leq j \leq M \quad (3)$$

Thus, $R(C_j, n)$ is determined by summing the profile values $F(i, C_j)$, i.e. probabilities of the selected content items, over all slots following slot n and within a window size, W, which constitutes a parameter of the system and represents item "freshness". Next, we determine the content item candidate with the maximum reward in slot n, that is,

$$Max(R(n)) = \max_j R(C_j, n) \quad (4)$$

In the second step of one algorithm round, the Utility U of a given slot n having a maximum Reward $Max(R(n))$ is given by

$$U(n) = S(n).Max(R(n)) \quad (5)$$

The slot utility incorporates the resource availability side, captured by $S(n)$, along with the content side, captured by the maximum Reward, $Max(R(n))$. The success probability, $S(n)$, is the probability of successfully retrieving an item in slot n and is defined as the probability that the average data rate and Battery state, at slot n, exceed pre-specified levels necessary for successful item retrieval over a slot. These levels were chosen to be in an "OK" or a "Charging" battery state and a data rate higher than 20 kbps. The slot with maximum Utility is assigned to the max Reward content item in this round and is removed from the current unscheduled slots list. This indicates the completion of one round of the algorithm.

The content item profiles are then updated based on the content item assigned to the scheduled slot. The profile values for the scheduled content item are decreased within a window just after the current slot. This update reflects the fact that a content item has just been scheduled and

would be fresh to use for a given window of time. The whole process is then repeated, iteratively, for all N slots in the segment (day).

For the proposed greedy scheduler algorithm, it is evident that in each round an assignment decision is taken for one slot. Hence, the number of algorithm rounds grows linearly with N. Thus, the computational complexity CC of the proposed greedy scheduler is given by

$$CC_{greedy} = O(N.v) \tag{6}$$

where v is the number of computations per round, that is the sum of number of computations in the Reward, R step in (6) and the number of computations in the Utility, U, step is as follows

$$v = O(M.N) + O(N) \tag{7}$$

Based on the above scaling behaviors, it is straightforward to establish the following complexity result for the PAUL scheduler

$$CC_{greedy} = N * [O(M.N) + O(N)] \tag{8}$$

$$= O(M.N^2) \tag{9}$$

Thus, we conclude that the greedy PAUL scheduler has only quadratic scaling behavior, with the number of slots per day, N, as opposed to exponential scaling behavior for the optimal scheduler. This renders the greedy scheduler a viable approach on smart phone and portable computing devices.

IV. Performance Evaluation

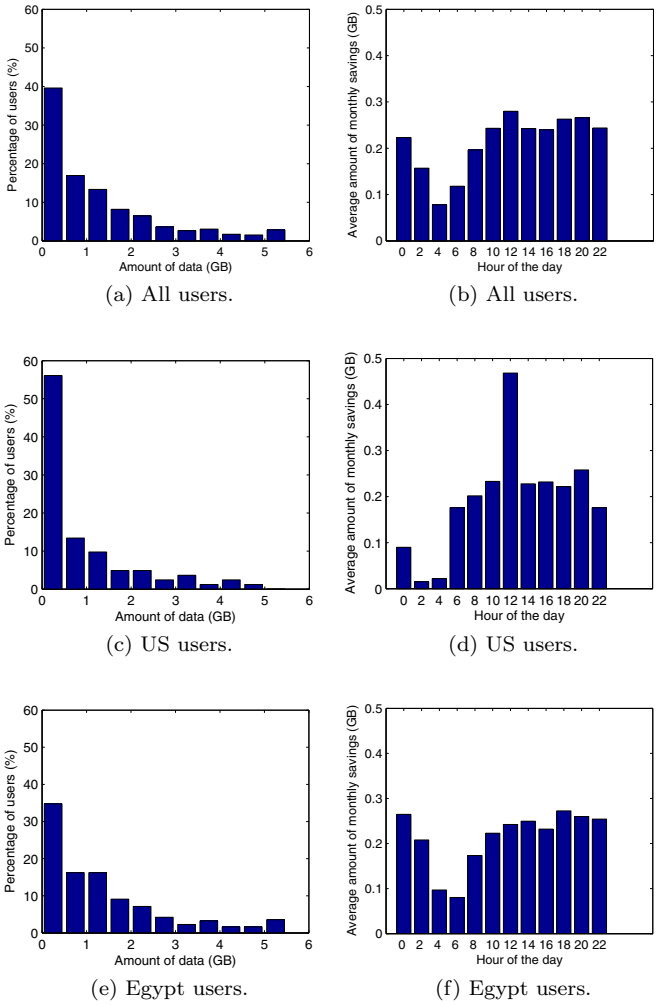
In this section we quantify the performance of the proposed PAUL system. First, we demonstrate the data saving gains which capture the amount of cellular network traffic that is effectively offloaded via PAUL over the WiFi networks. Then, we compare the hit ratio performance of the proposed greedy algorithm with a baseline simulated annealing (SA) scheme to expose its relative performance merits.

A. Performance of Data Savings

We consider the amount of data cached via PAUL on WiFi and consumed later on under **no** WiFi connectivity as savings. In particular, without PAUL, this data would have been consumed via the cellular network at a price paid by the user and a potential of low QoS due to buffering delays. Moreover, these savings also are considered beneficial for the cellular network operators since they eventually reduce the excessive peak hour demand on the network resources.

The results we present here are based on a real dataset collected from 777 users who have been using PAUL in the two months of May and June 2013. The users are located all over the world.

Fig. 2 illustrates the monthly data savings attained via PAUL. The savings frequency for all users in Fig. 2a reveals that more than 43% of all users save at least 1 GB of data. These savings are mainly concentrated in the period



(a) All users. (b) All users.

(c) US users. (d) US users.

(e) Egypt users. (f) Egypt users.

Fig. 2: Left figures (a), (c) and (e) depict the frequency of monthly savings, whereas right figures (b), (d) and (f) depict the distribution of monthly savings for an average user over the day.

of peak demand (from 10 a.m. to 10 p.m. [17]) as shown in Fig. 2b. Thus, it can be noted that PAUL yields a reduction of the total peak-hour load at the network operators. Not only does this reduction improve the QoS for PAUL users, but also it extends to non-PAUL users who will eventually witness lower congestion levels.

Figs. 2c, 2e compare the monthly savings frequency for the US and Egypt users. The relatively higher availability of WiFi coverage in the US yields a less percentage of users who save more than 1 GB as compared to Egypt. On the other hand, the savings during the lunch break in the US (from 12 p.m. to 1 p.m.) are remarkably larger than the savings at any period of the day both in the US and Egypt, as clarified in Figs. 2d, 2f.

B. Evaluation of the Greedy Algorithm

The baseline scheduler: is implemented to compare the results with the proposed greedy scheduling algorithm.

TABLE I: System Parameters

Parameter	Definition	Value
N	Number of slots per segment	48
M	Number of application categories	10
W	Window size (slots)	6
S	Item size (MB)	25
S_C	Cache size (MB)	250

SA [18] is a metaheuristic used for optimization of large search space problems, like the problem at hand. The probabilistic nature of this technique makes it more efficient than exhaustive search in solving certain problems that require finding a good solution rather than the optimal solution.

The main procedure of SA is to probabilistically decide which path between states shall be selected to move the whole system to a state of lower energy (analogous to a physical system state). At each step, the system moves to a neighboring state if a lower energy state can be achieved. Otherwise, the system remains in the current state. Different states are generated by altering given states, and a termination condition need to be satisfied to reach a prespecified "satisfactory" condition.

In our problem context, a random list of tasks is generated, i.e. assignment of content items to slots. The Utility of the schedule is calculated as shown before using the notion of Reward and the built user profiles. The total schedule utility is calculated by summing the utility of all assigned slots. In each iteration, a new randomized schedule is generated and selected if it has a higher total utility value than the previous schedule. This is repeated for the N times (the stopping condition).

Relative performance gains: Table I includes the numerical values of the PAUL system parameters used in our performance evaluation study. For all shown results, we assume the segment length is one day. We utilize the smart phone traces collected by Rice university LiveLab project [19], [20]. LiveLab is a methodology to measure real-world smart phone usage and wireless networks with a re-programmable, in-device, logger designed for long-term user studies. LiveLab was deployed for a number of iPhone 3GS users. This includes 24 Rice University students from February 2010 to February 2011, and 10 Houston Community College students from September 2010 to February 2011. While LiveLab logs a variety of measurements, for the sake of our system evaluation, we only use the application usage data, associated WiFi and data rate, and battery level data.

It is worth mentioning that some preprocessing on the data to fit our system has been performed. For the application data, we measured the user application usage frequency and duration and selected a set of applications to work with (CNN, ESPN, YouTube, Facebook, Twitter and LastFM) after removing the applications related to the OS and games. We focus on this set of application categories due to their popularity and widespread. For the associated WiFi data, we summarized the entries into a set of WiFi enter/exit entries. For the battery usage data, we worked on both the connectivity data and battery level data. We combined both data and processed them to get the entries representing the required battery activities (battery OK, low, and charging). Finally, we put all the processed application, WiFi and battery data into the format required by our user behavioral modeler to generate the user profiles needed by the scheduler.

TABLE II: Overall performance of the scheduling algorithm compared to the baseline scheduler for all users

Performance metric	Greedy	SA
Hit ratio	42 % ± 19%	28 % ± 17%
Cache utilization	64 % ± 12%	31% ± 15%
Time in cache	98.33 mins ± 25	121.58 mins ± 35

We implemented the greedy and baseline SA scheduler, using the system parameters in Table I for 25 users. This set was selected out of the 34 users available from the LiveLab data due to the availability of sufficient WiFi, battery and content logs over a sufficient period of time. Data was divided to a training set of 5 weeks and a testing set. A day was randomly picked from the available traces as the testing day and the previous 5 weeks were used for building the user behavioral models.

Table II outlines the performance results for all 25 users. These results show that an average value of 42% hit ratio can be achieved using the implemented greedy schedule. A maximum of 80% was achievable. The average cache utilization was found to be 64%. The time in cache value was measured to be the average time difference (in minutes) between the request time and the time of caching. This indicates how long the item resided in the memory before actually being consumed by the user. An average of hour and half is a satisfying result given that most of the cached items are videos. This value indicates that caching happens relatively just before consumption and not so soon to keep content as fresh as possible. It is worth mentioning that while the utilization of the cache reaches 100% in some cases, the memory resources were generally under utilized. This, in turn, motivates the quest for the optimal cache size for PAUL, which lies out of the scope of this work. The results also show that the greedy scheduler outperforms the SA w.r.t. the average hit ratio by 33%, and yields a higher cache utilization as well as lower time in cache per item.

Our scheduling algorithm would intuitively perform better in a resource abundant environment due to the WiFi availability and good battery conditions. To get further insights, users were classified into two main groups: Resource abundant (RA) users, and resource challenged (RC) users. Based on this classification, TableIII outlines the results using the proposed greedy algorithm and previously used system parameters.

The results confirm that our proactive scheduler yields a higher hit ratio (roughly doubled) in an RA environment as opposed to a RC one. The cache utilization is intuitively better since more items are cached in a RA environment.

TABLE III: Results on different user classifications, RA and RC

Performance metric	RA	RC
Hit ratio	63 % ± 9 %	27 % ± 7 %
Cache utilization	90 % ± 4 %	38% ± 11%
Time in cache	94.13 mins ± 19	72.33 mins ± 24

The time in cache, on the other hand, has a relatively lower value in the RC environment due to the fact that fewer number of items are cached (i.e. less competition), so the probability of caching it just before consumption is high.

Fig. 3a compares the hit ratio performance of the proposed greedy algorithm and SA. Different users have different behaviors concerning their available resource and demand behavior. This is obviously captured by having different hit ratio values for different users. User A10, for instance, had more resources (WiFi and battery) compared to user A04 which had a direct effect on the hit ratio, giving a higher value to the user with higher download data rate, on the average. The figure also shows that our greedy scheduling algorithm outperforms the SA in most of the cases by about 30%.

Fig. 3b quantifies the hit ratio while changing the slot duration, D. Users B08, B09 and B11 (as labeled in the LiveLab traces) had a higher hit ratio with smaller slot duration, while A04 experienced lower hit ratio with smaller period. This is due to the the sparse content demand and resources availability of A04 traces, as opposed to the other three users. This resource challenging environment is observed to perform better in a larger slot duration and less scheduling items. The reason behind this is due to the fact that a larger slot period, yields less items to cache, giving a higher probability to more important items than overwhelming the schedule with lots of items to cache as in the case of shorter slot period.

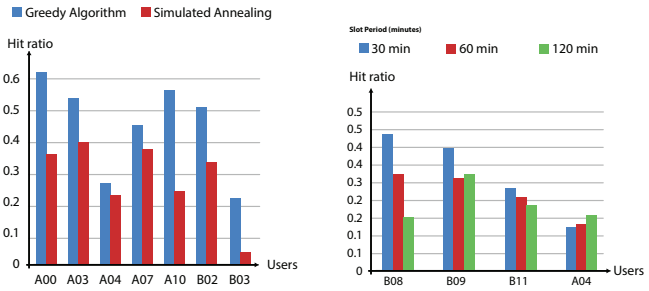


Fig. 3: Hit ratio performance: (a) comparison with SA, (b) effect of different slot size.

V. Conclusion

This paper addresses the aggravating congestion problem threatening cellular networks due to the excessive demand on multimedia applications. We propose a novel approach that pre-fetches multimedia content to mobile users before demand so as to ease the network congestion and enhance the user's quality of experience. The proposed system and proactive scheduler hinge upon constructing user behavioral models for content consumption, data rate, and battery level patterns. Due to the sheer complexity of the optimal scheduler, we developed a low-complexity greedy scheduler showing performance merits. When tested on a real dataset of 777 users, the developed framework yields significant data savings. These are successfully offloaded over WiFi and consumed under no WiFi connectivity. In particular, more than 43% of the users save at least 1 GB per month. Such a demand reduction on the cellular network renders the network conditions less strained at the peak hour. Moreover, the proposed greedy algorithm outperforms the baseline simulated annealing by 33%, on the average. The proposed scheduler can fulfill the users' requests with a high percentage (up to 70%) using real-life smart phone traces.

References

[1] A. Handa, *Mobile Data Offload for 3G Networks*, Intellinet-Technology 2009: Whitepaper.

[2] Cisco Technical Report *Cisco Visual Networking Index*, White Paper, Feburary 2012, available online at http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf.

[3] K. Lee, I. Rhee, *Mobile Data Offloading: How Much Can WiFi Deliver*, SIGCOMM 2010: 10th Annual SIGCOMM Conference on Computer and Data Communication Networks.

[4] Juniper Research, *Relief Ahead for Mobile Data Networks as 63% of Traffic to Move onto Fixed Networks Via WiFi and Femtocells by 2015*, April 2011, online at http://juniperresearch.com/viewpressrelease.php?pr=240.

[5] H. El Gamal, J. Tadrous, A. Eryilmaz, *Proactive Resource Allocation: Turning Predictable Behavior into Spectral Gain*, Allerton 2010 : 48th Annual Allerton Conference on Communication, Control, and Computing.

[6] D. Larrabeiti, R. Romeral, M. Uruea, A. Azcorr, P. Serrano, *Charging for Web Content Pre-fetching in 3G Networks*, ICQT 2004 : 4th International Workshop on Internet Charging and QoS Technology.

[7] C. Song, Z. Qu, N. Blumm, A. Barabas, *Limits of predictability in human mobility*, Science, vol. 327, pp. 1018-1021, Feb. 2010.

[8] J. Tadrous, A. Eryilmaz, H. El Gamal, *Proactive data download and user demand shaping for data networks*, http://arxiv.org/abs/1304.5745

[9] J. Tadrous, A. Eryilmaz, H. El Gamal, *Proactive content distribution for dynamic content*, to appear, the IEEE International Symposium on Information Theory (ISIT), July 2013.

[10] J. Tadrous, A. Eryilmaz, H. El Gamal, *Pricing for demand shaping and proactive download in smart data networks*, IEEE INFOCOM 2013, vol., no., pp.3189,3194, 14-19 April 2013.

[11] M. Johnston, *An Evolutionary Algorithm Approach to Multi-Objective Scheduling of Space Network Communications*, IASC: 2008: Intelligent Automation and Soft Computing.

[12] A. Kumar, *Multi-Objective Flow Shop Scheduling Using Meta-heuristics*, PHD thesis in Mechanical Engineering.

[13] H. Lee, V. Hail, K. Yum, E. Kim, *Bandwidth Estimation in Wireless Lans for Multimedia Streaming Services*, 2007: Hindawi Journal on Advances in Multimedis.

[14] K. Lakshminarayanan, V. Padmanabhan, J. Padhye, *Bandwidth Estimation in Broadband Access Networks*, IMC: 2004: ACM Internet Measurement Conference.

[15] S. Shah, K. Chen, K. Nahrstedt, *Dynamic Bandwidth Management in Single-Hop Ad Hoc Wireless Networks*, 2005: Kluwer Academic Publishers MONET.

[16] A. Garcia, *Probability, Statistics, and Random Processes For Electrical Engineering*, 2008: 3rd. Ed. Prentice Hall.

[17] http://oss.oetiker.ch/rrdtool/gallery/index.en.html

[18] B. Suman, P. Kumar, *A survey of simulated annealing as a tool for single and multiobjective optimization*, 2006: Journal of the Operational Research Society.

[19] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, P. Kortum, *LiveLab: measuring wireless networks and smartphone users in the field*, 2010: ACM SIGMETRICS Perform.

[20] Z. Wang, F. Xiaozhu Lin, L. Zhong, M. Chishtie, *How far can client-only solutions go for mobile browser speed?*, 2012: Int. World Wide Web Conf.