

Mobile Ad Prefetching and Energy Optimization via Tail Energy Accounting

Yongbo Li, *Student Member, IEEE*, Yimeng Wang, *Student Member, IEEE*, and Tian Lan, *Member, IEEE*,

Abstract—Accurately determining the network energy consumption of each software principal when multiple ones are active is the key to mobile energy optimization. *Tail energy accounting*, which attributes tail energy to individual software principals, remains an open problem. Besides, tail energy has also become a major energy drain, especially in mobile ad modules that generate frequent, intermittent network traffics by on-demand ad downloading. In this paper, we propose a systematic framework for mobile ad prefetching and energy optimization, based on a novel tail energy accounting policy using cooperative game theory. In particular, we maximize the sum of *deadline- and energy-aware ad utility*, by jointly determining apps' aggressiveness in ad prefetching. The proposed tail energy accounting not only characterizes the energy profile of each app's ad module, a crucial input in energy optimization, but also enables an efficient solution by decoupling decision making of individual apps. The proposed framework is implemented on Android with negligible performance/network overhead. Using real-world apps and usage traces, we demonstrate a significant reduction in mobile network energy consumption by up to 45% compared with existing approaches. To the best of our knowledge, it is the first fully implemented ad management system transparent to apps and ad ecosystem.

Index Terms—Tail energy, Mobile systems, Energy accounting, Ad Prefetching

1 INTRODUCTION

Current app marketplaces are increasingly dominated by free apps relying on advertising for revenue. Ad modules have become one of the major energy drainers on mobile devices, taking up 65% of apps' total network energy, or 23% of an app's overall energy [1]. This inefficiency mainly comes from the fact that mobile apps typically refresh their ads every 12 to 120 seconds [2], resulting in frequent, small transmissions. Since network interfaces often remain in *full power state* and *intermediate state* for a certain length of time after data transmission and before transitioning to *idle state* - e.g., 5 to 6 seconds in full power state, and 11.5 to 12 seconds in intermediate state in 3G [3] and LTE networks [4] - for the purpose of improving network responsiveness, such ad traffic causes network interfaces to constantly stay in full power or intermediate states, leading to considerable energy drain, commonly known as the *tail energy*. Also, when the size of data transfer is small like the ad traffic, the percentage of tail energy is larger [5]. It is shown that tail energy can contribute up to 48.2% of network energy consumption [4] on mobile devices. Recent study [6] even shows 89.2% out of the total cellular energy of 3G/LTE network is tail energy.

A natural approach to mitigating the tail energy (and reducing network bandwidth consumption) is to prefetch or cache a batch of ads locally. Since different ad libraries with different ad fetching patterns [7] may be imported by app developers, a centralized ad management proxy or middleware [8] seems promising to govern the system's ad fetching. However, prior works are often limited to measurement study of the energy saving [9] or simple heuristics-

based solution for tuning a single app's prefetching strategy [1]. Developing a quantitative framework for jointly optimizing multiple apps' ad prefetching strategies is an open problem. The first key to tackling the problem is that the timeliness of mobile ads must be modeled, as ads are often sold via real-time auctions freshly, hence, traditional techniques to cache and reuse content do not apply. For the same consideration, prefetching too aggressively, which is prefetching too many ads at a time, may result in missing display deadline, Service-Level-Agreement (SLA) violations and revenue loss [1]. Furthermore, despite energy accounting being an active research topic, there exists no theoretical framework to attribute tail energy - the main cause of mobile ad energy consumption - to individual apps and ad modules that are active during the same period of time.

In this paper, we propose a novel systematic framework for Mobile Ad Prefetching and Energy Optimization (MAPEO). Our solution consists of the following novel components.

To begin with, we formulate the MAPEO problem to jointly optimize all apps' prefetch strategies, in order to maximize their overall "energy-aware ad utility", defined as the sum of an app's ad utility normalized by its network energy consumption. The ad utility is time-dependent. Giving special consideration to SLA and without loss of generality, we model prefetched ads that are displayed after desired deadlines receive only partial or zero utility. This formulation allows us to leverage a tradeoff between energy saving and maintaining ad utility - because while on-demand ad downloads are not (tail) energy efficient, more aggressive prefetch strategies would lead to more residue ads with only reduced ad utilities. Our MAPEO problem captures this tradeoff and models the relevant system dynamics for a holistic optimization. Solving this MAPEO problem requires quantifying the energy consumption of each app's

• Yongbo Li, Yimeng Wang, and Tian Lan are with the Department of Electrical and Computer Engineering at George Washington University (email: {lib, wangyimeng, tlan}@gwu.edu).

ad module, which gives rise to the tail energy accounting problem.

Further, to tackle the challenge, we propose a new tail energy accounting framework relying on Shapley value [10] in cooperative game theory. We model the tail energy accounting problem as a cooperative game, where players are different software principals¹ that are active during the same period of time, and tail energy is the grand surplus. Existing energy accounting policies e.g., [11], [12], [13], [14], [15], [16], [17], [18], either (i) assume the synchronization between system activities and energy consumption, and thus cannot be applied to attribute asynchronous tail energy, or (ii) employ simple design heuristics without theoretical justification [19], [9]. Different from previous accounting policies that require measurement of an exponential number of system states [17], we develop a computation-efficient solution for attributing tail energy to individual software principals that are active concurrently.

We implement a prototype of our MAPEO framework for the proposed ad prefetching and energy optimization. A key feature is that the framework is completely transparent to mobile apps and compatible with contemporary ad ecosystem, supporting various ad libraries used in practice. This is achieved by our implementation of two key components: a Traffic Filter that selects and redirects ad-related traffic to our framework, and a central Ad Proxy that performs ad prefetching/caching and handles ad requests from apps. Our entire MAPEO framework is evaluated on Android devices including Galaxy Nexus, Samsung Galaxy Edge 6 and Galaxy Note 5, with real-world usage traces from Rice LiveLab [20].

The main contributions of our paper are as summarized below:

1) We propose a novel systematic approach to optimize mobile ad energy via SLA-aware prefetching. To the best of our knowledge, this is the first work to consider an optimization of both energy saving and ad utility, by jointly determining all apps' prefetch strategies.

2) We make it possible to decouple all apps' individual decision making and efficiently solve the joint system energy optimization problem by proposing a new tail energy accounting framework based on Shapley value [10]. It is shown to be the first tail energy accounting policy that satisfies two critical properties, *Positive Reward* and *Fairness*, making it suitable for identifying various software principals' energy profiles [21] and guiding energy optimization [9], [1].

3) We evaluate the implemented MAPEO system using top-free apps from Google Play, real-world usage traces [20], contemporary ad networks and ISPs. Results show that our proposed optimization framework using Shapley value tail energy accounting can outperform optimizations based on existing heuristics-based accounting policies by up to 45%. Further, the large variation of optimal prefetching strategies for 13 different apps suggests that jointly optimizing all apps' prefetch strategies is necessary to unlock the potential of mobile ad prefetching and energy optimization. Our

1. For ad prefetching in this paper, we consider each app's ad module as a software principal whose energy contribution is to be determined, while background traffic generated by apps and system threads are treated as separate software principals.

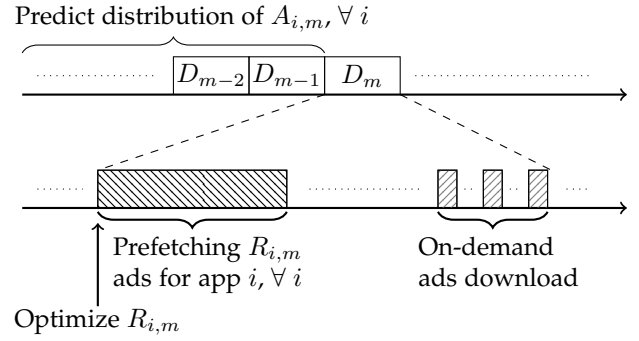


Fig. 1: An illustration of our system model.

solution only introduces an average of 1.72% performance overhead (since the computation of the optimal prefetching strategies can be cached and reused). The network RTT overhead is only 4.54% in terms of on-demand ad traffic while most ad requests are handled by prefetched content requiring no network connectivity.

2 SYSTEM MODEL AND OBJECTIVE

Mobile ads are typically downloaded on-demand. Once triggered (e.g., when first launching the app), a request is sent to an ad server. Then, ad content downloaded (often via HTTP or HTTPS) is displayed to the end user. While such ad traffic has short durations, mobile apps typically refresh ads every 12 to 120 seconds [2], resulting in heavy energy tails that have become a major energy drain and can take up almost 65% of an app's total network energy [1]. Traditional content caching and prefetching techniques [22], [23], [24] seem promising in improving energy efficiency. However, a unique challenge lies in ad prefetching problem that ads are normally delivered via real-time auctions and with deadlines required. Ignorant ad prefetching without considering the deadlines not only triggers SLA violations, it also consumes an excessive amount of energy.

In this paper, we consider a mobile system with K active apps, which periodically prefetch a bulk of ads every T minutes², and we consider a sequence of length- T intervals, $\mathcal{D} = \{D_1, D_2, \dots, D_m, \dots\}$.

As shown in Figure 1, at the beginning of each interval D_m , $R_{i,m}$ ads are prefetched for app i ³. Let $A_{i,m}$ be the number of ads actually displayed by app i during interval D_m . If $A_{i,m}$ exceeds $R_{i,m}$, additional ads (denoted by $O_{i,m}$) must be downloaded on-demand. Otherwise, surplus ads (denoted by $S_{i,m+1}$) will be rolled over to the next period D_{m+1} .

Taking into consideration of the SLA requirements in real-time ad auction, without loss of generality, we assume that SLA of the prefetched ads is preserved within T , each ad displayed in D_m receives a unit utility. Residual ads displayed in the next period D_{m+1} only receive a diminished

2. Our system model and optimization algorithm in this paper can be easily extended to time-varying T that changes according to network state or end-users' activity level.

3. The optimal value of $R_{i,m}$ may also be affected by background traffic. But it is shown in [2] that 68% to 81% of ad-related traffic is isolated from app traffic, unlikely to significantly change the optimization. Thus, we leave this to future work.

utility $v_i < 1$ per ad, and all residual ads are discarded after D_{m+1} ends due to severe SLA violation. v_i can be inferred from ad networks' SLA requirements, and is considered given in the problem. Therefore, the utility achieved by app i in interval D_m (denoted by $U_{i,m}$) depends on the number of residual ads from previous period $S_{i,m}$, the number of displayed ads $A_{i,m}$, and the prefetching decision $R_{i,m}$. Note that our model can be extended to accommodate other time-varying utility functions.

The goal of our mobile ad prefetching and energy optimization problem is to maximize the overall utility achieved by the ad display while minimizing the total energy consumed by ad fetching, i.e. maximize the overall "energy-aware ad utility" for each interval D_m .

$$\max_{\{R_{i,m}\}} \left(\frac{U_m}{E_m} \right), \text{ s.t. } R_{i,m} \geq 0, \forall i. \quad (1)$$

However, the main difficulty in directly optimizing the above objective is that the energy cost E_m is collectively determined by all apps' ad prefetching decisions. The number of feasible solutions is of the order of M^n for a mobile system with n ad-embedded apps where M is the max possible number of ads can be displayed by an app during the interval D_m .

In this paper, to tackle the above challenge, we make novel use of Shapley value based accounting to allocate the energy cost to individual software principals, thus decoupling the whole system optimization to each software principal's individual ad prefetching optimization. Our goal thus becomes the maximization of the sum of per-app "energy-aware ad utility", and we define our **Mobile Ad Prefetching and Energy Optimization (MAPEO)** problem for each interval D_m as follows:

$$\text{MAPEO} : \max_{\{R_{i,m}\}} \sum_i \left[\mathcal{E} \left(\frac{U_{i,m}}{E_{i,m}} \right) \right], \text{ s.t. } R_{i,m} \geq 0, \forall i. \quad (2)$$

Here, $E_{i,m}$ is the network energy consumed by app i during interval D_m . Note that both energy and ad utility $U_{i,m}$ are dependent on the ad prefetch decision $R_{i,m}$. The expectation is taken over the distribution of $A_{i,m}$.

Problem MAPEO allows us to explore the tradeoff between energy saving and ad utility, and it will be solved repeatedly for each interval to update prefetch strategy.

3 PROBLEM FORMULATION AND ALGORITHMIC SOLUTION

We start this section by formulating MAPEO problem. Then, we propose our tail energy accounting policy and finally develop a distributed solution to the MAPEO problem.

3.1 MAPEO Problem Formulation

We first analyze the ad utility $U_{i,m}$ of app i , along with the evolution of $A_{i,m}$, $S_{i,m}$, and $R_{i,m}$.

As shown in Figure 2, there exists 3 different cases. For *case 1*, $A_{i,m} > S_{i,m} + R_{i,m}$, more ads are displayed than what are available from rollover and prefetching. Thus, an additional $O_{i,m} = A_{i,m} - S_{i,m} - R_{i,m}$ number of ads must be downloaded on-demand. The total utility achieved $U_{i,m} = 1 \cdot (A_{i,m} - S_{i,m}) + v_i \cdot S_{i,m}$. Similarly, the $O_{i,m}$, $S_{i,m+1}$, and

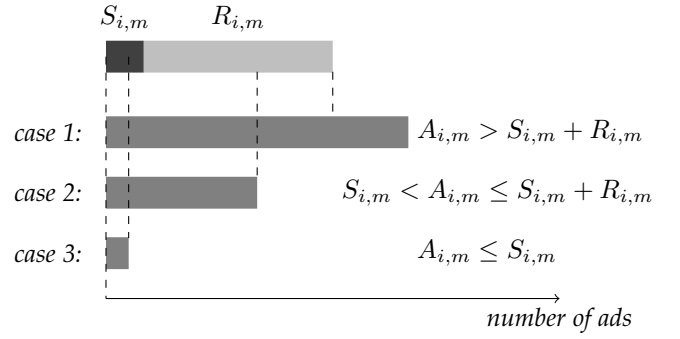


Fig. 2: Three cases of the comparisons between numbers of ads prefetched $R_{i,m}$, residue ads from previous period $S_{i,m}$, and ads displayed $A_{i,m}$ in interval D_m .

$U_{i,m}$ can be inferred for *case 2* and *case 3*. Combining the three cases with truncate function $[x]_+$, i.e., $\max(0, x)$, we derive the following equations:

During D_m , the utility achieved by app i is given by

$$U_{i,m} = 1 \cdot [A_{i,m} - S_{i,m}]_+ + v_i \cdot \min(A_{i,m}, S_{i,m}). \quad (3)$$

Further, the number of residual ads rolled over to next interval D_{m+1} is

$$S_{i,m+1} = [S_{i,m} + R_{i,m} - A_{i,m}]_+ - [S_{i,m} - A_{i,m}]_+. \quad (4)$$

Next, we construct an estimate for each app's energy consumption $E_{i,m}$ for varying prefetch decision $R_{i,m}$. We consider each app with network traffic as three (logical) software principals: (i) one involving all transmissions for ad prefetch, (ii) one for ad on-demand download, and (iii) one for all other network activities such as system events. For given prefetch decision $R_{i,m}$, we estimate $E_{i,m}$ using a hierarchical model:

$$\hat{E}_{i,m} = E_{1,i,m} \cdot R_{i,m} + E_{2,i,m} \cdot O_{i,m} + E_{0,i,m}, \quad (5)$$

where $E_{1,i,m}$ is the average energy per ad-prefetch, $E_{2,i,m}$ is the average energy per ad on-demand download, and $E_{0,i,m}$ is the energy for other network activities. Based on equations (3) and (4), we have

$$O_{i,m} = [A_{i,m} - S_{i,m} - R_{i,m}]_+. \quad (6)$$

Combining all these steps, for given distribution $\mathcal{P}_{i,m}(x)$ of displayed ads $A_{i,m}$, we rewrite MAPEO problem as

$$\begin{aligned} \max \quad & \sum_i \sum_{A_{i,m}} \mathcal{P}_{i,m}(A_{i,m}) \cdot \frac{U_{i,m}}{\hat{E}_{i,m}} \\ \text{s.t.} \quad & U_{i,m} = 1 \cdot [A_{i,m} - S_{i,m}]_+ + v_i \cdot \min(A_{i,m}, S_{i,m}) \\ & \hat{E}_{i,m} = E_{1,i,m} \cdot R_{i,m} + E_{2,i,m} \cdot O_{i,m} + E_{0,i,m} \\ & O_{i,m} = [A_{i,m} - S_{i,m} - R_{i,m}]_+ \\ & R_{i,m} \geq 0 \\ \text{var.} \quad & \{R_{i,m}, \forall i\}. \end{aligned} \quad (7)$$

The MAPEO problem explores the tradeoff between energy cost and ad utility, and the optimal prefetch decision $R_{i,m}$ maximizes the energy-aware ad utility.

However, the difficulty for solving the MAPEO problem is that $E_{1,i,m}$, $E_{2,i,m}$, and $E_{0,i,m}$ are unknown for the

current interval D_m during the time of optimization. To estimate the values from history intervals, the challenge lies in the fact that all apps' activities interweave and that multiple apps may all have ad fetching behaviors. All the apps' ad fetching behaviors and other non-ad network activities can jointly determine the amount of network energy and are simultaneously responsible for the tail energy consumed. In the next subsection, we are going to address this challenge.

3.2 Tail Energy Accounting

Now we consider the fundamental problem needs to be answered for estimation of $E_{1,i,m}$, $E_{2,i,m}$, and $E_{0,i,m}$ used in MAPEO problem: Given a set \mathbb{N} of active software principals in a system, let T_G be the time interval for tail energy accounting, i.e., from the start of data transmission to the end of energy tail. How to determine the energy contribution $\phi(i, T_G)$ of each software principal i (that have network activities during the period of T_G) in total energy consumption $E(\mathbb{N}, T_G)$?

3.2.1 Summarizing Existing Policies

We summarize four tail energy accounting policies reported in the literature.

Policy I: Last Trigger. This policy used by Eprof [19] states that the entire tail energy should be assigned to the last software principal that made data transmission before the tail.

Policy II: First Trigger. Opposite to Policy I, it assigns all tail energy to the software principal, which first started data transmission that generated the tail. This policy is considered but deemed to be less intuitive than Policy I in [19].

Policy III: App First. The policy used in [9], [1] computes the energy consumption of an ad module by measuring the marginal energy increase when enabling an ad module.

Policy IV: Tail Unaware. It ignores the existence of tail energy and proportionally assigns network energy to each app based on the length of its active transmission. This policy is used on Android [11] and most existing works on energy accounting [13], [17], [12], [15] do not consider tail energy.

Remark: App First policy can only attribute energy consumption to a single app and its ad module. It fails to determine energy accounting when there exist multiple active apps and ad modules, because the marginal energy increase depends on the order that different ad modules are enabled.

3.2.2 Analyzing Desirable Key Properties

Now we introduce two properties that are desirable for all energy accounting policies: *Positive Reward* and *Fairness*.

Definition 1. Positive Reward: If an app adopts a more tail-energy-friendly transmission strategy, it should receive a positive reward, i.e., a smaller share of energy consumption. This means $\phi(i', T_{G'}) \leq \phi(i, T_G)$ if there exists $E(\mathbb{S} \cup \{i'\}, T_{G'}) \leq E(\mathbb{S} \cup \{i\}, T_G)$, while all other energy consumptions remain the same in T_G and $T_{G'}$.

The *Positive Reward* property implies that a tail energy accounting policy should be able to quantify energy efficiency in accordance with software principals' transmission

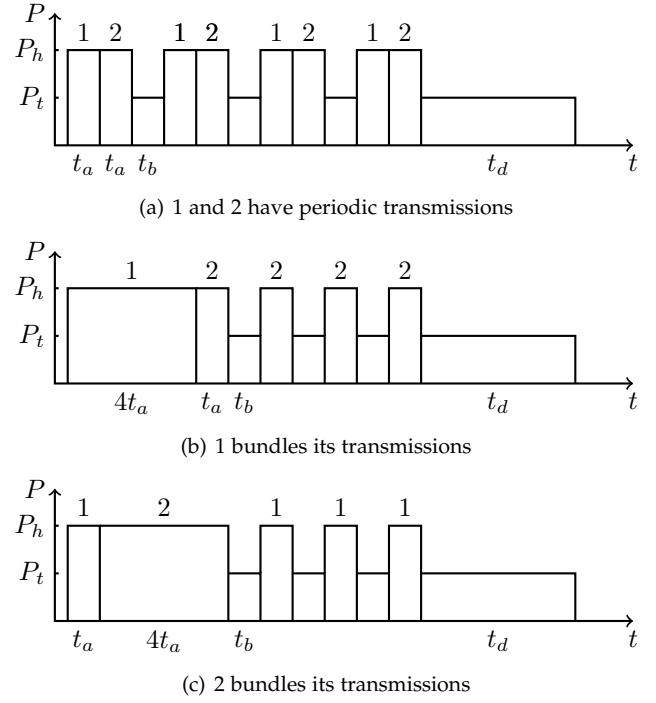


Fig. 3: Illustrative example for *Positive Reward* property with two software principals 1 and 2.

strategy, providing faithful input to energy optimization. Consider an illustrative example depicted in Figure 3. In Figure 3(a), both app 1 and 2 have periodic transmissions (each transmission session lasts for t_a , t_b is the idle time length between transmissions, and t_d is the length of the tail state), while app 1 bundles its transmissions in Figure 3(b) to allow tail energy saving, which would have materialized if app 2 is absent. Thus, app 1 now should be assigned a smaller energy consumption.

Remark: None of Policies I, II, III, IV fulfills the *Positive Reward* property. Under Last Trigger policy, software principal 2 receives all tail energy in both Figures 3(a) and 3(b). Thus, energy consumption attributed to app 1 is always $\phi_1 = 4P_h t_a$. P_h denotes power consumption at the high power state during transmission, and P_t denotes that at the intermediate state causing energy tail after transmissions. Similarly, under First Trigger policy, although app 2 bundles its transmissions in Figure 3(c), the energy attributed to it remains $\phi_2 = 4P_h t_a$ as in Figure 3(a). App First policy always assigns $\phi_1 = 4P_h t_a$, because energy consumption increment due to activating software principal 1 (assuming software principal 2 is already running) is the same in both cases. Finally, energy consumption is unchanged under Tail Unaware policy as transmission time of apps 1 and 2 remains constant.

Definition 2. Fairness: If replacing one software principal by another does not impact system energy consumption in any circumstances, the two software principals are symmetric and should receive equal energy share, i.e., $\phi(i', T_{G'}) = \phi(i, T_G)$ if $E(\mathbb{S} \cup \{i'\}, T_{G'}) = E(\mathbb{S} \cup \{i\}, T_G) \forall \mathbb{S} \in \mathbb{N}$.

Consider software principals 1 and 2 shown in Figure 4

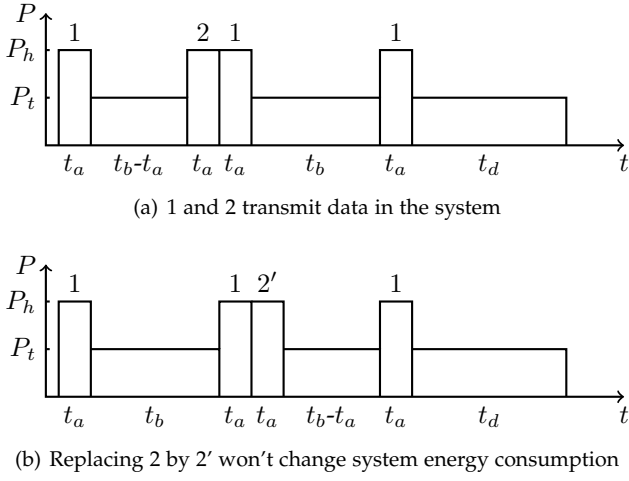


Fig. 4: Illustrative example for *Fairness* property involving two software principals 1 and 2.

(a). By replacing software principal 2 in Figure 4(a) with 2' in Figure 4(b), the system energy consumption including tail energy is unchanged. The software principals 2 and 2' are symmetric in terms of energy efficiency, and should receive the same energy allocation.

Remark: *None of Policies I, II fulfills the Fairness property.* Using Last Trigger policy, software principal 2 receives no tail energy in Figure 4(a), i.e., $\phi_2 = P_h t_a$, whereas software principal 2' is assigned $\phi_{2'} = P_h t_a + P_t(t_b - t_a)$ in Figure 4(b). Similarly, First Trigger policy allocates different amount of tail energy in the two scenarios.

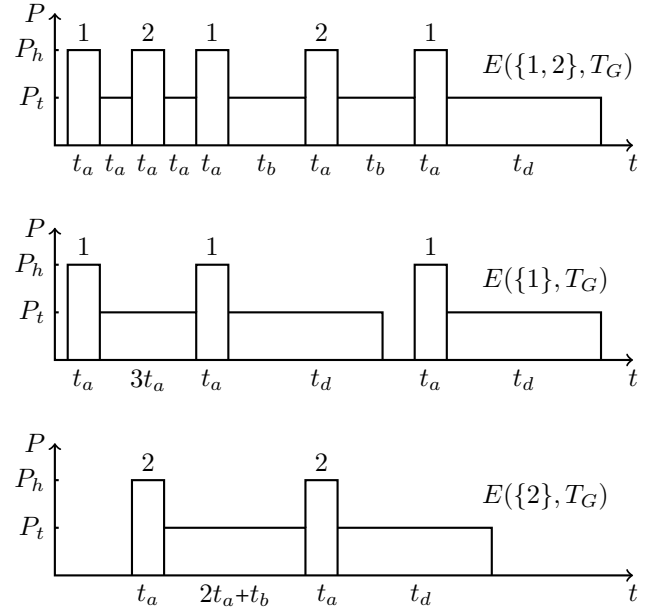
3.2.3 Our Tail Energy Accounting Policy

Showing all the existing accounting policies violate the desirable properties, we now propose our tail energy accounting policy based on cooperative game, which is widely used to study cost allocation problems in economics and engineering [25]. In a tail energy accounting game \mathcal{G} , we define the active software principals as players $\mathbb{N} = \{1, 2, \dots, n\}$, and total energy consumption $E(\mathbb{N}, T_G)$ as the grand surplus to allocate in the game. **The word “cooperative” here refers to the fact that the players (i.e., software principals) are not playing in isolation, rather, they can be active simultaneously. The grand surplus (i.e., total energy consumption) is determined jointly by the activities of all players that are active.**

We use Shapley value to calculate tail energy allocation $\phi(i, T_G)$ for each software principal i in the tail energy accounting game:

$$\phi(i, T_G) = \sum_{\mathbb{S} \subseteq \mathbb{N} \setminus \{i\}} \frac{E(\mathbb{S} \cup \{i\}, T_G) - E(\mathbb{S}, T_G)}{(|\mathbb{N}| - |\mathbb{S}|) \binom{|\mathbb{N}|}{|\mathbb{S}|}}, \quad (8)$$

where $\mathbb{S} \subseteq \mathbb{N} \setminus \{i\}$ is a subset of the players not including player i , $|\mathbb{S}|$ and $|\mathbb{N}|$ are the number of players in \mathbb{S} and \mathbb{N} respectively. $\binom{|\mathbb{N}|}{|\mathbb{S}|}$ is the number of different ways choosing $|\mathbb{S}|$ from $|\mathbb{N}|$. $E(\mathbb{S}, T_G)$ denotes the energy consumption if only players in \mathbb{S} are active during T_G . Basically, Shapley value as calculated by Equation (8) measures the contribution of



$$E(\emptyset, T_G) = 0$$

Fig. 5: Example for Shapley value calculation using Equation (8) for two active software principals 1 and 2.

one player averaged over all the possible permutations in which the coalition \mathbb{N} of the game can be formed starting with empty coalition [10].

Efficient computation of the proposed policy: Using Shapley value for energy accounting seems to require knowledge of energy consumption $E(\mathbb{S}, T_G)$ for all $\mathbb{S} \subseteq \mathbb{N}$, which has exponential complexity $2^{|\mathbb{N}|}$. However, existing work [4], [3] shows that for cellular networks, the network interface working states are determined by the data transmission size and rate, and can be modeled as state machines. For given network types and carriers, the interface's power consumption can be estimated by constant values for each working state, and the values can be profiled [13] for different phone models. Inspired by these facts, for tail energy accounting, given $E(\mathbb{N}, T_G)$, $E(\mathbb{S}, T_G)$ can be easily estimated by eliminating $\mathbb{N} \setminus \mathbb{S}$ players' transmission activities and inferring new energy tails. This method does not require additional measurements and can be efficiently calculated during runtime based on network activities and network interface working states.

An illustrative example involving two software principals is shown in Figure 5 for $2t_b + t_a > t_d$ and $2t_a + t_b < t_d$. The total energy is $E(\{1, 2\}, T_G) = 5P_h t_a + 2P_t t_a + 2P_t t_b + P_t t_d$. To find $E(\{1\}, T_G)$, we conceptually remove software principal 2's activities, generate new energy tails based on the state transfers for the active network interfaces using the models discussed in the existing work [4], [3], [13] mentioned above and obtain $E(\{1\}, T_G) = 3P_h t_a + 3P_t t_a + 2P_t t_d$. Similarly, we get $E(\{2\}, T_G)$ and $E(\emptyset, T_G)$. Then, from Equation (8), we can calculate the individual energy allocations: $\phi(1, T_G) = 3P_h t_a + 1.5P_t t_a + 0.5P_t t_b + P_t t_d$ and

$$\phi(2, T_G) = 2P_h t_a + 0.5P_t t_a + 1.5P_t t_b.$$

Remark: Plugging the conditions into (8), we can show the proposed Shapley value based tail energy accounting policy satisfy both *Positive Reward* and *Fairness* properties.

For the illustrative example in Figure 3, both software principals receive in Scenario (a) $\phi(1, T_G) = \phi(2, T_G) = 4P_h t_a + 1.5P_t t_b + 0.5P_t t_d$. When software principal 1 bundles data transmission in Scenario (b), its accounted energy share reduces to $\phi(1, T_G) = 4P_h t_a + 0.5P_t t_d$. For the example in Figure 4, both software principal 2 and the symmetric 2' receive $P_h t_a - 0.5P_t t_a + 0.5P_t t_d$, and this satisfies the *Fairness* property.

The capability to properly attribute tail energy consumption to different software principals is the key to addressing a wide range of problems, including evaluating apps' energy profile [1], identifying energy-friendly coding behaviors [21], energy optimization through prefetching [9], [1], proxy-assisted browsing [26], content pre-staging [27], and coalesced offloading [28], [29].

3.3 Proposed Algorithmic Solution for MAPEO

Solving MAPEO problem to jointly optimize all apps' prefetching strategies is not trivial. While the number of prefetched ads ($R_{i,m}$) should not exceed maximum number of ads that can be displayed during an interval (denoted by $n_i = \sup\{x : \mathcal{P}_{i,m}(x) > 0\}$), the joint optimization of all K apps' prefetching strategy still has an exponential complexity of $\prod_{i=1}^K n_i$. Inspired by our tail energy accounting policy, we propose an algorithmic solution that makes each app i independently maximize its energy-aware utility $U_{i,m}/E_{i,m}$ over its own prefetching decision $R_{i,m}$. This decouples MAPEO problem into K problems, which are easier to solve (over a space of size n_i only) and can be computed in parallel.

In practice, the actual number of displayed ads $A_{i,m}$ in interval D_m are unknown beforehand, however, its distribution can be estimated from history app trace, as evidenced in [1], [30].

Our proposed algorithm works as follows. At the beginning of each interval D_m , we estimate the distribution $\mathcal{P}_{i,m}(x)$ of $A_{i,m}$ for each app for D_m . In specific, we first check what time during a day (e.g. 9 : 30AM to 10AM) does the D_m correspond to. Then we query the user's history traces to find the intervals representing the same time of the day and summarize the distributions of $A_{i,m}$ for all i . In parallel, we apply our Shapley value-based tail energy accounting policy to estimate per-ad energy $E_{1,i,m}$ and $E_{2,i,m}$, as well as $E_{0,i,m}$ from previous intervals with the available energy measurement using our software-based Energy Meter module. This allows us to estimate energy consumption $\hat{E}_{i,m}$ for varying $R_{i,m}$ in interval D_m . Next, we optimize each app's prefetch decision $R_{i,m}$ to maximize its energy-aware utility $U_{i,m}/E_{i,m}$. This is an optimization involving only a single variable $R_{i,m}$ and can be solved efficiently and recursively using exhaustive search for all $R_{i,m} \leq n_i$ until the solution converges. Finally, we prefetch $R_{i,m}$ ads for each app i .

The algorithm is summarized in Figure 6. It finds optimal prefetch decision $R_{i,m}$ in real-time. Because distribution

```

// (a) Account network energy of  $D_{m-1}$ 
Apply Equation (8)
// (b) Optimize  $R_{i,m}$  for interval  $D_m$ 
for all  $i \in K$ :
  (b.0) Initialize:  $R_{i,m} = 0, Max = 0$ 
  (b.1) Predict  $\mathcal{P}_{i,m}(x)$  from  $i$ ' history app usage
  Assign  $n_i = \sup\{x : \mathcal{P}_{i,m}(x) > 0\}$ 
  Get the probability of  $A_{i,m} = x, \forall x < n_i$ 
  (b.2) Estimate  $E_{1,i,m}, E_{2,i,m}$  and  $E_{0,i,m}$  from (a)
  (b.3) Optimize  $R_{i,m}$ 
  for all  $r < n_i - S_{i,m}$ :
    Calculate  $\mathcal{E}[U_{i,m}/E_{i,m}]$  by  $\mathcal{P}_{i,m}(x), U_{i,m}, E_{i,m}$ :
    Calculate  $U_{i,m}$  using Equation (3);
    Calculate  $E_{i,m}$  using Equation (5);
    if  $\mathcal{E}[U_{i,m}/E_{i,m}] > Max$ :
       $Max = \mathcal{E}[U_{i,m}/E_{i,m}], R_{i,m} = r$ 
    end if
  end for
end for
// (c) Record relevant information during period  $D_m$ 
for all active app  $i \in D_m$ 
  Record  $A_{i,m}$ 
  Record app  $i$ 's ad fetching sessions
end for

Calculate  $S_{i,m+1}$  using Equation (4)
 $m = m+1$ 
Go to (a)

```

Fig. 6: Proposed algorithm for MAPEO problem.

$\mathcal{P}_{i,m}(x)$ and energy consumption $\hat{E}_{i,m}$ are updated dynamically for each interval, the proposed online algorithm has the ability to adapt to changes in user behavior and network states.

4 IMPLEMENTATION

We fully implement the ad prefetching and management system, with key modules and workflow shown in Figure 7. In specific, the implementation includes four key modules: Traffic Filter, Ad Proxy, Ad Counter, and Optimization Engine.

4.1 Traffic Filter

Traffic Filter selects ad-related traffic and redirects to Ad Proxy. The difficulty to directly apply *iptables* for this goal is that possible IP addresses for all the ad libraries are prohibitively many. We tackle this problem by jointly leveraging *iptables* and governing DNS service by *DNSmasq* [31]. Our governed DNS assures that all requests targeting a specific ad server will be sent to one single IP, so they can be captured by a single entry in *iptables*. The hostname-IP pairs maintained for DNS service are automatically and periodically updated to guarantee the validity of the IPs. In this way, we guarantee that the ad traffic is directed to Ad Proxy by minimal number of *iptables* rules, so that the overhead is minimized.

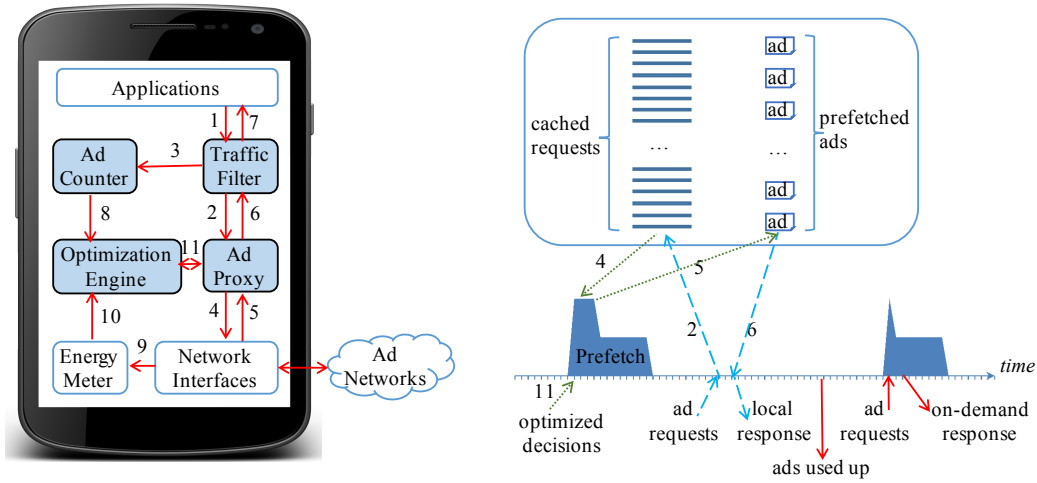


Fig. 7: Overview of our ad management system

4.2 Ad Proxy

The workflow of Ad Proxy is depicted in the right part of Figure 7. The indexes on arrows correspond to those in the left part of the figure. At the beginning of an optimization interval, optimized decisions from the Optimization Engine govern Ad Proxy to prefetch certain numbers of ads and store them in the ad cache for corresponding apps. Later, when an app requests an ad, if unused ads are available, the ad request is firstly parsed by Ad Proxy, which will select a best-match of ad response to process the request without triggering any network activities. The matching of ad responses to requests is a non-trivial task since poor matching will make the ad fail to display. We leverage the requests' destination address and detailed header information to conduct best matching. For HTTPS traffic, the certificates are specially cached. To prevent the same ad being reused, the proxy will mark any used ad as unavailable for future. Before next interval of prefetching, if an ad request arrives after all prefetched ads are used up, Ad Proxy fetches an ad from the external ad networks in an on-demand manner.

4.3 Ad Counter

The numbers of ads ($A_{i,m}$, $R_{i,m}$, and $O_{i,m}$) are key parameters in our MAPEO formulation (7). Difficulties exist for recording the number of ads, because the number of request/response pairs can be random for a single ad. Existing ad counting method [32] requires hardware/system modification and cannot easily apply to existing mobile platforms.

We use machine learning approach [33] to extract the indicating features for identifying the starting requests of ads. We build a Python script to crawl the top ad-embedded free apps from Google Play, and use Apktool [34] to decompile the packages of apps to check whether ad libraries are imported into the package. Afterward, we use MonkeyRunner [35] to run those apps for an extensive length of time on experiment phones to obtain an ad dataset for the training of the feature extraction algorithm. We intentionally prolong the period length between two consecutive launches of a tested app to separate groups of request/response pairs belonging to different ads, in order to facilitate marking of

starting requests for ads in the collected dataset. Then the starting requests in the collected dataset are automatically tagged by utilizing the synchronized timestamps between Ad Proxy and MonkeyRunner. After the training phases, we utilize the extracted features and mark the ad requests with matched features as the starting requests. In this lightweight way of ad counting, we guarantee the overhead introduced is minimal.

4.4 Optimization Engine

The governing module is Optimization Engine, which coordinates with Ad Counter and Energy Meter, and provides optimized decisions for Ad Proxy. The data provided by Ad Counter to Optimization Module includes statistics about numbers of ads. The Energy Meter provides timestamped power traces consumed by network activities to Optimization Engine. Optimization Engine takes care of both Shapley value-based tail energy accounting and energy optimization for ad prefetching.

4.4.1 Energy Accounting

To account network energy and enable real-time energy optimization, we need to first estimate the overall network energy. We infer the state transfer delays of different networks (3G, 4G, and LTE) and service providers based on results revealed in prior network characterization work [3], [4], [36]. For different phone models and carriers, we calibrate the network interfaces' average power consumption P_t and P_h in different working states by regression-based method [36], [13]. The regression uses training data obtained from real measurements by external Power Monitor [37], which is popularly used in existing work [13]. The state machine based approach is implemented in our Energy Meter module to record the total network power consumption.

The power measurement obtained from Energy Meter module in Figure 7 consists of timestamp and power consumption value tuples. With the timestamps information, we are able to divide the whole measurement into separate energy accounting games, each of which starts with the first data transmission that drives the network interface to leave the idle state, and ends when both the data transmission

and a whole tail is finished. Finally, combining the power measurement with numbers of each app's prefetched ads and on-demand fetched ads from Ad Counter, we are able to apply the efficient method for computing Shapley value discussed in Section 3.2.3 and account the total network energy to individual apps.

4.4.2 Energy Optimization

We implement the algorithmic solution (Figure 6) to the deadline-aware MAPEO problem in Optimization Engine, which also maintains the statistics and accounted energy of history optimization intervals in the database for each app. At the beginning of each optimization session, the Optimization Engine will query the history database, and apply the implemented algorithm to predict the optimal number of ads to prefetch for each app for next interval. The optimized decisions are fed into the Ad Proxy. At the end of each optimization interval, the Optimization Engine will summarize the statistics during the interval and store the data in the database for later use in following intervals.

5 EVALUATION

In this section, we evaluate the performance of our ad prefetching system, comparing our proposed energy accounting policy against different accounting policies from existing work and the Tail Unaware policy, which is also the default on-demand ad fetching mechanism adopted by most contemporary mobile ad libraries. We first evaluate these policies in the context of energy management, i.e. governing ad prefetch optimization. Further, using our proposed Shapley value-based accounting policy, we also illustrate varied optimal prefetching strategies for different apps. Finally, we evaluate the runtime system overhead.

5.1 Experiment Setup

5.1.1 Hardware Setup

Our implementation and evaluation are based on Android and we tune the parameters used in our Energy Meter module for three different phone models: Samsung Galaxy Nexus, Galaxy Edge 6, and Galaxy Note 5. We use the Monsoon Power Monitor [37] to measure the system power consumption. To infer the network interface power/energy consumption from network activities, we adopt a similar regression model used in existing work [36], [13]. All experiments are performed with contemporary AT&T LTE network.

5.1.2 Test User Traces and Apps Preparation

We collect 7 user traces out of Rice LiveLab traces [20], most of which are more than 150 days. The traces record the timestamps a user launches each app and the lengths of usage durations after app launches. We collect the top-ranking apps from Google Play as the testing apps.

Monkey Training: To minimize environment variation among experiments, we use the automated testing tool Monkey [35] to replay user traces. Although apps normally make the first ad request during launch time [38], for some apps, ad requests can only be triggered by certain

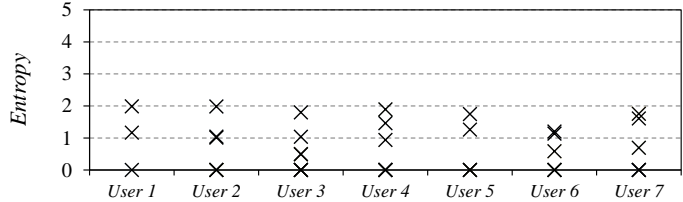


Fig. 8: Entropy of the actual number of ads used for different users (each point denotes a separate app)

user actions. Also, the frequency of ads depends on user behaviors. For example, in a game app, a new ad may be shown only when users restart a game or enter the next level. For such cases, we tune the combination and density of Monkey events for each app to best emulate a real user's behaviors. With a separate Monkey trace constructed for each app, we write a script to extract the list of apps and timing information from user traces, then utilize the separate Monkey traces to construct a combined trace for each user.

Predictability Study: We conduct a study on the predictability of the number of ads used by apps. Existing work [1] studied on raw user traces (with only theoretical number of ad slots) and showed the average entropy⁴ for 60-minute periods is around 3.5. With system implementation, we are able to replay real user traces on real devices and get the actual numbers of ads displayed, rather than simply trace-driven simulations as in existing works [9], [1], [24]. Since MAPEO formulation and our energy accounting policy enables a distributed manner (Section 3.3) to optimize the decisions separately for individual apps. We get the probability mass function $\mathcal{P}_{i,m}(x)$ of the number of ads displayed via each app i for consecutive intervals $\mathcal{D} = \{D_1, D_2, \dots, D_m, \dots\}$. The length of each interval is our optimization period length 30 minutes. Afterward, we calculate the entropy for each app by $H = -\sum_x \mathcal{P}_{i,m}(x) \log_2[\mathcal{P}_{i,m}(x)]$. Assuming 60-second [2] ad refreshing period, the theoretical upper bound for entropy within a 30-minute period is $\log_2(31) = 4.9542$. The entropies for individual apps are marked separately for each user in Figure 8. The entropies vary for different apps and users, but are generally closer to 0 than the upper bound, implying the predictability.

5.1.3 Ad Offline Display Verification

To fully verify that our system can enable offline ad functionality, we disconnect the phone from any network connections. We notice that some ad libraries only send ad requests when networks are available. To solve this issue, we implement a hook to Android's *WifiManager* using *Xposed* [39] to emulate a fake Wi-Fi status⁵. When prefetched ads are available for an app Compass [40], we start the app and verify ad is successfully shown when network is unavailable, as seen in the left of Figure 9. Such

4. Smaller entropy implies smaller uncertainty within the variable and hence, higher predictability.

5. The hook to *WifiManager* is only for demonstrating the effectiveness of the prefetching scheme when no network is available, our whole system doesn't rely on such hook.

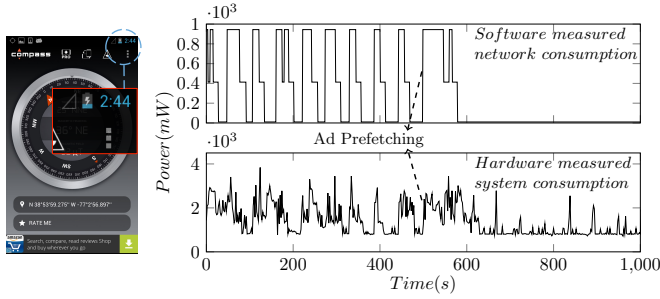


Fig. 9: Screenshot of shown ad when no network is available (left), and network interface power (top right) measured by software-based Energy Meter along with total system power (bottom right) measured by Monsoon Power Monitor.

feature is desirable for ad libraries, which enables offline ad impressions when network is unavailable but users tend to check their devices.

5.1.4 Energy Measurement

To test the correctness of energy measurement, and preliminarily study the effect of ad prefetching on energy, we pick a segment of trace when a user performs consistent actions in an app so that ads are also requested every minute. We enable our design in the middle of the trace.

The network power measured by the software-based Energy Meter is depicted at the top right of Figure 9, along with the hardware measured system power, which is depicted at the bottom right of Figure 9. As seen in the figure, in the first 500 seconds without prefetching, the hardware power measurement shows that each time an ad is requested, relatively consistent amount of energy is generated, and the software measured power shows a tail is triggered. At the beginning of the second half (at 501 seconds) of the trace, a relatively large amount of energy is generated in a short time (marked by arrow), and this is when optimization takes effect and prefetching happens. Afterward, when an ad is requested, the system power consumption is much smaller than that in the first half, and ads are successfully shown in the app without triggering any network energy tail. Summing up separately the energy consumption of the first and second periods, we found that during the second period (i.e. 501 - 1000 seconds), system overall energy and network energy are respectively $225.66J$ and $147.85J$ less than the first period.

5.2 Evaluating MAPEO Energy Management under Different Accounting Policies

The preliminary experiment in Figure 9 (right) shows ad prefetching is effective to save network and system energy. However, in real world scenarios, usage of different apps overlaps. To evaluate the efficiency of different policies in per-process energy management, we plug in our proposed Shapley value-based energy accounting and three accounting policies discussed in Section 3.2.1 into our system to consist of four experiment setups: (a) Tail Unaware, (b) First Trigger, (c) Last Trigger, and (d) Shapley. App First policy is left out because it doesn't apply when multiple apps are active.

TABLE 1: Breakdown of network energy and achieved utility (normalized by Shapley) for two users

Policy	User 1		User 2	
	Energy (J)	Utility	Energy (J)	Utility
Tail Unaware	2558.515	116%	2706.076	111%
First Trigger	2056.923	83%	2136.691	97%
Last Trigger	2188.082	79%	2268.402	102%
Shapley	2057.826	100%	2150.725	100%

We replay the 7 user traces for each group and set ad deadline as 30 minutes⁶. The evaluation metric is the ratio of network energy to the utility achieved by displayed ads.

As shown in Figure 10, Shapley value-based accounting policy consistently outperform the other three. For example, for user 1, First Trigger and Last Trigger policies are even worse than Tail Unaware policy, and respectively consume 20% and 34% more energy to achieve the same amount of utility as Shapley value-based policy. For the sake of space, we show the breakdown details for the first two representative users in Table 1 (for user 1, Shapley largely outperforms First Trigger and Last Trigger, for user 2, these three policies are close, but all outperform Tail Unaware policy). As seen in the table, First Trigger, Last Trigger, and Shapley all consume much less energy than Tail Unaware policy, but also get penalized utility because of ad deadline violations. This is because, under Tail Unaware policy, the cost to prefetch one ad is equivalent to fetch one on-demand, apps never prefetch ads to avoid deadline violations. For user 1, First Trigger and Last Trigger respectively achieve 17% and 21% less utility than Shapley. Careful investigations of the user 1's evaluation data traces in the database reveal that under Last Trigger and First Trigger policies, when apps have overlapped network activities, tail energy is always accounted to a single app. This leads to an unfair situation that one app tends to under-prefetch and the others over-prefetch ads for next interval. The under-prefetched apps later need to fetch more ads on-demand, and the over-prefetched apps fail to display prefetched ads before the deadlines, thus get penalized for utility. This confirms that the *Fairness* property discussed in Section 3.2.2 should hold for proper accounting policies. Investigation of user 2's usage traces shows that this user has much less overlaps among different apps' usage, in which case, apps have less shared network tail energy. As a result, First Trigger, Last Trigger, and Shapley are close in terms of utility achieved, while all save a certain amount of energy than Tail Unaware policy. Among all the traces evaluated, unlike user 2, most users have a large amount of overlapped usage of different apps.

Note that the only difference between the four experiment groups is the energy accounting policy. The proposed MAPEO algorithm and ad prefetching are performed in each group. Such controlled experimental comparison demonstrates that the smaller Energy/Utility ratio achieved by Shapley group is due to the Shapley value-based energy accounting policy used to guide energy optimization.

6. It is observed in prior work [1] that more than 99.5% of ads keep the dynamic price unchanged in 30 minutes.

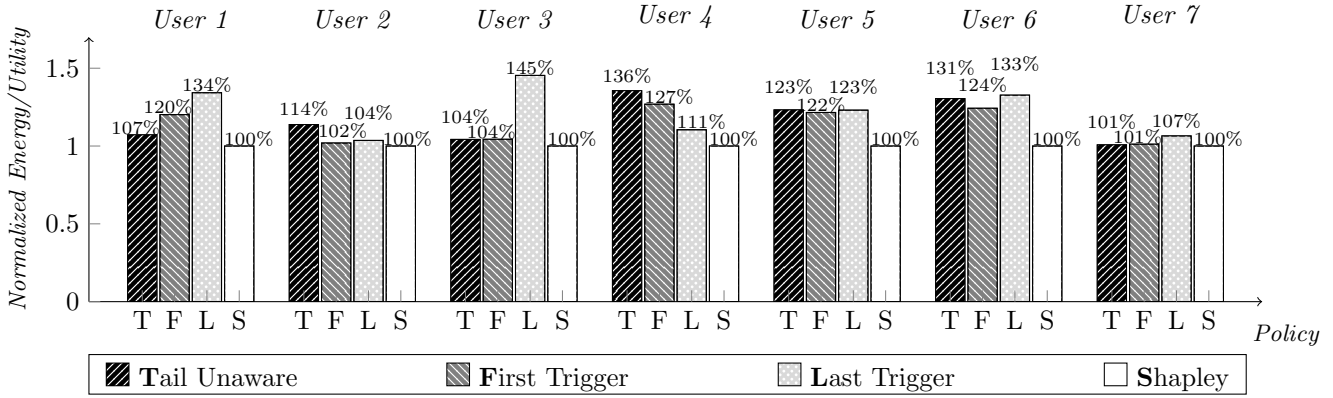


Fig. 10: Energy/Utility ratios archived by different accounting policies, normalized by these achieved by Shapley(S)

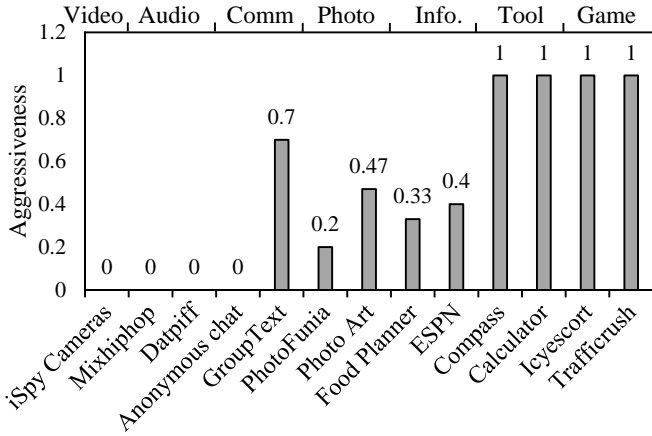


Fig. 11: Prefetching aggressiveness measured for different apps under Shapley value-based accounting

5.3 Illustrating Optimal Strategies for Different Apps

We investigate the optimal ad prefetching strategies of different apps under Shapley value-based accounting policy. We preselect 13 top free apps with ads from Google Play and play the same set of 7 user traces in Section 5.1.2. For each app i and intervals D_m , we study the number of residue ads from previous interval $S_{i,m}$, number of ads prefetched $R_{i,m}$, and maximum number of ads used in history sessions from 1 to $m - 1$, as denoted by $\sup\{x : \mathcal{P}_{i,m}(x) > 0\}$ in Section 3.3. Calculating the “aggressiveness” of app i for D_m by $\frac{S_{i,m} + R_{i,m}}{\sup\{x : \mathcal{P}_{i,m}(x) > 0\}}$, we summarize the average for each app from all user traces in Figure 11.

As seen from the figure, the optimal prefetching *aggressiveness* varies significantly for different apps. The interesting pattern is that the *aggressiveness* level is large for those Tool and Game apps whose network activities purely originate from ad modules. For the apps that generate consistent network activities such as Audio, Video and online chatting app (Anonymous chat), the *aggressiveness* level is 0, which imply ad prefetching is unnecessary for these apps. The other apps all have diverse optimal *aggressiveness* levels. Hence, this verifies a single *percentile* chosen for the whole system [1] fails to meet the requirements from all apps for

exploring optimal utility-energy tradeoff. With our formally formulated systematic framework, the implemented Optimization Engine makes such optimal decisions for every single app distributively and effectively, without the need for knowing the details of the apps.

5.4 Measuring Overhead on Real Devices

5.4.1 Overall System Overhead

To measure our design’s overhead on real devices, we use a popular benchmark *softweg* [41]. We run *softweg* for 20 times on Android, with and without our design. The performance differences are listed in Table 2. ‘RSD’ refers to scores’ relative standard deviation. It is seen that with our implementation, only 0.74% and 6.87% overhead are introduced to CPU and file system. The average of all performance overhead scores is 1.72%. The negative memory overhead observed results from score dispersion and is insignificant compared with the ‘RSD’, which is 6.3%.

5.4.2 Network Latency Overhead

To measure the effects of our design on network latency, we measure the delay between SYN and SYN-ACK packets as in [4], i.e. RTT for TCP connections. We again replay the user traces with and without our system design enabled, on Samsung Galaxy Note 5 with LTE network. We intentionally increase time intervals between Monkey events to facilitate pairing of SYN and SYN-ACK packets for the same connection. The measured mean RTT for LTE with original Android is 68.911 ms, and it becomes 72.042 ms with our system design, resulting only 4.54% network latency overhead. Note that such overhead is only introduced when network connection is necessary, however, when prefetched ads are available, both the whole RTT and the energy tail are avoided.

6 RELATED WORK

Energy accounting policies studied (explicitly or implicitly) in prior work [14], [13], [16], [15], [12] often ignore tail energy consumption, while [19], [9], [1] are notable exceptions, but they rely on simple heuristics and fail to satisfy certain key properties as identified in this paper. Tail energy accounting remains an open problem, and there is no known ground truth against which a policy could be evaluated.

TABLE 2: Overall system performance overhead

	with our design		original Android		Overhead
	mean	RSD	mean	RSD	
graphics score	19.59	1.88%	19.77	1.91%	0.91%
cpu score	3398.74	2.53%	3424.10	1.76%	0.74%
memory score	531.34	10.48%	522.70	6.30%	-1.65%
file system score	222.89	8.80%	239.34	7.17%	6.87%

Our work is also different from [17], which rely on external measurements and is not applicable to asynchronous tail energy consumption.

Various techniques are proposed to optimize mobile energy consumption in different problem domains, such as proxy-assisted browsing [26], content pre-staging [27], transmission protocol improvement [42], energy management [43], [17], coalesced offloading [29], [28], prefetching [22], [44], [23], caching [24], background workload management [45], [46], [22], [47], [48], study of app and user behaviors [49], [21], [30], [50], [51]. Work in [52] decreases latency by prefetching, but consumes more energy. Existing work on mobile ad prefetching [9], [1] often use simple heuristics, yet make strong assumptions such as single active app, fixed ad refreshing rate, and same ad type. There is no formal systematic framework for mobile ad prefetching and energy optimization before.

Another line of work on ad detection [33], [32], recommendation [38], [53] and pricing mechanisms [8] are complementary with our work, and can be easily plugged into our system, potentially opening up interesting avenues for researches. Existing work on network measurement [3], [36], [4], [54], [22], [55], [56], [57] can also provide useful input to our optimization.

7 CONCLUSION

In this paper, we propose a novel framework for mobile ad prefetching and energy optimization, which maximizes the sum “energy-aware ad utility” of all apps. The utility takes into account of the timeliness of mobile ads, since any prefetched ads that are displayed after desired deadlines only receive diminished utility. Our solution consists of a Shapley value-based policy for tail energy accounting in multiprocessing mobile systems, and an efficient distributed algorithm jointly optimizing all apps’ prefetching strategies. The proposed policy is proven to satisfy two crucial properties - *Positive Reward* and *Fairness*, which are violated by existing policies in prior work. We fully implement the framework on Android and show that the system and network overhead is almost negligible. Our work achieves up to 45% energy savings than existing policies using real-world user traces and it is transparent to mobile apps and contemporary ad ecosystem.

REFERENCES

[1] Prashanth Mohan, Suman Nath, and Oriana Riva. Prefetching mobile ads: Can advertising systems afford it? In *EuroSys*. ACM, 2013.

[2] Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. Breaking for commercials: Characterizing mobile advertising. In *IMC*. ACM, 2012.

[3] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Characterizing radio resource allocation for 3G networks. In *IMC*. ACM, 2010.

[4] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *MobiSys*. ACM, 2012.

[5] Ana Nika, Yibo Zhu, Ning Ding, Abhilash Jindal, Y Charlie Hu, Xia Zhou, Ben Y Zhao, and Haitao Zheng. Energy and performance of smartphone radio bundling in outdoor environments. In *Proceedings of the 24th International Conference on World Wide Web*, pages 809–819. International World Wide Web Conferences Steering Committee, 2015.

[6] Xiaomeng Chen, Ning Ding, Abhilash Jindal, Y Charlie Hu, Maruti Gupta, and Rath Vannithamby. Smartphone energy drain in the wild: Analysis and implications. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):151–164, 2015.

[7] Alok Tongaonkar, Shuaifu Dai, Antonio Nucci, and Dawn Song. Understanding mobile app usage patterns using in-app advertisements. In *Passive and Active Measurement*, pages 63–72. Springer, 2013.

[8] Azeem J Khan, Kasthuri Jayarajah, Dongsu Han, Archan Misra, Rajesh Balan, and Srinivasan Seshan. CAMEO: A middleware for mobile advertisement delivery. In *MobiSys*. ACM, 2013.

[9] Xiaomeng Chen, Abhilash Jindal, and Y. Charlie Hu. How much energy can we save from prefetching ads?: Energy drain analysis of top 100 apps. In *HotPower*. ACM, 2013.

[10] Lloyd S Shapley. A value for n-person games. Technical report, DTIC Document, 1952.

[11] Mobile Radio Power Calculator in Android . <http://goo.gl/UnmaBG>.

[12] Jason Flinn and Mahadev Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Second IEEE Workshop on Mobile Computing Systems and Applications*. IEEE, 1999.

[13] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Z. Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *CODES/ISSS*. ACM, 2010.

[14] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *EuroSys*. ACM, 2011.

[15] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulwoo Kang, and Hojung Cha. AppScope: Application energy metering framework for android smartphones using kernel activity monitoring. In *ATC. USENIX*, 2012.

[16] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *MobiCom*. ACM, 2012.

[17] Mian Dong, Tian Lan, and Lin Zhong. Rethink energy accounting with cooperative game theory. In *MobiCom*. ACM, 2014.

[18] Yongbo Li and Tian Lan. Multichoice games for optimizing task assignment in edge computing. In *2018 IEEE Global Communications Conference*. IEEE, 2018.

[19] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with Eprof. In *EuroSys*. ACM, 2012.

[20] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Phillip Kortum. LiveLab: measuring wireless networks and smartphone users in the field. In *ACM SIGMETRICS Performance Evaluation Review*, 2011.

[21] Abhijeet Banerjee, Lee Kee Chong, Sudipta Chattopadhyay, and Abhik Roychoudhury. Detecting energy bugs and hotspots in mobile apps. In *SIGSOFT FSE*. ACM, 2014.

[22] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *IMC*. ACM, 2009.

[23] Chao Wu, Xu Chen, Yuezhi Zhou, Ningyuan Li, Xiaoming Fu, and Yaoyue Zhang. Spice: Socially-driven learning-based mobile media prefetching. In *INFOCOM*. IEEE, 2016.

[24] Maria Carpen Amarie, Ioannis Pefkianakis, and Henrik Lundgren. Mobile video ad caching on smartphones. In *UbiComp*. ACM, 2014.

[25] Martin Shubik. Incentives, decentralized control, the assignment of joint costs and internal pricing. *Management science*, 8(3):325–343, 1962.

[26] Ashiwan Sivakumar, Shankaranarayanan Puzhavakath Narayanan, Vijay Gopalakrishnan, Seungjoon

- Lee, Sanjay Rao, and Subhabrata Sen. PARCEL: Proxy assisted browsing in cellular networks for energy and latency reduction. In *CoNEXT*. ACM, 2014.
- [27] Alessandro Finamore, Marco Mellia, Zafar Gilani, Konstantina Papagiannaki, Vijay Erramilli, and Yan Grunenberger. Is there a case for mobile phone content pre-staging? In *CoNEXT*. ACM, 2013.
- [28] Ali Sehati and Majid Ghaderi. Energy-delay tradeoff for request bundling on smartphones. In *INFOCOM*. IEEE, 2017.
- [29] Liyao Xiang, Shiwen Ye, Yuan Feng, Baochun Li, and Bo Li. Ready, set, go: Coalesced offloading from mobile devices to the cloud. In *INFOCOM*. IEEE, 2014.
- [30] Abouzar Rahmati and Lin Zhong. Studying smartphone usage: Lessons from a four-month field study. *TMC*, 12(7):1417–1427, 2013.
- [31] Dnsmaq. <http://www.thekelleys.org.uk/dnsmaq/doc.html>.
- [32] Wenhao Li, Haibo Li, Haibo Chen, and Yubin Xia. Adattester: Secure online mobile advertisement attestation using trustzone. In *MobiSys*. ACM, 2015.
- [33] Jonathan Crussell, Ryan Stevens, and Hao Chen. MAdFraud: Investigating ad fraud in android applications. In *MobiSys*. ACM, 2014.
- [34] Reverse engineering tool Apktool. <http://ibotpeaches.github.io/Apktool/>.
- [35] UI/Application exerciser monkey. <https://goo.gl/yi99ig>.
- [36] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. An in-depth study of LTE: Effect of network protocol and application behavior on performance. In *SIGCOMM*. ACM, 2013.
- [37] Monsoon Powermeter. <https://goo.gl/bB96ta>.
- [38] Suman Nath. Madscope: Characterizing mobile in-app targeted ads. In *MobiSys*. ACM, 2015.
- [39] Xposed framework. <https://goo.gl/NgfpCJ>.
- [40] App Compass from google Play Store. <https://play.google.com/store/apps/details?id=com.gn.android.compass>.
- [41] Mobile benchmark suite softweg. <https://goo.gl/8G3lDm>.
- [42] Wei Wang, Yingjie Chen, Lu Wang, and Qian Zhang. From rateless to sampleless: Wi-fi connectivity made energy efficient. In *INFOCOM*. IEEE, 2016.
- [43] Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. ECOSystem: Managing energy as a first class operating system resource. In *ASPLOS X*. ACM, 2002.
- [44] Brett D Higgins, Jason Flinn, Thomas J Giuli, Brian Noble, Christopher Peplin, and David Watson. Informed mobile prefetching. In *MobiSys*. ACM, 2012.
- [45] Abhijnan Chakraborty, Vishnu Navda, Venkata N Padmanabhan, and Ramachandran Ramjee. Coordinating cellular background transfers using loadsense. In *MobiCom*. ACM, 2013.
- [46] Xiaomeng Chen, Abhilash Jindal, Ning Ding, Yu Charlie Hu, Maruti Gupta, and Rath Vannithamby. Smartphone background activities in the wild: Origin, energy drain, and optimization. In *MobiCom*. ACM, 2015.
- [47] Fengyuan Xu, Yunxin Liu, Thomas Moscibroda, Ranveer Chandra, Long Jin, Yongguang Zhang, and Qun Li. Optimizing background email sync on smartphones. In *MobiSys*. ACM, 2013.
- [48] Shravan Aras and Chris Gniady. Greentouch: Transparent energy management for cellular data radios. In *UbiComp*. ACM, 2016.
- [49] Lenin Ravindranath, Sharad Agarwal, Jitendra Padhye, and Chris Riederer. Procrastinator: pacing mobile apps' usage of the network. In *MobiSys*. ACM, 2014.
- [50] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *MobiSys*. ACM, 2010.
- [51] Wenjie Hu and Guohong Cao. Energy-aware video streaming on smartphones. In *INFOCOM*. IEEE, 2015.
- [52] Yichuan Wang, Xin Liu, David Chu, and Yunxin Liu. Earlybird: Mobile prefetching of social network feeds via content preference mining and usage pattern analysis. In *MobiHoc*. ACM, 2015.
- [53] Suman Nath, Felix Xiaozhu Lin, Lenin Ravindranath, and Jitendra Padhye. SmartAds: bringing contextual ads to mobile apps. In *MobiSys*. ACM, 2013.
- [54] Ashkan Nikraves, Hongyi Yao, Shichang Xu, David Choffnes, and Z. Morley Mao. Mobilyzer: An open platform for controllable mobile network measurements. In *MobiSys*. ACM, 2015.
- [55] Andrius Aucinas, Narseo Vallina-Rodriguez, Yan Grunenberger, Vijay Erramilli, Konstantina Papagiannaki, Jon Crowcroft, and

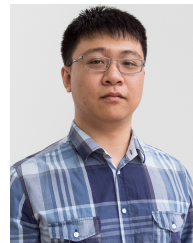
David Wetherall. Staying online while mobile: The hidden costs. In *CoNEXT*. ACM, 2013.

- [56] Duc Hoang Bui, Yunxin Liu, Hyosu Kim, Insik Shin, and Feng Zhao. Rethinking energy-performance trade-off in mobile web page loading. In *MobiCom*. ACM, 2015.

- [57] Shuai Wang, Song Min Kim, Yunhuai Liu, Guang Tan, and Tian He. Corlayer: A transparent link correlation layer for energy efficient broadcast. In *MobiCom*. ACM, 2013.



Yongbo Li received his Ph.D. degree from the Department of Electrical and Computer Engineering at the George Washington University in 2018. His research interests include mobile energy optimization, edge computing, and system security. He will join Facebook as a research scientist in 2018.



Yimeng Wang received the BS degree in communication engineering from Beijing Jiaotong University, in 2012. He received the MS degree from the George Washington University in 2015. He is pursuing the PhD degree in GWU, focusing on mobile energy optimization and mobile edge computing.



Tian Lan received the BS degree in Electrical Engineering from Tsinghua University in 2003, MS degree from the Department of Electrical and Computer Engineering at the University of Toronto in 2005. He received the PhD degree from the Department of Electrical Engineering at the Princeton University in 2010. He is an Associate Professor in the Department of Electrical and Computer Engineering at the George Washington University, which he joined in 2010. He received the best paper award from IEEE

INFOCOM 2012, IEEE GLOBECOM 2009 and IEEE Signal Processing Society 2008. His research interests include mobile energy accounting, cloud computing, and cyber security.