Can Mobile-to-Mobile Browser Cache Cooperation Reduce Energy Consumption of Internet Access?

Abdul Alim Abd Karim[1] Ajay Sharma[1] Mahbub Hassan[1]

Aruna Seneviratne[2]

[1] University of New South Wales, Australia
{aaak656,ajays,mahbub,aruna}@cse.unsw.edu.au
[2] University of New South Wales, Australia
a.seneviratne@unsw.edu.au

Technical Report
UNSW-CSE-TR-201302
January 2013

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

This paper investigates the possibility of reducing 3G browsing energy consumption of a mobile device by opportunistically fetching browser cache contents of a nearby device over the low-energy Bluetooth connection. By analysing a generic model of component-based web pages, we show that energy reduction is dependent on the fraction of dynamic components of a loaded page (dynamic page factor) which cannot be cached effectively. We find that energy reduction for a given page is possible only if the dynamic page factor is below a given threshold. We have designed and implemented an Android-based cooperative browser caching prototype and collected energy data in two different locations for 16 popular pages whose dynamic page factors ranged from zero to 0.6. Our experimental results confirm that cooperative browsing reduces energy cost only if the dynamic page factor is below a threshold and that this threshold is very small (about 0.03). This finding suggests that cooperative browsing may be counter productive if browsing has a bias toward pages with large dynamic page factors and vice-versa. For unbiased browsing, however, we find that short-range cache cooperation can reduce 3G browsing energy consumption by 13%. Finally, we propose a dynamic decision making algorithm that switches between non- cooperative and cooperative browsing adaptively based on the dynamic page factor. For our empirical data, the proposed algorithm could potentially achieve a 17% reduction in browsing energy consumption.

1 Introduction

Recent studies have reported that Internet browsing is one of the major sources of energy consumption in mobile devices [6, 16]. This is especially the case when browsing over 3G, which requires the mobile to use high power due to the long distance between the mobile and the radio tower. In general terms, 3G or any cellular-based radio is called long-range (LR) and Bluetooth or WiFi is called short-range (SR). Data access over SR typically consumes much less energy compared to LR. Due to this reason, and given the fact that most mobile devices now include both LR and SR radios, researchers have been investigating the possibility of opportunistically offloading data access from LR to SR. A particular trend has been to use mobile-to-mobile cooperative caching, where data accessed over LR and cached by one mobile is accessed by another nearby mobile over the SR [15, 7]. The main focus of these works has been on developing cache management algorithms that improve the cache hit ratio thereby increasing the potential for energy saving.

Although previous works [15, 7] show that by improving the cache hit ratio it is possible to achieve energy saving in cooperative caching systems, our research indicates that it is not always the case when accessing most of todays Internet content. This is because of the presence of dynamic components, such as frequently changing advertisements, in many popular pages, which cannot be cached effectively. The ratio of dynamic components over all the components in an web page, which we refer to as the *dynamic page factor*, therefore, acts as a limiting factor or an upper bound for how much energy savings that can be achieved through cache cooperation over SR. We, therefore, believe that the *dynamic page factor* is crucial for determining the energy performance of mobile-to-mobile cooperative caching schemes. However, to the best of our knowledge, no study has been reported in the literature that explores the effect of dynamic page factor on the energy consumption of cooperative caching. This motivates our work.

The novelty and contribution of this paper can be summarized as follows:

- A theoretical analysis of the effect of dynamic page factor on the energy consumption of cooperative browser caching. We show that energy reduction for a given page is possible only if the dynamic page factor is below a given threshold.
- We have designed and implemented an Android-based cooperative browser caching prototype. Using our prototype, we have collected energy data in two different locations for 16 popular pages whose dynamic page factors ranged from zero to 0.6.
- Our experimental results confirm that cooperative browsing reduces energy cost only if the dynamic page factor is below a threshold and that this threshold is very small (about 0.03). This finding suggests that cooperative browsing may be counter productive if browsing has a bias toward pages with large dynamic page factors and vice-versa. For unbiased browsing, however, we find that short-range cache cooperation can reduce 3G browsing energy consumption by 13%.
- Finally, we propose a dynamic decision making algorithm that switches between non-cooperative and cooperative browsing adaptively based on

the dynamic page factor. We show that such dynamic decision could potentially achieve a 17% reduction in browsing energy consumption.

The rest of the paper is organized as follows. Section 2 presents the proposed theoretical model to study energy consumption of cooperative caching as a function of dynamic page factor. We explain our Android prototype in Section 3 followed by data collection experiments in Section 4. Empirical results along with the proposed dynamic decision making algorithm and its performance evaluation are presented in Section 5 . Related work is analysed in Section 6 before concluding the paper in Section 7.

2 Theoretical analysis

The system model consists of two mobile devices each equipped with a LR and a SR interface. The mobiles communicate with the cellular base station using the LR interface, but uses the SR interface to communicate between them. For energy consumption modelling, we focus on one device, which we call device under evaluation (DUE). We assume that DUE is about to access a web page, which has been recently accessed and cached by the other mobile. We further assume that the page has multiple components of equal size, some of which are dynamic. We use the following notations:

- K: Total number of components in the page
- α: Dynamic page factor
- e_L: Average energy consumed by LR interface when receiving a component
- I_L: Average energy consumed by LR interface when a component is being received by SR interface
- e_S: Average energy consumed by SR interface when receiving a component
- I_S: Average energy consumed by SR interface when a component is being received by LR interface
- E^N: Energy consumed by DUE to download the entire page using non-cooperative browsing (LR remains turned off)
- E^C: Energy consumed by DUE to download the entire page using cooperative browsing (LR remains turned on)
- $G = E^N - E^C$: Energy saving due to cooperative browsing

Note that the following inequalities are assumed to hold: $e_L > I_L$, $e_S > I_S$, and $e_L > e_S$. Basically, these inequalities imply that (i) the SR has a smaller energy footprint than LR, and (ii) an interface consumes more energy when it is receiving a page component compared to when it is not receiving (the actual difference would vary depending on specific power management policies).

Because SR is turned off for non-cooperative browsing, all K components are received over the LR interface. Hence, E^N is simply given by:

$$E^N = K.e_L \tag{2.1}$$

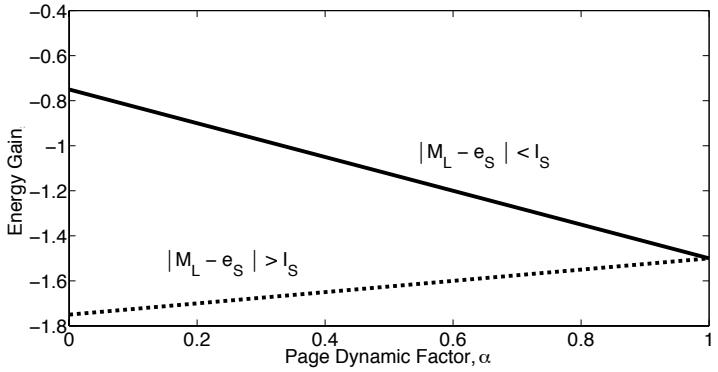


Figure 2.1: A numerical example of the case ($M_L < e_S$) when cache cooperation can never reduce browsing energy ($K = 50, e_L = 0.05, e_S = 0.04, I_S = 0.03$). $I_L = 0.025$ (for $|M_L - e_S| < I_S$) and $I_L = 0.045$ (for $|M_L - e_S| > I_S$).

For cooperative browsing, only the dynamic components are received over LR, while all other components are received over SR. E^C, therefore, is obtained as:

$$E^C = \alpha K.e_L + (1-\alpha)K.I_L + ((1-\alpha)K.e_S + \alpha K.I_S \tag{2.2}$$

Finally, through algebraic manipulation, it can be shown that:

$$G = K[(M_L - e_S)(1-\alpha) - \alpha I_S] \tag{2.3}$$

where $M_L = e_L - I_L$ is the marginal energy consumption of LR. Note that the actual value of M_L depends on what power level the LR interface can be brought to (and how fast) when it is not receiving a component, which would vary from one power management policy to another. For example, [10] shows that there are three different levels of power for a 3G interface and there are significant difference between the highest and the lowest levels as the lowest levels are usually near negligible. It is also known that a LR interface is usually power managed using a timer meaning that the interface may still remain at the highest power level for some time even when it ceases to receive data [17]. Therefore, I_L and hence M_L can vary considerably.

Due to these variations in M_L, we may observe two distinct cases:

- Case 1 ($M_L < e_S$): In this case, the marginal energy cost of LR is so small that G is always negative, irrespective of α. Therefore, no energy saving is possible in this case. Figure 2.1 shows a numerical example of this case, where we can see that G is either an increasing ($|M_L - e_S| > I_S$) or decreasing ($|M_L - e_S| < I_S$) function of α, but it remains negative for both conditions.

- Case 2 ($M_L > e_S$): Here we can achieve some energy saving as long as α remains less than a threshold value of α', where $\alpha' = \frac{M_L - e_S}{M_L - e_S + I_S}$. A numerical example in Figure 2.2 illustrates this case.

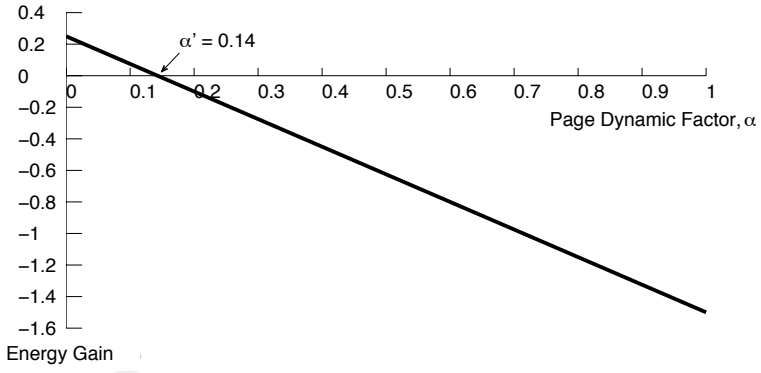


Figure 2.2: A numerical example of the case ($M_L > e_S$) when cache cooperation may help reduce browsing energy ($K = 50, e_L = 0.05, e_S = 0.04, I_S = 0.03, I_L = 0.005$)

If Bluetooth, which has a very small energy footprint compared to 3G, is used as SR, Case 2 is more likely to be observed than Case 1. Let us, therefore, examine the implications of Case 2. From Case 2, we can conclude the following:

- For a biased browser, i.e., where pages browsed predominantly have large ($> \alpha'$) or small ($< \alpha'$) page dynamic factors, expected performance of cooperative caching is very clear, it would be either positive energy saving or negative. A biased browser would therefore either choose non-cooperative or cooperative browsing depending on the estimated value of α'.
- For an unbiased browser, cooperative caching would yield positive energy saving if $\alpha' > 0.5$, negative otherwise. It is, therefore, unwise to select cooperative caching if α' is estimated to be less than 0.5. It should be noted, however, that this result is due to the linear dependency of G on α.

In the following sections, we are going to collect some practical data and examine how they fit to these theoretical observations. But first we are going to explain how we built our Android-based prototype, which was subsequently used for all data collection and experimentations.

3 Android Prototype

We have selected Android version 4.0 because it has all the required libraries to build cooperative browser caching. For example, shouldInterceptRequest(), which is needed to intercept all HTTP requests for searching the item in a neighbour's cache, is only available for Android 3.0 and above. We implemented a cooperative browser system, called CoopBrowser, by extending the default Android Webkit-based browser, WebView [2]. WebChromeClient and WebSettings are used to customize the WebViews features such as enabling JavaScript and cookies. The extended browser consists of four components, CBWebViewClient, ICBCacheManager, ConnectionManager and CBLogger. The interconnection of these components in an example setting is shown in Figure 3.1.

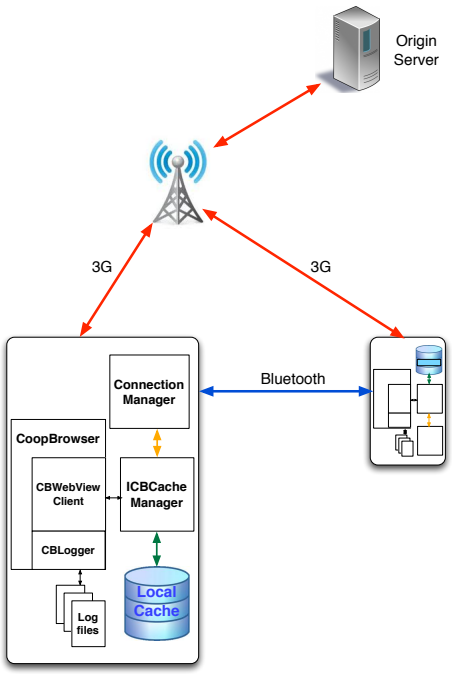


Figure 3.1: Android prototype architecture

CBWebViewClient handles all the HTTP requests and responses within the browser. It interacts with ICBCacheManager to obtain the requested items. ICBCacheManager, designed as an interface allows implementation of any cache management scheme in the application. We use this interface to implement two different caching schemes, non-cooperative and cooperative browser caching. The non-cooperative caching scheme implements the standard caching scheme, which only checks the local cache of the device and forwards the request to the origin server if there is a local cache miss. The cooperative caching scheme, on the other hand, searches the requested item in one-hop [1] neighbour devices' caches after a local cache miss.

Caches are implemented as dual-record cache items, where the first record holds the URL of a page component and the second holds the actual data. Therefore, to store a complete page consisting of many components, a large number of items will have to be cached. We have a cache size of 2MB which is sufficient to hold any of the 16 pages used in our experiments and data collection (most pages are smaller than 1MB and component sizes varied between 5-200 KB). Items are sorted with the least recently used (LRU) items stored at the bottom of the cache. When a new item is to be stored and the cache is full, the LRU item is replaced. When searching the cache, it is searched from the top (most recently used items are searched first). While items received over 3G are always cached, items received over Bluetooth from a peer is never cached to avoid data redundancy in the neighbourhood.

[1] We decided not to implement multi-hop search because of the additional complexity involved in wireless relaying and group management, which has the potential to mask some of the advantages of cooperative caching [11, 5]

ConnectionManager is designed as an interface to allow different implementations of peer-to-peer connections for cooperative cache management. Bluetooth is used as the connection as it is the most widely used mobile-to-mobile connectivity method [13].

Lastly, CBLogger is responsible for logging all relevant events such as the starting time of a page load, local cache request and global cache miss (i.e., the item is found neither in the local cache nor in the neighbours' caches). As an example, based on Log 1, in sequence, after a web address is entered in the address bar of the browser, CoopBrowser sends a HTTP request for all the contents of the pages at *11:29:13:586*. These requests are intercepted by the shouldInterceptRequest() function inside CBWebViewClient, which in turn calls ICBCacheManager. ICBCacheManager initiates its isLocalCacheHit() function at *11:29:13:594*, checks whether the requested resource exists in the local cache, and returns a miss at *11:29:13:596*. ICBCacheManager calls the function sendGlobalCacheRequest() to send global cache request to cooperative peers at *11:29:13:598*. This invokes the ConnectionManagers send() at *11:29:13:598*. If it receives a cache hit from a peer, it will return the retrieved resource to be displayed by CoopBrowser. However, if it is a cache miss, it will then request the resource from the origin server using HTTPUrlConnection. In each of the events, CBLogger will log them along with their related values. We use these log files to collect data as explained in Section 4.

```
1 2012.10.09-11:29:13:586 URL_LOAD_START http://gizmodo.com.au
2 2012.10.09-11:29:13:594 LOCAL_CACHE_REQUEST
      http://gizmodo.com.au/
3 2012.10.09-11:29:13:596 LOCAL_CACHE_MISS
      http://gizmodo.com.au/
4 2012.10.09-11:29:13:598 GLOBAL_CACHE_REQUEST
      http://gizmodo.com.au/
5 2012.10.09-11:29:14:321 GLOBAL_CACHE_MISS
      http://gizmodo.com.au/
6 2012.10.09-11:29:14:326 ORIGIN_SERVER_REQUEST
      http://gizmodo.com.au/ origin 4744
```

Log 1: Example of a CBLogger generated log file.

4 Data Collection

We use our Android-based prototype to collect the following data:

- Dynamic page factor (α): As explained earlier, CBLogger logs various events including which components were received over which interface (3G vs Bluetooth). When using cooperative caching, a component is considered dynamic if it is received over the 3G (it would have been found in the peer cache and received over Bluetooth otherwise). From these logs, we compute the total size of all dynamic components (D). page dynamic factor is then computed as $\alpha = \frac{D}{P}$, where P is the total page size. Table 4.1 shows the dynamic factor of all the pages used in our experiments and data collection.

Table 4.1: Variability in web page dynamic factor.

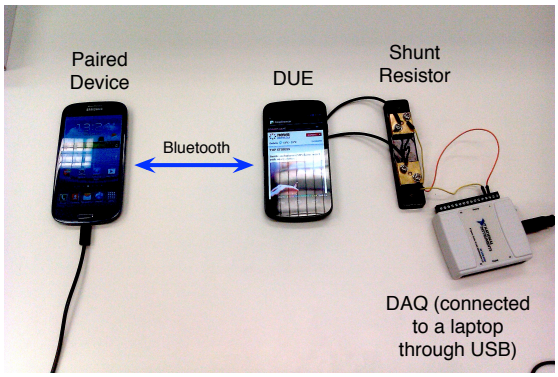
Web page	Total size of dynamic components (kB)	Total page size (kB)	Page dynamic factor
australia.gov.au	0.035	740.4	0.004727
www.m.webmd.com	0.043	892.0	0.004821
www.reddit.com/.compact	0.035	359.2	0.009743
m.usa.gov	0.086	366.0	0.0234
m.foxsports.com	0.557	575.6	0.0968
m.accenture.com	0.998	848.5	0.12
mobile.news.com.au	8.32	729.1	1.14
m.espn.go.com	11.896	566.1	2.1
m.weatherzone.com.au	7.557	287.0	2.63
m.9gag.com	52.0	977.6	5.32
www.unimelb.edu.au	38.9	478.6	8.12
www.nicta.com.au	117.4	1069.8	10.98
m.smh.com.au	214.2	899.3	23.82
www.lifehacker.com.au	322.3	1156.7	27.9
www.unsw.edu.au	249.2	849.3	29.3
www.engadget.com	242.3	574.7	42.17

- Page energy: Page energy is obtained as the total battery energy drawn during downloading of that page. We obtain page energy for both non-cooperative and cooperative caching. Energy measurement is achieved through a dedicated hardware that is interfaced with our Android prototype as explained below.

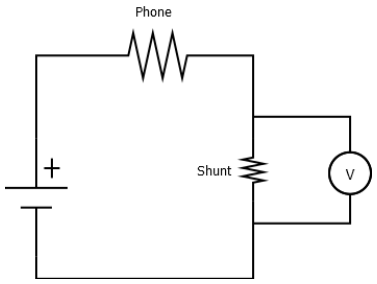
4.1 Energy Measurement

Although there exist several software power profilers for Android such as BatteryManager [1] and CurrentWidget [3], they only measure power values at fixed system dependent intervals. For example, BatteryManager only gives the current voltage reading whenever there is a change of a fixed percentage in the battery level. Such 'coarse' energy measurement does not meet our requirements since a page load causes a very small change in battery level and takes only a few seconds. To measure page energy, which requires a much finer energy measurement, we have built a custom hardware setup.

Figure 4.1a shows the hardware setup (and the corresponding circuit diagram in Figure 4.1b) used to measure energy consumption of the DUE during a given page download. The circuit setup treats the DUE's battery as an open battery, hijacked at one of its terminal and then connected in series with a shunt resistor



(a) Energy measurement hardware.



(b) Circuit diagram of energy measurement hardware.

Figure 4.1: Energy measurement.

so that the current through the circuit can be calculated later [1]. In Figure 4.1a, DUE is connected to the shunt resistor and the data acquisition device(DAQ), whereas the other device is used as a peer for cooperative browser caching.

The DAQ used in this circuit is the National Instrument's NI-USB 6008. It is connected in parallel to the shunt resistor to measure the voltage drop across it at a 1 kS/s sampling rate, a measurement every 1 milliseconds. These measurements are logged to the laptop (not shown) it is connected to through USB interface using NI LabView SignalExpress that logs all the measured voltage drop on the shunt resistor together with timestamps for each measurements into a comma separated value(CSV) file. From these values, the energy consumption of the Android device for a given time period t is calculated as follows:

$$I = V_r \times \frac{5}{50} \tag{4.1}$$

$$P = I \times V_b \tag{4.2}$$

$$E(t) = P \times t \tag{4.3}$$

where

[1] Similar setup has been used in the past by other researchers [16, 9, 18] to measure energy consumption of mobile devices.

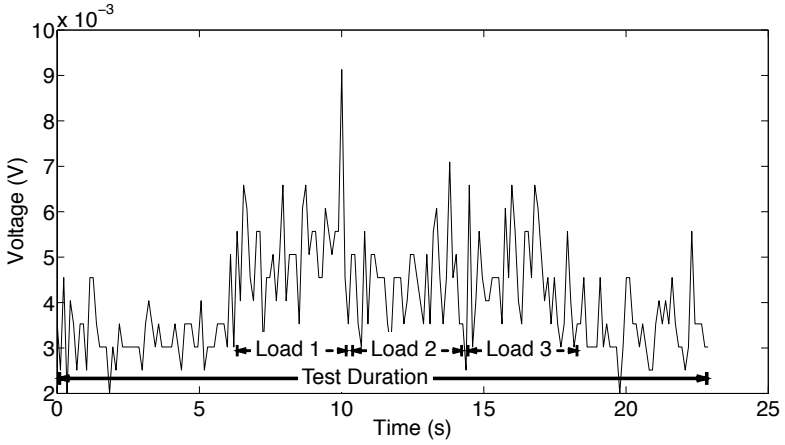


Figure 4.2: Voltage sample from data collection.

V_r = average voltage over t across the shunt resistor in mV

I = current across the circuit

$\frac{5}{50}$ = current rating of the shunt resistor

P = power rate in W

V_b = battery voltage in V

t = measurement time in s

$E(t)$ = energy consumed by the smartphone during the test in J

Figure 4.2 plots power values from one sample experiment where a single page is downloaded three times back to back. We can see that power consumption goes up each time a page is downloaded. It is important to note that page download in the Android and energy measurement at the DAQ are controlled separately. Therefore, the event log files generated by the CBLogger and the voltage log files generated by the DAQ needs to be time synchronized, so we can reliably match energy values to page downloads. We achieved time synchronization by running Network Time Protocol (NTP) [4] in both the laptop and the Android DUE. Furthermore, in order to ensure that the energy consumption value is not affected by other processes in the background, we have set the *background process limit* to 'no background processes' in Android's Developer options.

4.2 Experiment Design

A variety of pages have been used in the data collection since the number of dynamic components in a page varies between different web pages. Table 4.1

Table 4.2: Constant parameters of data collection experiments.

Parameter	Values
Ad-hoc connection	Bluetooth 3.0
Distance	approximately 15cm apart
Internet connection	Telstra 3G NextG HSPA 850 MHz network
Browser application	CoopBrowser
Test mobile device	Google Galaxy Nexus (Android 4.1) as the test device and Samsung Galaxy S3 (Android 4.0) as the paired device
Cache size	2 MB

gives an overview of the dynamic factors of the 16 web pages used in our evaluation. These pages are chosen based on their varying dynamic factors and different website categories (enterprise, news, sports, entertainment, education, portal and blogs) that are typically visited by mobile Internet users.

In the data collection, several fixed variables are identified and are ensured to be fixed in ensuring reliable sets of data are collected and preventing skewed results. Table 4.2 shows the parameters that are kept constant throughout the data collection process.

We first collected data at a residential location in Sydney, Australia. For each experiment, a test page is loaded three times back to back and the relevant data, such as page energy, is averaged over these three downloads. It is also important that the cache is always cleared after each page load to ensure that no page components in the consequent loads are loaded from local cache due to previous loads. Bluetooth is only turned on when running the test with cooperative browser caching scheme.

For each test pages, the page is first loaded using non-cooperative browser caching followed by cooperative browser caching. Before loading using CBC, it is ensured that the page has been loaded on the paired device to ensure that the DUE can obtain static components of the test page from the paired device over Bluetooth while dynamic components will be downloaded over 3G as a result of cache miss from the paired device.

After the tests have finished, we obtained log files for each test pages from the DUE and DAQ. There are a total of 16 CBLog files and 16 DAQ log files for each category (non-cooperative vs cooperative) of experiments. A post processing application uses these log files to mine all important data for analysis such as dynamic factor and page energy.

Data are collected within a three hour time window on a same day where each test pages are run back to back to ensure they belong to the same time window and as a result, in the same network state or traffic. After completing data collection in the residential area, we repeat the data collection on another day at a different location, namely in the headquarter of National ICT Australia (NICTA) in Sydney. We create a third data set by combining all data from these two locations.

5 Data Analysis and Results

5.1 Correlation between energy and page dynamic factor

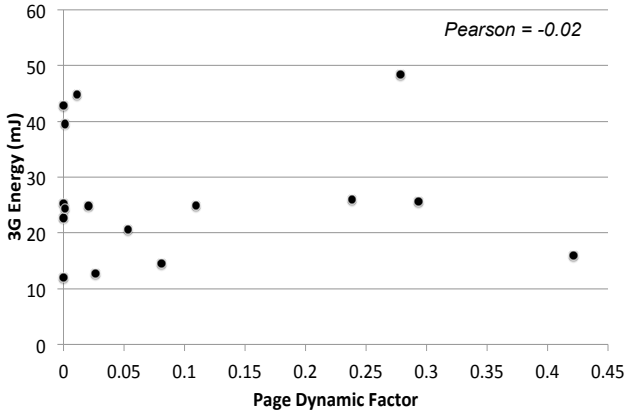
In this section, we analyse empirical data that we collected using our Android prototype. Figures 5.1 and 5.2 plot energy consumption of non-cooperative and cooperative caching, respectively, as a function of dynamic page factor α. It is clear that there is no correlation (Pearson correlation coefficient is close to zero) between energy consumption and α for non-cooperative caching, which is an expected result as all page components are to be received over the 3G interface irrespective of α (Bluetooth is turned off in this case). We, however, do see significant positive correlation (Pearson correlation coefficient is well above zero) for cooperative caching, suggesting that a decreasing α would result in deceasing energy consumption. Therefore, we can expect that a decreasing alpha would increase the energy difference between non-cooperative and cooperative caching.

Indeed, when energy saving is plotted against dynamic factor (Figure 5.3), we find that energy saving is strongly negatively correlated with dynamic factor confirming the theoretical result that we discussed earlier (Section 2) for a simple model of equal-size page components. We also find that that our empirical data fits the second case of the theoretical analysis, i.e., the case when marginal energy cost of 3G is larger than the energy cost of receiving data over Bluetooth ($M_L > e_S$). As discussed earlier, in this case we can expect to see some positive energy saving due to cooperation as long as the percentage of dynamic components in the page does not exceed a threshold. Figure 5.3 confirms this behaviour, but it also shows that the threshold, α', is very small (around 0.03). Such a small α' means that for many of today's web pages with dynamic components, cooperative caching may not yield any energy saving.

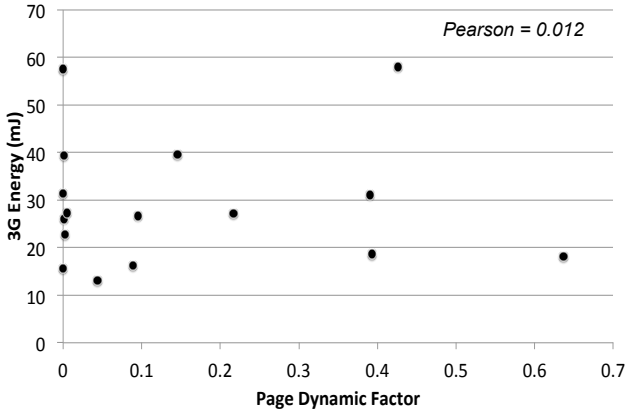
We, however, notice that energy gain decreases *logarithmically* instead of *linearly* as predicted earlier by the theoretical model. This is perhaps due to the fact that the theoretical model is based on equal-size components, whereas the component sizes vary over different pages (see Table 4.1). Therefore, the model outcome, which stipulates that cooperative caching would yield negative energy saving for $\alpha' < 0.5$ may not hold anymore. We investigate this issue in the following section.

5.2 Static Decision Making

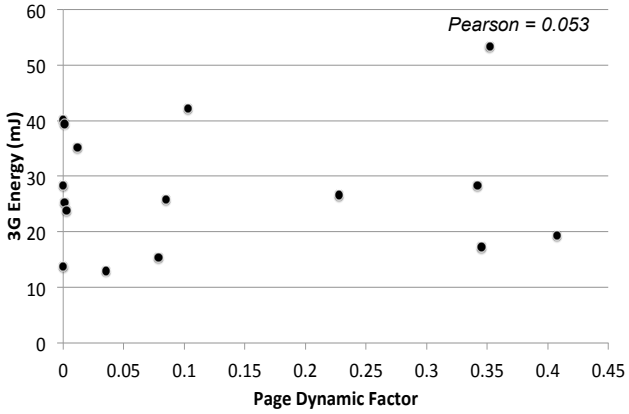
In static decision making, the decision to use either non-cooperative or cooperative caching is made once at the beginning of a browsing session that consists of multiple page downloads. The Bluetooth interface is turned on or off throughout the session according to this decision. We compare total energy consumption between these two caching schemes for a browser session consisting of the 16 pages we collected energy data for (Figure 5.4). We can see that despite having a very small $\alpha' \approx 0.03$, cooperative browsing consumes about 13% less energy than non-cooperative browsing. This is because, for our data, the positive gains are higher in magnitude than the negative gains.



(a) Residential

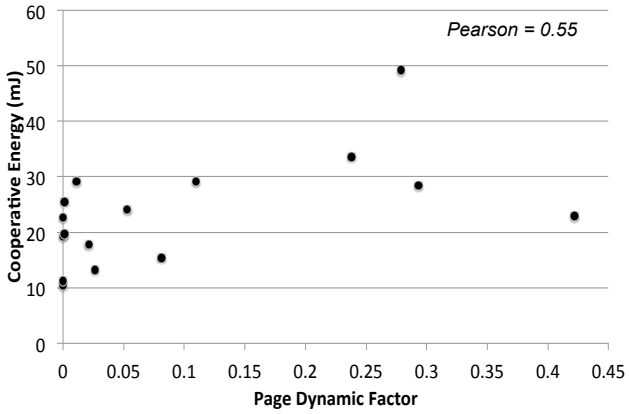


(b) NICTA

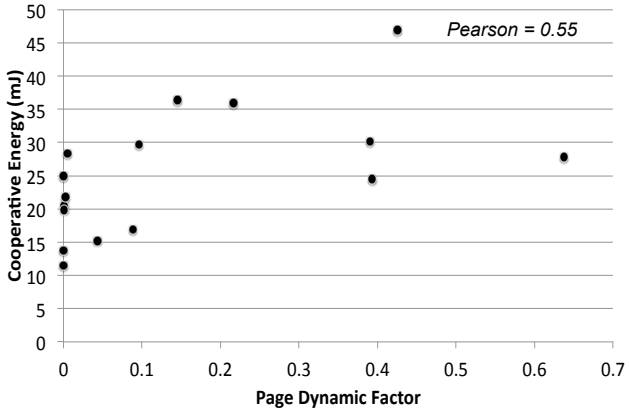


(c) Combined

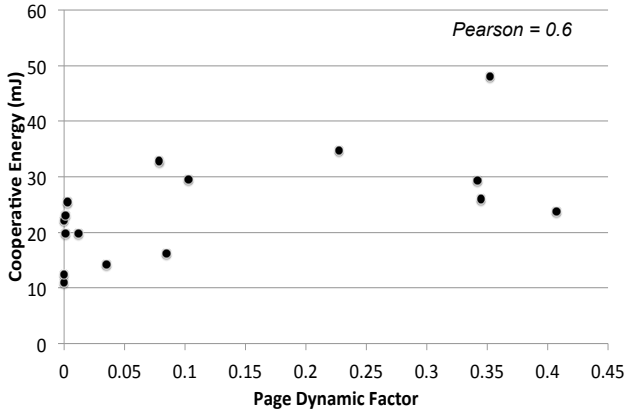
Figure 5.1: Correlation between non-cooperative energy consumption and page dynamic factor



(a) Residential

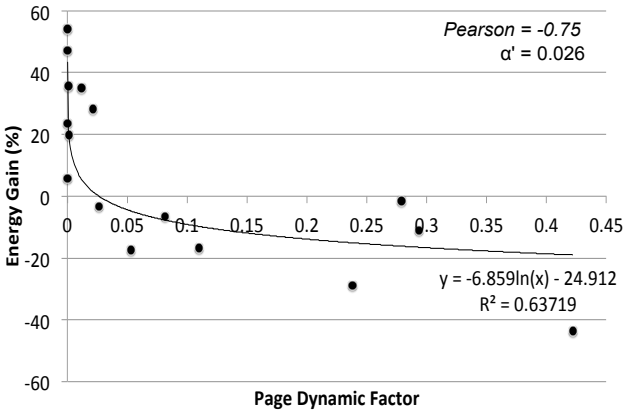


(b) NICTA

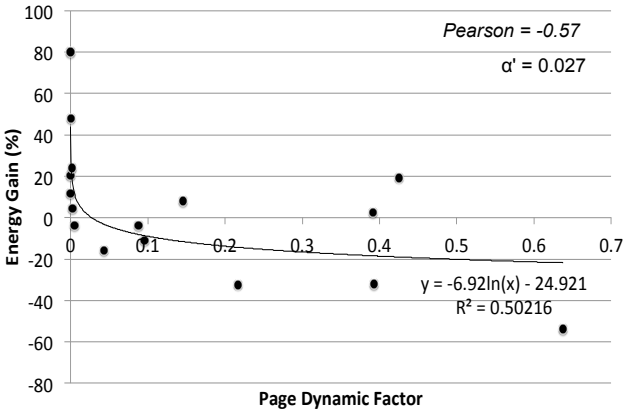


(c) Combined

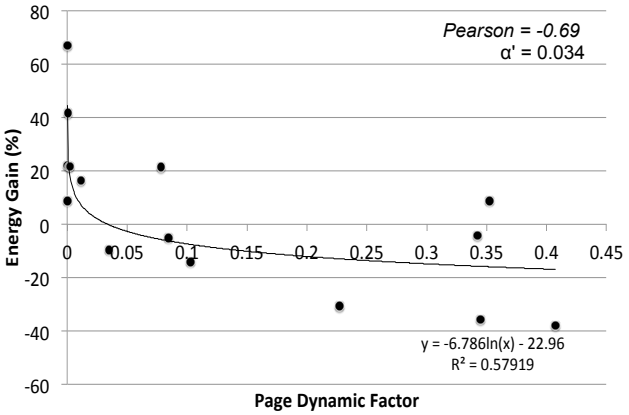
Figure 5.2: Correlation between energy consumption and page dynamic factor for cooperative caching.



(a) Residential



(b) NICTA



(c) Combined

Figure 5.3: Energy gain from cache cooperation as a function of page dynamic factor.

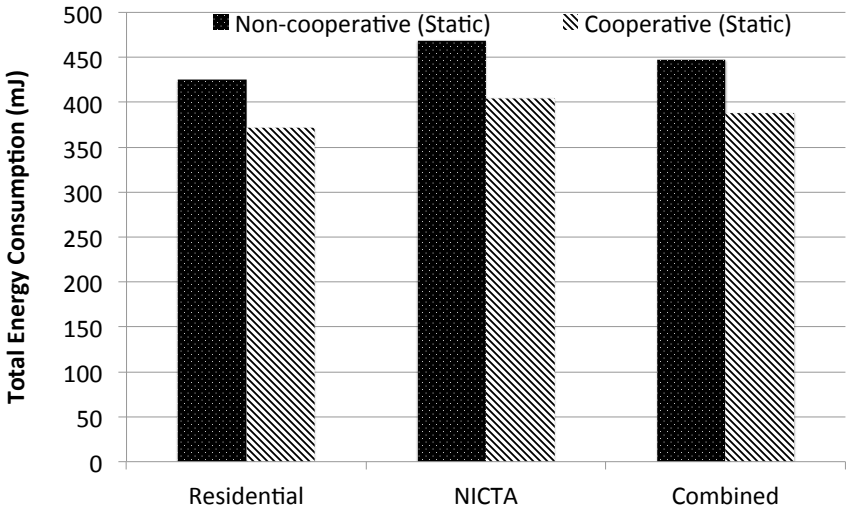


Figure 5.4: Comparison of total energy consumption between static non-cooperative and cooperative caching scheme.

5.3 Dynamic Decision Making

In this section, we investigate and evaluate a dynamic decision making (DDM) algorithm, which makes decision about whether to use cooperative caching (and whether to turn on the Bluetooth interface) dynamically page-by-page based on the predicted α of a page and a known α', which can be learned through historical data. The DDM algorithm works as follows. When it receives the complete list of components (list of URLs) from the origin server, it predicts the α for that page and decides to use cooperative caching (turns on Bluetooth) if $\alpha < \alpha'$, use non-cooperative (turn off Bluetooth) otherwise.

We consider two versions of this algorithm. For the ideal DDM, we assume that the browser can predict α very accurately, which would be possible if the origin server provided clear indication of which component is dynamic and what their sizes are, for example. However, with current version of HTTP, the browser cannot obtain these parameters, making ideal DDM less practical at the moment. We, therefore, propose a more practical algorithm, which we call *string matching and number of components based DDM* (SN-DDM). SN-DDM uses the strings "ad" and "analytic" as the targeted keywords since we have found most dynamic components contain these keywords in their URLs. Page dynamic factor is then estimated as the fraction of number of dynamic components to the total number of components in the page (assuming all components have equal size).

Total energy consumed by the proposed DDM scheme for a given browsing session consisting of loading M web pages is derived as:

$$E^D = \sum^{M} E_i \tag{5.1}$$

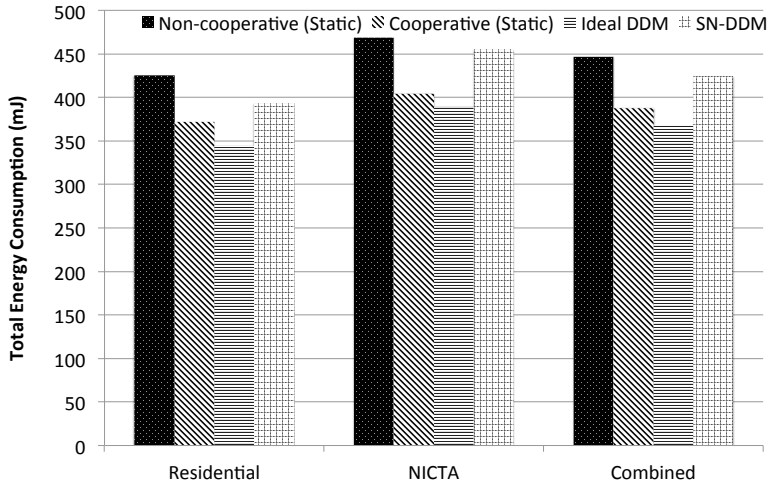


Figure 5.5: Comparison of dynamic decision making algorithms.

where the energy consumed in loading the ith page is

$$E_i = \begin{cases} E^C & \text{if } \alpha < \alpha' \\ E^N & \text{otherwise} \end{cases} \tag{5.2}$$

We apply both ideal and SN-DDM to our log files to obtain the total energy consumption for browsing the 16 pages. Figure 5.5 shows the results for both dynamic and static decision making. We see that ideal DDM achieves a 17% reduction in energy consumption compared to non-cooperative and it also outperforms cooperative static decision making (by 4.7%). However, SN-DDM performs worse than both ideal DDM and the cooperative static decision making. The reason for the poor performance of SN-DDM is explained below.

Figure 5.6 shows the comparison of computed dynamic factor of using both ideal and SN-DDM scheme. It is clear that SN-DDM suffers from significant prediction inaccuracy for the page dynamic factor, which ultimately leads to poor energy performance. From our analysis, we have found that there are false detections of dynamic components in SN-DDM, which relies on only "ad" and "analytics" strings to identify a dynamic component. False positives occurred for URLs with words such as "loaders" and "uploads", that contain the targeted strings, but actually are not dynamic components. On the other hand, false negatives occurred for components that are dynamic due to "no-cache" value in the HHTP response header from the origin server, but contain neither "ad" nor "analytics". Clearly, accurate prediction of dynamic page factor is a challenging problem, which requires further research.

6 Related Work

Cooperative browser caching is a popular research area and field due to its potential in improving mobile Internet browsing despite of various challenges in the method itself. Although handful of cooperative browser caching scheme has

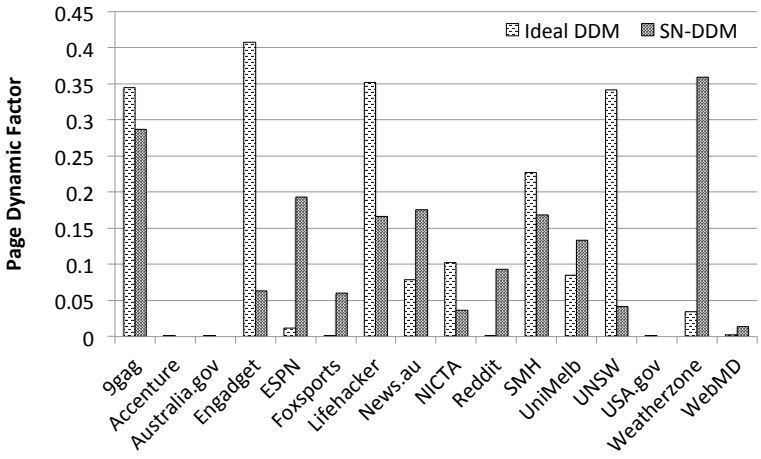


Figure 5.6: Inaccuracies in predicting page dynamic factor using SN-DDM.

been proposed in the past [12, 8, 5] with most of them however, only focusing on other performance metrics such as latency and cache hit ratio rather than energy efficiency. Among these, several energy efficient cooperative browser caching algorithms has been introduced in the past such as ECORP [7], PReCinCt [15] and by Sailhan et al. [14]. Although they did not focus on what factor affects the energy efficiency of cooperative browser caching, they did found factors affecting it along the way. For example, Huaping et al. [15] found that energy savings depends on the number of devices in the cooperative group.

Furthermore, these past researches uses simulations such as on CSIM [12, 8], ns2 [15] and trace simulations [7] to analyse and quantify the energy efficiency of their cooperative browser cache scheme. Although theoretically it is correct, however, practical analysis is sometimes more favorable since it allows the evaluation to include random variables in controlled environment and provide a general picture of a practical efficiency of the evaluated scheme. Hence, using a prototype implementation on Android mobile devices, we offer a practical analysis of the energy efficiency of cooperative browser caching.

7 Conclusion

Using both analysis and practical experiments with typical web pages, we have demonstrated that the dynamic page factor has a major influence on the energy consumption of cache cooperation between mobile devices. Our results show that cooperative browsing reduces energy cost only if the dynamic page factor is below a threshold and that this threshold is very small (about 0.03). This finding suggests that cooperative browsing may be counter productive if browsing has a bias toward pages with large dynamic page factors and vice-versa. For unbiased browsing, we find that Bluetooth-based cache cooperation can reduce 3G browsing energy consumption by 13%. If, however, the device switched between non-cooperative and cooperative browsing adaptively based on the dynamic page factor of every page that is downloaded, cooperative browsing could potentially achieve a 17% reduction in energy consumption. However,

for such dynamic decision making, one would need a mechanism to predict the page dynamic factor in advance, which would be an interesting future work to pursue.

Bibliography

[1] Android SDK reference of BatteryManager. `http://developer.android.com/reference/android/os/BatteryManager.html`.

[2] Android SDK reference of WebView. `http://developer.android.com/reference/android/webkit/WebView.html`.

[3] currentwidget : An Android widget that display the electric current usage of the device. `http://code.google.com/p/currentwidget/`.

[4] Network Time Protocol page. `http://www.ntp.org`.

[5] Y. Abdelmalek, A. Abd El Al, and T. Saadawi. Collaborative Content Caching Algorithms in Mobile Ad Hoc Networks Environment. In *International Symposium on Integrated Network Management (IM)*, pages 311–314, New York, USA, June 2009.

[6] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *ICM '09*, Chicago, USA, November 2009.

[7] Edward Chan, Wenzhong Li, and Daoxu Chen. Energy saving strategies for cooperative cache replacement in mobile ad hoc networks. *Pervasive and Mobile Computing*, 5:77–92, 2009.

[8] Chi-Yin Chow, Hong Va Leong, and Alvin Chan. Peer-to-peer cooperative caching in mobile environments. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, Tokyo, Japan, March 2004.

[9] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile systems, applications, and services*, MobiSys '10, pages 49–62, New York, USA, 2010.

[10] Shuo Deng. Reducing 3G Energy Consumption on Mobile Devices. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, February 2012.

[11] Jingwei Li and S-H. Gary Chan. Optimizing Segment Caching for Mobile Peer-to-Peer Interactive Streaming. In *International Conference on Communications (ICC)*, pages 1–5, Cape Town, South Africa, May 2010.

[12] S. Lim, W.-C. Lee, G. Cao, and C.R. Das. A novel caching scheme for internet based mobile ad hoc networks. In *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on*, pages 38 – 43, October 2003.

[13] Beatrice Pietrarca, Giovanni Sasso, Gian Paolo Perruci, and Frank H. P. Fitzek. Measurement Campaign on Connectivity of Mesh Networks formed by Mobile Devices. In *International Conference on Mobile Adhoc and Sensor System (MASS)*, pages 1–6, Pisa, Italy, October 2007.

[14] Franoise Sailhan and Valrie Issarny. Cooperative caching in ad hoc networks. In Ming-Syan Chen, PanosK. Chrysanthis, Morris Sloman, and Arkady Zaslavsky, editors, *Mobile Data Management*, volume 2574 of *Lecture Notes in Computer Science*, pages 13–28. Springer Berlin Heidelberg, 2003.

[15] Huaping Shen, M.S. Joseph, M. Kumar, and S.K. Das. Precinct: A scheme for cooperative caching in mobile peer-to-peer systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, page 57a, Denver, USA, April 2005.

[16] Narendran Thiagarajan, Gaurav Aggarwal, and Angela Nicoara. Who Killed My Battery : Analyzing Mobile Browser Energy Consumption. In *Proceedings of WWW 2012 Session : Mobile Web Performance*, Lyon, France, April 2012.

[17] Jul-Hung Yeh, Chi-Chen Lee, and Jyh-Cheng Chen. Performance analysis of energy consumption in 3gpp networks. In *Wireless Telecommunications Symposium, 2004*, pages 67 – 72, Pomona, USA, May 2004.

[18] Bo Zhao, Byung Chul Tak, and Guohong Cao. Reducing the delay and power consumption of web browsing on smartphones in 3g networks. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS)*, Minneapolis, USA, June 2011.